이학석사 학위논문

Detection of pancreatic cancer biomarkers using mass spectrometry

질량 분석법을 활용한 췌장암 바이오마커에 대한 연구

2015년 2월

서울대학교 대학원

통계학과

김 기 연

Detection of pancreatic cancer biomarkers using mass spectrometry

지도교수 장 원 철

이 논문을 이학석사 학위논문으로 제출함

2014년 10월

서울대학교 대학원

통계학과

김 기 연

김기연의 이학석사 학위논문을 인준함

2014년 12월

위 원 장	김 용 대	(인)
부위원장	장 원 철	(인)
위 원	이 재 용	(인)

Detection of pancreatic cancer biomarkers using mass spectrometry

by

Kiyoun Kim

A Dissertation
submitted in fulfillment of the requirement
for the degree of
Master of Science
in
Statistics

The Department of Statistics
College of Natural Sciences
Seoul National University
February, 2015

Abstract

Kiyoun Kim
The Department of Statistics
The Graduate School
Seoul National University

Mass spectrometry (MS) data analysis has been utilized to detect biomarkers for early cancer detection. In this research, the aim of analysis was to search biomarkers for pancreatic cancer that is a highly lethal malignancy and the fourth leading cause of cancer-related deaths. With the biomarkers detected in this study, we could improve the prognosis and survivals of pancreatic cancer patients.

MS data analysis typically consisted of three steps: preprocessing, classification, and variable selection. The results of preprocessing were usually non-fully populated data structure, so it was essential to substitute missing values for further advanced statistical inference. We used three methods to address missing issues.

We used four classification methods to discriminate the case and control group. Based on the prediction error, we compared the results of classification methods. We concluded that Peak Probability Contrast (PPC) method with bagging and random forest outperformed the others. Based on the variable importance measure in bagging and random forest, we selected the m/z 1,206, 1,465 as biomarkers for pancreatic cancer.

Keywords : biomarker, classification, missing values, pancreatic cancer, mass spectrometry

Student Number : 2013-20211

Contents

Chapter 1

Introduction

Pancreatic cancer, which is one of the most lethal malignancies, is the fourth leading cause of cancer-related deaths. Its five-year survival rate is just around 5 % because it is usually diagnosed at a late stage, and thus it is often too late to perform a curative surgery. In addition, even after surgery, it has a high recurrent rate and resistance to chemotherapy or radiation therapy (Siegel et al., 2014). Therefore, major advances in survivals of patients will be greatly aided by early detection of diagnosing and treating pancreatic cancer. Carbohydrate antigen 19-9 (CA 19-9) has been widely used for the biomarker of the pancreatic cancer. Unfortunately, CA 19-9 is not currently adequate as a early biomarker because of low sensitivity and specificity (Misek et al., 2007). Hence, this study focuses on the identification of the new serum protein biomarkers for the early detection of pancreatic cancer.

Mass spectrometry-based proteomics has been an indispensable tool for analyzing biological samples to conduct cancer-related studies (Aebersold et al., 2003). Especially, cancer biomarker discovery from mass spectrometry data has always been an active research area in proteomics. A large body

of research has been done in analyzing mass spectrometry data derived from Matrix-assisted laser desorption and ionization, time-of-flight (MALDI-TOF), or Surface-enhanced laser desorption and ionization, time-of-flight (SELDI-TOF). In this study, MS data obtained by MALDI-TOF are used as a essential screening tool for early detection of pancreatic cancer.

Many causes have been known as main risk factors for pancreatic cancer, including genetics, diabetes, smoking, and obesity. Among these risk factors, many results from meta-analysis supported a causal relationship between diabetes and pancreatic cancer. In Chari et al. (2005) and Huxley et al. (2005), recent-onset diabetes patients had 50 % greater risk of the malignancy compared with those who had long-term diabetes. Consequently, within the high-risk population in whom diabetes was already diagnosed, we want find novel biomarkers that differentiate the patients with both pancreatic cancer and diabetes and those with only diabetes.

It is very important to conduct preprocessing steps in mass spectrometry data analysis. Some changes in the preprocessing steps can affect the result of the analysis, thus we focus on the optimal choice for methods and parameter of the preprocessing steps. These steps include transformation, smoothing, normalization, baseline subtraction, peak detection, and peak alignment. Details are given in Chapter 2. Another major issue in MS data analysis is missing. In general, 10-20% of MS data from an experiment are missing. Simply ignoring all the variables with missing values might be a naive solution for the problem, only if a small number of variables are affected by missing. However, in most MS data analysis, the fraction of missing values are not negligible. To handle the missing data, we propose to use the following methods. i) Peak probability contrast ii) missing value imputation based on nonparametric methods such as k-nearest neighbor and random forest. Chapter 2 introduce precisely

these methods. To impute missing values, we generated random noises from the normal distribution with mean 0 and variance σ^2. To estimate σ^2, we considered the following two methods. First, to alleviate the high-fluctuation of the sample variances across the m/z values due to sample size, we used the loess estimator for estimating the variance in each m/z value. Second, we conducted wavelet transforms and computed the MAD variance estimator. The main theme of this study is to compare the performance of classification among the methods that address the missing problems.

For the classification, the matrix with full data is produced after preprocessing and imputation. We spilt the training data and test data for calculating the test error. To compare the results of the methods clearly, we repeat the separation of training and test data 100 times. Consequently, the prediction error and Area Under the Curve (AUC) on the ROC curve are computed as the criterion of performance on every iteration. We compare the classification and missing data imputation with these performance criterion.

Chapter 2

Material and Methods

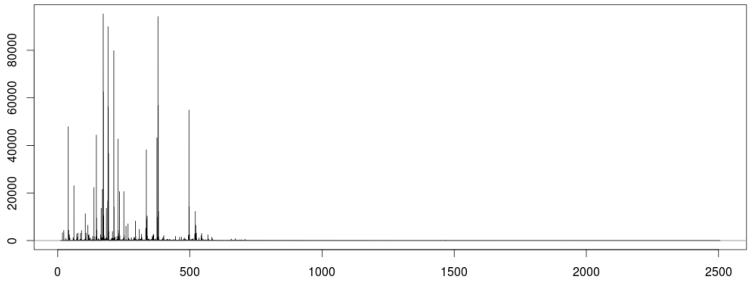
In this chapter, we will introduce the material of analysis and methods used in preprocessing, solution for missing data problem and classification. The data were collected at Seoul National University Bundang Hospital between September 2009 and December 2011. It consisted of MALDI-TOF/MS spectra that have mass-to-charge ratio (m/z value) and its corresponding intensity.

2.1 Experimental protocol and data description

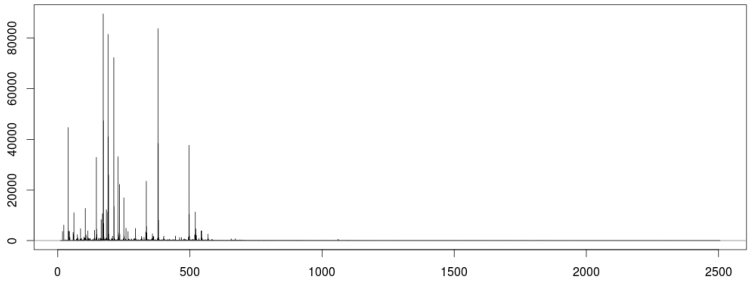
At a experimental level, the following procedures were implemented in each serum sample for MALDI-TOF/MS data. Approximately 5-10 mL of blood was obtained from each patient. Blood was drawn into serum-separating tubes and centrifuged at 2,000 $\times g$ for 10 minutes at room temperature. The serum was removed and transferred to a polypropylene, capped tube in aliquots, and the samples were stored at –70 °C. Sera (25 μL) were incubated with 100 μL methanol/ chloroform (2:1, v/v) for 10 minutes at the room temperature after

vortexing. The mixture was centrifuged at 6,000 $\times g$ for 10 minutes at 4°C. The supernatant was completely dried in a concentrator for 1 hour and resolved in 30 μL of 50% acetonitrile/0.1% trifluoroacetic acid (TFA) on a vortexer for 30 minutes. The methanol/ chloroform extract was mixed (1:12, v/v) with an α-cyano-4-hydroxycinnamic acid solution in 50% acetonitrile/0.1% TFA, and 1 μL of the mixture was spotted on the MALDI target for analysis. The mass spectral data represent the average of 20 accumulated spectra. To minimize experimental errors, variable factors including focus mass, laser intensity, target plate, and data acquisition time were tested. Ideal focus mass and laser intensity were fixed at 500 m/z and 5,000 m/z, respectively. For the fixed focus mass and laser intensity, each sample was analyzed 6 times using different extractions and data acquisition times.

We had 118 patients' serum samples, consisting of 93 pancreatic cancer patients and 25 diabetes patients. Among pancreatic cancer patients, 53 patients had diabetes and 25 patients did not have diabetes. Each sample was collected before curative surgery and was analyzed 6 times to reduce the experimental fluctuations. Because diabetes has been known as a high-risk factor for pancreatic cancer, we limited the population to the high-risk group in which individuals had diabetes in advance. Note that the 53 patients that had pancreatic cancer and diabetes were used for the case group, and the 25 diabetes-only patients were used for the control group. The number of patients used for MS analysis was 78, hence the total number of spectra was 468 because of replicates. The MS spectra were measured at different sites, extending from 9 to 2,500 Da. The mean length of spectra was 165,977. The average of peak widths was approximately 0.015 Da. Note that the increment of m/z values was slightly increasing with a linear trend.



(a) Case group



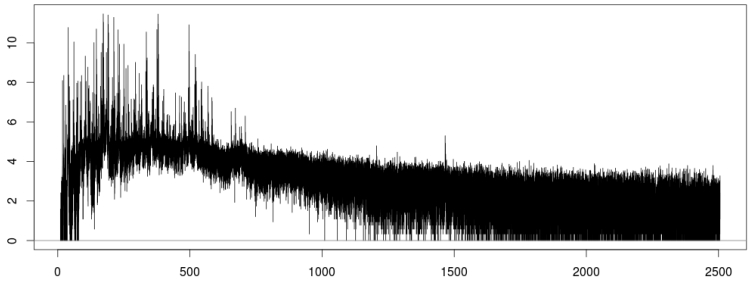
(b) Control group

Figure 2.1: Raw spectra

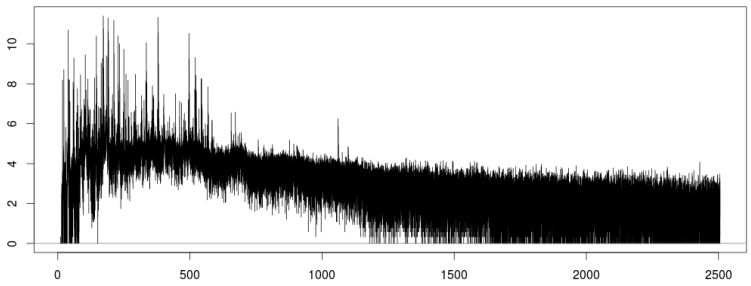
2.2 Preprocessing

The preprocessing step in MS data analysis is one of the most important steps. The results of analysis crucially depend on the quality of these processes, we try to conduct the preprocessing carefully. We begin with the raw MS data. First, we split the training and test sets to calculate the prediction error. If the training and test sets are processed separately, m/z value sites are not same between the training and test sets in the final data matrix. In this case, we cannot apply the classification model fitted in training set to the test set. Thus, we compute the common peaks from the training set. Given the com-

mon peaks, we extract the features from test set. The followings are typical preprocessing steps in MS data analysis. i) Transformation ii) Smoothing iii) Baseline subtraction and Normalization iv) Peak detection v) Peak alignment. Figures 2.2-2.6 present a typical preprocessing step in one spectrum of the case and control group.



(a) Case group



(b) Control group

Figure 2.2: Preprocessing: Log Transformation

2.2.1 Transformation

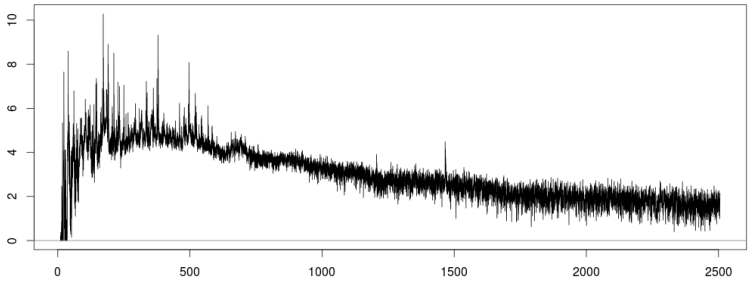
To begin with raw spectra, we can see that the intensity has high fluctuations across the m/z values. See Figure 2.1. We can show that it has high intensity values in the area of low m/z. In contrast, a low level of intensity is in the m/z values above almost 600 Da, even non-visible in the plot. We have to implement the transformation step so that spectrum has the adequate variability across m/z values. A typical method of transformation is the square root transformation or logarithm transformation. In this case, we use the $\log(x+1)$ transformation to deal with the case about the zero value of intensity. The result of transformed spectrum is presented in Figure 2.2.

2.2.2 Smoothing and Baseline subtraction

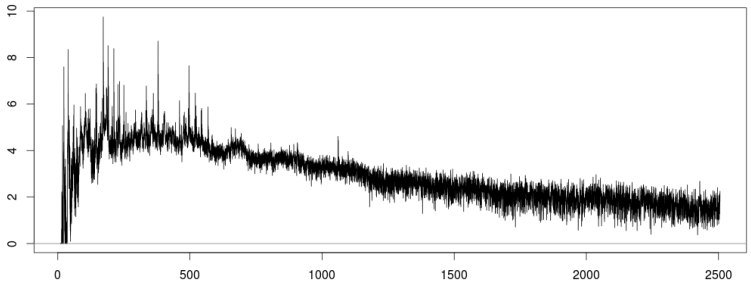
Smoothing process is normally used in MS data analysis. In MALDI-TOF/MS, this step is conducted for smoothing over the isotopic envelop. Tibshirani et al. (2004) used a super-smoother with a span of specific value. The methods encompass popular methods such as Moving average smoother and the Savitzky-Golay smoother. The result of smoothed spectrum is plotted in Figure 2.3.

Furthermore, MALDI spectra often contain a large amount of chemical or electronical background noises. To estimate the background noise for spectra, non-parametric regression methods are used for eliminating the baseline fluctuations. After estimating the baseline for the spectra, the baseline corrected spectra are calculated by subtracting from the smoothed data. Wu et al. (2003) used local regression to estimate the baseline intensity values. Ryan et al. (1988) introduced the Statistics-sensitive Non-linear Iterative Peak-clipping (SNIP) algorithm for baseline estimation. We used Moving average for smooth-

ing and the SNIP algorithm for the baseline estimation. Figure 2.4 shows the estimated baseline by the SNIP algorithm.



(a) Case group

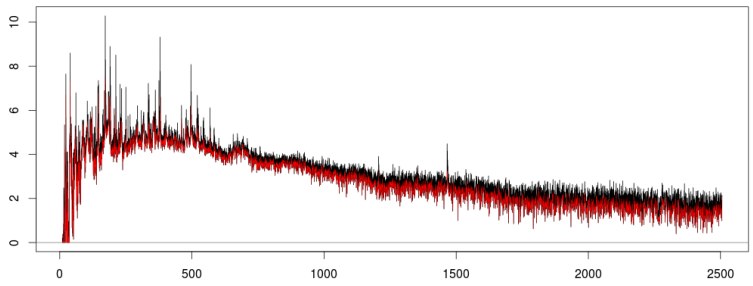


(b) Control group

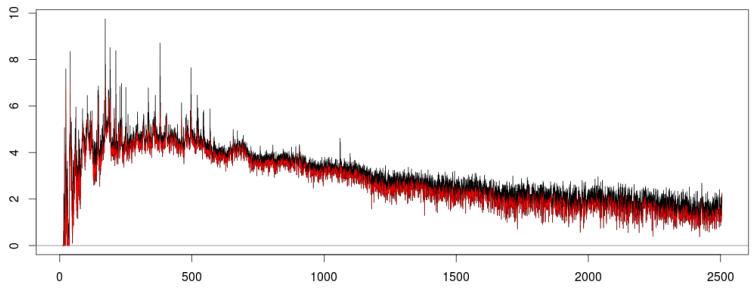
Figure 2.3: Preprocessing: Moving average Smoothing

2.2.3 Normalization

The purpose of normalization is to remove systematic differences between spectra due to varying the total amount of proteins, degradation over time in samples, or variation in the instrument sensitivity. A number of different normalization methods are provided. Total Ion Current (TIC) is a common choice for normalization. First, the Area Under the Curve (AUC) is measured in each spectrum. Second, all the intensities in the spectra are divided by the



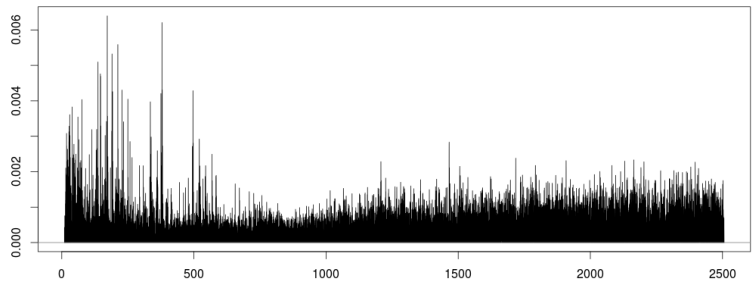
(a) Case group



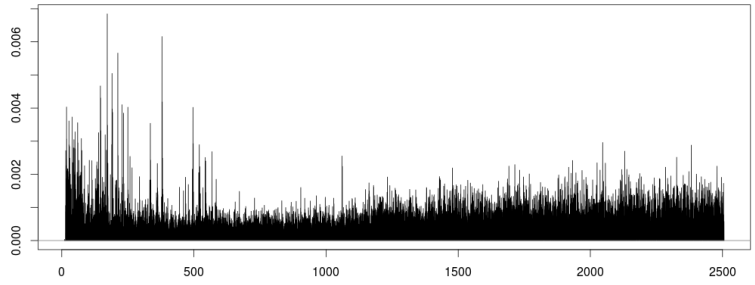
(b) Control group

Figure 2.4: Preprocessing: Estimated Baseline by using SNIP algorithm

calculated AUC. After that, they are adjusted by the rescaling coefficients. The common choice for the rescaling coefficient is set to one. An alternative choice for the coefficient is the average AUC over all spectra. A quantity of the normalized intensity at specific m/z value means the proportion of abundance within the spectra (Sauve et al., 2004). Probabilistic quotient normalization could be a robust method for the normalization (Dieterle et al., 2006). We chose the TIC method for the normalization and set rescaling coefficient to be one. Note that the scale of intensity is altered by normalization, as shown in Figure 2.5.



(a) Case group

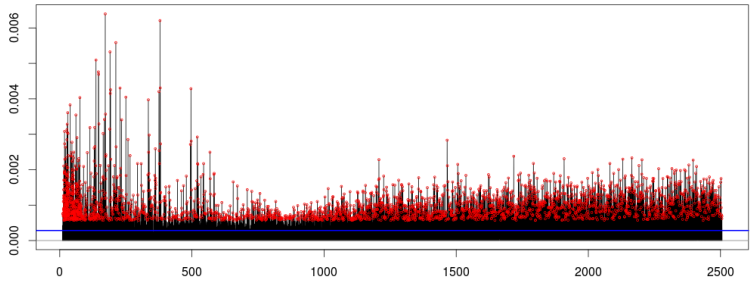


(b) Control group

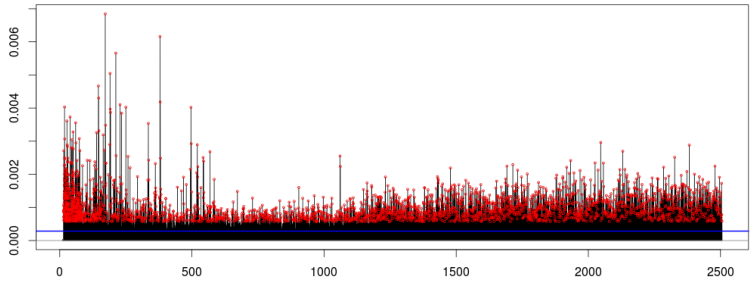
Figure 2.5: Preprocessing: Baseline Subtraction and TIC normalization

2.2.4 Peak detection

In this step, we extract peaks from the normalized spectra. The intuitive idea of peak detection is regarding peaks as local maxima in the spectra. Yasui et al. (2003) proposed to find m/z sites, whose intensity is highest among its neighborhoods. An alternative is to find peaks that exceed a certain threshold in the spectra. Signal-to-Noise Ratio (SNR) is used to define the threshold, which is the value of estimated noise $\times$ SNR. The Median absolute deviation (MAD) is often used for estimating noise in the spectra. Figure 2.6 presents detected peaks that are local maxima above a SNR threshold.



(a) Case group



(b) Control group

Figure 2.6: Preprocessing: Peak Detection : Red points are the detected peaks and Blue line is the estimated noise by MAD

2.2.5 Peak alignment

Due to the existence of chemical and electronic noises, the same peaks may be horizontally shifted in different spectra, and not match in the all spectra. In addition, because of the detection limit in peak detection step, the deleted peaks (m/z values) differ between the spectra. Thus, to do the further statistical inference, it is necessary to align the peaks across multiple spectra. This step includes the 3-step method for the alignment. At first, the peaks from spectra are aligned by the Self-Calibrated warping (SCW) algorithm (He et al., 2011). Roughly speaking, the idea of algorithm is the modification of peak

sites based on the reference peak by estimating the warping function. To estimate the warping function, we use the Loess estimation. In the second step, peaks that present in at least 3 out of 6 technical replication are retained, and the intensities corresponding to the retained peaks are averaged to create a single spectrrum per patient. Finally, after intra-sample alignment, the peaks that present in at least 50% in the two groups, the control and case group, are retained.

Although we use reasonable methods and parameter for the preprocessing, it is not easy to choose the processes that are best suited for the present data. These approach work well on some data structure, but may not effective on the others. So, several trials for the suitable methods would be essential for MS data preprocessing.

If the entire preprocessing step are performed in the training set and test set respectively, derived common peaks are not same between the training set and test set. If so, we cannot apply the classification models (from the training set) to the test set. In this case, the test set is pre-processed until the normalization step and we opt the peak extraction based on the common peaks from the training set. In the test set, we attain the highest intensity among peaks that lies with $\pm$s position of common peak (Tibshirani et al., 2004).

2.3 Missing value in mass spectrometry data

In MS data analysis, 10-20% missing data generally exist in the final rectangle matrix. The result of preprocessing is the matrix that is not fully populated because of the biological or technical reason. The nature of missing data can

be derived from the heterogeneity between patients' sample. For example, particular proteins (represented by m/z value in our study) of the one serum sample are large enough to be detected as peaks. However, the other sample may not have those or have in relatively small quantity, though it exists. Thus, missing data can be generated because of the difference among samples at specific m/z values. On the other hand, missing data can be caused by the elimination of proteins due to the detection limit in the peak detection step. In practice, the origin of missing data is not easily identified and both are often present in the data simultaneously (Hrydziuszko, O., and Viant, M. R., 2012).

A simple solution for the problem with missing data is to consider solely all the variables that do not have the missing data. This method can be a quite reasonable solution when only small proportion of the missing data is present. However, it is not a typical case in most MS data analysis. Therefore, We consider three other solutions for missing data problem. i) Peak Probability contrast (PPC) ii) missing value imputation by using non-parametric methods. iii) Random noise imputation generated from normal distribution. Tibshirani et al. (2004) introduced the Peak Probability Contrast (PPC) method. In brief, this method is the process to make the binary, 0-1 matrix based on the estimated split point for each peak. The binary information means the presence or absence of the peaks in the matrix. The following algorithm is given. At first, if there is no peak in the matrix, we take zero for the substitutes. Then the details are given in the following.

1. Let $P_{ik}(\alpha)$ be the proportion of spectra in group k with a peak at site i larger than $q(\alpha, i)$

$$P_{ik}(\alpha) = \sum_{j \in G_k} \mathrm{I}\left(y_{ij} - \mathrm{q}(\alpha, i)\right) / n_k$$

Where G_k is the case and control group of size n_k, $q(\alpha, i)$ is the α quantile

of the peaks y_{ij} at site i, and I(x) is the indicator function which equals one if x is larger than 0 and zero otherwise.

2. Find the $\widehat{\alpha}(i)$ to maximize $|p_{i2}(\alpha) - p_{i1}(\alpha)|$ and set $\hat{p}_{ik} = p_{ik}[\hat{\alpha}(i)]$

This method means that we find the quantile that maximally discriminates between the two groups, the case and control groups. By using the computed quantile $q(\hat{\alpha}(i), i)$ as the threshold at site i, the element y_{ij} (i-th m/z site and j-th sample) is set to 1 if its intensity is above the threshold and set to 0 otherwise. The test data are also produced by applying the PPC idea. Finally, the resulted binary matrix contains only the information of peak presence or absence. Some missing value imputation approaches have been developed to handle nontrivial missing problems. The objective of the missing value imputation is to produce the complete dataset for further statistical analysis by replacing the missing value with plausible substitutes. In our study, we apply the two non-parametric methods for missing value imputations, the weighted K-nearest neighbor and random forest methods (Hrydziuszko, O., and Viant, M. R., 2012).

The weighted K-nearest neighbor idea was suggested in the paper of Troyanskaya et al. (2001) and usually implemented in the gene expression microarray data. Recently, it is also used in the proteomics and metabolomics data. To estimate the missing values, we first calculate the similarity measure between variables (m/z values) and select the K m/z values whose intensity vector is similar to the each m/z value. Generally, the reciprocal of Euclidean distance is used as a similarity measure. The small distance between two m/z values means that they have high similarity. Then, the weighted average intensity

of K neighbors in terms of similarity is substituted with the missing value. The similarity is used as the weights and if y_{jh} is missing at j site, it is excluded from N_i. The K value should be determined, but the KNN imputation has no theoretical basis for choosing optimal K. So, we choose the K value empirically (Troyanskaya et al., 2001).

$$\text{Similarity measure} : s(i,j) = \frac{1}{\sum\limits_{k \in O_i \cap O_j} (y_{ik} - y_{jk})^2}$$

$$O_i = \{\ k \mid \text{the } k-\text{thcomponentof } y_i \text{ is observed}\}$$

$$\text{Estimated missing entry} : \hat{y}_{ih} = \frac{\sum\limits_{j \in \mathrm{N_i}} s(i,j)\ y_{jh}}{\sum\limits_{j \in N_i} s(i,j)}$$

N_i = the index set of K-nearest neighbor of i-th m/z value, h = missing entry in i-th m/z value

Random Forest is one of the classification and regression technique, ensemble methods based on the tree algorithm. Iterative imputation method based on the random forest was implemented in various papers to handle the missing value. This approach constitutes the N bootstrapped samples and without a need of test set for calculating imputation error, the out-of-bag (OOB) error estimates is used for estimating the imputation performance. By averaging over the N unpruned tree, the RF imputation algorithm could be considered as the multiple imputation scheme. To begin with, we have to construct the proximity matrix. The proximity matrix is an $n \times n$ symmetric matrix with the number of sample n. At one growing tree, If sample i and j both land in the same terminal node, the (i, j) element of proximity matrix increases by one.

After fitting all N bootstrapped samples, it is divided by the total number of growing tree in the run. Finally, the element of proximity matrix indicates the proportion of trees over which each pair of samples fall in the same terminal node. And it also means the proximity between the two samples. The more proximity is, the closer the two samples are (He et al., 2011).

The followings are the details of algorithm. The first step is to replace the missing values with the column (m/z value) medians as an initial value. The next thing is to compute the proximity matrix, and by using the proximity as the weight, we calculate the weighted average of non-missing samples at each variable. This weighted average is used to estimate missing values and update missing entry in the matrix. This process continues until some stopping criterion is satisfied. Consequently, we can obtain full data without missing values. `rfImpute` function in `randomForest` R library provides missing value imputation algorithm by using the random forest.

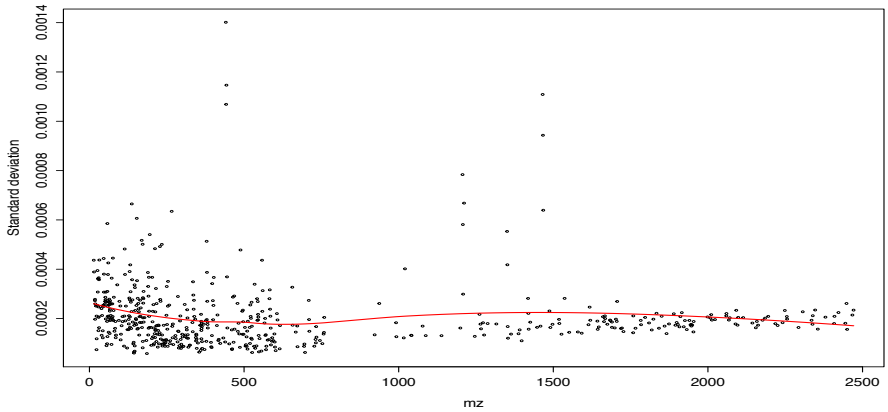


Figure 2.7: variance estimation by using Loess

Final methods for imputation is to generate random noise from the normal distribution with mean 0 and variance σ^2. To estimate the variance of the normal distribution, we implement two robust variance estimation methods. First, Loess estimation is used for estimating the sample variance of each peak. As shown in Figure 2.7, each point means sample variance in each m/z value. It has high fluctuation because of small sample size and variability. To alleviate its variability, we use fitted variance by using Loess for random noise imputation.

Second, median absoulte devation (MAD) estimator in Wavelet regression is used for estimating the variance. we fit the wavelet regression to each spectra and compute a robust variance estimator based on the wavelet coefficient (Johnstone, I. M., and Silverman, B. W. , 1997). After variance estimation for normal distribution, missing value is substituted with the random noise which is extracted from the normal distribution with mean 0 and estimated variance.

$$\hat{\sigma} = \sqrt{n}\ \frac{\text{median}(|\tilde{\beta}_{J-1,k} - m| : k = 0, \cdots, 2^{J-1} - 1)}{0.6745}$$

$$\text{where}\ \ m = \text{median}(\tilde{\beta}_{J-1,k} : k = 0, \cdots, 2^{J-1} - 1$$

2.4 Classification

We apply four classification methods, CART, Bagging, Random Forest, and Lasso regression with the complete data (Breiman et al., 1984; Breiman, 1996,

2001; Tibshirani, 1996).

Classification And Regression Tree (CART) is a decision tree algorithm for classification and regression. We select the best split to differentiate observations of two groups based on explanatory variables. More precisely, to start with a root node, we find the set that minimize the impurities in two child nodes for each variables (m/z values) and choose the split that gives the minimum impurity on overall variables and its corresponding set. The algorithm ends when stopping criterion is satisfied, otherwise same procedure is applied to each child node. CART is commonly used for classification and easily interpretable. However, it is certainly unstable for the classification performance when some changes exist in the training and test data. It is expected that the classification results produced by CART is not relatively superior to other classification methods.

Bagging and Random Forest are ensemble techniques that combine many weak learners fitted in the bootstrap samples to produce better classification performance than the single tree algorithm. The bootstrap samples are obtained by drawing training samples repeatedly with replacement. We fit the decision trees, usually consist of single split tree(stump), on each bootstrapped sampled data sets, and obtain the predictions by using fitted model on each bootstrapped samples. Finally, we classify the samples by the majority vote in the test data. By aggregating the results of the predictions, we can reduce the variance of the predictions. In this case, all variables are used to fit the model for Bagging. Random Forest also uses the bootstrapped samples for reducing its variance. In contrast to Bagging, RF draws only a limited random selection of m variables within p total variables and only those m variables are considered for splitting. Typically, $m = \sqrt{p}$, or $\log_2 p$, where p is the number of variables. By using this method, we can reduce the correlation between

decision trees fitted in bootstrapped samples.

Lastly, it is possible approach to apply the logistic regression for the classification. If we construct the ordinary logistic regression, we have

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \sum_{k=1}^{p} \beta_k x_{ik}, \quad \text{where } \pi_i = \mathrm{P}(Y_i = 1|x)$$

However, in our study, ordinary logistic regression can not be used for classification because the number of variables are greater than the sample size. As an alternative solution, Least Absolute Shrinkage and Selection Operator (LASSO) are used for logistic regression. The Lasso is shrinkage and selection method for regression, and compared to ordinary least squares (OLS) estimator of coefficients, it shrinks the estimated coefficient toward zero by using L_1 penalty function. The implement of Lasso necessitates the imposing L_1 constraint to the log-likelihood. The L_1 constraint is expressed by

$$\sum_{k=1}^{p} |\beta_k| \leq \lambda, \ \lambda > 0$$

Here, λ is the tuning parameter, which has to be selected via the methods including 10 fold cross-validation. The key property of Lasso is to produce a sparse model consisting the variables with non-zero estimated coefficients.

To evaluate performance among classification methods, we randomly split the original data into the training and test data with the ratio 70% to 30%, respectively. The training data includes 37 cases and 18 controls, and the test data includes 16 cases and 7 controls. To impartially compare the classification results among solutions with missing problem, we run 100 times iterations for training and test data separation. In each iteration, we compute the prediction error, and also Area Under the Curve (AUC). A Receiver Operating Characteristic (ROC) curve is also created by the sensitivity and specificity on every iteration.

2.5 Software tools

All data analysis including preprocessing and classification were conducted in R version 3.0.1. `MALDIquant` library was employed for most of preprocessing step (Gibb, S., and Strimmer, K., 2012). And `Tree, randomForest, glmnet` was used for the classification methods, which is CART, Bagging, randomForest, and Lasso regression. The algorithms of others were produced by author's work.

Chapter 3

Results

At the beginning of analysis, the raw MS data were imported by `MALDIquant Foreign` into `MassSpectrum` object in R for implementing the `MALDIquant` preprocessing. The total number of spectra was 468 : 78 biological samples with 6 technical replicates. With the randomly choosing split criterion, the training and test data were separated into 78 training and 23 test samples, respectively. As described in Chapter 2, we independently conducted the preprocessing of the training data and test data. The reason for separately preprocessing between training and test data was due to prevent the preprocessing from changing classification results. For training data, the preprocessing was conducted from transformation to peak alignment. Test data was carried out from transformation to normalization. According to peak extraction idea, the test data could be obtained by the non-missing data based on common peaks (m/z values) derived from the training data.

Here we had the training data with p variables (m/z values in our study) and 55 samples with some missing values, and test data with p variables and 23 samples with complete-case data. In our study, the proportion of missing

Method	PPC	KNNimpute	RFimpute	Loess	Wavelet
CART	0.1348	0.1330	0.1409	0.1340	0.1400
	(0.0704)	(0.0585)	(0.0684)	(0.0632)	(0.0635)
Bagging	0.0713	0.0887	0.0921	0.0809	0.0800
	(0.0410)	(0.0448)	(0.0486)	(0.0430)	(0.0434)
Random Forest	0.0748	0.0887	0.0896	0.0791	0.0817
	(0.0422)	(0.0430)	(0.0537)	(0.0409)	(0.0478)
Lasso	0.1052	0.1269	0.1435	0.0879	0.0835
	(0.0457)	(0.0949)	(0.1202)	(0.0523)	(0.0401)

Table 3.1: Prediction error. Standard deviation is given in parenthesis.

values in the training data matrix was about 20%. We used 5 methods to address the imputation of missing data.

Through randomly separation of the training and test data repeatedly 100 times, we computed the prediction error to use classification performance criterion. Note that we obtained 100 misclassification rates, with mean and standard deviation error. Also, the performance of methods were evaluated in terms of the Area Under the Curve (AUC) and the Receiver-Operating Characteristics (ROC) curve, as shown in Figures 3.1 and 3.2.

Table 3.1 provided the prediction error for each method. Peak Probability Contrast (PPC) methods outperformed the others when the bagging and random forest were used for classification. In contrast, loess and wavelet worked better with lasso. We concluded that CART yielded the highest prediction error among missing imputation methods. Relatively high standard deviation means that CART algorithm was not robust when some changes existed in the data.

Hrydziuszko, O., and Viant, M. R. (2012) recommended that KNN imputation. Gromski et al. (2014) showed that RF imputation outperformed KNN imputation with zero or median replacement. However, we found that these methods were not significantly better than PPC or replacing random noises from normal distribution. Thus, there's no unique optimal solution for missing data in MS data.

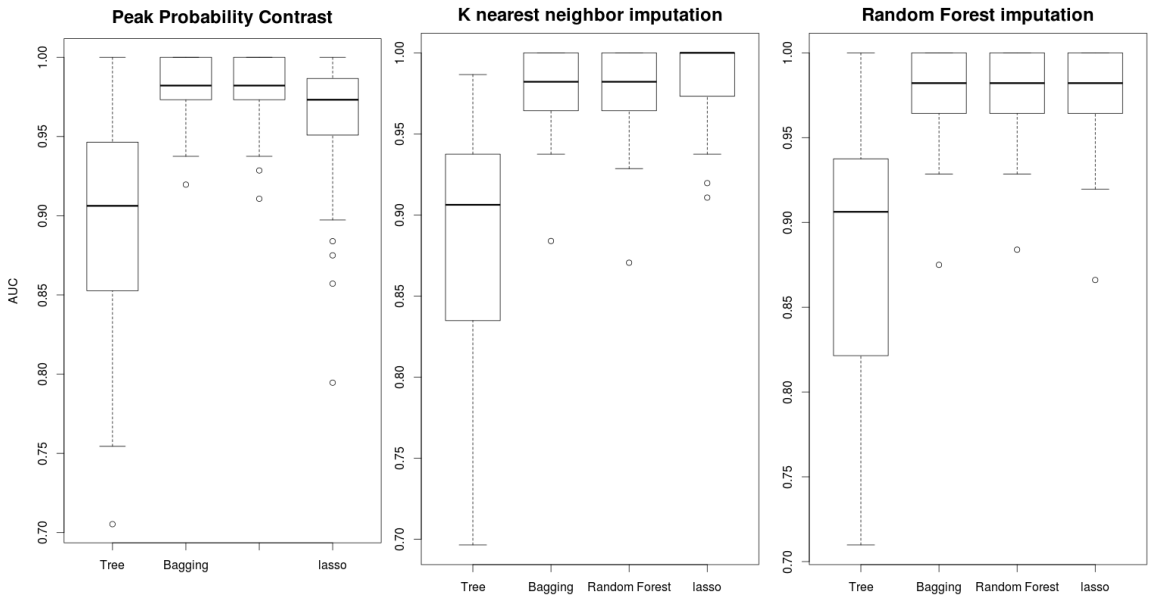
If missing entries scarcely existed in the data, the imputation of missing data was not needed for the analysis. Simply ignoring the variables with missing data could be a reasonable solution. However, it could rather generate a bias to the inference, or some significant peaks might be removed due to missing data. The noise imputations with small arbitrarily chosen numbers would be a reasonable choice for the problem when the majority of missing data were generated from the measurements below the SNR noise (Detection limit). On the other hand, if the majority of missing data were largely unknown, or non-detects rather than measurements below SNR noise, KNN or RF imputation might be superior to other imputation methods (Hrydziuszko, O., and Viant, M. R., 2012). Consequently, to find the appropriate solution for missing data, we should focus on the nature of the missing data generation.

Peak Probability Contrast (PPC) method had more advantages than other noise imputation methods. Its calculation was simple and easily understandable. Also, its estimated difference of peak probability $|\hat{p}_{i2} - \hat{p}_{i1}|$ roughly meant the relative importance in discriminating the case and control groups. Above all, it resulted in the best classification performance in this study.

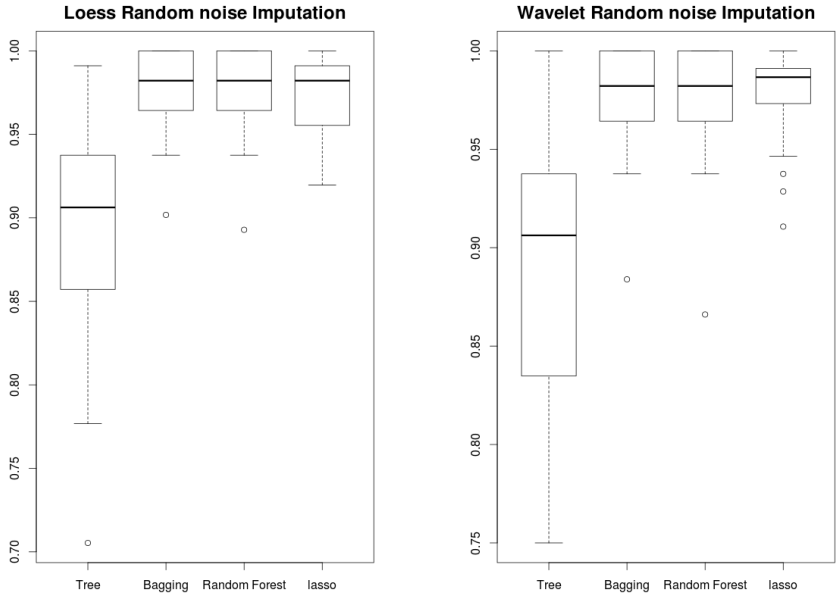
The purpose of this study was to select the biomarkers for detecting the pancreatic cancer. Thus, based on the optimal methods for classification re-

sults, we calculated variable importance measures by computing the mean decrease of prediction accuracy or Gini index in bagging and random forest. We concluded that the results of bagging and random forest with PPC method outweighed the others. Figure 3.3 shows that the top 10 m/z values is plotted in terms of mean decrease accuracy and Gini index.

Many m/z values were commonly considered as the selected biomarkers. According to variance importance, m/z 175, 212, 1,206, 1,207, 1,351, 1,465, 1,466, 1,467 were detected as biomarkers for pancreatic cancer. For internal validation for biomarkers, we simply plotted all the intensities on those biomarkers between two groups. See Figure 3.4, 3.5. Based on confirmation for the visually difference in the raw intensity plot between case and control groups, we finally observed at m/z 1,206, 1,207, 1,465, 1,466, and 1,467 as biomarkers for pancreatic cancer.

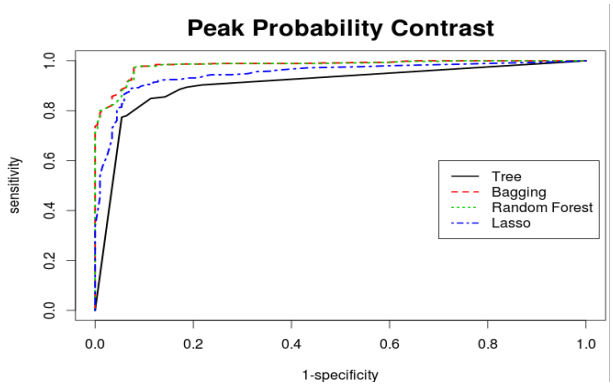


(a) PPC (b) KNNimputation (c) RFimputation

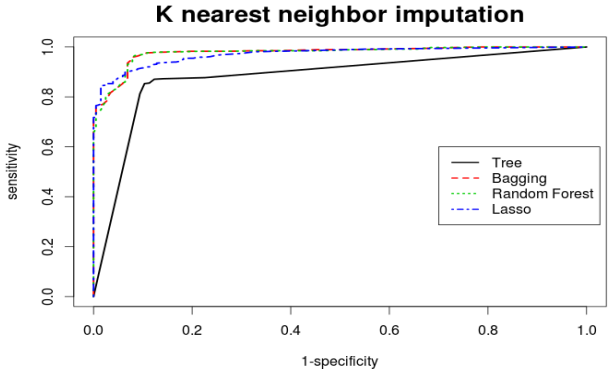


(d) Loess noise imputation (e) Wavelet noise imputation

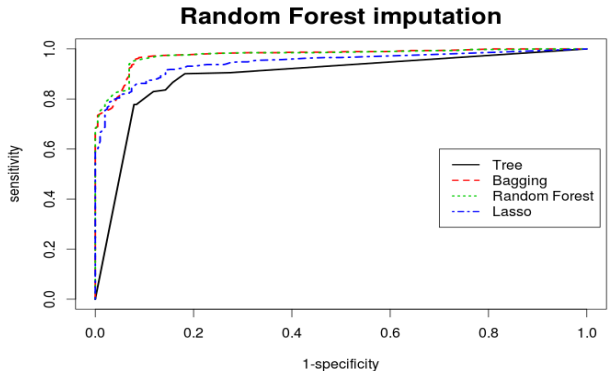
Figure 3.1: Area Under the Curve (AUC)



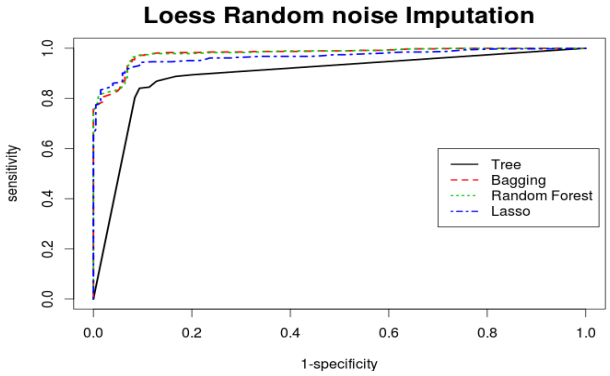
(a) PPC



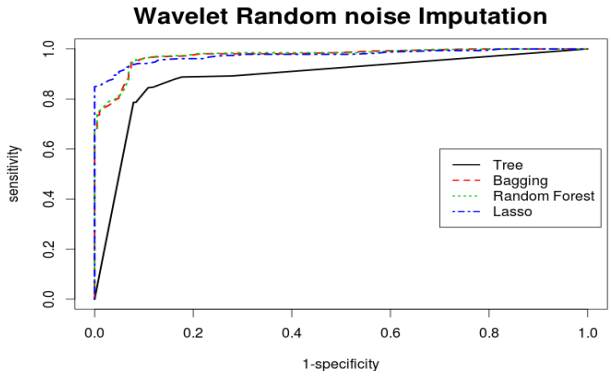
(b) KNNimputation



(c) RFimputation

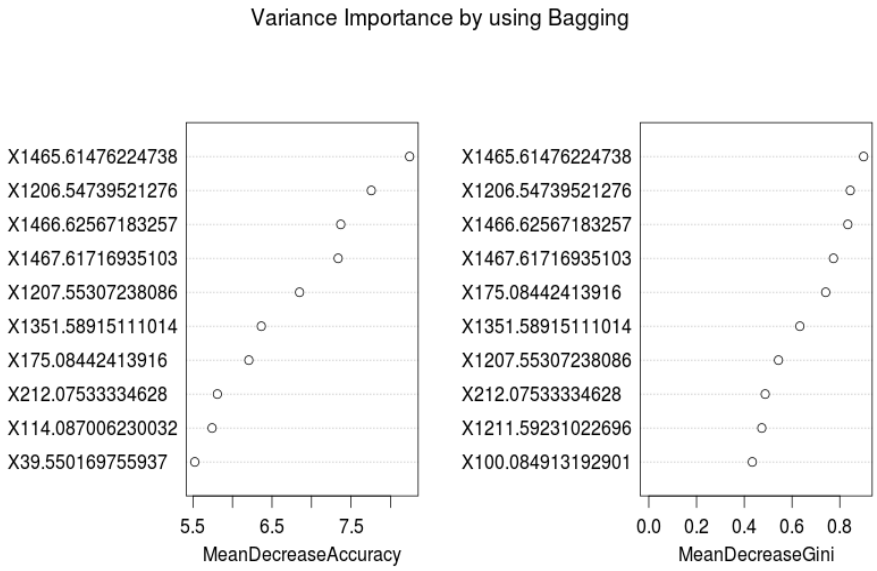


(d) Loess noise imputation

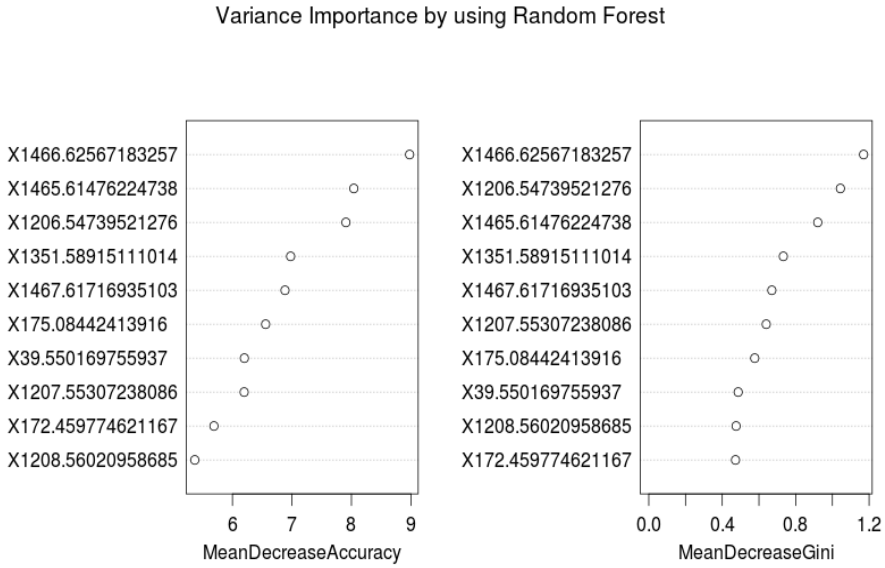


(e) Wavelet noise imputation

Figure 3.2: Receiver operating characteristic (ROC) Curve



(a) Bagging



(b) Random Forest

Figure 3.3: Mean decrease accuracy and Gini index by using PPC

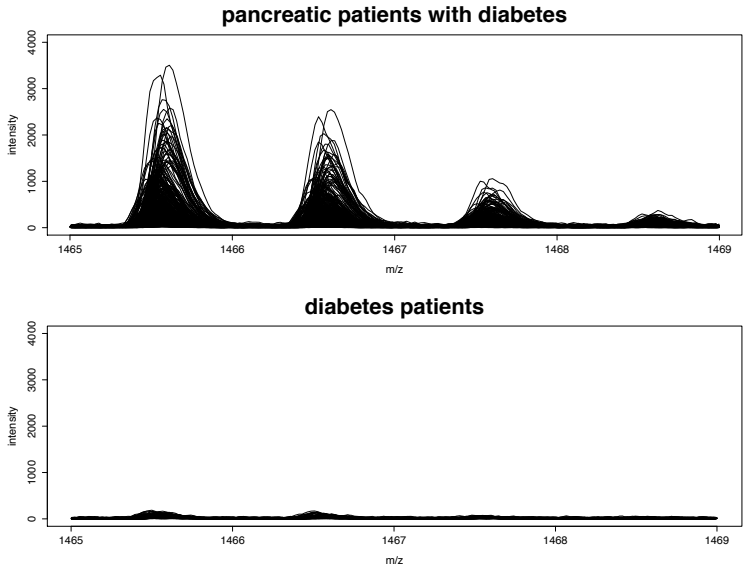


Figure 3.4: Raw intensities between cases and controls from m/z 1,465 to 1,468

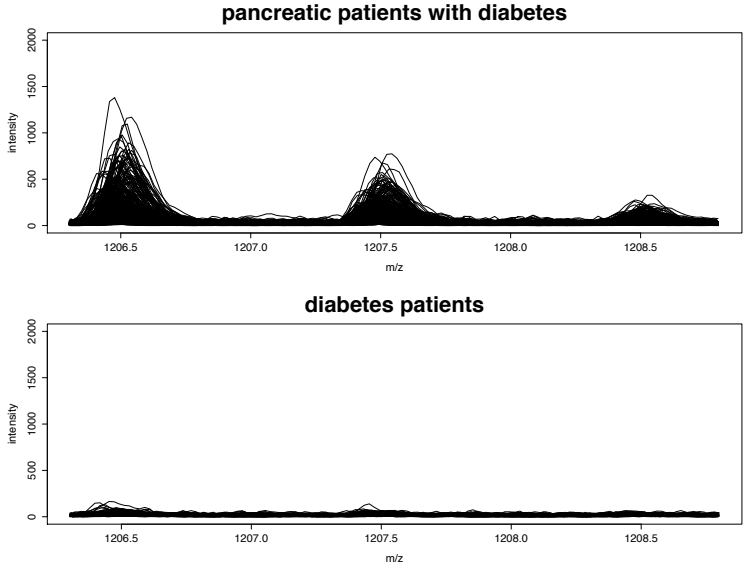


Figure 3.5: Raw intensities between cases and controls from m/z 1,206 to 1,207

Chapter 4

Conclusion

To detect biomarkers for early detection of pancreatic cancer, we analyzed MS data. A typical MS analysis procedure included the preprocessing, classification, and variable selection. A variety of methods about preprocessing have been proposed in many papers. Choosing the optimal preprocessing is crucial in MS data analysis.

To differentiate the case and control groups, we performed CART, bagging, random forest, and lasso regression. After fitting the classification models, we could extract several serum proteins represented by m/z values through computations of the variance importance. The missing data problem in the MS data analysis was not trivial in most practical cases. Similar to most MALDI-TOF/MS analysis, non-trivial missing problems arose in this study. In general, when missing data did not largely exist in the data or was derived from the intensity below estimated noise, replacing the missing data with small predefined value, mean, median, or random noise from the normal distribution for the continuity could be a reasonable solution. In contrast, if the nature of missing data is not due to detection limit, it was rational to use KNN or RF

imputation.

In our study, we concluded that PPC method was the best solution for the missing data problem. It was also quite simple and easy to be implemented and conduct the computing. Many cases with MS data analysis could be well suited for PPC method and it will produce reasonable results for the analysis.

In conclusion, we concluded that m/z 1,206, 1207, 1,465, 1,466, and 1467 was considered as biomarkers for early detection of pancreatic cancer. Because these m/z sites was too close to identify as separate and ionized protein could be distributed in range of m/z vales, we merged them and concluded m/z 1,206, 1,465 as biomarkers.

Reference

Aebersold, R., and Mann, M. (2003). Mass spectrometry-based proteomics. *Nature*, **422**(6928), 198-207.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. Chapman & Hall

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**(2), 123-140.

Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5-32.

Chari, S. T., Leibson, C. L., Rabe, K. G., Ransom, J., de Andrade, M., and Petersen, G. M. (2005). Probability of pancreatic cancer following diabetes: a population-based study. *Gastroenterology*, **129**(2), 504-511.

Dieterle, F., Ross, A., Schlotterbeck, G., and Senn, H. (2006). Probabilistic quotient normalization as robust method to account for dilution of complex biological mixtures. Application in 1H NMR metabonomics. *Analytical Chemistry*, **78**(13), 4281-4290.

Gibb, S., and Strimmer, K. (2012). MALDIquant: a versatile R package for the analysis of mass spectrometry data. *Bioinformatics*, **28**(17), 2270-2271.

Gromski, P. S., Xu, Y., Kotze, H. L., Correa, E., Ellis, D. I., Armitage, E. G., Turner, M.L., and Goodacre, R. (2014). Influence of missing values sub-

stitutes on multivariate analysis of metabolomics data. *Metabolites*, **4**(2), 433-452.

He, Y. (2006). Missing data imputation for tree-based models (Doctoral dissertation, University OF California Los Angeles).

He, Q. P., Wang, J., Mobley, J. A., Richman, J., and Grizzle, W. E. (2011). Self-calibrated warping for mass spectra alignment. *Cancer informatics*, **10**, 65.

Hilario, M., Kalousis, A., Pellegrini, C., and Mueller, M. (2006). Processing and classification of protein mass spectra. *Mass spectrometry reviews*, **25**(3), 409-449.

Hrydziuszko, O., and Viant, M. R. (2012). Missing values in mass spectrometry based metabolomics: an undervalued step in the data processing pipeline. *Metabolomics*, **8**(1), 161-174.

Huxley, R., Ansary-Moghaddam, A., De González, A. B., Barzi, F., and Woodward, M. (2005). Type-II diabetes and pancreatic cancer: a meta-analysis of 36 studies. *British journal of cancer*, **92**(11), 2076-2083.

Johnstone, I. M., and Silverman, B. W. (1997). Wavelet threshold estimators for data with correlated noise. *Journal of the royal statistical society: series B (statistical methodology)*, **59**(2), 319-351.

Misek, D. E., Patwa, T. H., Lubman, D. M., and Simeone, D. M. (2007). Early detection and biomarkers in pancreatic cancer. *Journal of the National Comprehensive Cancer Network*, **5**(10), 1034-1041.

Ryan, C. G., Clayton, E., Griffin, W. L., Sie, S. H., and Cousens, D. R. (1988). SNIP, a statistics-sensitive background treatment for the quantitative analysis of PIXE spectra in geoscience applications. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, **34**(3), 396-402.

Sauve, A. C., and Speed, T. P. (2004). Normalization, baseline correction and alignment of high-throughput mass spectrometry data. *Proceedings Gensips*.

Siegel, R., Ma, J., Zou, Z., and Jemal, A. (2014). Cancer statistics, 2014. *CA: a cancer journal for clinicians*, **64**(1), 9-29.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267-288.

Tibshirani, R., Hastie, T., Narasimhan, B., Soltys, S., Shi, G., Koong, A., and Le, Q. T. (2004). Sample classification from protein mass spectrometry, by 'peak probability contrasts'. *Bioinformatics*, **20**(17), 3034-3044.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics*, **17**(6), 520-525.

Wagner, M., Naik, D., and Pothen, A. (2003). Protocols for disease classification from mass spectrometry data. *Proteomics*, **3**(9), 1692-1698.

Wu, B., Abbott, T., Fishman, D., McMurray, W., Mor, G., Stone, K., Ward, D., Williams, K., and Zhao, H. (2003). Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics*, **19**(13), 1636-1643.

Yasui, Y., Pepe, M., Thompson, M. L., Adam, B. L., Wright, G. L., Qu, Y., Potter, J.D., Winget, M., Thornquist M., and Feng, Z. (2003). A data-analytic strategy for protein biomarker discovery: profiling of high-dimensional proteomic data for cancer detection. *Biostatistics*, **4**(3), 449-463.

국문초록

최근 암 조기 진단을 위한 바이오마커를 찾기 위하여 질량 분석법(Mass spectrometry)이 활용되고 있다. 본 연구에서는 췌장암의 조기 진단을 위한 바이오마커를 찾기 위하여 환자 78명의 데이터를 분석하였다. 분석을 통해 구한 바이오마커를 활용하여 췌장암 환자의 예후 및 생존률 향상에 기여할 수 있다.

분석은 전처리, 분류, 변수선택 등 총 3단계로 진행된다. 전처리를 통해 얻어진 데이터는 결측치가 존재하므로 이 문제를 해결하기 위하여 Peak Probability Contrast (PPC), 비모수적 방법을 활용한 결측치 대체, 정규 분포로 부터 랜덤 잡음 생성을 활용한 결측치 대체 등 세 가지 방법을 제시하였다. 의사결정나무, 배깅, 랜덤 포레스트, Lasso 방법을 적용하여 당뇨병이 있는 췌장암 환자와 순수 당뇨병 환자를 분류하는 모형을 적합하였다.

결과적으로 PPC 방법을 활용한 배깅 및 랜덤 포레스트 모형에서 가장 낮은 예측 오차를 구할 수 있었다. 변수 중요도를 기준으로 바이오마커를 찾은 결과, m/z 1,206, 1,465가 고위험군인 당뇨병 환자그룹에서 췌장암을 구별해주는 중요한 바이오마커로 도출되었다.

주요어 : 바이오마커, 분류, 결측치, 췌장암, 질량분석법
학　번 : 2013-20211

Appendix A

R code

```
############
## library
############

library(MALDIquant)
library(MALDIquantForeign)
library(plyr)
library(gdata)
library(locfit)
library(randomForest)
library(gbm)
library(glmnet)
library(tree)
library(class)
library(MASS)
library(wavethresh)
```

```
library(stats)
library(ROCR)
library(parallel)

############
## setting working directory
############
dir.path <- "/home/rlarus/PC txt/PC txt"
setwd(dir.path)

############
## input data (sample list) ##
############

temp<-c(12181159, 17283733, 12966019, 14188550) ## remove 4 sample
because of sampling after surg
sample <- read.xls("Pancreatic cancer_coding_20131011.xlsx",  header=T
, sheet=1, stringsAsFactors=F)
sample_train <- sample[which(sample$채혈횟수==1),] ## before surgery sample

############
## input data (ASCII file) & transform format to maldiquant
############
## 수술전 샘플
num <- rep(0,4)
for(i in 1:length(temp))
{
```

```
  num[i] <- which(sample_train$환자번호== temp[i] )
}

sample_train <- sample_train[-num,]
seq<-c(1:96, 98:192)
match<-data.frame(seq, samp_num=sort(sample$sample번호))
sample_num<-vector()
for(i in 1:length(sample_train$sample번호))
{
  sample_num[i] <- match$seq[(match$samp_num == sample_train$sample
번호[i])]
}

############
## contingency table
############
str(sample_train)
contingency <- sample_train[,c(14,15)]
ftable(contingency)

############
## input data (ASCII files)

dir.list <- list.dirs()[-1]
sample_data <- list()
for(i in 1:length(sample_num))
{
```

```
  k <- sample_num[i]
  setwd(dir.list[k])
  file.list <- list.files()
  for(j in 1:6)
  {
    sample_data[6*(i-1)+j] <- importTxt(file.list[j], skip=2)
  }
  setwd(dir.path)
}

############
## data structure modify

p1.0 <- which(sample_train$종양종류==2) ## 순수 당뇨환자 index
p2.1 <- which(sample_train$종양종류==1 & sample_train$당뇨유무==1) ## 당
뇨있는 췌장암환자 index

idx1=NULL
for(i in 1:length(p2.1)){
  idx1=c(idx1,(6*(p2.1[i]-1)+1):(6*p2.1[i]))
}

for(i in 1:length(p1.0)){
  idx1=c(idx1,(6*(p1.0[i]-1)+1):(6*p1.0[i]))
}
sample_data1=sample_data[idx1]
```

```
###########
## collection of functions
###########

## preprocessing
preprocessing <- function(trainmass, testmass, plot=TRUE, plot_num=1)

{
 ###########
  ## transformation
 ###########
  for(i in 1:length(trainmass)) trainmass[[i]]@intensity
  <- trainmass[[i]]@intensity + 1
  for(i in 1:length(testmass)) testmass[[i]]@intensity
  <- testmass[[i]]@intensity + 1

  sample_tranf1 <- transformIntensity(trainmass, method="log")
  sample_tranf2 <- transformIntensity(testmass, method="log")

  ###########
  ## smoothing spectra
  ##########
  sample_smooth1 <- smoothIntensity(sample_tranf1, method="MovingAverage"
  , halfWindowSize=10)
  sample_smooth2 <- smoothIntensity(sample_tranf2, method="MovingAverage"
  , halfWindowSize=10)
```

```
############
## baseline estimate & subtraction
############
sample_base1 <- removeBaseline(sample_smooth1, method="SNIP"
, iterations=10)
sample_base2 <- removeBaseline(sample_smooth2, method="SNIP"
, iterations=10)
est.base <- estimateBaseline(sample_smooth1[[plot_num]], method="SNIP"
, iterations=10)

############
## spectra normalization
############
sample_normal1 <- calibrateIntensity(sample_base1, method="TIC")
sample_normal2 <- calibrateIntensity(sample_base2, method="TIC")

############
## peak detection
############
peaks1 <- detectPeaks(sample_normal1, method="MAD", SNR=2, halfWindowSize=10)
noise <- rep(0,length(trainmass))
for(i in 1:length(trainmass)) noise[i] <- unique(estimateNoise
(sample_normal1[[i]], method ="MAD", halfWindowSize=10)[,2])

############
## peak alignment
############
```

```
refPeaks1 <- referencePeaks(peaks1)
warping1 <- determineWarpingFunctions(peaks1, reference=refPeaks1)
peaks_align1 <- warpMassPeaks(peaks1, warping1)
peaks_align1 <- binPeaks(peaks_align1)

############
## merge replicate
############
nrep <- 6
nsample1 <- length(peaks_align1)/nrep
samples1 <- factor(rep(1:nsample1, each=nrep), levels=1:nsample1)

peaks_filter1 <- filterPeaks(peaks_align1, labels= samples1, minNumber=3)
peaks_merge1 <- mergeMassPeaks(peaks_filter1, labels=samples1, method="mean")

if(plot == TRUE)
{
  plot(trainmass[[plot_num]])
  plot(sample_tranf1[[plot_num]])
  plot(sample_smooth1[[plot_num]])
  plot(sample_smooth1[[plot_num]])
  lines(est.base, col=2, lwd=0.5)
  plot(sample_base1[[plot_num]])
  plot(sample_normal1[[plot_num]])
  plot(sample_normal1[[plot_num]])
  points(peaks1[[plot_num]], col=2, cex=0.4)
  abline(h=noise, col=4, lwd=3)
```

```
  }
  return(list(peaks_merge1, sample_normal2, noise))
}

## split training and test set
p <- function(p0, p1, seed)
{
  n <- length(p0) + length(p1)
  trn <- round(n*0.7)
  tst <- n - trn

  set.seed(seed)
  trn.0 <- sample(p0, round(trn*(length(p0)/n)))
  set.seed(seed)
  trn.1 <- sample(p1, round(trn*(length(p1)/n)))
  index1 <- c(sort(trn.1), sort(trn.0))
  index2 <- c(p1[!(p1 %in% sort(trn.1))], p0[!(p0 %in% sort(trn.0))])
  list(index1, index2)
}

## peak probability contrast
ppc <- function(intensity){
  G1 <- intensity[1:37,]
  G2 <- intensity[38:55,]
  n1 <- nrow(G1)
  n2 <- nrow(G2)
```

```
  alpha <- seq(0.01,1,0.01)

  for(k in 1:ncol(intensity))
  {
    temp <- data.frame(q1=rep(0, length(alpha)), q2=0)
    for(i in 1:length(alpha))
    {
      q <- quantile(intensity[,k], alpha[i])
      temp[i,1] <- sum(G1[,k] > q) / n1
      temp[i,2] <- sum(G2[,k] > q) / n2
    }
    alphahat <- alpha[which.max(apply(temp, 1, function(i) {abs(i[1]-i[2])}))]
    phat <- temp[which.max(apply(temp, 1, function(i) {abs(i[1]-i[2])})),]
    if(k == 1) phat_data <- c(phat, alphahat, qhat=quantile(intensity[,1]
    , alphahat))
    if(k > 1) phat_data <- rbind(phat_data, c(phat, alphahat, quantile
    (intensity[,k], alphahat)))
  }

  #row.names(phat_data) <- colnames(intensity)
  return(phat_data)
}

## test data
testdata <- function(test) {
  search <- function(ma) {
    ind <- (mass_test < (ma + tol)) & (mass_test > (ma - tol))
```

```
    if(sum(ind) > 0) return(max(intensity_test[ind]))
  }

  tol <- 0.02
  mass_test <- mass(test)
  intensity_test <- intensity(test)
  test_vec <- lapply(refmass, search)
  return(as.numeric(test_vec))
}


## kNNimpute
knn <- function(n) {
  dist_n <- vector()
  ind_n <- vector()
  ind_na <- is.na(intensity_mat[,n])
  for(m in 1:ncol(intensity_mat)) dist_n[m] <- sum((na.omit(intensity_mat[,n]
  - intensity_mat[,m]))^2)
  s <- sort(dist_n)[2:(K+1)]
  inv_s <- 1/s
  for(i in 1:K) ind_n[i] <- which(dist_n == s[i])

  intensity_imp <- intensity_mat[ind_na,ind_n]
  imp <- vector()
  for(j in 1:nrow(intensity_imp)) {
    na.imp <- !is.na(intensity_imp[j,])
    intensity_imp_1 <- intensity_imp[j,]
```

```
    imp[j] <- sum(intensity_imp_1[na.imp] * inv_s[na.imp]) / sum(inv_s[na.imp])
  }
  imp_final <- intensity_mat[,n]
  imp_final[ind_na] <- imp
  return(as.numeric(imp_final))
}

############
## preprocessing
############

train2_to <- list() ## ppc
train4_to <- list() ## knnImpute
train5_to <- list() ## rfImpute
train6_to <- list() ## loess
train7_to <- list() ## wavelet
test1_to1 <- list() ## ppc
test2_to1 <- list() ## imputation
percent_missing <- rep(0,100)
seed_u <- sample(1:100000, 100)

for(u in 1:100){
  ############
  ## dataset에 맞추어 수정(dataset은 cbind(p2.1, p1.0)순서로 저장)
  ############

  ## generate index
```

```
p.index <- p(p1.0, p2.1, seed_u[u])
cls.p <- factor(c(p2.1, p1.0) %in% p2.1)
levels(cls.p) <- c(0, 1)
sample_data1_tr <- sample_data1[rep(c(p2.1,p1.0) %in% p.index[[1]],each=6)]
sample_data1_ts <- sample_data1[rep(c(p2.1,p1.0) %in% p.index[[2]],each=6)]
############

## 수술 전
result <- preprocessing(sample_data1_tr, sample_data1_ts, plot=F)
############
## output
############
intensity_mat <- data.frame(intensityMatrix(result[[1]]))
## inter_sample
idx2.1<-which(apply(apply(intensity_mat, 1, is.na)[,1:37],1,sum) <= 37/2)
idx1.0<-which(apply(apply(intensity_mat, 1, is.na)[,-c(1:37)],1,sum) <= 18/2)
intensity_mat=intensity_mat[,sort(intersect(idx2.1,idx1.0))]
mass <- as.numeric(substr(colnames(intensity_mat), 2,15))
percent_missing[u] <- sum(apply(apply(intensity_mat, 2, is.na), 2, sum))
/ (dim(intensity_mat)[1]*dim(intensity_mat)[2])

############
## test sample
############

refmass <- mass
```

```
test2 <- mcmapply(testdata, result[[2]], mc.cores = getOption("mc.cores", 4)
, mc.preschedule = TRUE)
test2 <- as.data.frame(t(test2))
colnames(test2) <- colnames(intensity_mat)
sample_ind <- factor(rep(1:23, each=6))
test2 <- apply(test2, 2, FUN=function(x) tapply(x, sample_ind, mean))
test2 <- as.data.frame(test2)

############
## ppc
############

intensity_mat0 <- intensity_mat
intensity_mat0[is.na(intensity_mat0)] <- 0
intensity_ppc <- ppc(intensity_mat0)

row.names(intensity_ppc) <- colnames(intensity_mat0)
intensity_ppc <- as.data.frame(intensity_ppc)
intensity_ppc$abs_diff <- apply(intensity_ppc[,c(1,2)], 1, function(i)
{abs(i[[2]]-i[[1]])})
intensity_ppc <- cbind(mz=row.names(intensity_ppc), intensity_ppc)
intensity_ppc_desc <- arrange(intensity_ppc, desc(abs_diff))
for(i in 1:ncol(intensity_mat0))

{

  if(i==1)
```

```
  intensity_ppc_1 <- as.numeric(intensity_mat0[,i] >
  intensity_ppc[i,5])

  if(i>1) intensity_ppc_1 <- cbind(intensity_ppc_1,
  as.numeric(intensity_mat0[,i] > intensity_ppc[i,5]))

}

colnames(intensity_ppc_1) <- colnames(intensity_mat0)

train2 <- as.data.frame(intensity_ppc_1)

## test data : test1
qhat <- unlist(intensity_ppc[,5])
for(i in 1:(ncol(test2)))
{
  if(i==1) test1 <- as.numeric(test2[,i] > qhat[i])
  if(i>1) test1 <- cbind(test1, as.numeric(test2[,i] > qhat[i]))
}
test1 <- as.data.frame(test1)

colnames(test1) <- colnames(intensity_mat)

############
## KNN
############
K <- 10
```

```
intensity_mat_knn <- mclapply(1:ncol(intensity_mat), knn,  mc.cores =
getOption("mc.cores", 4), mc.preschedule = TRUE)
intensity_mat_knn <- as.data.frame(intensity_mat_knn)
colnames(intensity_mat_knn) <- colnames(intensity_mat)
train4 <- intensity_mat_knn

###########
## random forest
###########

intensity_mat_rf <- intensity_mat
intensity_mat_rf$Y <- cls.p[c(p2.1,p1.0) %in% p.index[[1]]]
imp_rf <- rfImpute(Y ~., data=intensity_mat_rf, ntree=500,iter=6)
imp_rf[,(ncol(imp_rf)+1)] <- imp_rf[,1]
imp_rf <- imp_rf[,-1]
colnames(imp_rf)[ncol(imp_rf)] <- "Y"
train5 <- imp_rf

###########
## loess
###########

spectra_std <- apply(intensity_mat, 2, sd, na.rm=T)
## NA 처리 방법 1.Imputation (random noise)
## 1.1 spectra site별로 local regression(LOESS) 추정량
loess_fit <- loess(spectra_std~mass)
loess_std <- predict(loess_fit,mass)
```

```
#lines(mz,loess_std,col=2,lwd=2)

intensity_mat_loess <- intensity_mat
for(i in 1:dim(intensity_mat)[2]){
  intensity_mat_loess[is.na(intensity_mat[,i]),i] <- abs(rnorm(sum(is.na
  (intensity_mat[,i])), 0 ,loess_std[i]))
}
train6 <- intensity_mat_loess

############
## wavelet
############
# intensity_mat_wav
## 1.2 wavelet regression MAD (median absolute deviation) 추정량
wav_std <- matrix(0,1,dim(intensity_mat)[1])
for(i in 1:dim(intensity_mat)[1]){
  idx <- which(!is.na(intensity_mat[i,]))
  idx <- idx[1:256]
  #plot(mz[idx],intensity_mat[i,idx])
  wav_fit <- wd(as.numeric(intensity_mat[i,idx]))
  FineCoefs <- accessD(wav_fit, lev=wav_fit$nlevels-1)
  wav_std[1,i] <- sqrt(256)*mad(FineCoefs)/0.6745
}

intensity_mat_wav <- intensity_mat
for(i in 1:dim(intensity_mat)[1]){
  intensity_mat_wav[i,is.na(intensity_mat[i,])] <- abs(rnorm(sum(is.na
```

```
    (intensity_mat[i,])), 0 ,wav_std[1,i]))
  }
  train7 <- intensity_mat_wav

  ## assign Y
  train2$Y <- cls.p[c(p2.1,p1.0) %in% p.index[[1]]]
  train4$Y <- cls.p[c(p2.1,p1.0) %in% p.index[[1]]]
  train6$Y <- cls.p[c(p2.1,p1.0) %in% p.index[[1]]]
  train7$Y <- cls.p[c(p2.1,p1.0) %in% p.index[[1]]]
  test0$Y <- cls.p[c(p2.1,p1.0) %in% p.index[[2]]]
  test1$Y <- cls.p[c(p2.1,p1.0) %in% p.index[[2]]]
  test2$Y <- cls.p[c(p2.1,p1.0) %in% p.index[[2]]]

  train2_to[[u]] <- train2 ## ppc
  train4_to[[u]] <- train4 ## knnImpute
  train5_to[[u]] <- train5 ## rfImpute
  train6_to[[u]] <- train6 ## loess noise imputation
  train7_to[[u]] <- train7 ## wavelet noise imputation
  test1_to1[[u]] <- test1 ## ppc
  test2_to1[[u]] <- test2 ## imputation

  if(u %in% seq(20,100,20)) save.image("~/MALDI_realfinal.RData")
  print(u)
}

############
### classification
```

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

train2_to ## ppc
train4_to ## knnImpute
train5_to ## rfImpute
train6_to ## loess
train7_to ## rf
test1_to1 ## ppc
test2_to1 ## imputation

#######

maxi<-100


## 2를 모두 4,5,6,7 로 바꿔서 실행!!!
## 2를 제외한 4,5,6,7 에서는 test2_to1을 test set으로 사용!

tree.pred.prob2 <- list()
bag.pred2 <- list()
rf.pred2 <- list()
pred.lasso.prob2 <- list()
test_y2 <- list()
mis_tree2 <- numeric(maxi)
mis_bag2 <- numeric(maxi)
mis_rf2 <-  numeric(maxi)
mis_lasso2 <- numeric(maxi)
```

```
auc_tree2 <- numeric(maxi)
auc_bag2 <- numeric(maxi)
auc_rf2 <- numeric(maxi)
auc_lasso2 <- numeric(maxi)
spec_tree2 <- numeric(maxi); sens_tree2 <- numeric(maxi)
spec_bag2 <- numeric(maxi); sens_bag2 <- numeric(maxi)
spec_rf2 <- numeric(maxi); sens_rf2 <- numeric(maxi)
spec_lasso2 <- numeric(maxi); sens_lasso2 <- numeric(maxi)

for(i in 1:maxi)
{
  n <- ncol(train2_to[[i]])

  ## pruning tree
  tree1 <- tree(Y~. , data=train2_to[[i]])

  tree.pred <- predict(tree1, newdata=test1_to1[[i]], type='class')
  tree.table <- table(tree.pred, test1_to1[[i]]$Y)
  mis.tree <- (tree.table[1,2]+tree.table[2,1])/sum(tree.table)

  tree.pred.prob2[[i]] <- predict(tree1, newdata=test1_to1[[i]])
  auc.tree <- performance(prediction(tree.pred.prob2[[i]][,2],
  test1_to1[[i]]$Y), "auc")
  spec_tree2[[i]] <- tree.table[1,1]/(tree.table[1,1]+tree.table[2,1])
  sens_tree2[[i]] <- tree.table[2,2]/(tree.table[1,2]+tree.table[2,2])

  ## bagging, random forest
```

```
bag <- randomForest(Y~., data=train2_to[[i]], mty=n, importance=T,
ntree=1000)
rf <- randomForest(Y~., data=train2_to[[i]], mty=round(sqrt(n)),
importance=T, ntree=1000)

bag.pred2[[i]] <- predict(bag, newdata=test1_to1[[i]], type="prob")
rf.pred2[[i]] <- predict(rf, newdata=test1_to1[[i]], type="prob")
auc.bag <- performance(prediction(bag.pred2[[i]][,2], test1_to1[[i]]$Y)
 , "auc")
auc.rf <- performance(prediction(rf.pred2[[i]][,2], test1_to1[[i]]$Y)
 , "auc")

bag.table <- table(predict(bag, newdata=test1_to1[[i]]), test1_to1[[i]]$Y)
rf.table <- table(predict(rf, newdata=test1_to1[[i]]), test1_to1[[i]]$Y)
mis.bag <- (bag.table[1,2]+bag.table[2,1])/sum(bag.table)
mis.rf <- (rf.table[1,2]+rf.table[2,1])/sum(rf.table)

spec_bag2[[i]] <- bag.table[1,1]/(bag.table[1,1]+bag.table[2,1])
sens_bag2[[i]] <- bag.table[2,2]/(bag.table[1,2]+bag.table[2,2])
spec_rf2[[i]] <- rf.table[1,1]/(rf.table[1,1]+rf.table[2,1])
sens_rf2[[i]] <- rf.table[2,2]/(rf.table[1,2]+rf.table[2,2])

## lasso
grid=10^seq(10,-2,length=100)
lasso <- glmnet(as.matrix(train2_to[[i]][,-n]) ,lambda=grid, train2_to[[i]]$Y
, alpha=1, family="binomial")
cv.out <- cv.glmnet(as.matrix(train2_to[[i]][,-n]) , train2_to[[i]]$Y,
```

```
 alpha=1, family="binomial", nfolds=10)
lamb <- cv.out$lambda.min

lasso.cv <- glmnet(as.matrix(train2_to[[i]][,-n]) , train2_to[[i]]$Y,
 lambda=lamb, alpha=1, family="binomial")
pred.lasso.prob2[[i]] <- predict(lasso.cv, s=lamb, newx=as.matrix
(test1_to1[[i]][,-n]), type="response")
auc.lasso <- performance(prediction(pred.lasso.prob2[[i]],
test1_to1[[i]]$Y), "auc")
pred.lasso <- predict(lasso.cv, s=lamb, newx=as.matrix(test1_to1[[i]][,-n])
, type="class")
lasso.table <- table(pred.lasso, test1_to1[[i]]$Y)
if(nrow(lasso.table)==2) mis.lasso <- (lasso.table[1,2]+lasso.table[2,1])
/sum(lasso.table)
if(nrow(lasso.table)==1) mis.lasso <- ifelse(row.names(lasso.table)=="0",
 lasso.table[1,2]/sum(lasso.table), lasso.table[2,1]/sum(lasso.table))

if(nrow(lasso.table)==2) {
  spec_lasso2[[i]] <- lasso.table[1,1]/(lasso.table[1,1]+lasso.table[2,1])
  sens_lasso2[[i]] <- lasso.table[2,2]/(lasso.table[1,2]+lasso.table[2,2])
}

if(nrow(lasso.table)==1) {
  spec_lasso2[[i]] <- ifelse(row.names(lasso.table)=="0", 1, 0)
  sens_lasso2[[i]] <- ifelse(row.names(lasso.table)=="0", 0, 1)
}
```

```
  mis_tree2[i] <- mis.tree
  mis_bag2[i] <- mis.bag
  mis_rf2[i] <- mis.rf
  mis_lasso2[i] <- mis.lasso

  test_y2[[i]] <- test1_to1[[i]]$Y
  auc_tree2[i] <- as.numeric(auc.tree@y.values)
  auc_bag2[i] <- as.numeric(auc.bag@y.values)
  auc_rf2[i] <- as.numeric(auc.rf@y.values)
  auc_lasso2[i] <- as.numeric(auc.lasso@y.values)
  print(i)

}

varImpPlot(bag, n.var=10, main="Variance Importance by using Bagging")
varImpPlot(rf, n.var=10, main="Variance Importance by using Random Forest")

mean(mis_tree1);mean(mis_bag1);mean(mis_rf1);mean(mis_lasso1)
sd(mis_tree1);sd(mis_bag1);sd(mis_rf1);sd(mis_lasso1);

mean(mis_tree2);mean(mis_bag2);mean(mis_rf2);mean(mis_lasso2)
sd(mis_tree2);sd(mis_bag2);sd(mis_rf2);sd(mis_lasso2)

mean(mis_tree4);mean(mis_bag4);mean(mis_rf4);mean(mis_lasso4)
sd(mis_tree4);sd(mis_bag4);sd(mis_rf4);sd(mis_lasso4)

mean(mis_tree5);mean(mis_bag5);mean(mis_rf5);;mean(mis_lasso5)
```

```
sd(mis_tree5);sd(mis_bag5);sd(mis_rf5);sd(mis_lasso5)

mean(mis_tree6);mean(mis_bag6);mean(mis_rf6);mean(mis_lasso6)
sd(mis_tree6);sd(mis_bag6);sd(mis_rf6);sd(mis_lasso6)

mean(mis_tree7);mean(mis_bag7);mean(mis_rf7);mean(mis_lasso7)
sd(mis_tree7);sd(mis_bag7);sd(mis_rf7);sd(mis_lasso7)

par(mfrow=c(1,1), mai=c(0.4,0.5, 0.7,0.26))

auc2 <- cbind(auc_tree2, auc_bag2, auc_rf2, auc_lasso2)
colnames(auc2) <- c("Tree","Bagging","Random Forest","lasso")
boxplot(auc2, main="Peak Probability Contrast", ylab="AUC", cex.main=1.5,
pars=list(boxwex=0.7, staplewex = 0.5, outwex = 0.5))
colMeans(auc2)
apply(auc2, 2, sd)

auc4 <- cbind(auc_tree4, auc_bag4, auc_rf4, auc_lasso4)
colnames(auc4) <- c("Tree","Bagging","Random Forest","lasso")
boxplot(auc4, main="K nearest neighbor imputation", ylab="AUC", cex.main=1.5,
pars=list(boxwex=0.7, staplewex = 0.5, outwex = 0.5))
colMeans(auc4)
apply(auc4, 2, sd)

auc5 <- cbind(auc_tree5, auc_bag5, auc_rf5, auc_lasso5)
colnames(auc5) <- c("Tree","Bagging","Random Forest","lasso")
boxplot(auc5, main="Random Forest imputation", ylab="AUC", cex.main=1.5,
```

```
pars=list(boxwex=0.7, staplewex = 0.5, outwex = 0.5))
colMeans(auc5)
apply(auc5, 2, sd)

auc6 <- cbind(auc_tree6, auc_bag6, auc_rf6, auc_lasso6)
colnames(auc6) <- c("Tree","Bagging","Random Forest","lasso")
boxplot(auc6, main="Loess Random noise Imputation", ylab="AUC", cex.main=1.5,
pars=list(boxwex=0.7, staplewex = 0.5, outwex = 0.5))
colMeans(auc6)
apply(auc6, 2, sd)

auc7 <- cbind(auc_tree7, auc_bag7, auc_rf7, auc_lasso7)
colnames(auc7) <- c("Tree","Bagging","Random Forest","lasso")
boxplot(auc7, main="Wavelet Random noise Imputation", ylab="AUC", cex.main=1.5,
pars=list(boxwex=0.7, staplewex = 0.5, outwex = 0.5))
colMeans(auc7)
apply(auc7, 2, sd)

## roc curve
roc <- function(test, tree.pr, bag.pr, rf.pr, las.pr)
{
  test <- lapply(test, as.factor)
  grid <- seq(1.01, 0, by=-0.01)

  specsens <- function(prd, grd)
  {
    prd.z <- lapply(prd, function(x) {
```

```
  if(ncol(x)==2) { q<-as.factor(ifelse(x[,2] >= grd, 1, 0)) }
  if(ncol(x)==1) { q<-as.factor(ifelse(x >= grd, 1, 0)) }
  return(q) })
  spec <- numeric(29)
  sens <- numeric(29)

  for(j in 1:29)
  {
    tb <- table(test[[j]], prd.z[[j]])
    if(ncol(tb)==2)
    {
      spec[j] <- tb[1,1] / (tb[1,1] + tb[1,2])
      sens[j] <- tb[2,2] / (tb[2,1] + tb[2,2])
    }

    if(ncol(tb)==1)
    {
      spec[j] <- ifelse(colnames(tb)=="0", 1, 0)
      sens[j] <- ifelse(colnames(tb)=="1", 1, 0)
    }
  }
  return(c(mean(spec), mean(sens)))
}

mean_specsens1 <- as.data.frame(t(sapply(grid, function(g)
specsens(tree.pr, g))))
colnames(mean_specsens1) <- c("spec", "sens")
```

```
  mean_specsens1$spec_1 <- 1- mean_specsens1$spec
  mean_specsens2 <- as.data.frame(t(sapply(grid, function(g)
  specsens(bag.pr, g))))
  colnames(mean_specsens2) <- c("spec", "sens")

  mean_specsens2$spec_1 <- 1- mean_specsens2$spec
  mean_specsens3 <- as.data.frame(t(sapply(grid, function(g)
  specsens(rf.pr, g))))
  colnames(mean_specsens3) <- c("spec", "sens")
  mean_specsens3$spec_1 <- 1- mean_specsens3$spec
  mean_specsens4 <- as.data.frame(t(sapply(grid, function(g)
  specsens(las.pr, g))))
  colnames(mean_specsens4) <- c("spec", "sens")
  mean_specsens4$spec_1 <- 1- mean_specsens4$spec

  plot(sens~spec_1, data =mean_specsens1, type='l', lwd=2.5, ylab="", xlab="")
  lines(sens~spec_1, data =mean_specsens2, col=2, lty=2, lwd=2.5)
  lines(sens~spec_1, data =mean_specsens3, col=3, lty=3, lwd=2.5)
  lines(sens~spec_1, data =mean_specsens4, col=4, lty=4, lwd=2.5)
  legend(0.7, 0.6, c("Tree","Bagging","Random Forest", "Lasso"),
  col=c(1,2,3,4), lwd=c(2,2,2,2), lty=c(1,2,3,4), cex=1)
}

roc(test_y2, tree.pred.prob2, bag.pred2, rf.pred2, pred.lasso.prob2)
title(main="Peak Probability Contrast", cex.main=1.7, xlab="1-specificity"
, ylab="sensitivity")
```

```
roc(test_y4, tree.pred.prob4, bag.pred4, rf.pred4, pred.lasso.prob4)
title(main="K nearest neighbor imputation", cex.main=1.7, xlab="1-specificity"
, ylab="sensitivity")

roc(test_y5, tree.pred.prob5, bag.pred5, rf.pred5, pred.lasso.prob5)
title(main="Random Forest imputation", cex.main=1.7, xlab="1-specificity"
, ylab="sensitivity")

roc(test_y6, tree.pred.prob6, bag.pred6, rf.pred6, pred.lasso.prob6)
title(main="Loess Random noise Imputation", cex.main=1.7, xlab="1-specificity"
, ylab="sensitivity")

roc(test_y7, tree.pred.prob7, bag.pred7, rf.pred7, pred.lasso.prob7)
title(main="Wavelet Random noise Imputation", cex.main=1.7, xlab="1-specificity"
, ylab="sensitivity")

############
## plotting
############

group_intensity_plot <- function(from, to, mz=vector(), ylim=3000)

{
  cancer <- seq(1,53*6)
  contr <- seq(53*6+1 , 468)

  cancer_data <- sample_data1[cancer]
```

```
contr_data <- sample_data1[contr]

for(i in 1:length(cancer_data))
{
  if(i==1)  plot(mass(cancer_data[[i]])[((mass(cancer_data[[i]])) > from &
 (mass(cancer_data[[i]]) < to))], intensity(cancer_data[[i]])
 [((mass(cancer_data[[i]])) > from & (mass(cancer_data[[i]]) < to))]
 , type='l', col=1, ylim=c(0,ylim), xlab="m/z", ylab="intensity"
 , main="pancreatic patients with diabetes", cex.main=2)

  lines(mass(cancer_data[[i]])[((mass(cancer_data[[i]])) > from & (mass
 (cancer_data[[i]]) < to))], intensity(cancer_data[[i]])[((mass
 (cancer_data[[i]]))> from & (mass(cancer_data[[i]]) < to))], col=1)
}

for(i in 1:length(contr_data))
{
  if(i==1)
 plot(mass(contr_data[[i]])[((mass(contr_data[[i]])) > from & (mass
 (contr_data[[i]]) < to))], intensity(contr_data[[i]])[((mass
 (contr_data[[i]])) > from & (mass(contr_data[[i]]) < to))],
 type='l', col=1, ylim=c(0,ylim), xlab="m/z", ylab="intensity"
 ,main="diabetes patients", cex.main=2)

  lines(mass(contr_data[[i]])[((mass(contr_
  data[[i]])) > from & (mass(contr_data[[i]]) < to))],
  intensity(contr_data[[i]])[((mass(contr_data[[i]]))
```

```
    > from & (mass(contr_data[[i]]) < to))], col=1)
  }
}

############
## 그룹별 plot
par(mfrow=c(2,1), mar=c(1.5,2,1.5,1), oma=c(1,0,0,0))
group_intensity_plot(from = 1465, to = 1469, ylim=4000)
group_intensity_plot(1206.3, 1208.8, ylim=2000)
group_intensity_plot(1020, 1021, ylim=1000)
```