5-2013

Compound identification using penalized linear regression.

Ruiqi Liu
University of Louisville

Recommended Citation

Liu, Ruiqi, "Compound identification using penalized linear regression." (2013). *Electronic Theses and Dissertations.* Paper 844.
https://doi.org/10.18297/etd/844

COMPOUND IDENTIFICATION USING PENALIZED LINEAR REGRESSION

By

Ruiqi Liu
B.S., China Jiliang University, 2010

A Thesis
Submitted to the Faculty of the
School of Public Health and Information Sciences of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science

Department of Bioinformatics and Biostatistics
School of Public Health and Information Sciences
University of Louisville
Louisville, Kentucky

May 2013

COMPOUND IDENTIFICATION USING PENALIZED LINEAR REGRESSION

By

Ruiqi Liu
B.S., China Jiliang University, 2010

A Thesis Approved on

April 17, 2013

by the following Thesis Committee:

Dr. Seongho Kim (Thesis Director)

Dr. Dongfeng Wu

Dr. Xiang Zhang

ACKNOWLEDGMENTS

I would like to thank all the people who made this work possible. First, I would like to thank my principle advisor, Dr. Seongho Kim, for his guidance and patience that he gave me throughout this research project. I would not have accomplished this project without his help and support. I would also like to thank Dr. Xiang Zhang for giving me an opportunity to work on this project. I appreciate his moral and financial support during this eight months. Also many thanks to Dr. Dongfeng Wu, for being my thesis committee member and assisting and supporting me during my Master's study at University of Louisville.

ABSTRACT

COMPOUND IDENTIFICATION USING PENALIZED LINEAR REGRESSION

Ruiqi Liu

April 17, 2013

In this study, we propose a new method for compound identification using penalized linear regression. Compound identification is often achieved by matching the experimental mass spectra to the mass spectra stored in a reference library based on mass spectral similarity. In the context of the linear regression, the response variable is an experimental mass spectrum (i.e., query) and all the compounds in the reference library are the independent variables. However, the number of compounds in the reference library is much larger than the range of m/z values so that the data become high dimensional data with suffering from singularity. For this reason, we use penalized linear regression such as ridge regression and the Lasso. Furthermore, we also propose two-step approaches using dot product and Pearson's correlation along with the penalized linear regression in this study.

TABLE OF CONTENTS

PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Metabolomics has been raised in recent years, it is an important part of systems biology. Metabolites can directly reflect the environmental conditions of cells, which have strong connections of nutritional status of cells, effects of medicines and environmental contaminants and influences from other factors. Metabolites are the end products of cellular regulatory processes, and their levels can be regarded as the ultimate response of biological systems to genetic or environmental changes.[1] Nowadays, the advance of analytical instruments, new software tools and algorithms has enabled new strategies for separation and identification of a manifold of individual metabolites. To this end, comprehensive analysis using gas chromatography coupled with mass spectrometry (MS) has been used as a ‘‘gold standard’’ in researches involving primary metabolism analysis.[2]

One of the critical analyses on GC-MS data is compound identification, which is often achieved by matching the experimental mass spectra to the mass spectra stored in a reference library based on mass spectral similarity.[3] To increase the accuracy of compound identification, various methods for the calculation of mass spectral similarity scores have been developed, including dot product,[4,5,6,7] composite similarity,[8] probability-based matching system,[9] Hertz similarity index,[10] normalized Euclidean distance (L_2-norm),[8,11,12] absolute value distance (L_1-norm) [5,12], and Fourier and wavelet-based composite similarity.[13]

Since some compounds have mass spectral information that is similar to that of other compounds, an experimental query spectrum of these compounds is often matched to multiple mass spectra in the reference library with high similarity scores, resulting in impeding the high confidence compound identification. In other words, the mass spectral similarity score of a true positive pair does not always have the top ranked score; and it is instead ranked as the second- or even the third-highest similarity score with an ignorable difference from the top-ranked score.

In order to circumvent the above issue, Kim et al. recently developed a novel similarity measure using partial and semi-partial correlations.[3] The partial correlation can be seen as the pure relation between two random variables after removing the effect of other random variables. While the semi-partial correlation eliminates the effect of a fraction of other random variables, in other words, just removing the effect of one random variable from a total of two random variables. When it comes to compound identification, these partial and semi-partial correlations can be applied to calculate the mass spectral similarity score. By removing the effect of other mass spectra over the two mass spectra of interest, the unique relationship between the mass spectra can be extracted. Using partial and semi-partial correlations can obtain high accuracy of compound identification. However, the performance of this method suffers from expensive calculation since the data are high-dimensional, which propels us to search for an alternative for compound identification.

Another way for compound identification is to use the multiple ordinary linear regression-based methods. In the context of linear regression, the response variable is an experimental mass spectrum (i.e., query) and all the compounds in the reference library are the independent variables. Each regression coefficient reflects the strength of their relationship with the response variable, so we could match the experimental compound with the reference compound which shows the strongest connection. In particular, the coefficients of the multiple ordinary linear regressions are proportional to the semi-partial correlation coefficient, meaning that both methods will give us the same result if the maximal coefficient is only considered. In other words, the ordinary linear regression is a great alternative to semi-partial correlation-based compound identification.

However, it is not feasible to apply ordinary linear regression in compound identification for two reasons. First, our data are high-dimensional data. Usually, the size of a reference library is much larger than the range of m/z values and the number of variables becomes much larger than the number of samples so that the ordinary linear regression will suffer from singularity. Second, it is possible that different compounds have identical mass spectra, such as isomers. Because of the existence of isomers, several predictors are highly correlated to each other so that their correlation coefficients become one. This also causes ordinary linear regression to suffer from singularity.

In order to elude this difficulty, we introduce penalized linear regression for the compound identification. Penalized linear regression can deal with high-dimensional data, and it is a trade-off between unbiasedness and a smaller estimation variance by putting a penalty constrain on coefficients. Different types of constrains will result in Lasso and ridge regression, which have L_1-norm and L_2-norm penalties, respectively. To improve the performance of penalized linear regression, we also present two-step approaches, using widely used mass spectral similarity scoring methods, dot product or Pearson's correlations as the first step, and then penalized linear regression as the second step.

Using the NIST mass spectral library, we further compare our proposed penalized linear regression approaches and two-step approaches with the dot product and Pearson's correlation for the accuracy of compound identification.

CHAPTER II

METHODS

2.1 Library-based compound identification

Library-based compound identification is achieved by matching the experimental mass spectra to the mass spectra stored in a reference library based on mass spectral similarity. In other words, all the pairwise similarity scores between an experimental mass spectrum and all library mass spectra in the library are first calculated. The library mass spectrum having the highest mass spectral similarity score will be assigned to the experimental mass spectrum. Each mass spectrum is composed of m/z values and their intensities, as shown in Figure 1. The intensities are used for calculation of the spectral similarity scores.

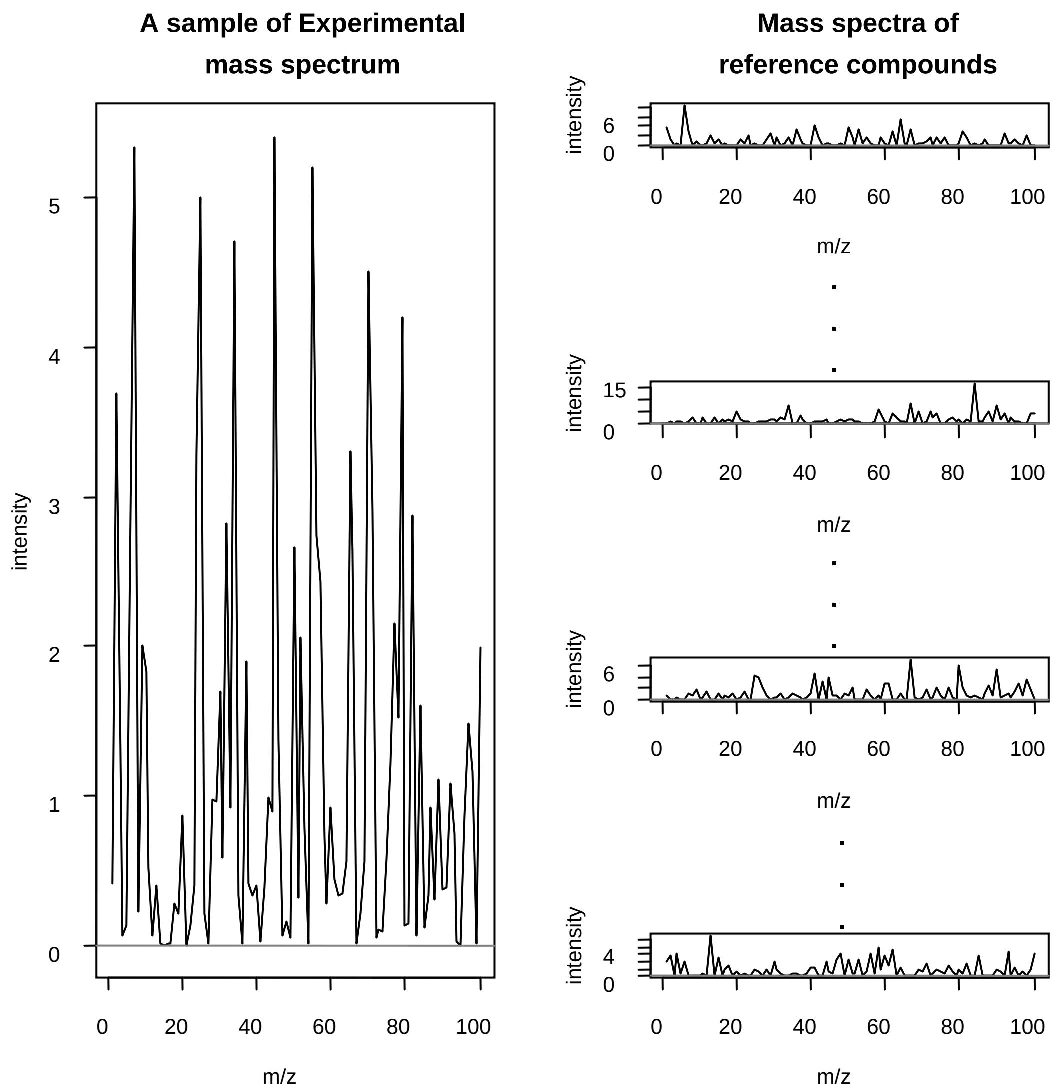

Figure 1. Mass spectral library-based compound identification.

In this study, the spectral similarity between experimental mass spectrum and each of the reference spectra is calculated. A reference compound is considered as the compound given rise to the experimental spectrum if its reference spectrum has the best similarity with the experimental spectrum. The following methods are applied to calculate the similarity scores between the experimental mass spectrum and each reference spectrum.

2.2 Dot product

The dot product, which is also known as the cosine correlation,[14] was used to obtain the cosine of the angle between two sequences of intensities, **x** = $(x_i)_{i=1, \dots, n}$ and y = $(y_i)_{i=1,\dots, n}$. It is defined as

$$S = S(x, y) = \frac{x^T y}{\| x \| \cdot \| y \|}$$

where $x^T y = \sum_{i=1}^{n} x_i y_i$, and $\|x\| = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}$.

We calculate the dot product of mass spectra for each experimental compound and each reference compound, and a greater value indicates a higher chance that the reference compound is the compound that gave rise to the experimental mass spectrum.

2.3 Ridge regression

Ridge regression is a shrinkage method which imposes a penalty on the size of regression coefficients. The ridge coefficients minimize a penalized residual sum of squares,

$$\beta^{ridge} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}.$$

Here λ ≥ 0, it is a complexity parameter and controls the amount of shrinkage, a larger value of λ will result in a greater amount of shrinkage. The coefficients are shrunk toward zero (and each other).[15] An equivalent way is to solve the following ridge problem,

$$\beta^{ridge} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j)^2,$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 \le t .$$

This form makes the size constraint on the parameters explicit, and there is a one-to-one correspondence between the parameters λ and t.

For ridge regression, we can also write the above criterion in matrix form, the ridge regression can be easily solved as

$$\beta^{ridge} = (X^T X + \lambda I)^{-1} X^T y ,$$

p is the number of variables, N is the number of observations, **I** is the p×p identity matrix. In our case, $p \gg N$, we use the singular-value decomposition of **X**,

$$\mathbf{X=UDV^T=RV^T}$$

to calculate the coefficients. **V** is p × N with orthonormal columns, U is N × N orthogonal, and **D** is a diagonal matrix with elements $d_1 \ge d_2 \ge d_N \ge 0$. The matrix **R** is N ×N, with rows r_i^T. Replacing **X** by **RV**^{T} and we have

$$\beta^{ridge} = V(R^T R + \lambda I)^{-1} R^T y .$$

2.4 Lasso regression

Lasso (Least Absolute Shrinkage and Selection Operator) regression was proposed by Tibshirani (1996), it is a shrinkage method like ridge, but it has subtle but important differences from ridge regression. The Lasso is a penalized least squares procedure that minimizes residual sum of squares (RSS) subject to the non-differentiable constraint expressed in terms of the L_1 norm of the coefficients.[16] That is, the Lasso estimator is given by

$$\beta^{lasso} = \arg\min_{\beta}\left\{\frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\left|\beta_j\right|\right\}.$$

For the Lasso, it uses L_1 Lasso penalty $\sum_{j=1}^{p}\left|\beta_j\right|$ to replace the L_2 ridge penalty $\sum_{j=1}^{p}\beta_j^2$. The L_1 norm constraint makes the solutions nonlinear in the y_i, and there is no easy form expression as in ridge regression.

2.5 Two-step approaches

To maximize the performance of compound identification and also reduce the data dimensionality, we propose two-step approaches by combining the dot product, Pearson's correlation and penalized linear regression. In the two-step procedure, the first step is made to proceed "the first match". Based on the first step, we then select a certain amount of the best "matches" and use them to conduct the second step—penalized linear regression.

2.5.1 Dot product and ridge/Lasso regression

In this two-step approach, after calculating the dot product of mass spectra for all the experimental compound and reference compound, we then rank the results of dot product and choose N reference compounds with top N largest dot product value. Then conduct ridge/ Lasso regression with only this N reference compounds. Normally, we could choose N=25, 50, 100 and so on. The flowchart is shown below.

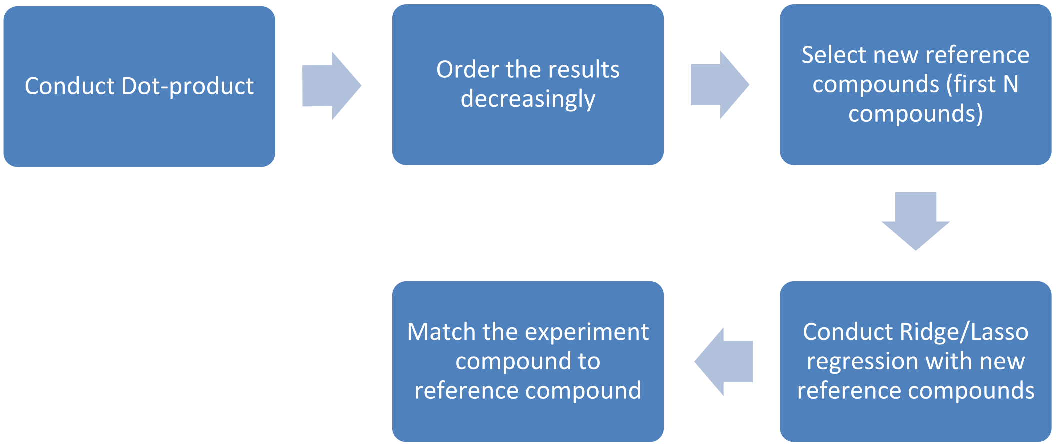
Conduct Dot-product
Order the results decreasingly
Select new reference compounds (first N compounds)
Conduct Ridge/Lasso regression with new reference compounds
Match the experiment compound to reference compound

Figure 2. Workflow of proposed two-step approach using dot product and ridge/Lasso regression.

2.5.2 Pearson's correlation and ridge/Lasso regression

In this two-step approach, after calculating the Pearson's correlation coefficients of a experimental spectrum and all the reference spectra, we order the correlation coefficients decreasingly, and calculate their (1-α)% confidence intervals respectively. Then we check if there is overlap between two adjacent intervals from the top, and stop at the Nth compound, if there is no overlap between the Nth interval and (N+1)th interval. By applying the selecting method stated above, we select N reference compounds then conduct ridge/ Lasso regression only with these N reference compounds. Normally, we could change α and obtain different amount of reference compounds for the second penalized regression step. The flow chart is shown below.

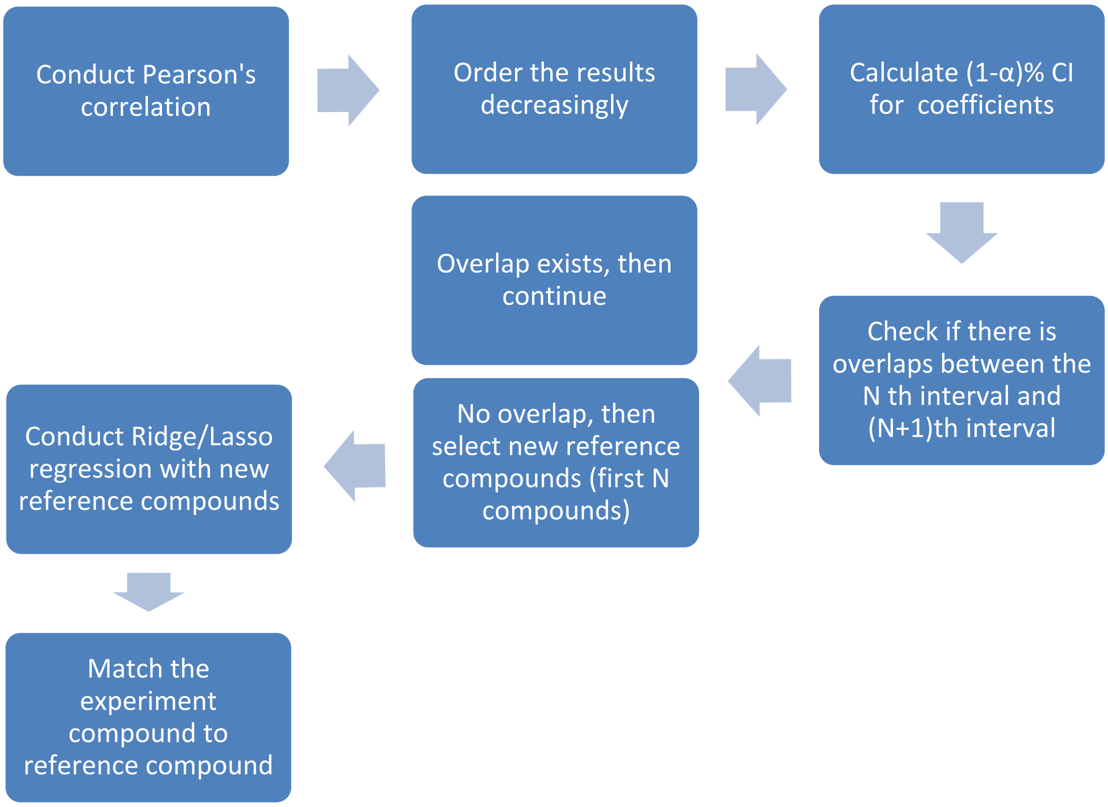
Conduct Pearson's correlation
Order the results decreasingly
Calculate (1-α)% CI for coefficients
Check if there is overlaps between the N th interval and (N+1)th interval
Overlap exists, then continue
No overlap, then select new reference compounds (first N compounds)
Conduct Ridge/Lasso regression with new reference compounds
Match the experiment compound to reference compound

Figure 3. Workflow of proposed two-step approach using Pearson's correlations and ridge/Lasso regression.

CHAPTER III

EXPERIMENTAL SECTION

3.1 Data

We use NIST (National Institute of Standards and Technology) mass spectrometry and repetitive database as the reference database and experimental spectra, respectively. The NIST Chemistry WebBook service (http://webbook.nist.gov/chemistry/) provides users with chemical and physical information for chemical compounds, including mass spectra generated by electron ionization mass spectrometry.[17]

For our reference library, the mass spectra of 2739 compounds were extracted from NIST Chemistry WebBook database. The fragment ion m/z values ranged from 1 to 1036 with a bin size of 1. The experimental library contains 1530 mass spectra of compounds extracted from the repetitive database.

The same chemical compounds are identified and grouped by Chemical Abstracts Service (CAS) registry number. In the simulation studies, we consider the mass spectra extracted from the NIST Chemistry WebBook (NIST library) as a reference library and the repetitive library as query (experimental) data. In addition, since we assume that the NIST library has the mass spectrum information for all the experimental compounds, all the compounds that were not present in the NIST library were removed from the repetitive library.

3.2 Performance evaluation

To evaluate the performance of compound identification of each similarity measure, we calculated the accuracy. The accuracy is the proportion of the spectra identified correctly in query data. In other words, if a pair of unknown and reference spectra have the same CAS index, we consider this pair as the correct match and if otherwise as the incorrect match. Then by counting all the correct matches, the accuracy of identification can be calculated by

$$\text{accuracy} = \frac{\text{number of spectra matched correctly}}{\text{number of spectra queried}}$$

3.3 Software

All the statistical analyses were performed using statistical software R version 2.15.3 (The R Foundation for Statistical Computing 2013). The comparison of the ridge and Lasso regression was performed by the R package *glmnet*.

CHAPTER IV

RESULTS

4.1 Comparison between ridge regression and Lasso regression

Since there is no easy solution form for the Lasso regression, to compare with ridge regression, we conducted these two penalized regressions using R package *glmnet*, whose calculation time is relatively shorter than those of others. We use 100 same shrinkage factor λ (range from 0.0001 to 1000000) to proceed ridge and Lasso regression, and calculated the correct matching and the accuracy for both regressions. The figure below shows the change of accuracy along with different shrinkage factor value for the two penalized linear regression.

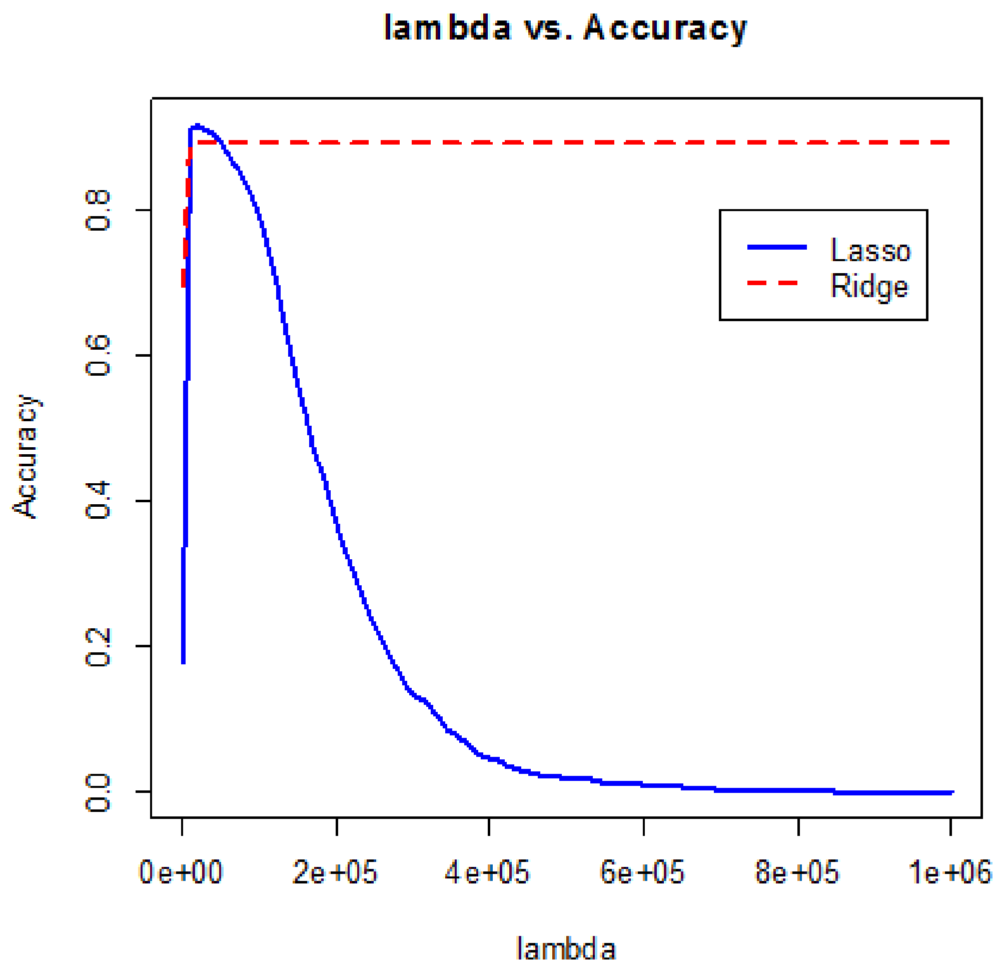

Figure 4. Lambda vs. Accuracy for ridge regression and Lasso regression.

From the figure shown above, we find the accuracy trend for Lasso regression is very different from ridge regression. When lambda value gets greater, though their accuracy both tend to be constant, the accuracy for Lasso regression tends to be 0 but for ridge regression, it levels off at 89.20%.

We then applied ridge regression and Lasso regression respectively to further check the specific trends of each regression.

4.2 Ridge regression

After conducting a ridge regression between query data and reference data with

100 different shrinkage factor λ (ranging from 0.1 to 5000), we calculated the correct matching and the accuracy. The best 10 results are shown in Table 1.

Table 1. Top 10 best accuracy and corresponding shrinkage factor for ridge regression.

Shrinkage factor (λ)	Number of query	Number of correct match	Accuracy
1363.70909	1530	1373	89.74%
1111.18889	1530	1372	89.67%
1161.69293	1530	1372	89.67%
1212.19697	1530	1372	89.67%
1313.20505	1530	1372	89.67%
3535.38283	1530	1372	89.67%
3585.88687	1530	1372	89.67%
3636.39091	1530	1372	89.67%
1060.68485	1530	1371	89.61%
1262.70101	1530	1371	89.61%

The highest accuracy from ridge regression is not over 90.00%, the largest accuracy appears when λ value is around 1360, which makes accuracy 89.74%.

The figure below shows the change of accuracy along with different shrinkage factor λ values.

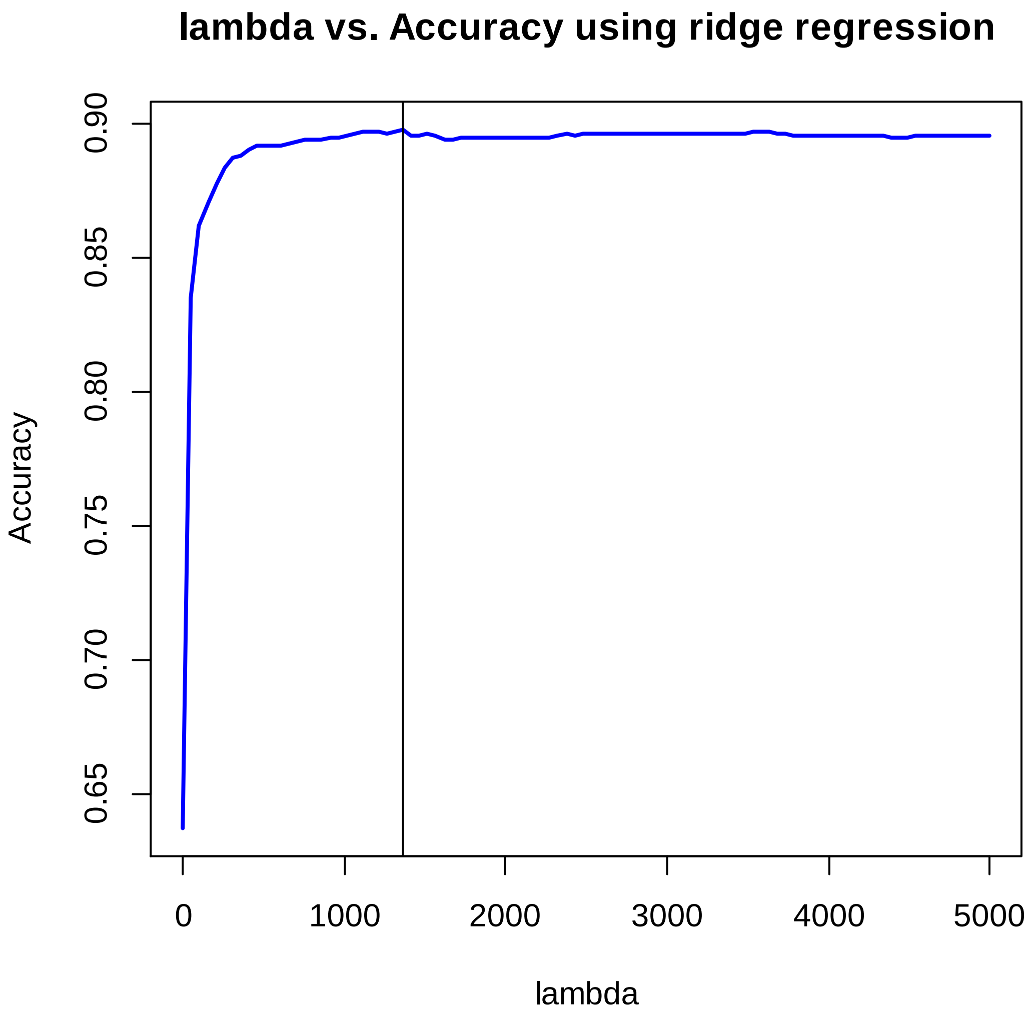

Figure 5. Lambda vs. Accuracy for ridge regression.

We could see the accuracy tends to be a constant when lambda value gets greater.

4.3 Lasso regression

After conducting a Lasso regression between query data and reference data with 100 different shrinkage factor λ (range from 0.1 to 5000), we calculated the correct matching and the accuracy. Table 2 shows the best 10 results.

Table 2. Top 10 best accuracy and corresponding shrinkage factor for Lasso regression.

Shrinkage factor (λ)	Number of query	Number of correct match	Accuracy
4646.472	1530	1400	91.50%
4595.968	1530	1398	91.37%
4696.976	1530	1398	91.37%
4747.480	1530	1398	91.37%
4898.992	1530	1398	91.37%
4797.984	1530	1397	91.31%
5000.000	1530	1397	91.31%
4343.447	1530	1396	91.24%
4545.464	1530	1396	91.24%
4848.488	1530	1396	91.24%

After a further check, the best accuracy for Lasso regression is 91.50% when λ=4646. This accuracy is higher than the highest accuracy from ridge regression. Figure 6 shows the change of accuracy corresponding to different shrinkage factors.

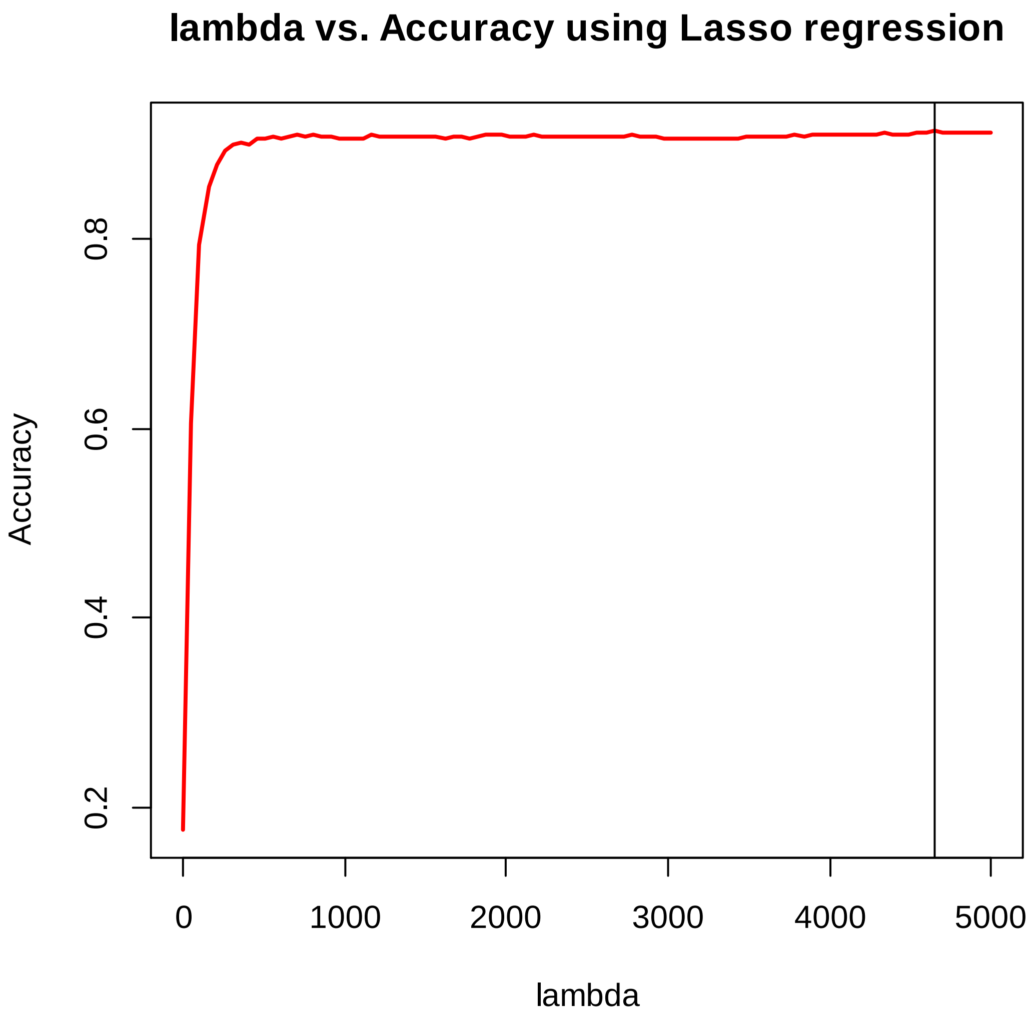

Figure 6. Lambda vs. Accuracy for Lasso regression.

4.4 Two-step approaches

4.4.1 Dot product and Ridge/Lasso regression

We then conducted dot product and ridge/ Lasso regression to optimize the performance of compound identification, and tried to find the relationship between accuracy and different rank levels and λ values. We choose 12 different rank levels ranging from 25 to 300. For λ, we also have 100 values ranging from 0.1 to 5000, the same with the identification using only ridge regression and only Lasso regression.

Part of the results are shown in Table 3 (ridge regression) and Table 4 (Lasso regression).

Table 3. Top 20 best accuracy and corresponding shrinkage factor for the dot product and ridge regression.

Rank	Shrinkage factor (λ)	Number of query	Number of correct match	Accuracy
25	0.1	1530	1380	90.20%
100	202.1162	1530	1380	90.20%
100	303.1242	1530	1380	90.20%
250	505.1404	1530	1380	90.20%
275	555.6444	1530	1380	90.20%
50	303.1242	1530	1379	90.13%
75	151.6121	1530	1379	90.13%
125	252.6202	1530	1379	90.13%
125	353.6283	1530	1379	90.13%
150	252.6202	1530	1379	90.13%
150	404.1323	1530	1379	90.13%
225	505.1404	1530	1379	90.13%
25	151.6121	1530	1378	90.07%
50	252.6202	1530	1378	90.07%
50	353.6283	1530	1378	90.07%
75	252.6202	1530	1378	90.07%
100	353.6283	1530	1378	90.07%
125	303.1242	1530	1378	90.07%
125	404.1323	1530	1378	90.07%
125	555.6444	1530	1378	90.07%

Table 4 Top 20 best accuracy and corresponding shrinkage factor for the dot product and Lasso regression

Rank	Shrinkage factor (λ)	Number of query	Number of correct match	Accuracy
200	3838.407	1530	1395	91.18%
300	1363.709	1530	1395	91.18%
300	1414.213	1530	1395	91.18%
300	1464.717	1530	1395	91.18%

300	1515.221	1530	1395	91.18%
175	3888.911	1530	1394	91.11%
175	3939.415	1530	1394	91.11%
175	3989.919	1530	1394	91.11%
200	3787.903	1530	1394	91.11%
300	1565.725	1530	1394	91.11%
300	1616.229	1530	1394	91.11%
300	1666.733	1530	1394	91.11%
300	1717.237	1530	1394	91.11%
150	3737.399	1530	1393	91.05%
200	2121.27	1530	1393	91.05%
200	2171.774	1530	1393	91.05%
200	3535.383	1530	1393	91.05%
200	3888.911	1530	1393	91.05%
200	3939.415	1530	1393	91.05%
200	3989.919	1530	1393	91.05%

The results for this two-step approach are not so clear to interpret, we use a contour plot to show the relationship among accuracy, rank levels and lambda values for ridge regression and Lasso regression, respectively.

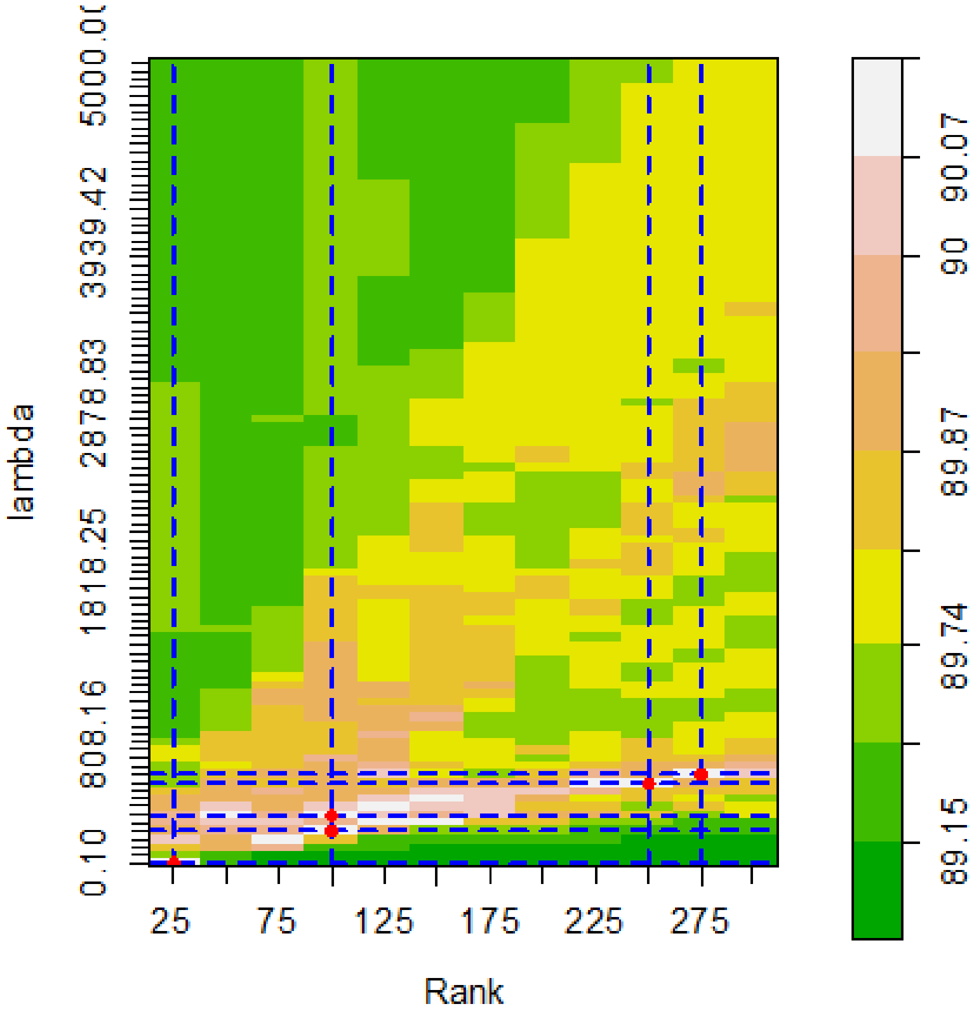

Figure 7. Accuracy of two-step approach using dot product and ridge regression.

As the Figure 7 shows, green color stands for relatively low accuracy and white and pink stands for high accuracy. The highest accuracy 90.20% appears at rank level=25 and λ=0.1, which shown as a red point. Along with other four red points, the accuracy is also relatively high. Comparing to ridge regression only, we are pleased to find this two-step approach performs better than ridge regression (accuracy=89.74%).

From Figure 7, in general, we could see when λ value is increasing, we also need to increase the rank correspondingly to guarantee better accuracy. Equally, if rank level changes from 100 to 250, we probably need to increase λ value from 500 to 800.

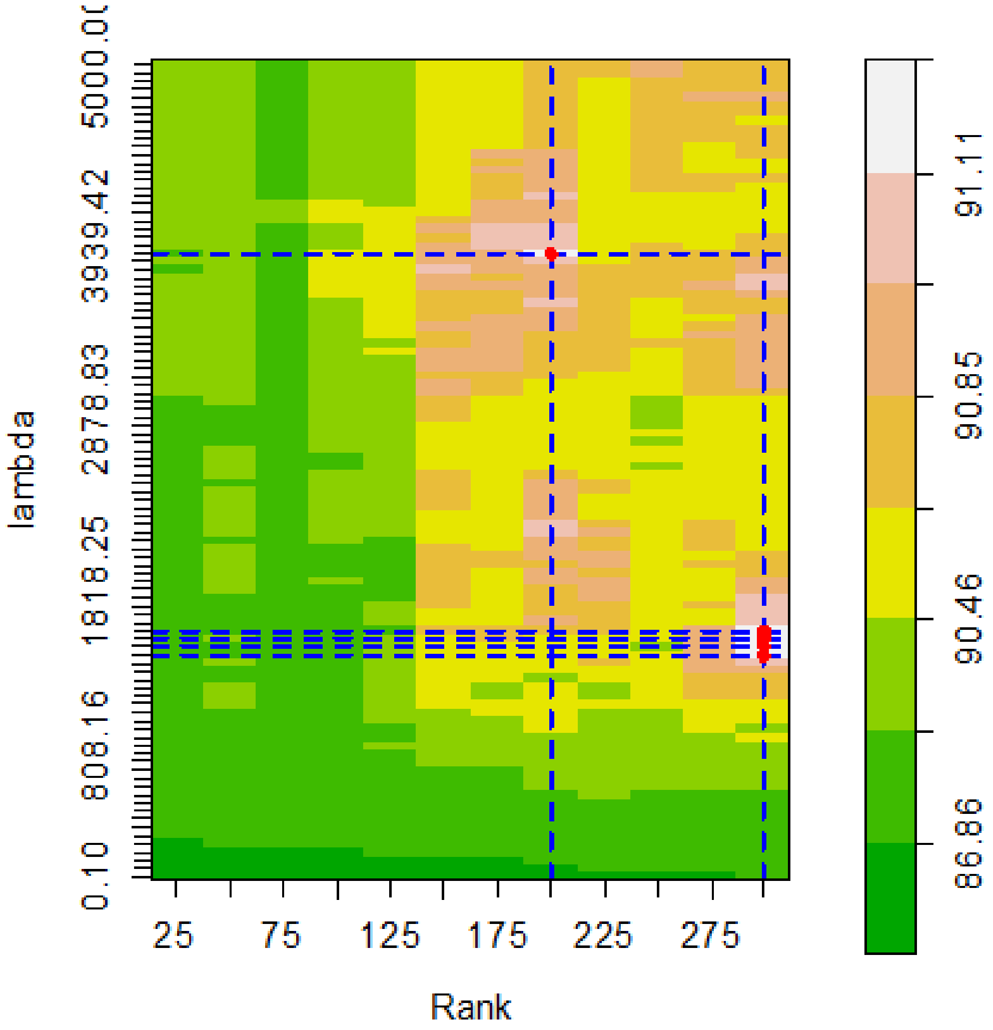

Figure 8 Accuracy of two-step approach using dot product and Lasso regression.

Figure 8 presents the relationship among accuracy, rank levels and λ values for the dot product and Lasso regression two-step approach. The highest accuracy 91.18% appears at rank level=200 and λ=3838, which shown as a red point. Along with other four red points, the accuracy is also relatively high. Comparing to Lasso regression only, this two-step approach has no improvement in accuracy, which is different from using ridge regression.

Figure 8 also shows a general trend that increasing the rank will result in improved accuracy, and this is different from the two-step approach using dot product and ridge regression.

4.4.2 Pearson’s correlation and ridge/ Lasso regression

For Pearson’s correlation and penalized linear regression two-step approach, we intend to find the relationship among accuracy, different confidence levels and λ values. We choose 0.01, 0.025, 0.05, 0.1 these four α levels, and 100 λ values ranging from 0.1 to 5000 as well. The top 20 highest accuracy and corresponding shrinkage factor are shown in Table 5 (ridge regression) and Table 6 (Lasso regression).

Table 5. Top 20 best accuracy and corresponding shrinkage factor for Pearson’s correlation and ridge regression.

α	Shrinkage factor (λ)	Number of query	Number of correct match	Accuracy
0.1	101.1081	1530	1368	89.41%
0.1	353.6283	1530	1368	89.41%
0.1	404.1323	1530	1368	89.41%
0.1	454.6364	1530	1368	89.41%
0.1	505.1404	1530	1368	89.41%
0.1	555.6444	1530	1368	89.41%
0.1	606.1485	1530	1368	89.41%
0.1	656.6525	1530	1368	89.41%
0.1	707.1566	1530	1368	89.41%
0.1	757.6606	1530	1368	89.41%
0.1	808.1646	1530	1368	89.41%
0.1	858.6687	1530	1368	89.41%
0.1	959.6768	1530	1368	89.41%
0.1	1010.1808	1530	1368	89.41%

0.1	1060.6848	1530	1368	89.41%
0.1	1111.1889	1530	1368	89.41%
0.1	1161.6929	1530	1368	89.41%
0.1	1212.1970	1530	1368	89.41%
0.1	1262.7010	1530	1368	89.41%
0.1	1313.2051	1530	1368	89.41%

Table 6 Top 20 best accuracy and corresponding shrinkage factor for Pearson's correlation and Lasso regression.

α	Shrinkage factor (λ)	Number of query	Number of correct match	Accuracy
0.1	0.10000	1530	1192	77.91%
0.1	50.60404	1530	1192	77.91%
0.1	101.10808	1530	1192	77.91%
0.1	151.61212	1530	1192	77.91%
0.1	202.11616	1530	1192	77.91%
0.1	252.62020	1530	1192	77.91%
0.1	303.12424	1530	1192	77.91%
0.1	353.62828	1530	1192	77.91%
0.1	404.13232	1530	1192	77.91%
0.1	454.63636	1530	1192	77.91%
0.1	505.14040	1530	1192	77.91%
0.1	555.64444	1530	1192	77.91%
0.1	606.14848	1530	1192	77.91%
0.1	656.65253	1530	1192	77.91%
0.1	707.15657	1530	1192	77.91%
0.1	757.66061	1530	1192	77.91%
0.1	808.16465	1530	1192	77.91%
0.1	858.66869	1530	1192	77.91%
0.1	909.17273	1530	1192	77.91%
0.1	959.67677	1530	1192	77.91%

We could see the best accuracies for this two-step approach using ridge and Lasso regression all appear at α=0.10, which is 89.41% (ridge regression) and 77.91%

(Lasso regression), respectively. While in this two-step approach, Lasso regression seems not as good as ridge regression. The same contour plots are shown below.

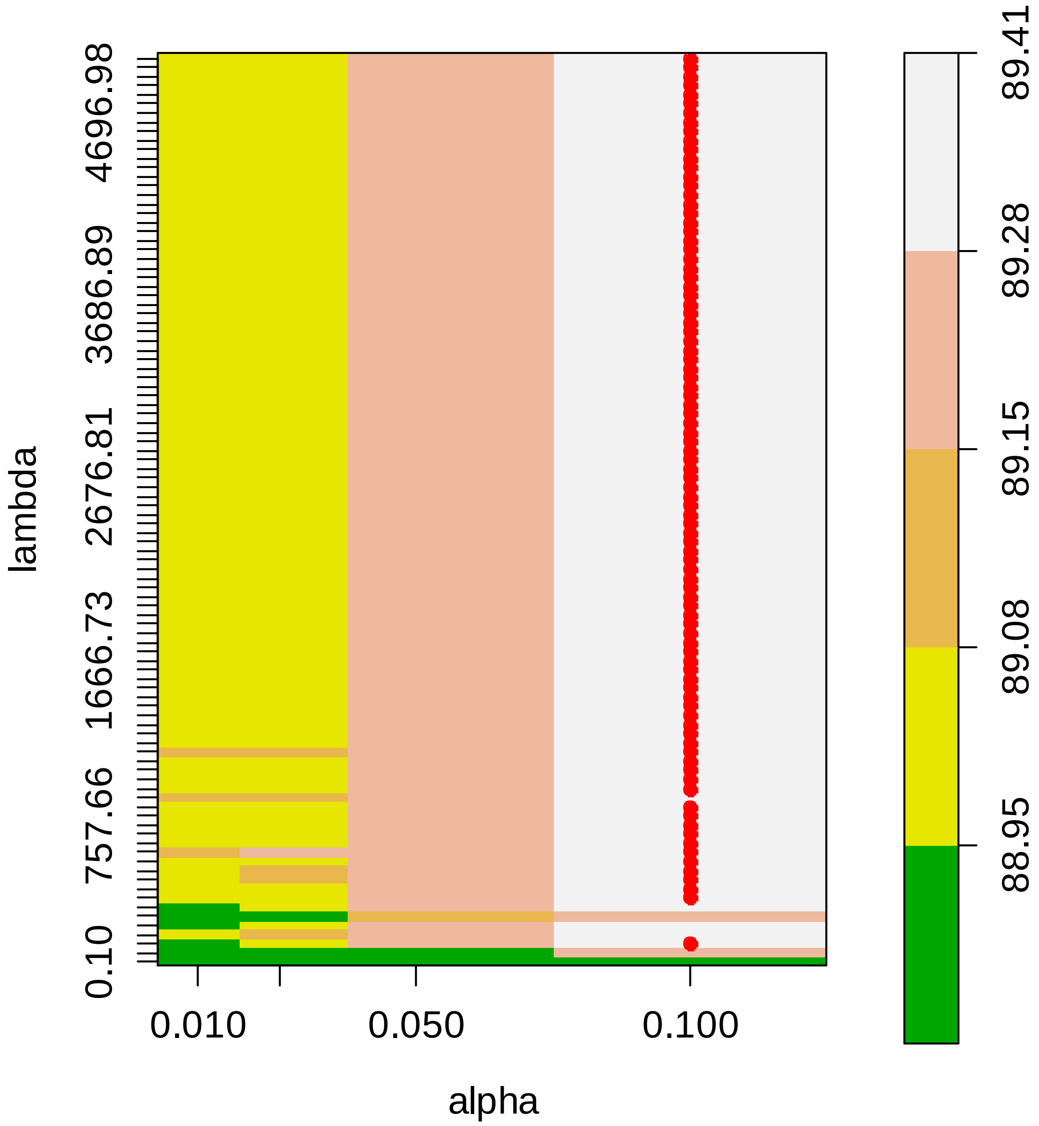

Figure 9. Accuracy of two-step approach using Pearson's correlation and ridge regression.

The relation among accuracy, α levels and λ values in this two-step approach seems much easier. When λ value is greater than certain value (around 300), it may not influence the accuracy so much. The red points stand for best accuracy, and they all appear at α=0.1, which make a red vertical line. While it is clear that greater α level results in higher accuracy.

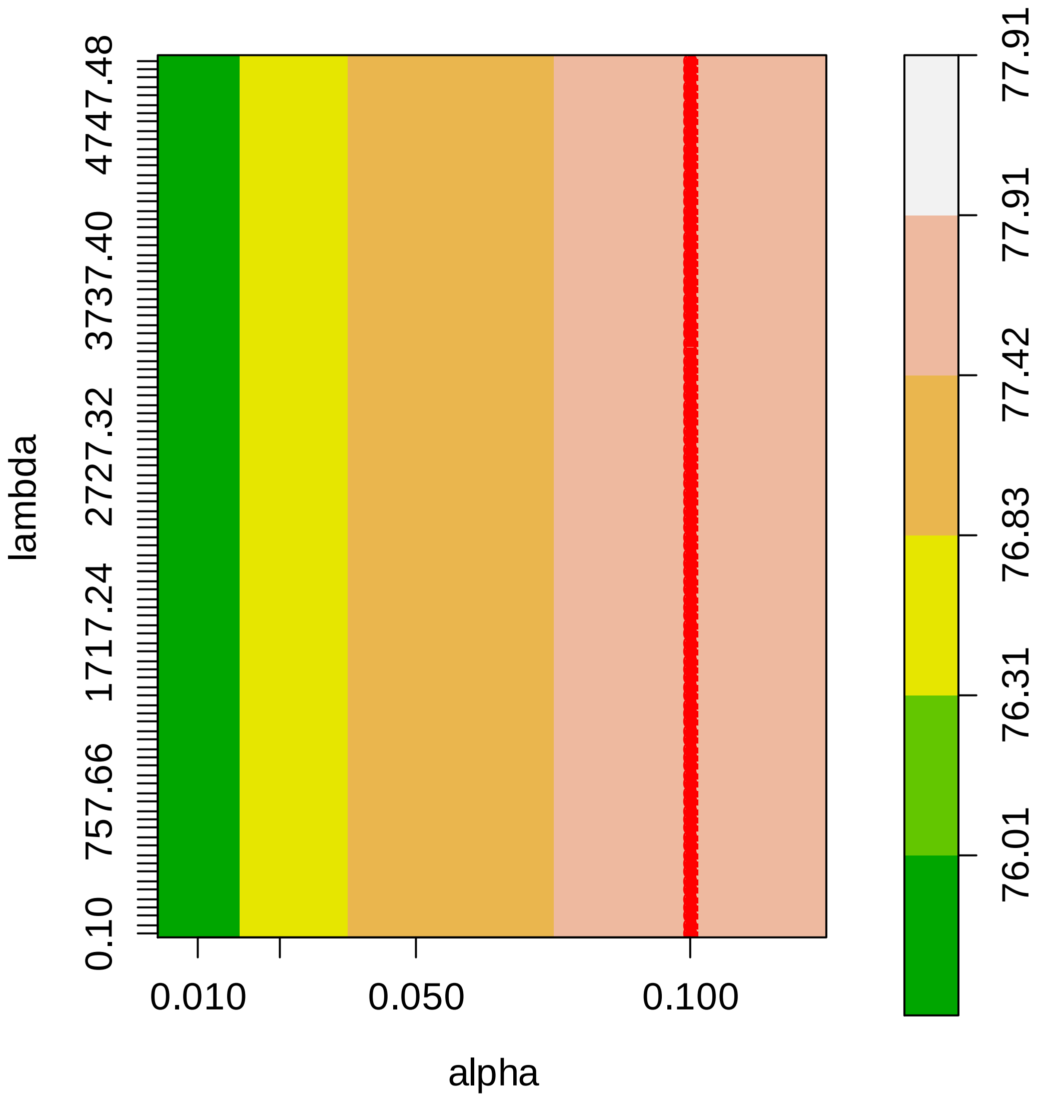

Figure 10. Accuracy of two-step approach using Pearson's correlation and Lasso regression.

The relation among accuracy, α levels and λ values in Pearson's correlation and Lasso regression two-step approach is the same with using ridge regression. As the two-step approach using ridge regression, the red points all appear at α=0.1, which make a red vertical line. The selection of λ value may not influence the accuracy, while it is clear that greater α level results in higher accuracy.

4.5 The best performance

We have tested the performance of four compound identification methods involving penalized linear regression. In addition, we also included previously widely used methods in our study. The table below shows these new methods and their best performance (accuracy) respectively, including the corresponding lambda value, rank selection (for dot product and ridge/ Lasso regression two-step approach) and alpha selection (for Pearson’s correlation and ridge/ Lasso regression two-step approach). Also, we list the performance of dot product and Pearson’s correlation in compound identification.

Table 7. Compound identification methods and their performance.

Method	Lambda	Rank (Alpha)	Accuracy (%)
Dot product	/	/	89.54
Pearson’s correlation	/	/	89.54
Ridge Regression	1363.7	/	89.74
Lasso Regression	4646.5	/	91.50
Dot product and ridge regression	0.1	25	90.20
Pearson’s correlation and ridge regression	353.6~858.7	0.1	89.41
Dot product and Lasso regression	3838.4 1363.7~1515.2	200 300	91.18

Pearson’s correlation and Lasso regression	0.1~960	0.1	77.91

CHAPTER V

CONCLUSION

In this study, we propose new approaches for compound identification using penalized linear regressions and introduce further two-step approaches. In particular, we pursue to find an alternative to the semi-partial correlation-based approach using multiple linear regressions.

From the results using a small data set, we can see that the Lasso regression achieves the highest accuracy of compound identification, which is 91.50% with λ of 4646.5, which is 1% larger than that of the dot product.

However, considering the overall performance of these methods, since the accuracy for Lasso regression is highly related to the selection of shrinkage factor λ, we have to do cross-validation using Lasso regression for compound identification, clearly, this will cause longer calculation time. While ridge regression shows a constant accuracy after a certain λ value, it might be a better choice. In addition, considering the two-step approaches using the dot product and ridge regression, its accuracy is 90.2%, which is respectively high. We consider this method has the best performance.

We might notice that the two-step approach using Pearson's correlation and ridge/ Lasso regression has no improvement in identification accuracy. While this approach shows the shrinkage factor selection has no effect upon the accuracy of compound identification, which means we do not have to concern about the selection of shrinkage factors. The accuracy is purely related to confidence levels.

Because the process of compound identification is very time-consuming, as we mentioned before, we only extracted 2739 mass spectra of compounds from the NIST database as the reference mass spectra. After demonstrating the effectiveness of our new approach, there is a lot of work to do in applying this approach to the entire NIST library. Since the first step in our two-step approaches is to reduce the dimension, we believe this entire library identification will gain more benefits from this new approach.

In addition, we also need to justify the relationship between the semi-partial correlation and linear regression. Because the ridge regression seems to conform to this relationship, while Lasso regression seems not.

REFERENCES

1 Fiehn, O. Plant Mol. Biol. 2002, 48, 155–171.

2 Denkert, C.; Budczies, J.; Kind, T.; Weichert, W.; Tablack, P.; Sehouli, J.; Niesporek, S.; Konsgen, D.; Dietel, M.; Fiehn, O. Cancer Res. 2006, 66, 10795–10804.

3 Seongho Kim; Imhoi Koo; Jaesik Jeong; Shiwen Wu; Xue Shi; Xiang Zhang. Anal. Chem.2012,84,6477-6487.

4 Tabb, D. L.; MacCoss, M. J.; Wu, C. C.; Anderson, S. D.; Yates, J. R. Anal. Chem. 2003, 75, 2470−2477.

5 Beer, I.; Barnea, E.; Ziv, T.; Admon, A. Proteomics 2004, 4, 950−960.

6 Craig, R.; Cortens, J. C.; Fenyo, D.; Beavis, R. C. J. Proteome Res. 2006, 5, 1843−1849.

7 Frewen, B. E.; Merrihew, G. E.; Wu, C. C.; Noble, W. S.; MacCoss, M. J. Anal. Chem. 2006, 78, 5678−5684.

8 Stein, S. E.; Scott, D. R. J. Am. Soc. Mass Spectrom. 1994, 5, 859−866.

9 Atwater, B. L.; Stauffer, D. B.; Mclafferty, F. W.; Peterson, D. W. Anal. Chem. 1985, 57, 899−903.

10 Hertz, H. S.; Hites, R. A.; Biemann, K. Anal. Chem. 1971, 43, 681

11 Julian, R. K.; Higgs, R. E.; Gygi, J. D.; Hilton, M. D. Anal. Chem. 1998, 70, 3249−3254.

12 Rasmussen, G. T.; Isenhour, T. L. J. Chem. Inf. Comput. Sci. 1979, 19, 179−186.

13 Koo, I.; Zhang, X.; Kim, S. Anal. Chem. 2011, 83, 5631−5638.

14 Stein, S.; Scott, D. J. Am. Soc. Mass Spectrom. 1994, 5, 859–866.

15 Hastie, Tibshirani and Friedman. The Elements of Statistical Learning (2nd edition); Springer-Verlag, 2008.

16 Minjung Kyung; Jeff Gill; Malay Ghosh; George Casella. Bayesian Analysis. 2010, 5, 2,369-412.

17 Linstrom, P.; Mallard, W. NIST Chemistry WebBook, NIST Standard Reference. Database 69, 2000. http://webbook.nist.gov/chemistry/; accessed 3/26/2013.

APPENDIX

R code for 4.1

```
load("resmallnist2.rda")
library(glmnet)


######################################################################

## Lasso (alpha=1)

Lassofit<function(vlambda=seq(0.0001,1000000,length=100),dy=rep,dx=nist,
           iddy=idrep, iddx=idnist){

    # normalization to make the size as one
        dy = dy/sqrt(apply(dy^2,1,sum))
        dx = dx/sqrt(apply(dx^2,1,sum))

        # standardization to make the mean as zero
        dy = t(dy)
        dx = t(dx)
        dy = t(t(dy)-c(apply(dy,2,mean)))
        dx = t(t(dx)-c(apply(dx,2,mean)))

    # identification by lambda
     rlt = c()
     for(lambda in vlambda){
                nacc = 0
        for (i in 1:dim(dy)[2]) {
            result <- glmnet(dx,dy[,i],alpha=1, lambda=lambda,standardize=FALSE)
              ## calculate accuracy
                idx<-which.max(as.numeric(result$beta))
                    if(iddy[i]==iddx[idx]){
                                 nacc = nacc + 1
                }
          }
     nacc
     tmp.rlt = c(lambda,dim(dy)[2],nacc,nacc/dim(dy)[2])
                rlt = rbind(rlt,as.numeric(tmp.rlt))
        }
        rlt=as.data.frame(rlt)
dimnames(rlt)[[2]] = c("lambda","nquery","nmatch","acc")
        rlt
}

Lasso <- Lassofit()
save(Lasso, file="Lasso.RData")

#########################################################################

## Ridge (alpha=0)

Ridgefit<-function(vlambda=seq(0.0001,1000000,length=100),dy=rep,dx=nist,
            iddy=idrep, iddx=idnist){

    # normalization to make the size as one
        dy = dy/sqrt(apply(dy^2,1,sum))
        dx = dx/sqrt(apply(dx^2,1,sum))

        # standardization to make the mean as zero
        dy = t(dy)
        dx = t(dx)
        dy = t(t(dy)-c(apply(dy,2,mean)))
        dx = t(t(dx)-c(apply(dx,2,mean)))

    # identification by lambda
     rlt = c()
     for(lambda in vlambda){
                nacc = 0
        for (i in 1:dim(dy)[2]) {
            result <- glmnet(dx,dy[,i],alpha=0, lambda=lambda,standardize=FALSE)
             ## calculate accuracy
               idx<-which.max(as.numeric(result$beta))
                    if(iddy[i]==iddx[idx]){
                                nacc = nacc + 1
               }
         }
     nacc
     tmp.rlt = c(lambda,dim(dy)[2],nacc,nacc/dim(dy)[2])
                rlt = rbind(rlt,as.numeric(tmp.rlt))
        }
        rlt = as.data.frame(rlt)
        dimnames(rlt)[[2]] = c("lambda","nquery","nmatch","acc")
        rlt
}

Ridge <- Ridgefit()
save(Ridge, file="Ridge.RData")
```

```
######################################################################
## Plot for comparison between ridge and Lasso regression

plot(Lasso$lambda, Lasso$acc, type="l", main="lambda vs. Accuracy",
   xlab="lambda", ylab="Accuracy", lwd=2, col=4)

lines(Ridge$lambda, Ridge $acc, lty=2, lwd=2, col=2)

legend(700000, 0.8, c("Lasso", "Ridge"), col = c(4, 2),lty = c(1,2), lwd=c(2,2))



######################################################################
```

R code for 4.2

```
## Ridge regression based identification

ridge.svd<-function(vlambda=seq(0.1,5000,length=100),dy=rep,dx=nist,
          iddy=idrep,iddx=idnist){

      # normalization to make the size as one
      dy = dy/sqrt(apply(dy^2,1,sum))
      dx = dx/sqrt(apply(dx^2,1,sum))

      # standization to make the mean as zero
      dy = t(dy)
      dx = t(dx)
      dy = t(t(dy)-c(apply(dy,2,mean)))
      dx = t(t(dx)-c(apply(dx,2,mean)))

      # SVD for the reference library
      dx = svd(dx)
      r2 = (dx$u) %*% diag(dx$d)
      v2 = dx$v

      # identification by lambda
      rlt = c()
      for(lambda in vlambda){
            nacc = 0
      beta = v2 %*% solve(t(r2)%*%r2+diag(lambda,dim(r2)[2]))%*% t(r2) %*% dy
            for (i in 1:dim(beta)[2]){
                  tmp = as.numeric(beta[,i])
                  idx = which.max(tmp)
                  if(iddy[i]==iddx[idx]){
                        nacc = nacc+1
                  }
```

```
                }
                nacc
                tmp.rlt = c(lambda,dim(dy)[2],nacc,nacc/dim(dy)[2])
                rlt = rbind(rlt,as.numeric(tmp.rlt))
        }
        rlt = as.data.frame(rlt)
        dimnames(rlt)[[2]] = c("lambda","nquery","nmatch","acc")
        rlt
}

ridge <- ridge.svd()
save(ridge, file="ridge.RData")

########################################################################

## Plot for ridge regression

plot(ridge$lambda, ridge$acc, type="l", main="lambda vs. Accuracy using ridge
regression", xlab="lambda", ylab="Accuracy", lwd=2, col=4)
abline(v = 1361.7)



########################################################################
```

R code for 4.3

```
## Lasso (alpha=1)

Lassofit<-function(vlambda=seq(0.1,5000,length=100),dy=rep,dx=nist,
           iddy=idrep,iddx=idnist){

    # normalization to make the size as one
        dy = dy/sqrt(apply(dy^2,1,sum))
        dx = dx/sqrt(apply(dx^2,1,sum))

        # standardization to make the mean as zero
        dy = t(dy)
        dx = t(dx)
        dy = t(t(dy)-c(apply(dy,2,mean)))
        dx = t(t(dx)-c(apply(dx,2,mean)))

    # identification by lambda
     rlt = c()
     for(lambda in vlambda){
                nacc = 0
        for (i in 1:dim(dy)[2]) {
           result <- glmnet(dx,dy[,i],alpha=1, lambda=lambda,standardize=FALSE)
             ## calculate accuracy
               idx<-which.max(as.numeric(result$beta))
```

```
                    if(iddy[i]==iddx[idx]){
                                nacc = nacc + 1
                }
          }
     nacc
     tmp.rlt = c(lambda,dim(dy)[2],nacc,nacc/dim(dy)[2])
                rlt = rbind(rlt,as.numeric(tmp.rlt))
        }
        rlt = as.data.frame(rlt)
        dimnames(rlt)[[2]] = c("lambda","nquery","nmatch","acc")
        rlt
}

lasso <- Lassofit()
save(lasso, file="lasso.RData")


######################################################################

## Plot for Lasso regression

plot(lasso$lambda, lasso$acc, type="l", main="lambda vs. Accuracy using Lasso
regression",  xlab="lambda", ylab="Accuracy", lwd=2, col=2)
abline(v = 4646)


######################################################################
```

R code for 4.4.1

```
## Two-step approach--- Dot product and ridge

dx1 <- nist
dy1 <- rep
idy = idrep
idx = idnist
vlambda=seq(0.1,5000,length=100)
vrank =seq(25,300, by=25)

## do dot product (first step)
                dx2 = dx1/sqrt(apply(dx1^2,1,sum))
                dy2 = dy1/sqrt(apply(dy1^2,1,sum))

                dot = dy2 %*% t(dx2)  ## dot product

## second step
  rlt=c()
  for (rank in vrank) {
      for (lambda in vlambda) {
        M=dim(dy2)[1]
        nacc2 = 0
         for (i in 1:M) {
           index <- order(dot[i,], decreasing =TRUE)[1:rank]
           newx <- t(dx2[index,])    ## new top x possible compound in lib
           svdnewx <- svd(newx)
           newr <-svdnewx$u%*%diag(svdnewx$d)
           newv <- svdnewx$v
           beta1<- newv %*% solve(t(newr)%*%newr + diag(lambda,dim(newr)[2])) %*%
t(newr) %*% t(dy2)[,i]

   # fit <- lm.ridge(dy2[i,]~newx, lambda = 1360)
   # beta1 <-fit$coef
           pos <- which.max(beta1)
           newidx <- index[pos]
           if(idrep[i]==idnist[newidx]){
                                       nacc2 = nacc2+1
                             }
           }
       nacc2
                    tmp.rlt = c(rank,lambda,M,nacc2,nacc2/M)
                    rlt = rbind(rlt,as.numeric(tmp.rlt))
           }
   }
          rlt = as.data.frame(rlt)
          dimnames(rlt)[[2]] = c("rank","lambda","nquery","nmatch","acc")
          rlt
save(rlt, file="dotridge.RData")



######################################################################

##  Plot for two-step approach--- Dot product and ridge


if(T){
          load("dotridge.RData")
}
if(T){
          td = rlt
          td.x = sort(unique(rlt$rank))
          td.y = sort(unique(rlt$lambda))
          n.x = length(td.x)
          n.y = length(td.y)
          td.z = matrix(0,n.x,n.y)
          for(i in 1:n.x){
                    for(j in 1:n.y){
```

```
            tmp = td$acc[td$rank==td.x[i] & td$lambda==td.y[j]]
            td.z[i,j] = tmp
        }
    }
}

cont.plot <- function(td=rlt,plot=F,fsize=1.2){
    td = rlt
    maxacc = td[td$acc==max(td$acc),]
    td.x = sort(unique(rlt$rank))
    td.y = sort(unique(rlt$lambda))
    n.x = length(td.x)
    n.y = length(td.y)
    td.z = matrix(0,n.x,n.y)
    for(i in 1:n.x){
        for(j in 1:n.y){
            tmp = td$acc[td$rank==td.x[i] & td$lambda==td.y[j]]
            td.z[i,j] = tmp
        }
    }
    if(plot){
        nf <- layout(matrix(c(1,2),1,2,byrow=TRUE), c(3.5,1), TRUE)

        x = td.x
        y = td.y
        volcano = td.z
        x.at <- x
        y.at <- round(y,2)

        # Using Terrain Colors
        par(mar=c(5,5,1.5,0))
vbreaks= sort(unique(c(0,quantile(td$acc,probs=c(.025,.25,.5,.75,.9,.95,.975,.99,1)))))

        print(vbreaks)

            ty = c(1:(length(vbreaks)-1))
            tz = matrix(vbreaks[-1],1,length(ty))
            tat = c(ty+.5)
            main.labs = " " #"(b)"

        image(x, y, volcano, col=terrain.colors(length(ty)),axes=FALSE
            ,breaks=vbreaks
            ,xlab="Rank",ylab="lambda"
            ,main=main.labs
            ,cex=fsize
            ,cex.axis=fsize
            ,cex.lab=fsize
            ,cex.main=fsize
            )
        abline(v=maxacc$rank,h=maxacc$lambda,col=4,lty=2,lwd=2)
```

```
                points(maxacc$rank,maxacc$lambda,pch=19,col=2)
                axis(1, at=x.at,cex.axis=fsize)
                axis(2, at=y.at,cex.axis=fsize)
                box()
                par(mar=c(3,2,1.5,3))
                aa=vbreaks
                image(x=1,y=ty,z=tz,col=terrain.colors(length(ty)),axes=F
                        ,breaks=vbreaks
                        ,xlab="Accuracy (%)",ylab=""
                        ,cex.lab=1.2
                        )
                        axis(4,at=tat,lab=round(aa[-1]*100,2),cex.axis=fsize)
                box()
        }
}
cont.plot(td=rlt,plot=T)



#####################################################################

## Two-step approach--- Dot product and Lasso

library(glmnet)

dx1 <- nist
dy1 <- rep
idy = idrep
idx = idnist
vlambda=seq(0.1,5000,length=100)
vrank =seq(25,300, by=25)

## do dot product
                dx2 = dx1/sqrt(apply(dx1^2,1,sum))
                dy2 = dy1/sqrt(apply(dy1^2,1,sum))

                dot = dy2 %*% t(dx2)  ## dot product

## second step
  rlt=c()
  for (rank in vrank) {
      for (lambda in vlambda) {
        M=dim(dy2)[1]
        nacc = 0
         for (i in 1:M) {
           index <- order(dot[i,], decreasing =TRUE)[1:rank]
           newx <- t(dx2[index,])    ## new top x possible compound in lib
           result <- glmnet(newx,t(dy2)[,i],alpha=1, lambda=lambda, standardize=FALSE)
                 ## calculate accuracy
                   idx<-which.max(as.numeric(result$beta))
```

```
          if(iddy[i]==iddx[idx]){
                nacc = nacc + 1
            }
    }
  nacc
        tmp.rlt = c(rank,lambda,M,nacc,nacc/M)
        rlt = rbind(rlt,as.numeric(tmp.rlt))
    }
 }
    rlt = as.data.frame(rlt)
    dimnames(rlt)[[2]] = c("rank","lambda","nquery","nmatch","acc")
    rlt
save(rlt, file="dotlasso.RData")



#####################################################################

##  Plot for two-step approach--- Dot product and Lasso

if(T){
    load("dotlasso.RData")
}
if(T){
    td = rlt
    td.x = sort(unique(rlt$rank))
    td.y = sort(unique(rlt$lambda))
    n.x = length(td.x)
    n.y = length(td.y)
    td.z = matrix(0,n.x,n.y)
    for(i in 1:n.x){
        for(j in 1:n.y){
            tmp = td$acc[td$rank==td.x[i] & td$lambda==td.y[j]]
            td.z[i,j] = tmp
        }
    }
}

cont.plot <- function(td=rlt,plot=F,fsize=1.2){
    td = rlt
    maxacc = td[td$acc==max(td$acc),]
    td.x = sort(unique(rlt$rank))
    td.y = sort(unique(rlt$lambda))
    n.x = length(td.x)
    n.y = length(td.y)
    td.z = matrix(0,n.x,n.y)
    for(i in 1:n.x){
        for(j in 1:n.y){
            tmp = td$acc[td$rank==td.x[i] & td$lambda==td.y[j]]
            td.z[i,j] = tmp
		}
	}
	if(plot){
		#par(mfrow=c(1,2))
		nf <- layout(matrix(c(1,2),1,2,byrow=TRUE), c(3.5,1), TRUE)

		x = td.x
		y = td.y
		volcano = td.z
		x.at <- x
		y.at <- round(y,2)

		# Using Terrain Colors

		par(mar=c(5,5,1.5,0))

		vbreaks= sort(unique(c(0,quantile(td$acc,probs=
					c(.025,.25,.5,.75,.9,.95,.975,.99,1)))))

		print(vbreaks)

			ty = c(1:(length(vbreaks)-1))
			tz = matrix(vbreaks[-1],1,length(ty))
			tat = c(ty+.5)
			main.labs = " " #"(b)"

		image(x, y, volcano, col=terrain.colors(length(ty)),axes=FALSE
			,breaks=vbreaks
			,xlab="Rank",ylab="lambda"
			,main=main.labs
			,cex=fsize
			,cex.axis=fsize
			,cex.lab=fsize
			,cex.main=fsize
			)
		abline(v=maxacc$rank,h=maxacc$lambda,col=4,lty=2,lwd=2)
		points(maxacc$rank,maxacc$lambda,pch=19,col=2)
		axis(1, at=x.at,cex.axis=fsize)
		axis(2, at=y.at,cex.axis=fsize)
		box()
		par(mar=c(3,2,1.5,3))
		aa=vbreaks
		image(x=1,y=ty,z=tz,col=terrain.colors(length(ty)),axes=F
			,breaks=vbreaks
			,xlab="Accuracy (%)",ylab=""
			,cex.lab=1.2
			)
			axis(4,at=tat,lab=round(aa[-1]*100,2),cex.axis=fsize)
		box()
	}
```

```
}
cont.plot(td=rlt,plot=T)
```

```
######################################################################
```

R code for 4.4.2

```
## Two-step approach--- Pearson’s correlation and ridge regression

dx1 <- nist
dy1 <- rep
idy = idrep
idx = idnist
vlambda=seq(0.1,5000,length=100)
valpha =c(0.01, 0.025, 0.05, 0.1)

M <- dim(rep)[1]
P <- dim(nist)[1]

sampleSize <- dim(nist)[2]

## Pearson's correlation

correlation <- cor(t(dy1),t(dx1))

rlt=c()
for (alpha in valpha) {
          for (lambda in vlambda) {
           nacc3=0
        for (i in 1:M) {
                    ## order the Pearson's correlation coef first
                    index1 <- order(correlation[i,],decreasing=TRUE)
                    dx1.order <- dx1[index1,]
                    new.idx <- 1
                    minimum.lower.int <- 100
                              for (j in 2:P) {
                    # test the overlaps
                    pearson1 <- cor.test(dy1[i,], dx1.order[j-1,], method = "pearson",
conf.level = 1-alpha,alternative = "two.sided")
                    pearson2 <- cor.test(dy1[i,], dx1.order[j,], method = "pearson",
conf.level = 1-alpha,alternative = "two.sided")
                    minimum.lower.int <- min(minimum.lower.int,pearson1$conf.int[1])
                              if (minimum.lower.int <= pearson2$conf.int[2]) {
                                        new.idx <- new.idx+1
                                                  }
```

```
            else { break}
        }
        if (new.idx>1) {
        ## fit ridge regression
        new.x <- t(dx1.order[1:new.idx,])   ## new x
       beta2 <- solve(t(new.x)%*% new.x + diag(lambda,dim(new.x)[2]))%*%
t(new.x)%*% t(dy1)[,i]
        pos2 <- which.max(beta2)
        new.idx2 <- index1[pos2]
        if(idrep[i]==idnist[new.idx2]){
                    nacc3 = nacc3+1
                }
        }
        else {  new.idx2 <-index1[1]
            if(idrep[i]==idnist[new.idx2]){
                    nacc3 = nacc3+1
            }
    }
   }
  nacc3
        tmp.rlt = c(alpha,lambda,M,nacc3,nacc3/M)
        rlt = rbind(rlt,as.numeric(tmp.rlt))
 }
}
    rlt = as.data.frame(rlt)
    dimnames(rlt)[[2]] = c("alpha","lambda","nquery","nmatch","acc")
    rlt

save(rlt, file="Pearsonridge.RData")



#################################################################

##  Plot for two-step approach--- Pearson's correlation and ridge regression

if(T){
    load("Pearsonridge.RData")
}
if(T){
    td = rlt
    td.x = sort(unique(rlt$alpha))
    td.y = sort(unique(rlt$lambda))
    n.x = length(td.x)
    n.y = length(td.y)
    td.z = matrix(0,n.x,n.y)
    for(i in 1:n.x){
        for(j in 1:n.y){
            tmp = td$acc[td$alpha==td.x[i] & td$lambda==td.y[j]]
            td.z[i,j] = tmp
        }
    }
}

cont.plot <- function(td=rlt,plot=F,fsize=1.2){
    td = rlt
    maxacc = td[td$acc==max(td$acc),]
    td.x = sort(unique(rlt$alpha))
    td.y = sort(unique(rlt$lambda))
    n.x = length(td.x)
    n.y = length(td.y)
    td.z = matrix(0,n.x,n.y)
    for(i in 1:n.x){
        for(j in 1:n.y){
            tmp = td$acc[td$alpha==td.x[i] & td$lambda==td.y[j]]
            td.z[i,j] = tmp
        }
    }
    if(plot){
        nf <- layout(matrix(c(1,2),1,2,byrow=TRUE), c(3.5,1), TRUE)

        x = td.x
        y = td.y
        volcano = td.z
        x.at <- x
        y.at <- round(y,2)

        # Using Terrain Colors

        par(mar=c(5,5,1.5,0))

        vbreaks                                                        =
sort(unique(c(0,quantile(td$acc,probs=c(.025,.25,.5,.75,.9,.95,.975,.99,1)))))

        print(vbreaks)

            ty = c(1:(length(vbreaks)-1))
            tz = matrix(vbreaks[-1],1,length(ty))
            tat = c(ty+.5)
            main.labs = " " #"(b)"

        image(x, y, volcano, col=terrain.colors(length(ty)),axes=FALSE
            ,breaks=vbreaks
            ,xlab="alpha",ylab="lambda"
            ,main=main.labs
            ,cex=fsize
            ,cex.axis=fsize
            ,cex.lab=fsize
            ,cex.main=fsize
            )
        points(maxacc$alpha,maxacc$lambda,pch=19,col=2)
        axis(1, at=x.at,cex.axis=fsize)
        axis(2, at=y.at,cex.axis=fsize)
        box()
        par(mar=c(3,2,1.5,3))
        aa=vbreaks
        image(x=1,y=ty,z=tz,col=terrain.colors(length(ty)),axes=F
            ,breaks=vbreaks
            ,xlab="Accuracy (%)",ylab=""
            ,cex.lab=1.2
            )
            axis(4,at=tat,lab=round(aa[-1]*100,2),cex.axis=fsize)
        box()
    }
}

cont.plot(td=rlt,plot=T)



####################################################################
##  Two-step approach--- Pearson’s correlation and Lasso regression

library(glmnet)
dx1 <- nist
dy1 <- rep
iddy = idrep
iddx = idnist
vlambda=seq(0.1,5000,length=100)
valpha =c(0.01, 0.025, 0.05, 0.1)

M <- dim(rep)[1]
P <- dim(nist)[1]

sampleSize <- dim(nist)[2]

## Pearson's correlation

correlation <- cor(t(dy1),t(dx1))

rlt=c()
for (alpha in valpha) {
    for (lambda in vlambda) {
     nacc=0
   for (i in 1:M) {
        ## order the Pearson's correlation coef first
        index1 <- order(correlation[i,],decreasing=TRUE)
        dx1.order <- dx1[index1,]
        new.idx <- 1
        minimum.lower.int <- 100
```

```
                              for (j in 2:P) {
                    # test the overlaps
                    pearson1  <-  cor.test(dy1[i,],  dx1.order[j-1,],  method  =  "pearson",
conf.level = 1-alpha,alternative = "two.sided")
                    pearson2  <-  cor.test(dy1[i,],  dx1.order[j,],  method  =  "pearson",
conf.level = 1-alpha,alternative = "two.sided")
                    minimum.lower.int <- min(minimum.lower.int,pearson1$conf.int[1])
                    #if (pearson1$conf.int[1] <= pearson2$conf.int[2]) {
                              if (minimum.lower.int <= pearson2$conf.int[2]) {
                                                  new.idx <- new.idx+1
                                                            }
                              else { break}
                    }
                    if (new.idx>1) {
                    ## fit lasso regression
             new.x <- t(dx1.order[1:new.idx,])   ## new x
           result<-glmnet(new.x,t(dy1)[,i],alpha=1,  lambda=lambda,standardize=FALSE)

                    ## calculate accuracy
                   idx<-which.max(as.numeric(result$beta))
                         if(iddy[i]==iddx[idx]){
                                        nacc = nacc + 1
                                        }
                    }
                    else {  new.idx2 <-index1[1]
                              if(idrep[i]==idnist[new.idx2]){
                                                  nacc = nacc+1
                              }
         }
        }
      nacc
                    tmp.rlt = c(alpha,lambda,M,nacc,nacc/M)
                    rlt = rbind(rlt,as.numeric(tmp.rlt))
   }
}
          rlt = as.data.frame(rlt)
          dimnames(rlt)[[2]] = c("alpha","lambda","nquery","nmatch","acc")
          rlt
save(rlt, file="Pearsonandlasso.RData")


####################################################################

##  Plot for two-step approach--- Pearson’s correlation and Lasso regression

if(T){
          load("Pearsonandlasso.RData")
}
if(T){
          td = rlt
```

```
		td.x = sort(unique(rlt$alpha))
		td.y = sort(unique(rlt$lambda))
		n.x = length(td.x)
		n.y = length(td.y)
		td.z = matrix(0,n.x,n.y)
		for(i in 1:n.x){
			for(j in 1:n.y){
				tmp = td$acc[td$alpha==td.x[i] & td$lambda==td.y[j]]
				td.z[i,j] = tmp
			}
		}
}

cont.plot <- function(td=rlt,plot=F,fsize=1.2){
		td = rlt
		maxacc = td[td$acc==max(td$acc),]
		td.x = sort(unique(rlt$alpha))
		td.y = sort(unique(rlt$lambda))
		n.x = length(td.x)
		n.y = length(td.y)
		td.z = matrix(0,n.x,n.y)
		for(i in 1:n.x){
			for(j in 1:n.y){
				tmp = td$acc[td$alpha==td.x[i] & td$lambda==td.y[j]]
				td.z[i,j] = tmp
			}
		}
		if(plot){
			#par(mfrow=c(1,2))
			nf <- layout(matrix(c(1,2),1,2,byrow=TRUE), c(3.5,1), TRUE)

			x = td.x
			y = td.y
			volcano = td.z
			x.at <- x
			y.at <- round(y,2)

			# Using Terrain Colors

			par(mar=c(5,5,1.5,0))

			vbreaks                                                    =
sort(unique(c(0,quantile(td$acc,probs=c(.025,.25,.5,.75,.9,.95,.975,.99,1)))))

			print(vbreaks)
				ty = c(1:(length(vbreaks)-1))
				tz = matrix(vbreaks[-1],1,length(ty))
				tat = c(ty+.5)
				main.labs = " " #"(b)"
```

```
                    image(x, y, volcano, col=terrain.colors(length(ty)),axes=FALSE
                              ,breaks=vbreaks
                              ,xlab="alpha",ylab="lambda"
                              ,main=main.labs
                              ,cex=fsize
                              ,cex.axis=fsize
                              ,cex.lab=fsize
                              ,cex.main=fsize
                              )

                    points(maxacc$alpha,maxacc$lambda,pch=19,col=2)
                    axis(1, at=x.at,cex.axis=fsize)
                    axis(2, at=y.at,cex.axis=fsize)
                    box()
                    par(mar=c(3,2,1.5,3))
                    aa=vbreaks
                    image(x=1,y=ty,z=tz,col=terrain.colors(length(ty)),axes=F
                              ,breaks=vbreaks
                              ,xlab="Accuracy (%)",ylab=""
                              ,cex.lab=1.2
                              )
                              axis(4,at=tat,lab=round(aa[-1]*100,2),cex.axis=fsize)
                    box()
          }

}

cont.plot(td=rlt,plot=T)
```

CURRICULUM VITAE

Ruiqi Liu
E-mail: r0liu009@louisville.edu
Address:332 Idlewylde Dr. ,Louisville, KY 40206

EDUCATION

Master of Science in Biostatistics, August 2011 - May 2013(expected)
Department of Biostatistics and Bioinformatics, University of Louisville
Relevant Courses: Probability, Biostatistical Methods, Mathematical Statistics, Clinical Trials, Advanced Statistical Computing, Survival Analysis, Multivariate Statistical Analysis, Biology of Disease in Populations
Cumulative GPA: 3.74
Bachelor of Science in Pharmacy, September 2006 - July 2010
Department of Pharmacy, College of Life Sciences, China Jiliang University
Relevant Courses: Molecule Biology, Biochemistry, Microbiology and Immunology, Cell Biology, Pharmacology, Medicinal Chemistry, Pharmaceutics, Human Anatomy and Physiology.
Cumulative GPA: 3.60

RESEARCH EXPERIENCE

Research Assistant, Department of Biostatistics and Bioinformatics, University of Louisville. Sept. 2012 – present

- Conducted literature review related to data normalization
- Normalized metabolomics data with different approaches

Research Assistant, School of Life Sciences, China Jiliang University. September 2009 – May 2010

- Conducted lab work related to pharmaceutical research projects independently
- Learned to use HPLC, and did HPLC analysis independently
- Conducted various literature review related to pharmaceutical research project

ACADEMIC HONORS

- Merit-based scholarship and “Excellent Student”, China Jiliang University, March 2010
- Merit-based scholarship and “Excellent Student”, China Jiliang University, March 2009
- Merit-based Self-renewal scholarship, China Jiliang University, September 2007

PUBLICATIONS

- Weifeng Xu, Qin Chen, Ruiqi Liu, Fengbo Ren, Yifeng Zhou, Xiulian Lu. Synthesis of β-Amino Cyclone Catalyzed by Alkaline Al2O3. Asian Journal of Chemistry 2011,9, 4165-4168.