# Metabolomic Data Analysis with MetaboAnalyst 6.0

Name: guest5394319678214572061

May 8, 2024

# 1 Data Processing and Normalization

## 1.1 Reading and Processing the Raw Data

MetaboAnalyst accepts a variety of data types generated in metabolomic studies, including compound concentration data, binned NMR/MS spectra data, NMR/MS peak list data, as well as MS spectra (NetCDF, mzXML, mzDATA). Users need to specify the data types when uploading their data in order for MetaboAnalyst to select the correct algorithm to process them. Table 1 summarizes the result of the data processing steps.

### 1.1.1 Reading Binned Spectral Data

The binned spectra data should be uploaded in comma seperated values (.csv) format. Samples can be in rows or columns, with class labels immediately following the sample IDs.

Samples are in rows and features in columns The uploaded file is in comma separated values (.csv) format. The uploaded data file contains 50 (samples) by 200 (spectra bins) data matrix.

### 1.1.2 Data Integrity Check

Before data analysis, a data integrity check is performed to make sure that all the necessary information has been collected. The class labels must be present and contain only two classes. If samples are paired, the class label must be from -n/2 to -1 for one group, and 1 to n/2 for the other group (n is the sample number and must be an even number). Class labels with same absolute value are assumed to be pairs. Compound concentration or peak intensity values should all be non-negative numbers. By default, all missing values, zeros and negative values will be replaced by the half of the minimum positive value found within the data (see next section)

### 1.1.3 Missing value imputations

Too many zeroes or missing values will cause difficulties for downstream analysis. MetaboAnalyst offers several different methods for this purpose. The default method replaces all the missing and zero values with a small values (the half of the minimum positive values in the original data) assuming to be the detection limit. The assumption of this approach is that most missing values are caused by low abundance metabolites (i.e.below the detection limit). In addition, since zero values may cause problem for data normalization (i.e. log), they are also replaced with this small value. User can also specify other methods, such as replace by mean/median, or use K-Nearest Neighbours (KNN), Probabilistic PCA (PPCA), Bayesian PCA (BPCA) method, Singular Value Decomposition (SVD) method to impute the missing values [1]. Please choose the one that is the most appropriate for your data.

[1] Stacklies W, Redestig H, Scholz M, Walther D, Selbig J. *pcaMethods: a bioconductor package, providing PCA methods for incomplete data.*, Bioinformatics 2007 23(9):1164-1167

Zero or missing values were replaced by 1/5 of the min positive value for each variable.

### 1.1.4 Data Filtering

The purpose of the data filtering is to identify and remove variables that are unlikely to be of use when modeling the data. No phenotype information are used in the filtering process, so the result can be used with any downstream analysis. This step can usually improves the results. Data filter is strongly recommended for datasets with large number of variables ($>$ 250) datasets contain much noise (i.e.chemometrics data). Filtering can usually improve your results[2].

*For data with number of variables $<$ 250, this step will reduce 5% of variables; For variable number between 250 and 500, 10% of variables will be removed; For variable number bwteen 500 and 1000, 25% of variables will be removed; And 40% of variabled will be removed for data with over 1000 variables. The None option is only for less than 5000 features. Over that, if you choose None, the IQR filter will still be applied. In addition, the maximum allowed number of variables is* ***10000***

No data filtering was performed.

Table 1: Summary of data processing results

| | Features (positive) | Missing/Zero | Features (processed) |
|---|---|---|---|
| C002 | 194 | 6 | 200 |
| C004 | 189 | 11 | 200 |
| C005 | 191 | 9 | 200 |
| C006 | 195 | 5 | 200 |
| C007 | 200 | 0 | 200 |
| C009 | 186 | 14 | 200 |
| C010 | 196 | 4 | 200 |
| C011 | 177 | 23 | 200 |
| C012 | 189 | 11 | 200 |
| C015 | 188 | 12 | 200 |
| C016 | 188 | 12 | 200 |
| C017 | 198 | 2 | 200 |
| C019 | 181 | 19 | 200 |
| C020 | 184 | 16 | 200 |
| C021 | 187 | 13 | 200 |
| C022 | 191 | 9 | 200 |
| C024 | 190 | 10 | 200 |
| C026 | 195 | 5 | 200 |
| C028 | 196 | 4 | 200 |
| C029 | 192 | 8 | 200 |
| C030 | 182 | 18 | 200 |
| C031 | 179 | 21 | 200 |
| C032 | 191 | 9 | 200 |
| C033 | 189 | 11 | 200 |
| C034 | 199 | 1 | 200 |
| P002 | 195 | 5 | 200 |
| P012 | 187 | 13 | 200 |
| P014 | 200 | 0 | 200 |
| P027 | 200 | 0 | 200 |
| P034 | 198 | 2 | 200 |
| P037 | 187 | 13 | 200 |
| P038 | 195 | 5 | 200 |
| P041 | 178 | 22 | 200 |
| P042 | 198 | 2 | 200 |
| P049 | 189 | 11 | 200 |
| P056 | 190 | 10 | 200 |
| P058 | 179 | 21 | 200 |
| P060 | 190 | 10 | 200 |
| P064 | 200 | 0 | 200 |
| P065 | 198 | 2 | 200 |
| P070 | 190 | 10 | 200 |
| P080 | 196 | 4 | 200 |
| P085 | 200 | 0 | 200 |
| P086 | 193 | 7 | 200 |
| P089 | 199 | 1 | 200 |
| P092 | 191 | 9 | 200 |
| P099 | 190 | 10 | 200 |
| P113 | 152 | 48 | 200 |
| P013b | 191 | 9 | 200 |
| P100b | 199 | 1 | 200 |

[2]Hackstadt AJ, Hess AM.*Filtering for increased power for microarray data analysis*, BMC Bioinformatics. 2009; 10: 11.

## 1.2 Data Normalization

The data is stored as a table with one sample per row and one variable (bin/peak/metabolite) per column. The normalization procedures implemented below are grouped into four categories. Sample specific normalization allows users to manually adjust concentrations based on biological inputs (i.e. volume, mass); row-wise normalization allows general-purpose adjustment for differences among samples; data transformation and scaling are two different approaches to make features more comparable. You can use one or combine both to achieve better results.

The normalization consists of the following options:

1. Row-wise procedures:
    - Sample specific normalization (i.e. normalize by dry weight, volume)
    - Normalization by the sum
    - Normalization by the sample median
    - Normalization by a reference sample (probabilistic quotient normalization)[3]
    - Normalization by a pooled or average sample from a particular group
    - Normalization by a reference feature (i.e. creatinine, internal control)
    - Quantile normalization
2. Data transformation :
    - Log transformation (base 10)
    - Square root transformation
    - Cube root transformation
3. Data scaling:
    - Mean centering (mean-centered only)
    - Auto scaling (mean-centered and divided by standard deviation of each variable)
    - Pareto scaling (mean-centered and divided by the square root of standard deviation of each variable)
    - Range scaling (mean-centered and divided by the value range of each variable)

Figure 1 shows the effects before and after normalization.

Row-wise normalization: Normalization to sample median; Data transformation: Log10 Normalization; Data scaling: Autoscaling.

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

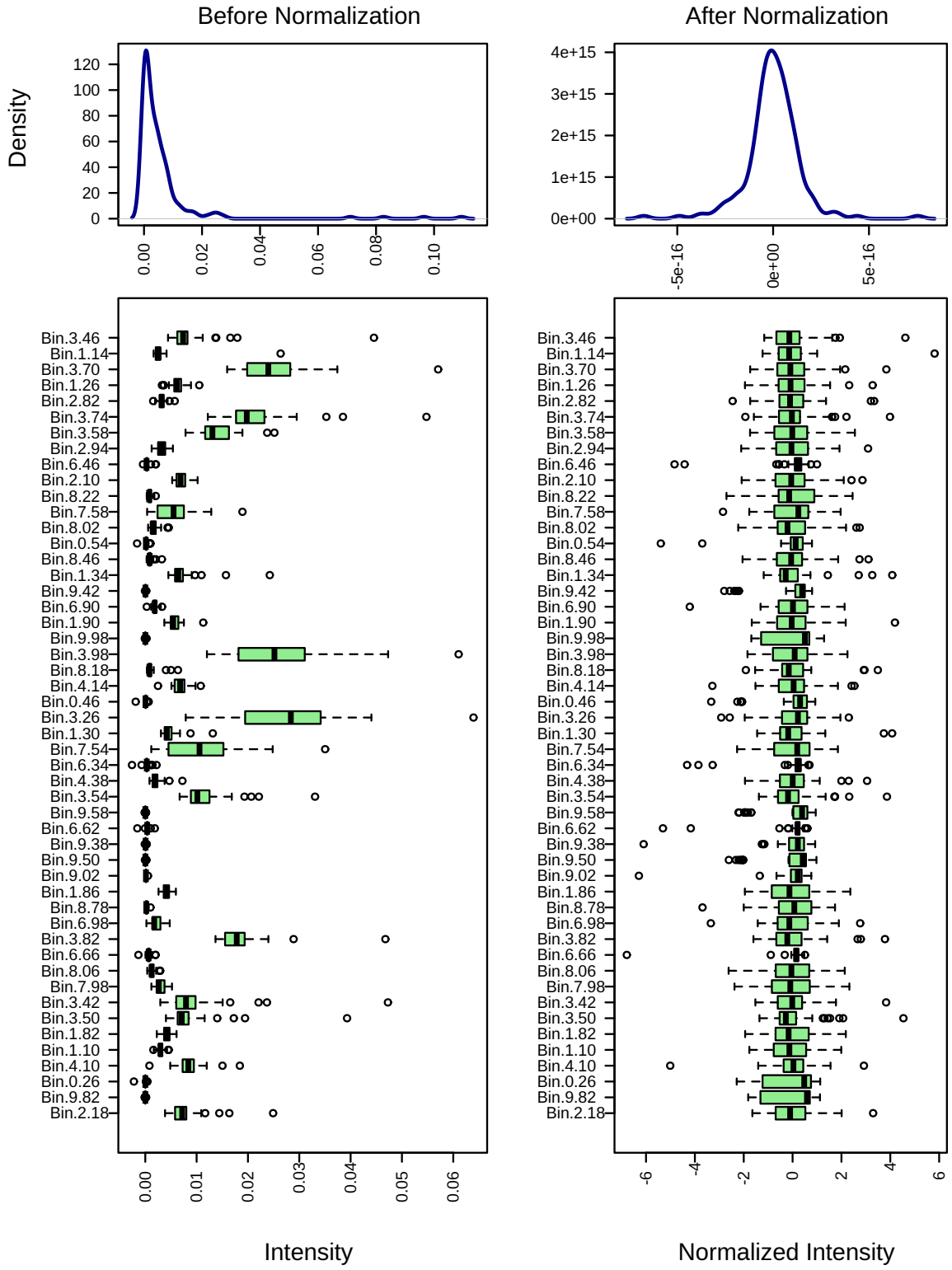


Figure 1: Box plots and kernel density plots before and after normalization. The boxplots show at most 50 features due to space limit. The density plots are based on all samples.

## 2 Statistical and Machine Learning Data Analysis

MetaboAnalyst offers a variety of methods commonly used in metabolomic data analyses. They include:

1. Univariate analysis methods:
    - Fold Change Analysis
    - T-tests
    - Volcano Plot
    - One-way ANOVA and post-hoc analysis
    - Correlation analysis
2. Multivariate analysis methods:
    - Principal Component Analysis (PCA)
    - Partial Least Squares - Discriminant Analysis (PLS-DA)
3. Robust Feature Selection Methods in microarray studies
    - Significance Analysis of Microarray (SAM)
    - Empirical Bayesian Analysis of Microarray (EBAM)
4. Clustering Analysis
    - Hierarchical Clustering
        - Dendrogram
        - Heatmap
    - Partitional Clustering
        - K-means Clustering
        - Self-Organizing Map (SOM)
5. Supervised Classification and Feature Selection methods
    - Random Forest
    - Support Vector Machine (SVM)

`Please note: some advanced methods are available only for two-group sample analyais.`

## 2.1 Principal Component Analysis (PCA)

PCA is an unsupervised method aiming to find the directions that best explain the variance in a data set (X) without referring to class labels (Y). The data are summarized into much fewer variables called *scores* which are weighted average of the original variables. The weighting profiles are called *loadings*. The PCA analysis is performed using the `prcomp` package. The calculation is based on singular value decomposition.

The Rscript `chemometrics.R` is required. Figure 2 is pairwise score plots providing an overview of the various seperation patterns among the most significant PCs; Figure 3 is the scree plot showing the variances explained by the selected PCs; Figure 4 shows the 2-D scores plot between selected PCs; Figure 5 shows the biplot between the selected PCs. Interactive 3-D scores plots are not included here and can be directly downloaded from website.

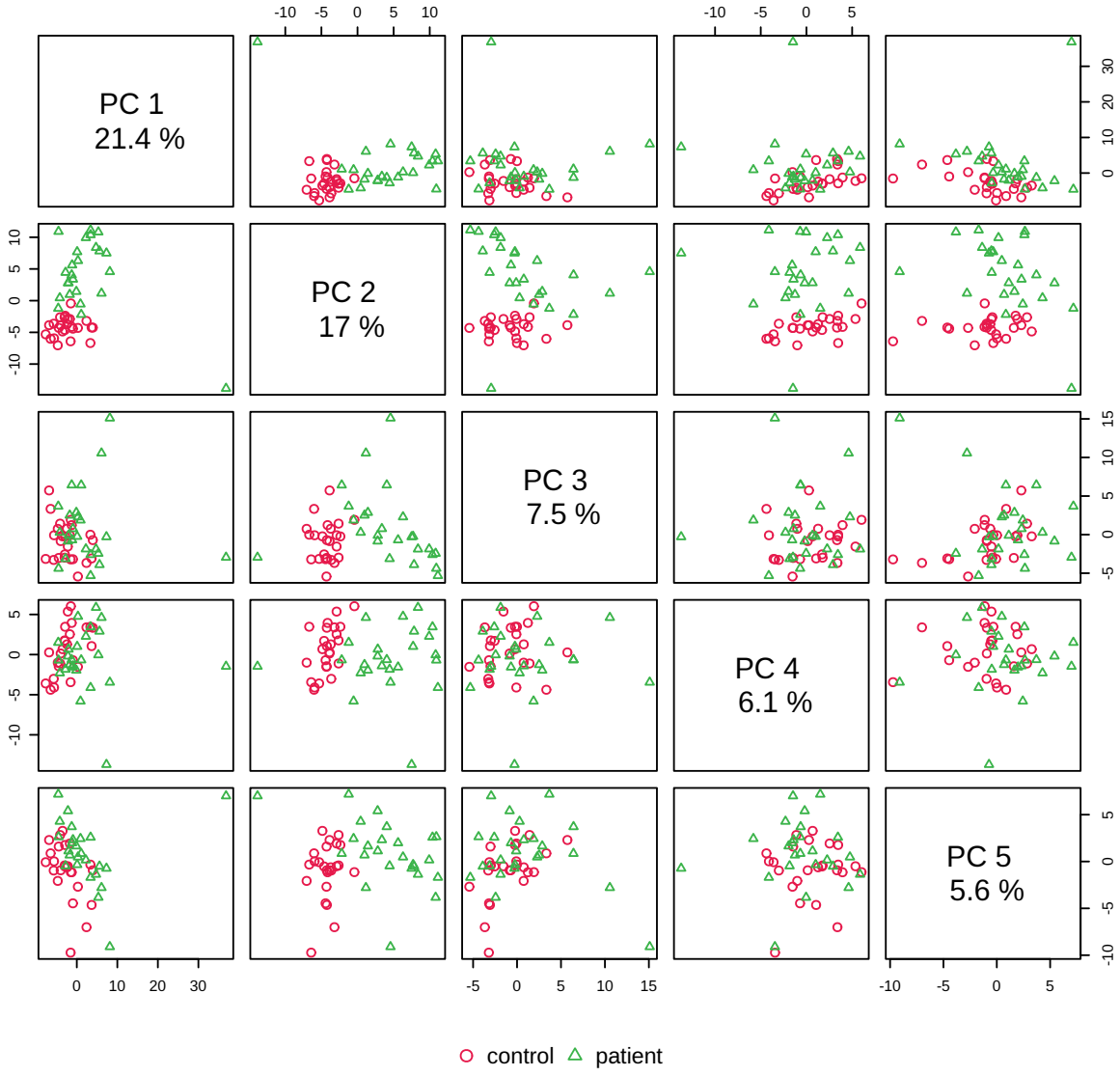


Figure 2: Pairwise score plots between the selected PCs. The explained variance of each PC is shown in the corresponding diagonal cell.

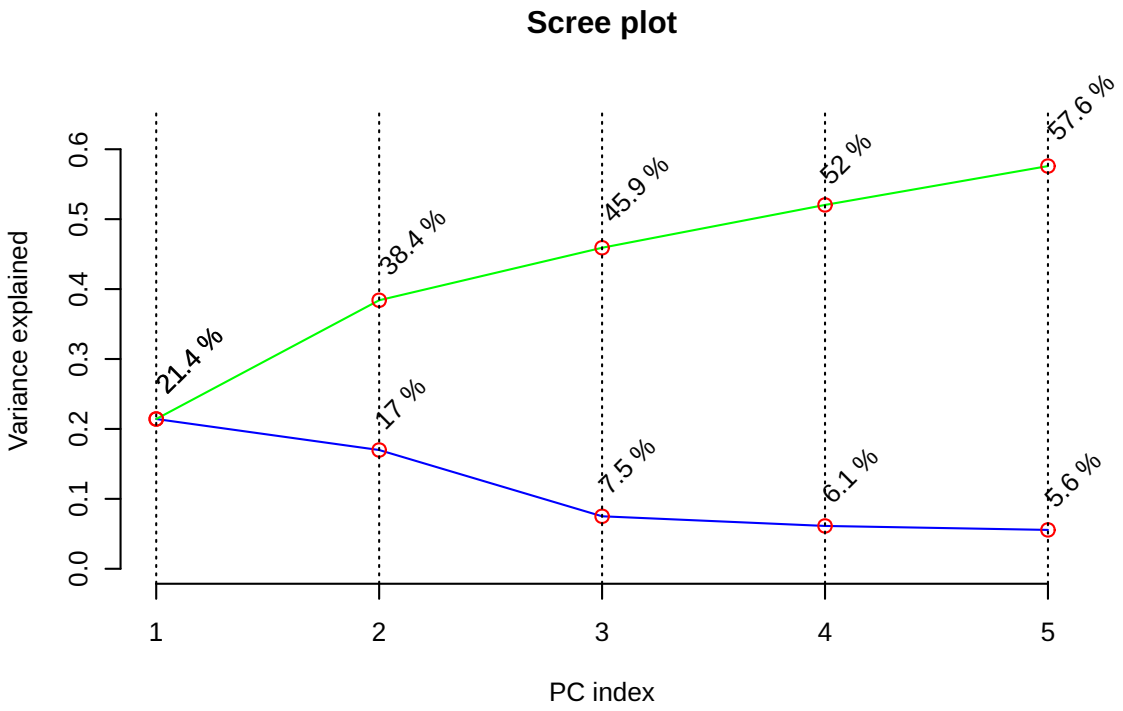


Figure 3: Scree plot shows the variance explained by PCs. The green line on top shows the accumulated variance explained; the blue line underneath shows the variance explained by individual PC.

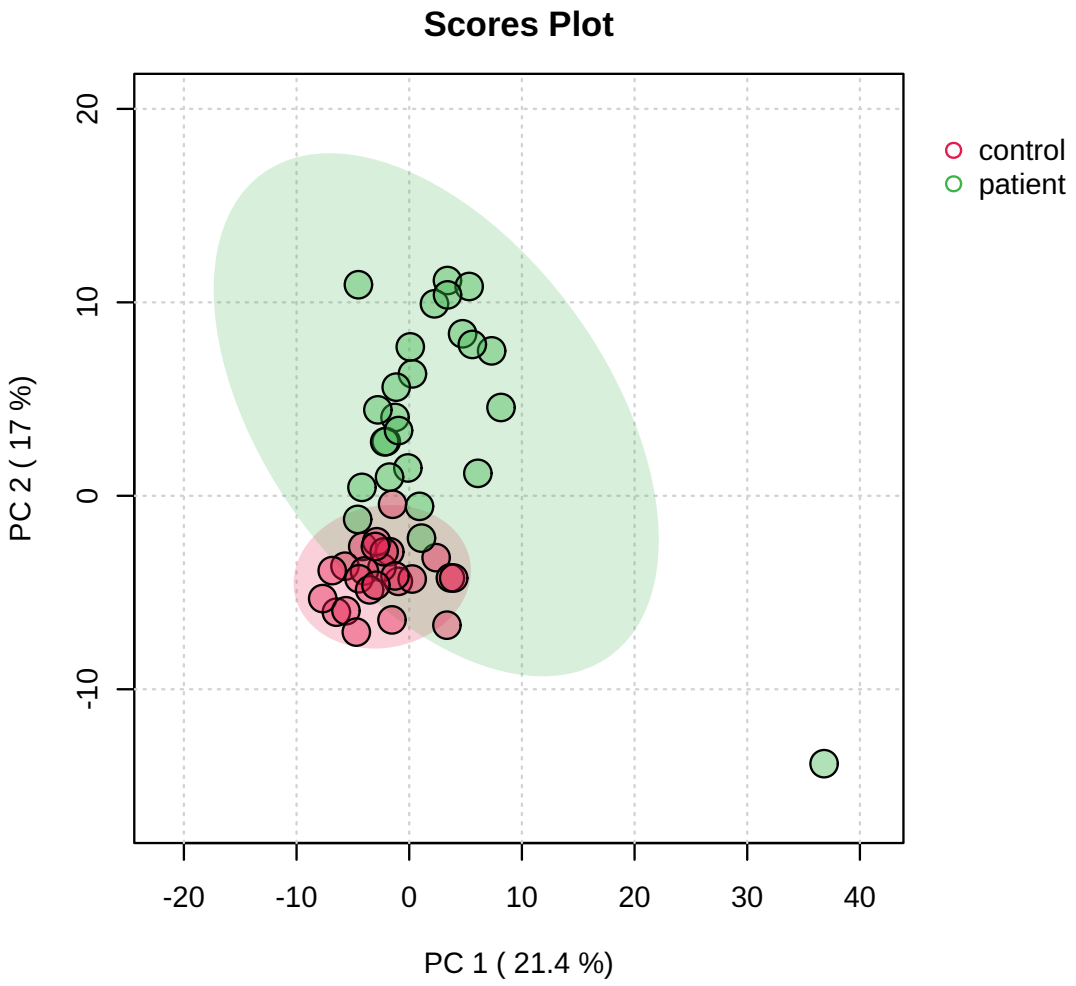


Figure 4: Scores plot between the selected PCs. The explained variances are shown in brackets.

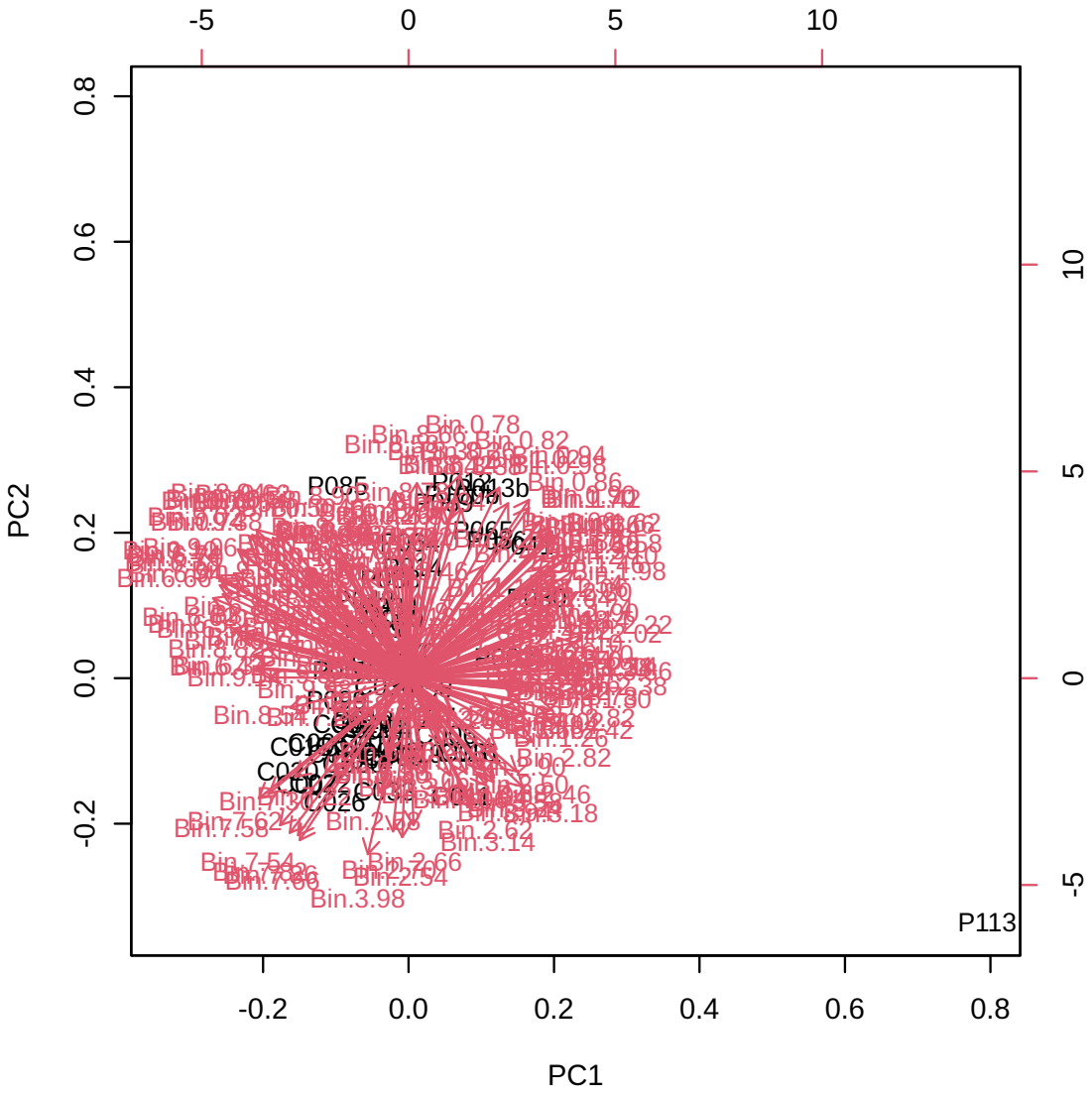


Figure 5: PCA biplot between the selected PCs. Note, you may want to test different centering and scaling normalization methods for the biplot to be displayed properly.

## 2.2 Hierarchical Clustering

In (agglomerative) hierarchical cluster analysis, each sample begins as a separate cluster and the algorithm proceeds to combine them until all samples belong to one cluster. Two parameters need to be considered when performing hierarchical clustering. The first one is similarity measure - Euclidean distance, Pearson's correlation, Spearman's rank correlation. The other parameter is clustering algorithms, including average linkage (clustering uses the centroids of the observations), complete linkage (clustering uses the farthest pair of observations between the two groups), single linkage (clustering uses the closest pair of observations) and Ward's linkage (clustering to minimize the sum of squares of any two clusters). Heatmap is often presented as a visual aid in addition to the dendrogram.

Hierachical clustering is performed with the `hclust` function in package `stat`. Figure 6 shows the clustering result in the form of a dendrogram. Figure 7 shows the clustering result in the form of a heatmap.

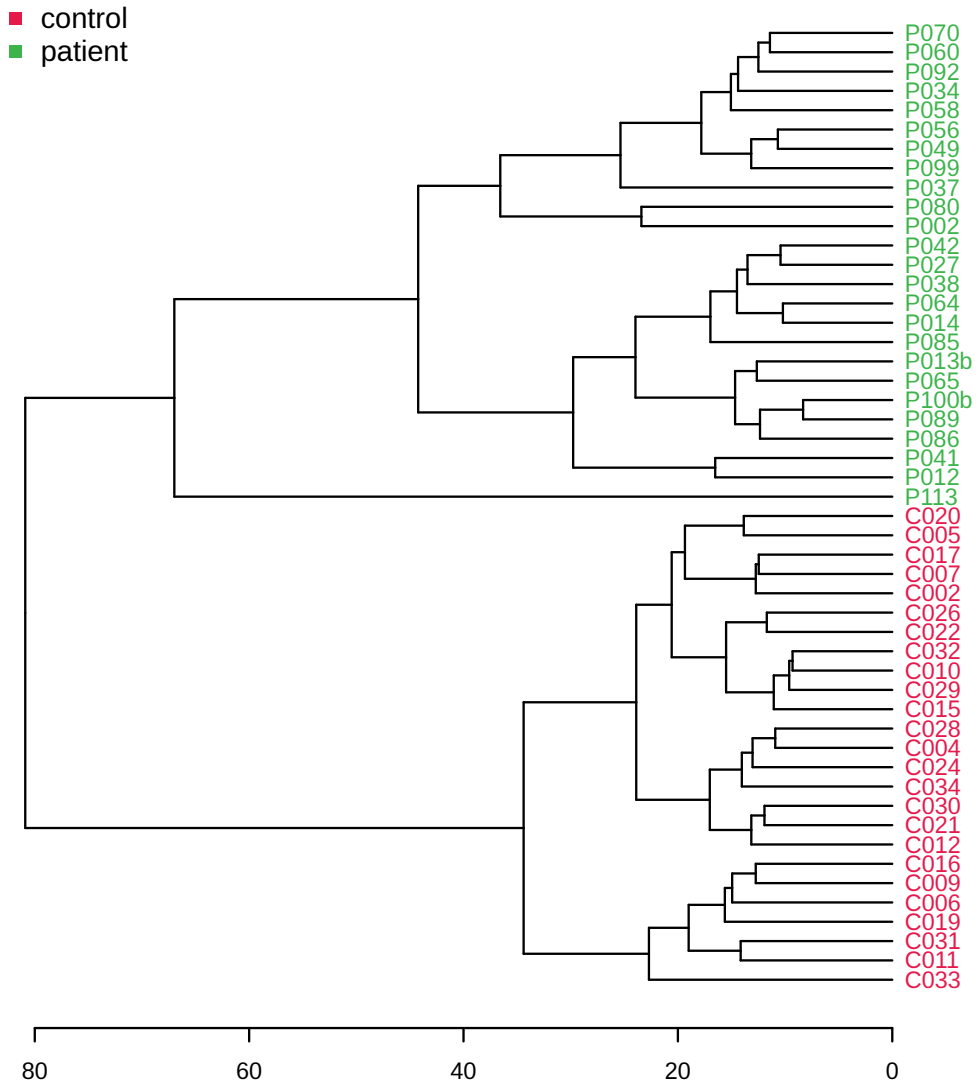


Figure 6: Clustering result shown as dendrogram (distance measure using `euclidean`, and clustering algorithm using `ward.D`).

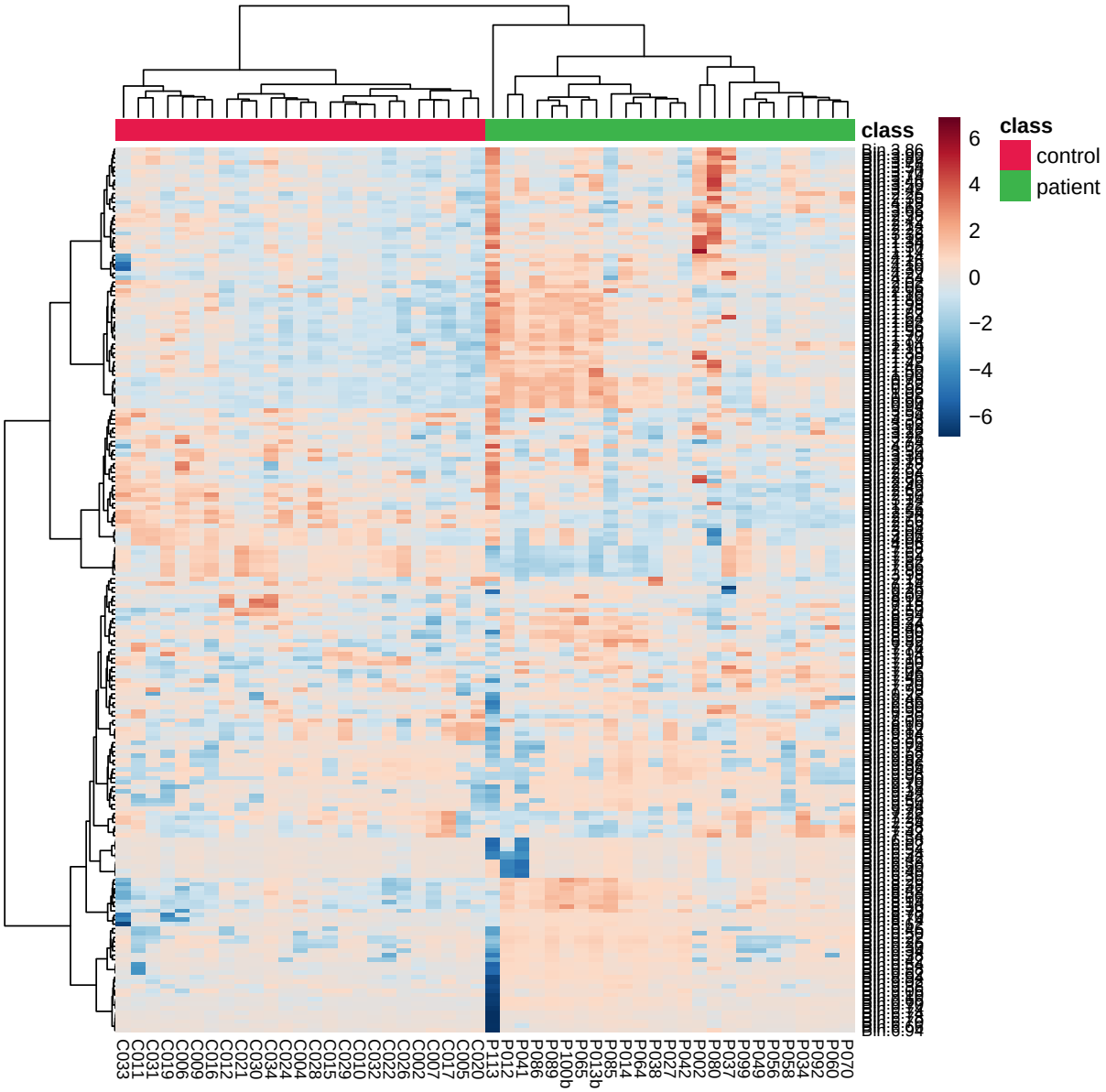


Figure 7: Clustering result shown as heatmap (distance measure using `euclidean`, and clustering algorithm using `ward.D`).

## 2.3 K-means Clustering

K-means clustering is a nonhierarchical clustering technique. It begins by creating k random clusters (k is supplied by user). The program then calculates the mean of each cluster. If an observation is closer to the centroid of another cluster then the observation is made a member of that cluster. This process is repeated until none of the observations are reassigned to a different cluster.

K-means analysis is performed using the `kmeans` function in the package `stat`. Figure 8 shows clustering the results. Table 2 shows the members in each cluster from K-means analysis.

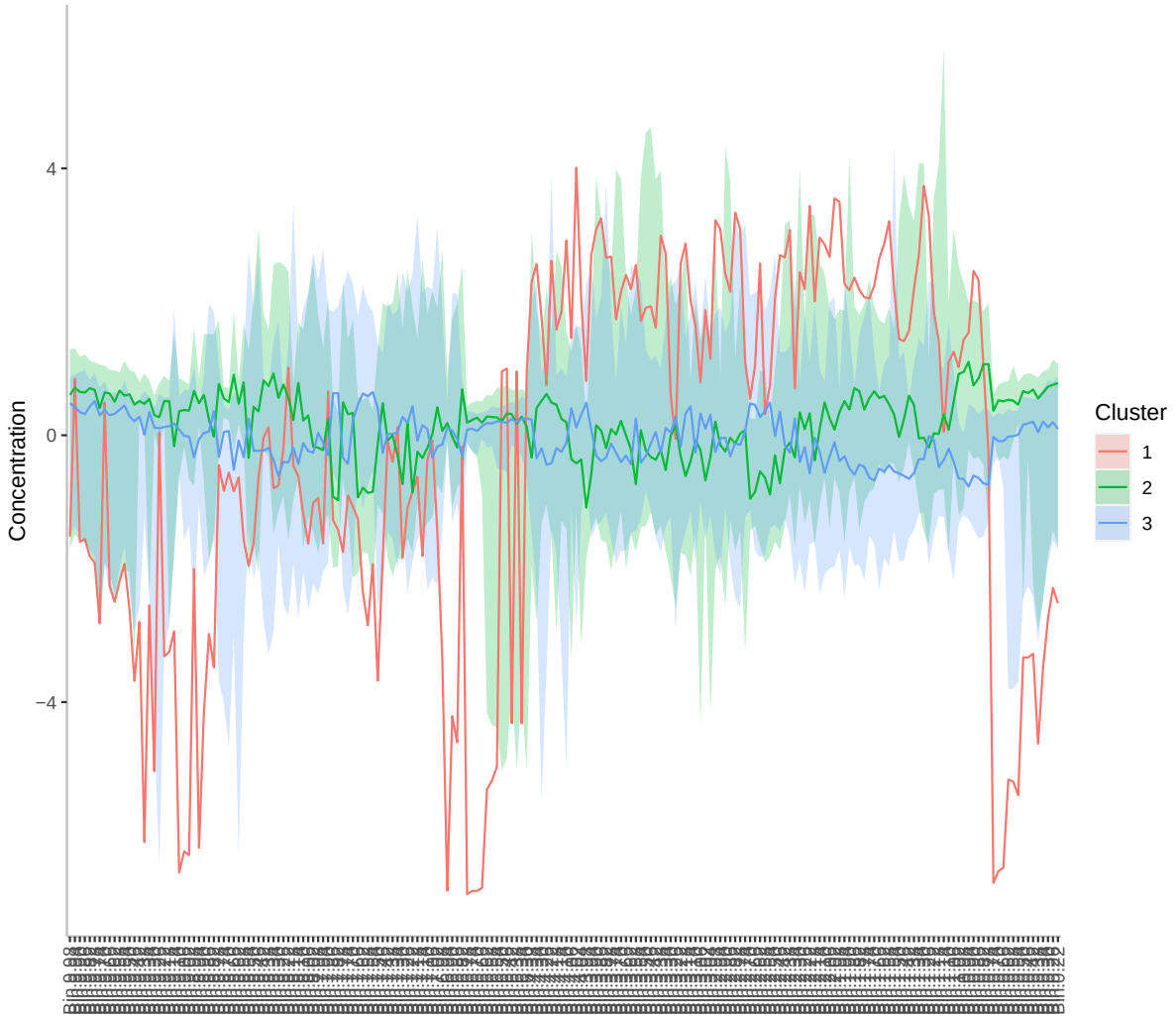


Figure 8: K-means cluster analysis. The x-axes are variable indices and y-axes are relative intensities. The blue lines represent median intensities of corresponding clusters

Table 2: Clustering result using K-means

| | Samples in each cluster |
|---|---|
| Cluster( 1 ) | P113 |
| Cluster( 2 ) | P002 P012 P014 P027 P034 P038 P041 P042 P049 P060 P064 P065 P070 P080 P085 P086 P089 P013b P100b |
| Cluster( 3 ) | C002 C004 C005 C006 C007 C009 C010 C011 C012 C015 C016 C017 C019 C020 C021 C022 C024 C026 C028 C029 C030 C031 C032 C033 C034 P037 P056 P058 P092 P099 |

## 2.4 Random Forest (RF)

Random Forest is a supervised learning algorithm suitable for high dimensional data analysis. It uses an ensemble of classification trees, each of which is grown by random feature selection from a bootstrap sample at each branch. Class prediction is based on the majority vote of the ensemble. RF also provides other useful information such as OOB (out-of-bag) error, variable importance measure, and outlier measures. During tree construction, about one-third of the instances are left out of the bootstrap sample. This OOB data is then used as test sample to obtain an unbiased estimate of the classification error (OOB error). Variable importance is evaluated by measuring the increase of the OOB error when it is permuted. The outlier measures are based on the proximities during tree construction.

RF analysis is performed using the `randomForest` package[4]. Table 3 shows the confusion matrix of random forest. Figure 9 shows the cumulative error rates of random forest analysis for given parameters. Figure 10 shows the important features ranked by random forest. Figure 11 shows the outlier measures of all samples for the given parameters. The OOB error is 0.06

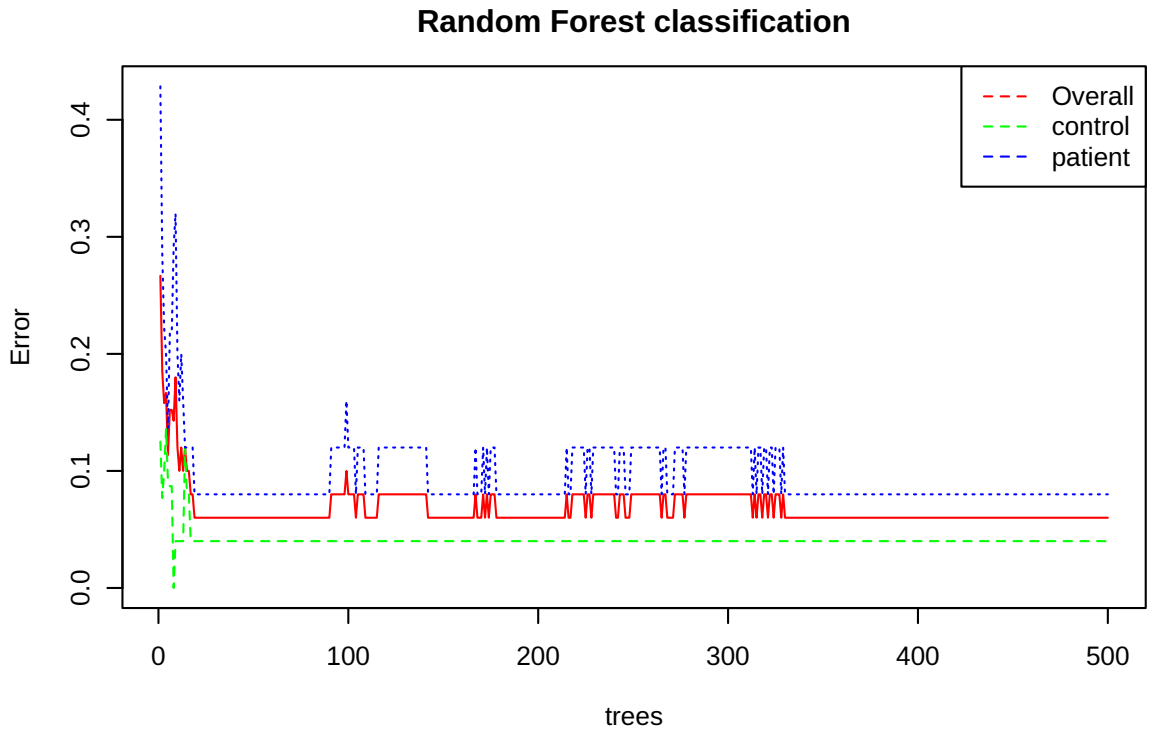


Figure 9: Cumulative error rates by Random Forest classification. The overall error rate is shown as the black line; the red and green lines represent the error rates for each class.

| | control | patient | class.error |
|---|---|---|---|
| control | 24.00 | 1.00 | 0.04 |
| patient | 2.00 | 23.00 | 0.08 |

Table 3: Random Forest Classification Performance

[4] Andy Liaw and Matthew Wiener. *Classification and Regression by randomForest*, 2002, R News

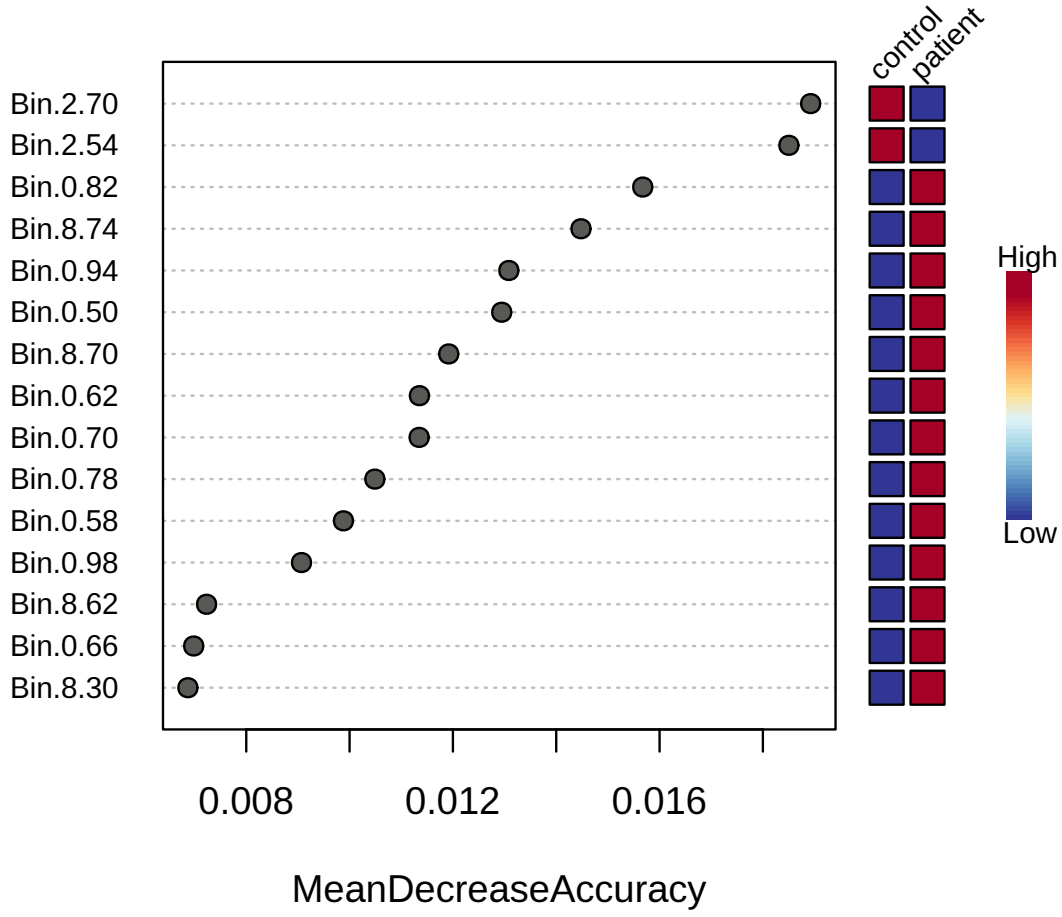


Figure 10: Significant features identified by Random Forest. The features are ranked by the mean decrease in classification accuracy when they are permuted.

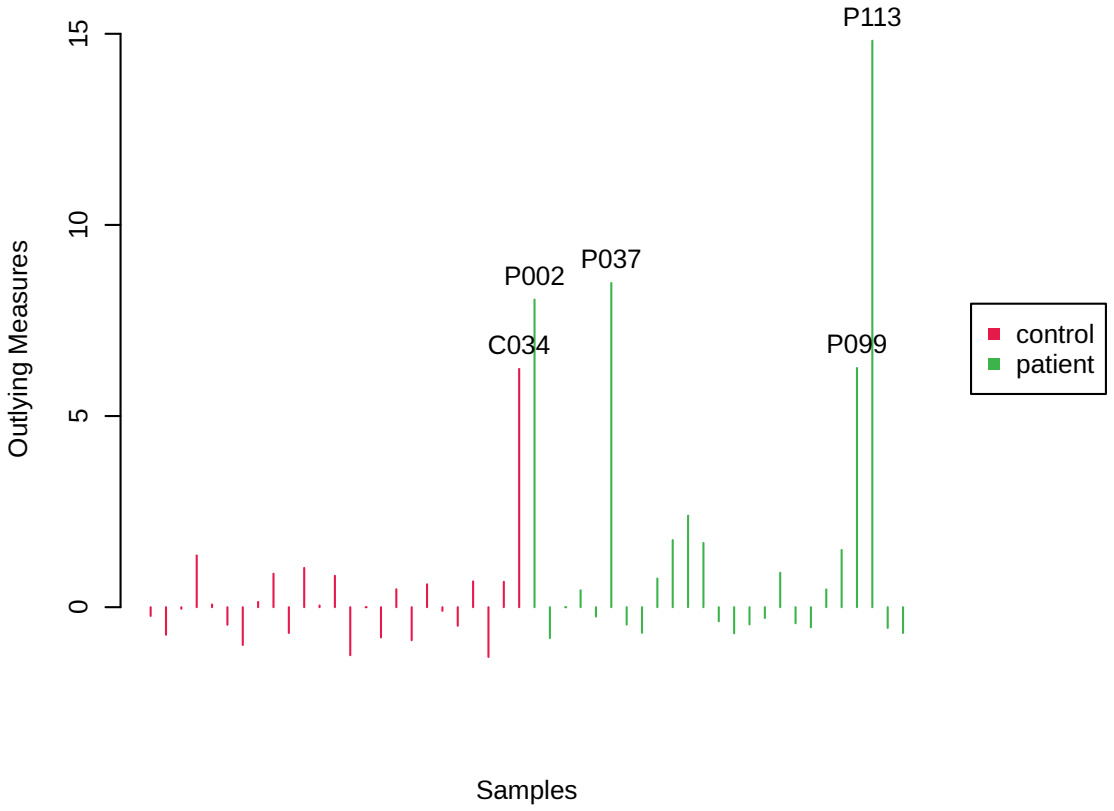


Figure 11: Potential outliers identified by Random Forest. Only the top five are labeled.

## 2.5 Support Vector Machine (SVM)

SVM aims to find a nonlinear decision function in the input space by mapping the data into a higher dimensional feature space and separating it there by means of a maximum margin hyperplane. The SVM-based recursive feature selection and classification is performed using the `R-SVM` script[5]. The process is performed recursively using decreasing series of feature subsets (`ladder`) so that different classification models can be calculated. Feature importance is evaluated based on its frequencies being selected in the best classifier identified by recursive classification and cross-validation. Please note, R-SVM is very computationally intensive. Only the top 50 features (ranked by their p values from t-tests) will be evaluated.

In total, 9 models (levels) were created using 200, 100, 50, 30, 18, 14, 10, 8, 6 selected feature subsets. Figure 12 shows the SVM classification performance using recursive feature selection. Figure 13 shows the signicant features used by the best classifiers.

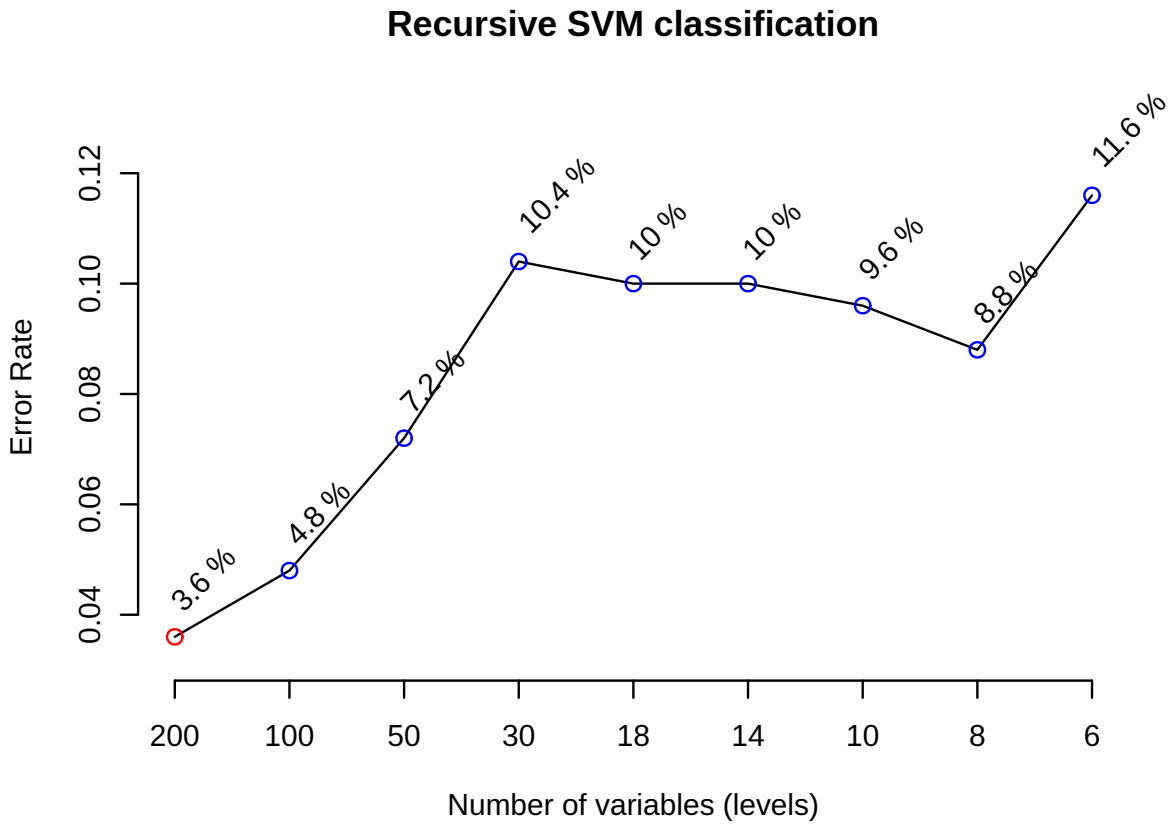


Figure 12: Recursive classification with SVM. The red circle indicates the best classifier.

[5] http://www.hsph.harvard.edu/bioinfocore/RSVMhome/R-SVM.html

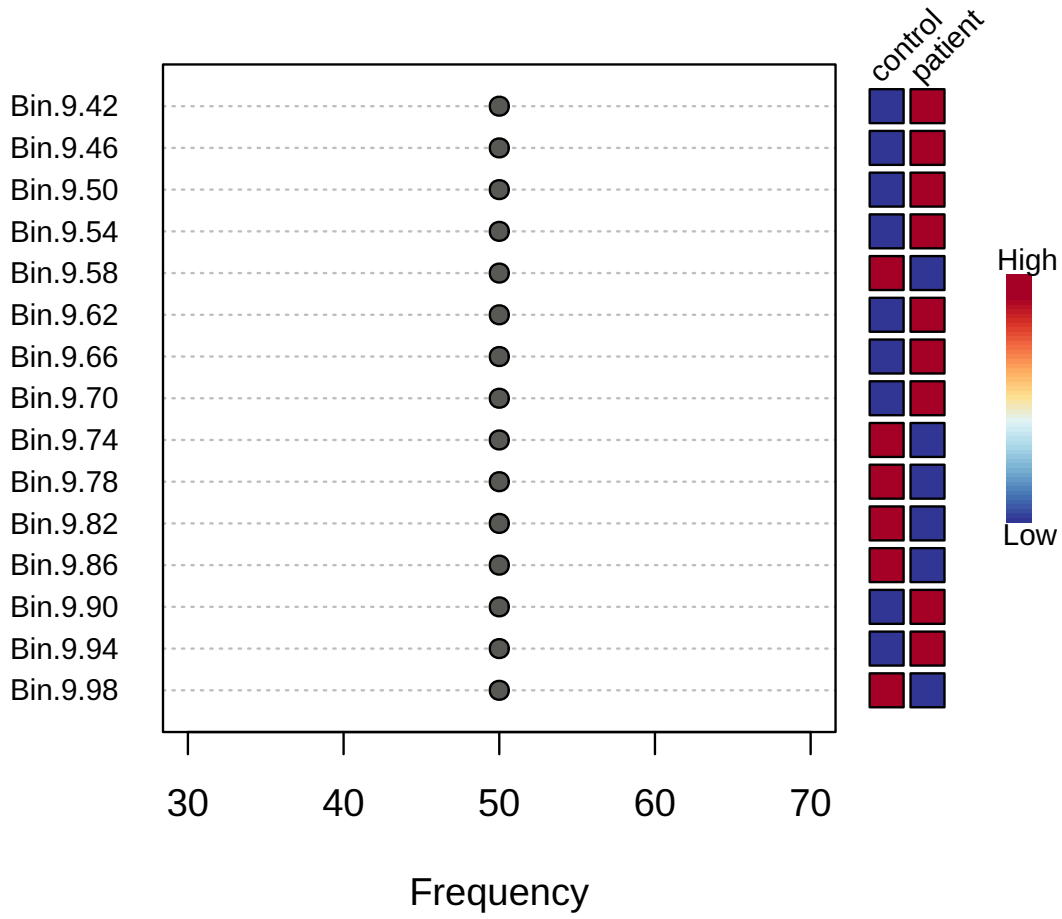


Figure 13: Significant features identified by R-SVM. Features are ranked by their frequencies of being selected in the classifer.

# 3 Appendix: R Command History

```
 [1] "mSet<-InitDataObjects(\"specbin\", \"stat\", FALSE)"
 [2] "mSet<-Read.TextData(mSet, \"Replacing_with_your_file_path\", \"rowu\", \"disc\");"
 [3] "mSet<-SanityCheckData(mSet)"
 [4] "mSet<-ReplaceMin(mSet);"
 [5] "mSet<-SanityCheckData(mSet)"
 [6] "mSet<-FilterVariable(mSet, \"F\", 25, \"none\", -1, \"mean\", 0)"
 [7] "mSet<-PreparePrenormData(mSet)"
 [8] "mSet<-Normalization(mSet, \"MedianNorm\", \"LogNorm\", \"AutoNorm\", ratio=FALSE, ratioNum=20)
 [9] "mSet<-PlotNormSummary(mSet, \"norm_0_\", \"png\", 72, width=NA)"
[10] "mSet<-PlotSampleNormSummary(mSet, \"snorm_0_\", \"png\", 72, width=NA)"
[11] "mSet<-Kmeans.Anal(mSet, 3)"
[12] "mSet<-PlotKmeans(mSet, \"km_0_\", \"png\", 72, width=NA, \"default\", \"F\")"
[13] "mSet<-PlotClustPCA(mSet, \"km_pca_0_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[14] "mSet<-Kmeans.Anal(mSet, 5)"
[15] "mSet<-PlotKmeans(mSet, \"km_1_\", \"png\", 72, width=NA, \"default\", \"F\")"
[16] "mSet<-PlotClustPCA(mSet, \"km_pca_1_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[17] "mSet<-Kmeans.Anal(mSet, 2)"
[18] "mSet<-PlotKmeans(mSet, \"km_2_\", \"png\", 72, width=NA, \"default\", \"F\")"
[19] "mSet<-PlotClustPCA(mSet, \"km_pca_2_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[20] "mSet<-Kmeans.Anal(mSet, 3)"
[21] "mSet<-PlotKmeans(mSet, \"km_3_\", \"png\", 72, width=NA, \"default\", \"F\")"
[22] "mSet<-PlotClustPCA(mSet, \"km_pca_3_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[23] "mSet<-Kmeans.Anal(mSet, 2)"
[24] "mSet<-PlotKmeans(mSet, \"km_4_\", \"png\", 72, width=NA, \"default\", \"F\")"
[25] "mSet<-PlotClustPCA(mSet, \"km_pca_4_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[26] "mSet<-Kmeans.Anal(mSet, 3)"
[27] "mSet<-PlotKmeans(mSet, \"km_5_\", \"png\", 72, width=NA, \"default\", \"F\")"
[28] "mSet<-PlotClustPCA(mSet, \"km_pca_5_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[29] "mSet<-Kmeans.Anal(mSet, 2)"
[30] "mSet<-PlotKmeans(mSet, \"km_6_\", \"png\", 72, width=NA, \"default\", \"F\")"
[31] "mSet<-PlotClustPCA(mSet, \"km_pca_6_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[32] "mSet<-Kmeans.Anal(mSet, 3)"
[33] "mSet<-PlotKmeans(mSet, \"km_7_\", \"png\", 72, width=NA, \"default\", \"F\")"
[34] "mSet<-PlotClustPCA(mSet, \"km_pca_7_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[35] "mSet<-Kmeans.Anal(mSet, 3)"
[36] "mSet<-PlotKmeans(mSet, \"km_8_\", \"png\", 72, width=NA, \"default\", \"T\")"
[37] "mSet<-PlotClustPCA(mSet, \"km_pca_8_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[38] "mSet<-Kmeans.Anal(mSet, 3)"
[39] "mSet<-PlotKmeans(mSet, \"km_9_\", \"png\", 72, width=NA, \"default\", \"F\")"
[40] "mSet<-PlotClustPCA(mSet, \"km_pca_9_\", \"png\", 72, width=NA, \"default\", \"km\", \"T\")"
[41] "mSet<-PlotHeatMap(mSet, \"heatmap_1_\", \"png\", 72, width=NA, \"norm\", \"row\", \"euclidean\
[42] "mSet<-RF.Anal(mSet, 500,7,1)"
[43] "mSet<-PlotRF.Classify(mSet, \"rf_cls_0_\", \"png\", 72, width=NA)"
[44] "mSet<-PlotRF.VIP(mSet, \"rf_imp_0_\", \"png\", 72, width=NA)"
[45] "mSet<-PlotRF.Outlier(mSet, \"rf_outlier_0_\", \"png\", 72, width=NA)"
[46] "mSet<-RF.Anal(mSet, 500,7,1)"
[47] "mSet<-PlotRF.Classify(mSet, \"rf_cls_1_\", \"png\", 72, width=NA)"
[48] "mSet<-PlotRF.VIP(mSet, \"rf_imp_1_\", \"png\", 72, width=NA)"
[49] "mSet<-PlotRF.Outlier(mSet, \"rf_outlier_1_\", \"png\", 72, width=NA)"
[50] "mSet<-RSVM.Anal(mSet, 10)"
[51] "mSet<-PlotRSVM.Classification(mSet, \"svm_cls_0_\", \"png\", 72, width=NA)"
[52] "mSet<-PlotRSVM.Cmpd(mSet, \"svm_imp_0_\", \"png\", 72, width=NA)"
[53] "mSet<-PlotHCTree(mSet, \"tree_0_\", \"png\", 72, width=NA, \"euclidean\", \"ward.D\")"
[54] "mSet<-PlotHCTree(mSet, \"tree_1_\", \"png\", 72, width=NA, \"spearman\", \"complete\")"
[55] "mSet<-PlotHCTree(mSet, \"tree_2_\", \"png\", 72, width=NA, \"spearman\", \"single\")"
[56] "mSet<-PlotHCTree(mSet, \"tree_3_\", \"png\", 72, width=NA, \"euclidean\", \"single\")"
```

```
[57] "mSet<-PlotHCTree(mSet, \"tree_4_\", \"png\", 72, width=NA, \"euclidean\", \"ward.D\")"
[58] "mSet<-PCA.Anal(mSet)"
[59] "mSet<-PlotPCAPairSummary(mSet, \"pca_pair_0_\", \"png\", 72, width=NA, 5)"
[60] "mSet<-PlotPCAScree(mSet, \"pca_scree_0_\", \"png\", 72, width=NA, 5)"
[61] "mSet<-PlotPCA2DScore(mSet, \"pca_score2d_0_\", \"png\", 72, width=NA, 1,2,0.95,0,0, \"na\")"
[62] "mSet<-PlotPCALoading(mSet, \"pca_loading_0_\", \"png\", 72, width=NA, 1,2);"
[63] "mSet<-PlotPCABiplot(mSet, \"pca_biplot_0_\", \"png\", 72, width=NA, 1,2)"
[64] "mSet<-PlotPCA3DLoading(mSet, \"pca_loading3d_0_\", \"json\", 1,2,3)"
[65] "mSet<-SaveTransformedData(mSet)"
[66] "mSet<-PreparePDFReport(mSet, \"guest5394319678214572061\")\n"
```

---

The report was generated on Wed May 8 06:47:56 2024 with R version 4.3.2 (2023-10-31), OS system: Linux, version: -Ubuntu SMP Tue Mar 5 20:16:58 UTC 2024 .