Metabolomic Data Analysis with MetaboAnalyst 6.0

Name: guest12006273676888371586

March 20, 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: Square Root Transformation; Data scaling: Range Scaling.

[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

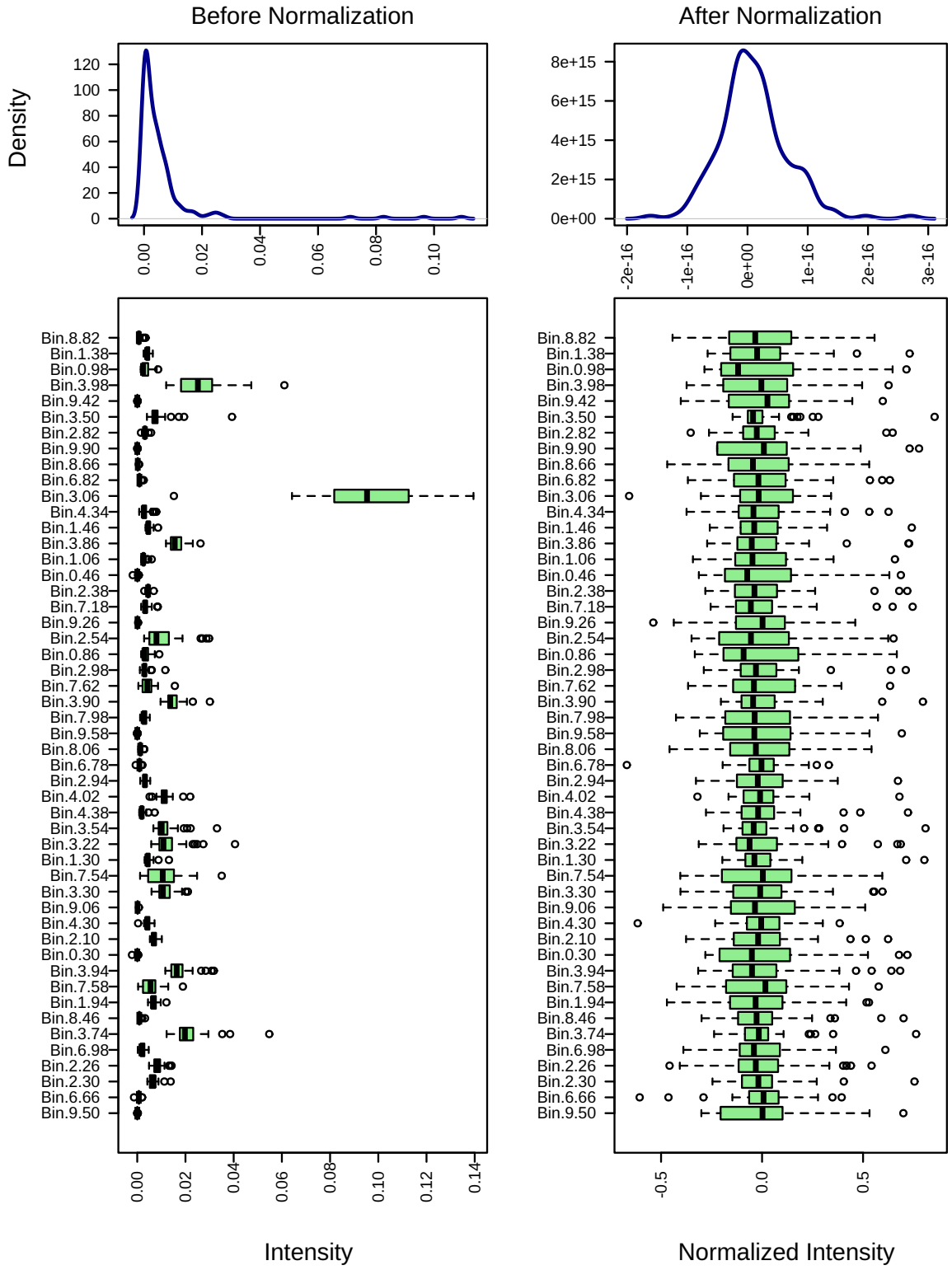

Figure 1: Box plots and kernel density plots before and after normalization. The boxplots show at most 50 features due to space limit. The density plots are based on all samples.

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 Univariate Analysis

Univariate analysis methods are the most common methods used for exploratory data analysis. For two-group data, MetaboAnalyst provides Fold Change (FC) analysis, t-tests, and volcano plot which is a combination of the first two methods. All three these methods support both unpaired and paired analyses. For multi-group analysis, MetaboAnalyst provides two types of analysis - one-way analysis of variance (ANOVA) with associated post-hoc analyses, and correlation analysis to identify signficant compounds that follow a given pattern. The univariate analyses provide a preliminary overview about features that are potentially significant in discriminating the conditions under study.

For paired fold change analysis, the algorithm first counts the total number of pairs with fold changes that are consistently above/below the specified FC threshold for each variable. A variable will be reported as significant if this number is above a given count threshold (default $>$ 75% of pairs/variable)

Figure 2 shows the important features identified by fold change analysis. Table 2 shows the details of these features; Figure 3 shows the important features identified by t-tests. Table 3 shows the details of these features;

Please note, the purpose of fold change is to compare absolute value changes between two group means. Therefore, the data before column normalization will be used instead. Also note, the result is plotted in log2 scale, so that same fold change (up/down regulated) will have the same distance to the zero baseline.

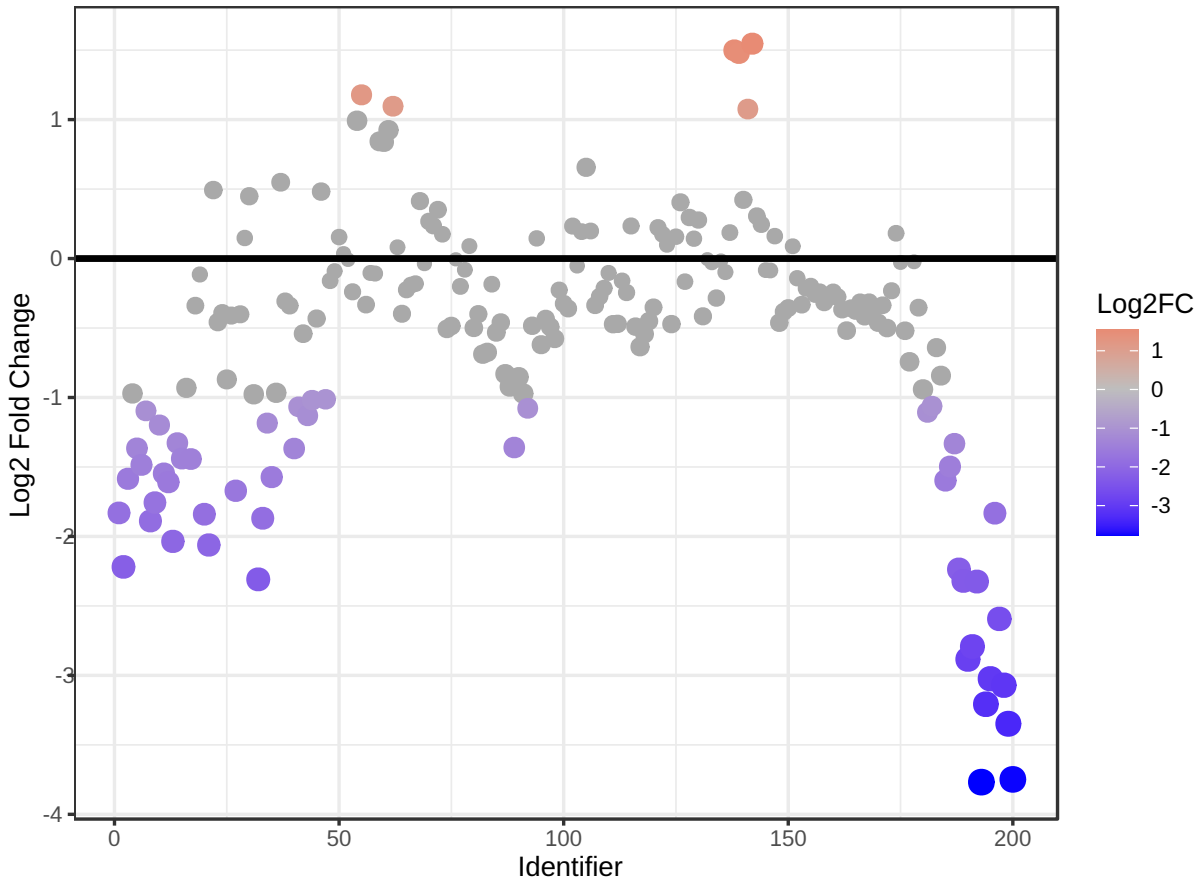

Figure 2: Important features selected by fold-change analysis with threshold 2. The red circles represent features above the threshold. Note the values are on log scale, so that both up-regulated and down-regulated features can be plotted in a symmetrical way

Table 2: Top 50 features identified by fold change analysis

	Spectra Bins	Fold Change	log2(FC)
1	Bin.0.50	0.073381	-3.7684
2	Bin.0.22	0.074396	-3.7486
3	Bin.0.26	0.098188	-3.3483
4	Bin.0.46	0.10825	-3.2076
5	Bin.0.30	0.11901	-3.0708
6	Bin.0.42	0.12291	-3.0243
7	Bin.0.62	0.13546	-2.8841
8	Bin.0.58	0.14434	-2.7925
9	Bin.0.34	0.16568	-2.5935
10	Bin.0.54	0.19961	-2.3248
11	Bin.0.66	0.20028	-2.3199
12	Bin.8.74	0.2018	-2.309
13	Bin.0.70	0.21209	-2.2373
14	Bin.9.94	0.21475	-2.2193
15	Bin.9.18	0.23955	-2.0616
16	Bin.9.50	0.2439	-2.0356
17	Bin.9.70	0.27004	-1.8887
18	Bin.8.70	0.27372	-1.8692
19	Bin.9.22	0.27928	-1.8402
20	Bin.0.38	0.28071	-1.8328
21	Bin.9.98	0.2811	-1.8308
22	Bin.9.66	0.29597	-1.7565
23	Bin.8.94	0.31398	-1.6713
24	Bin.9.54	0.32773	-1.6094
25	Bin.0.82	0.33046	-1.5975
26	Bin.9.90	0.33333	-1.585
27	Bin.8.62	0.33612	-1.573
28	Bin.9.58	0.34217	-1.5472
29	Bin.2.54	2.9209	1.5464
30	Bin.2.70	2.8251	1.4983
31	Bin.0.78	0.35423	-1.4972
32	Bin.9.78	0.35721	-1.4851
33	Bin.2.66	2.788	1.4792
34	Bin.9.34	0.36779	-1.443
35	Bin.9.42	0.3686	-1.4399
36	Bin.8.42	0.3876	-1.3674
37	Bin.9.82	0.38804	-1.3657
38	Bin.6.46	0.38956	-1.3601
39	Bin.0.74	0.39715	-1.3323
40	Bin.9.46	0.39847	-1.3274
41	Bin.9.62	0.43577	-1.1984
42	Bin.8.66	0.44002	-1.1844
43	Bin.7.82	2.263	1.1782
44	Bin.8.30	0.45651	-1.1313
45	Bin.0.98	0.46426	-1.107
46	Bin.9.74	0.46771	-1.0963
47	Bin.7.54	2.1377	1.0961
48	Bin.6.34	0.47367	-1.078
49	Bin.2.58	2.1077	1.0757
50	Bin.8.38	0.47723	-1.0672

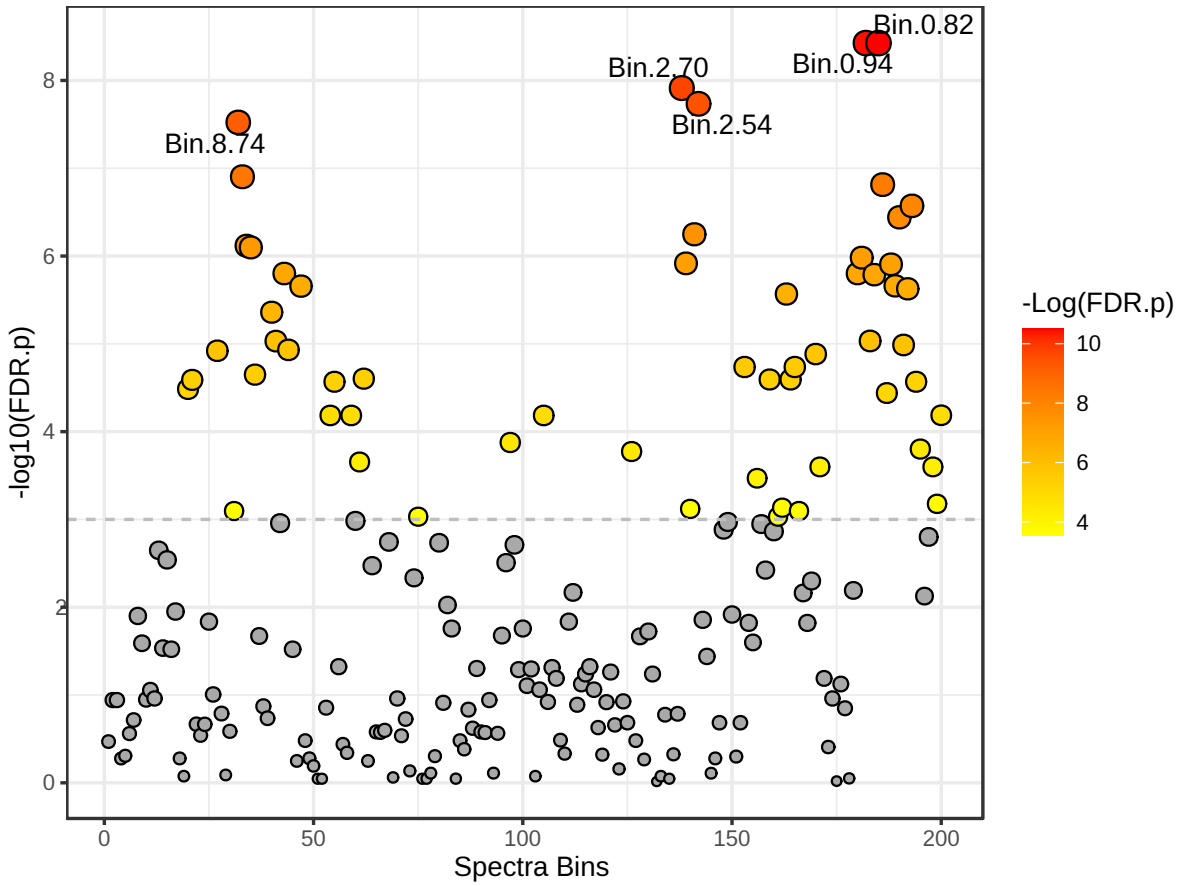

Figure 3: Important features selected by t-tests with threshold 0.001. The red circles represent features above the threshold. Note the p values are transformed by -log10 so that the more significant features (with smaller p values) will be plotted higher on the graph.

Table 3: Top 50 features identified by t-tests

	Spectra Bins	t.stat	p.value	-log10(p)	FDR
1	Bin.0.82	-8.5573	3.2089e-11	10.494	3.753e-09
2	Bin.0.94	-8.5117	3.753e-11	10.426	3.753e-09
3	Bin.2.70	8.053	1.8312e-10	9.7373	1.2208e-08
4	Bin.2.54	7.8517	3.6913e-10	9.4328	1.8456e-08
5	Bin.8.74	-7.6489	7.5003e-10	9.1249	3.0001e-08
6	Bin.8.70	-7.1909	3.7498e-09	8.426	1.2499e-07
7	Bin.0.78	-7.0887	5.3764e-09	8.2695	1.5361e-07
8	Bin.0.50	-6.8932	1.072e-08	7.9698	2.6801e-07
9	Bin.0.62	-6.7748	1.6289e-08	7.7881	3.6199e-07
10	Bin.2.58	6.6189	2.8257e-08	7.5489	5.6515e-07
11	Bin.8.66	-6.5071	4.1947e-08	7.3773	7.6267e-07
12	Bin.8.62	-6.4696	4.7886e-08	7.3198	7.9809e-07
13	Bin.0.98	-6.3712	6.7777e-08	7.1689	1.0427e-06
14	Bin.2.66	6.3077	8.482e-08	7.0715	1.2117e-06
15	Bin.0.70	-6.2811	9.3151e-08	7.0308	1.242e-06
16	Bin.8.30	-6.1791	1.3348e-07	6.8746	1.5808e-06
17	Bin.1.02	-6.1772	1.3437e-07	6.8717	1.5808e-06
18	Bin.0.86	-6.1525	1.466e-07	6.8339	1.6289e-06
19	Bin.0.66	-6.0538	2.0751e-07	6.683	2.1843e-06
20	Bin.8.14	-6.0371	2.2004e-07	6.6575	2.2004e-06
21	Bin.0.54	-6.0032	2.4794e-07	6.6057	2.3613e-06
22	Bin.1.70	-5.951	2.9783e-07	6.526	2.7076e-06
23	Bin.8.42	-5.8025	5.0151e-07	6.2997	4.361e-06
24	Bin.0.90	-5.5743	1.1128e-06	5.9536	9.273e-06
25	Bin.8.38	-5.5616	1.1635e-06	5.9343	9.3076e-06
26	Bin.0.58	-5.5212	1.3388e-06	5.8733	1.0298e-05
27	Bin.8.26	-5.4734	1.5809e-06	5.8011	1.171e-05
28	Bin.8.94	-5.4561	1.6783e-06	5.7751	1.1988e-05
29	Bin.1.42	-5.421	1.8958e-06	5.7222	1.3074e-05
30	Bin.2.10	-5.3052	2.8303e-06	5.5482	1.836e-05
31	Bin.1.62	-5.3036	2.8459e-06	5.5458	1.836e-05
32	Bin.8.58	-5.2363	3.5882e-06	5.4451	2.2427e-05
33	Bin.7.54	5.1967	4.1125e-06	5.3859	2.4924e-05
34	Bin.1.66	-5.1807	4.3447e-06	5.362	2.5467e-05
35	Bin.1.86	-5.1733	4.4568e-06	5.351	2.5467e-05
36	Bin.9.18	-5.162	4.6329e-06	5.3342	2.5738e-05
37	Bin.7.82	5.1335	5.1088e-06	5.2917	2.7035e-05
38	Bin.0.46	-5.1319	5.1366e-06	5.2893	2.7035e-05
39	Bin.9.22	-5.0687	6.3764e-06	5.1954	3.2699e-05
40	Bin.0.74	-5.0304	7.2692e-06	5.1385	3.6346e-05
41	Bin.0.22	-4.8515	1.3342e-05	4.8748	6.5084e-05
42	Bin.3.98	4.835	1.4106e-05	4.8506	6.5414e-05
43	Bin.7.86	4.8298	1.4358e-05	4.8429	6.5414e-05
44	Bin.7.66	4.8291	1.4391e-05	4.8419	6.5414e-05
45	Bin.4.26	-4.6109	2.9914e-05	4.5241	0.00013295
46	Bin.0.42	-4.5523	3.6335e-05	4.4397	0.00015798
47	Bin.3.14	4.5264	3.9595e-05	4.4024	0.00016849
48	Bin.7.58	4.4369	5.32e-05	4.2741	0.00022167
49	Bin.0.30	-4.3903	6.1985e-05	4.2077	0.00025212
50	Bin.1.38	-4.3852	6.303e-05	4.2005	0.00025212

2.2 Correlation Analysis

Correlation analysis can be used to visualize the overall correlations between different features It can also be used to identify which features are correlated with a feature of interest. Correlation analysis can also be used to identify if certain features show particular patterns under different conditions. Users first need to define a pattern in the form of a series of hyphenated numbers. For example, in a time-series study with four time points, a pattern of of `1-2-3-4` is used to search compounds with increasing the concentration as time changes; while a pattern of `3-2-1-3` can be used to search compounds that decrease at first, then bounce back to the original level.

Figure 4 shows the overall correlation heatmap.

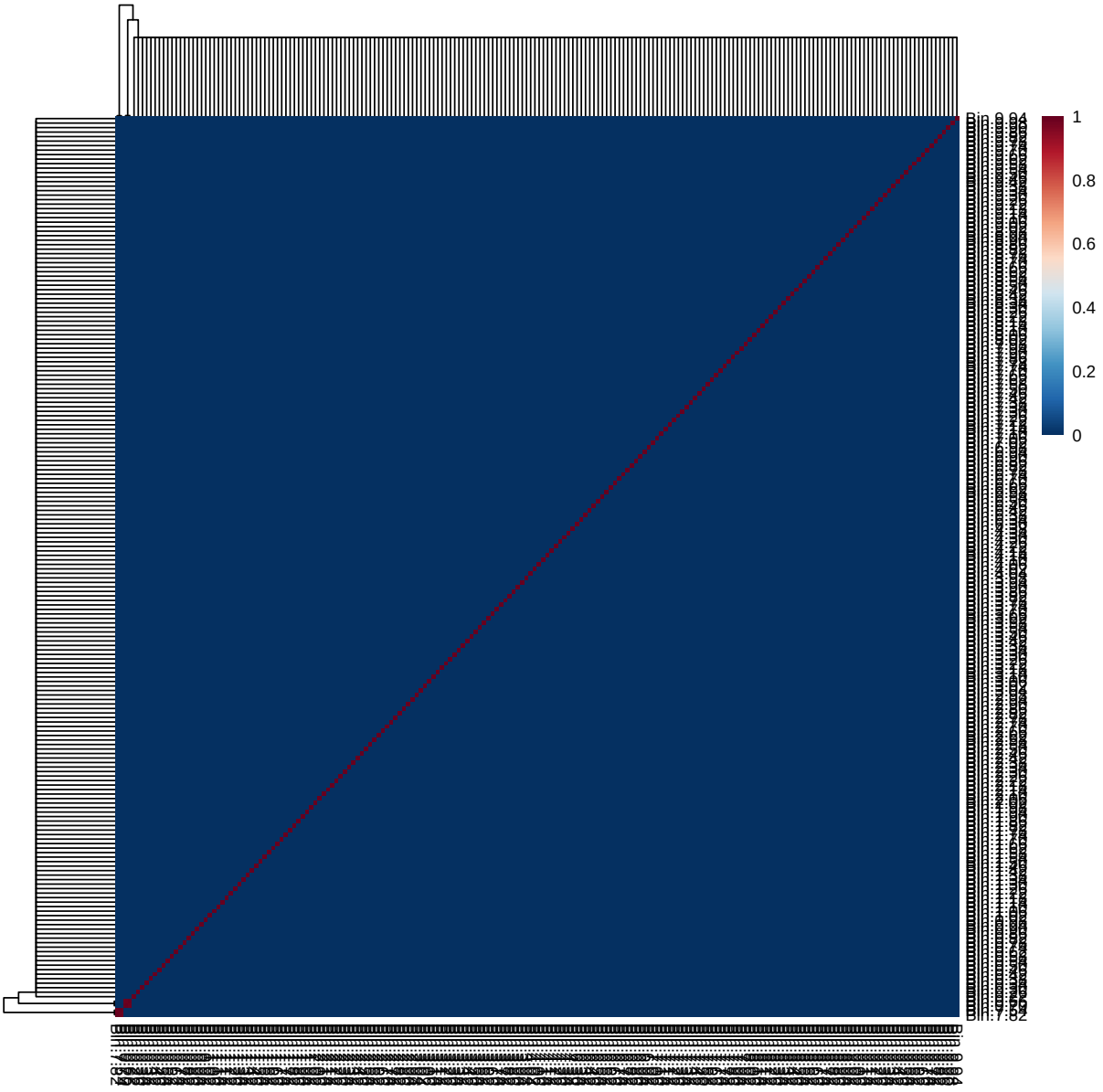

Figure 4: Correlation Heatmaps

2.3 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 5 is pairwise score plots providing an overview of the various seperation patterns among the most significant PCs; Figure 6 is the scree plot showing the variances explained by the selected PCs; Figure 7 shows the 2-D scores plot between selected PCs; Figure 8 shows the biplot between the selected PCs. Interactive 3-D scores plots are not included here and can be directly downloaded from website.

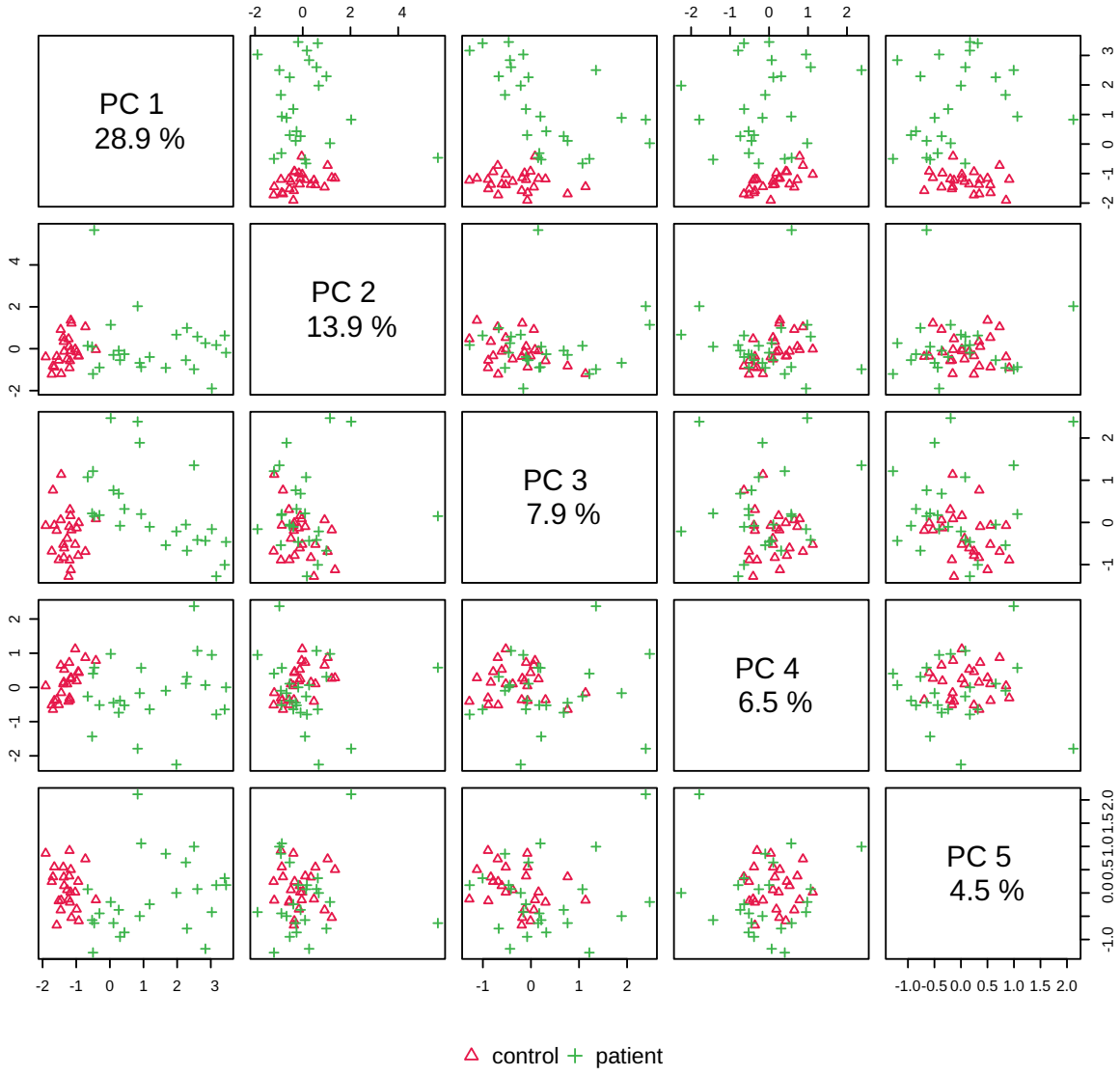

Figure 5: Pairwise score plots between the selected PCs. The explained variance of each PC is shown in the corresponding diagonal cell.

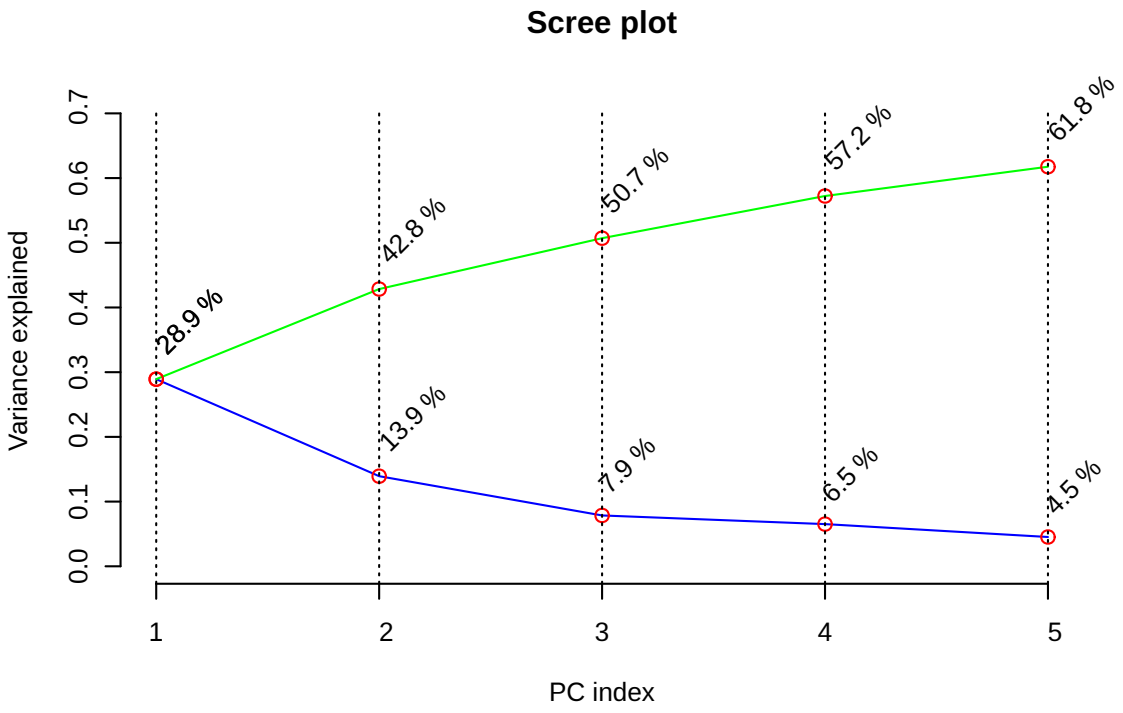

Figure 6: Scree plot shows the variance explained by PCs. The green line on top shows the accumulated variance explained; the blue line underneath shows the variance explained by individual PC.

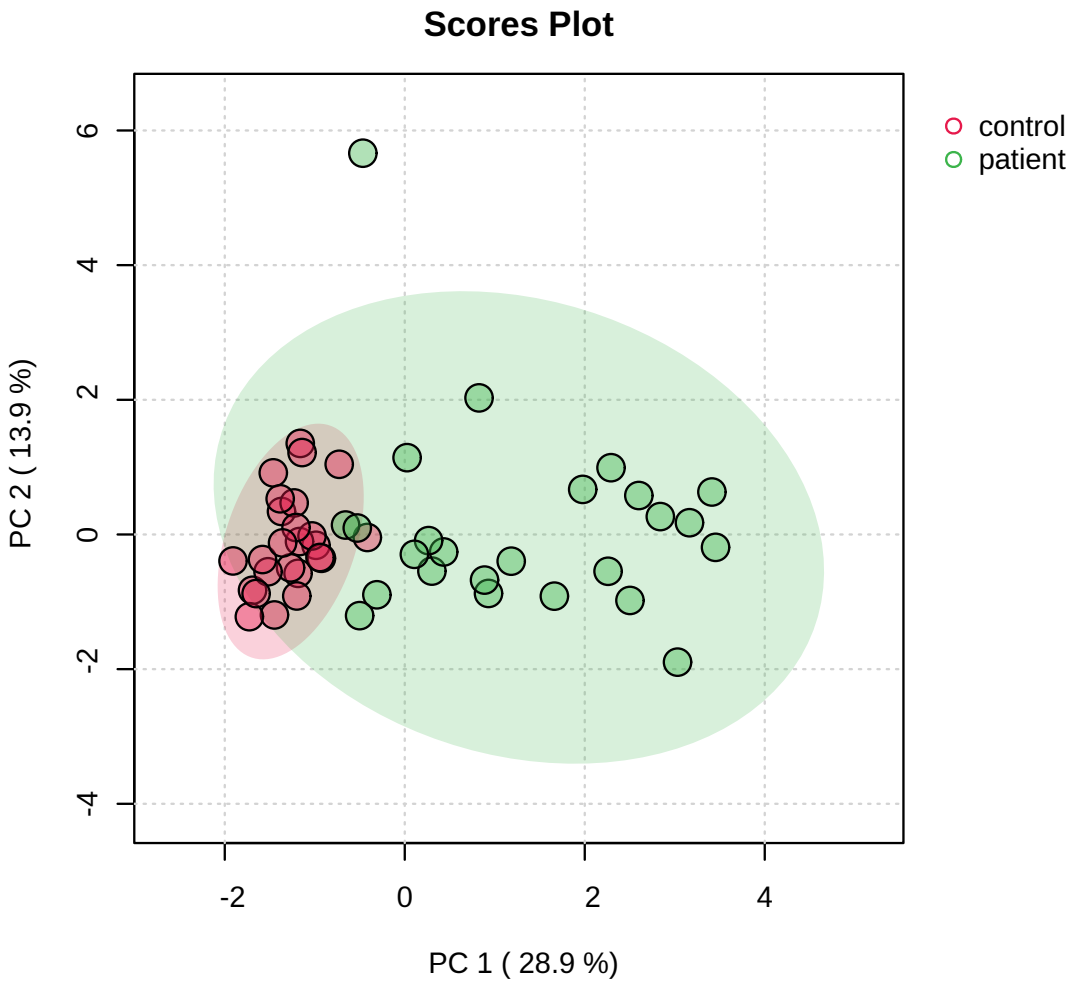

Figure 7: Scores plot between the selected PCs. The explained variances are shown in brackets.

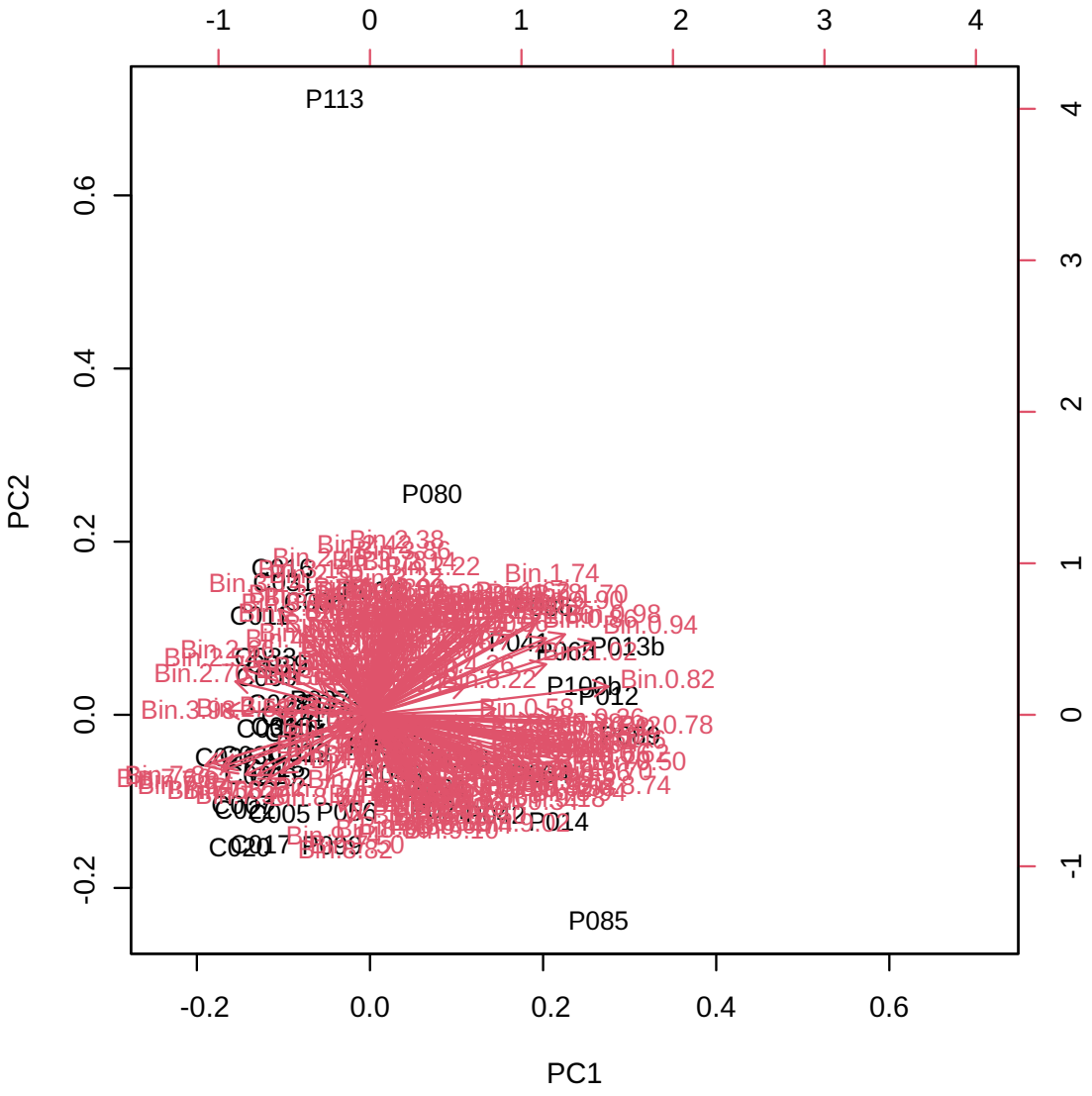

Figure 8: PCA biplot between the selected PCs. Note, you may want to test different centering and scaling normalization methods for the biplot to be displayed properly.

2.4 Partial Least Squares - Discriminant Analysis (PLS-DA)

PLS is a supervised method that uses multivariate regression techniques to extract via linear combination of original variables (X) the information that can predict the class membership (Y). The PLS regression is performed using the `plsr` function provided by R `pls` package[4]. The classification and cross-validation are performed using the corresponding wrapper function offered by the `caret` package[5].

To assess the significance of class discrimination, a permutation test was performed. In each permutation, a PLS-DA model was built between the data (X) and the permuted class labels (Y) using the optimal number of components determined by cross validation for the model based on the original class assignment. MetaboAnalyst supports two types of test statistics for measuring the class discrimination. The first one is based on prediction accuracy during training. The second one is separation distance based on the ratio of the between group sum of the squares and the within group sum of squares (B/W-ratio). If the observed test statistic is part of the distribution based on the permuted class assignments, the class discrimination cannot be considered significant from a statistical point of view.[6].

There are two variable importance measures in PLS-DA. The first, Variable Importance in Projection (VIP) is a weighted sum of squares of the PLS loadings taking into account the amount of explained Y-variation in each dimension. Please note, VIP scores are calculated for each components. When more than components are used to calculate the feature importance, the average of the VIP scores are used. The other importance measure is based on the weighted sum of PLS-regression. The weights are a function of the reduction of the sums of squares across the number of PLS components. Please note, for multiple-group (more than two) analysis, the same number of predictors will be built for each group. Therefore, the coefficient of each feature will be different depending on which group you want to predict. The average of the feature coefficients are used to indicate the overall coefficient-based importance.

Figure 9 shows the overview of scores plots; Figure 10 shows the 2-D scores plot between selected components; Figure 11 shows the 3-D scores plot between selected components; Figure 12 shows the loading plot between the selected components;Figure 13 shows the classification performance with different number of components; Figure 14 shows the results of permutation test for model validation; Figure 15 shows important features identified by PLS-DA.

[4]Ron Wehrens and Bjorn-Helge Mevik.*pls: Partial Least Squares Regression (PLSR) and Principal Component Regression (PCR)*, 2007, R package version 2.1-0

[5]Max Kuhn. Contributions from Jed Wing and Steve Weston and Andre Williams.*caret: Classification and Regression Training*, 2008, R package version 3.45

[6]Bijlsma et al.*Large-Scale Human Metabolomics Studies: A Strategy for Data (Pre-) Processing and Validation*, Anal Chem. 2006, 78 567 - 574

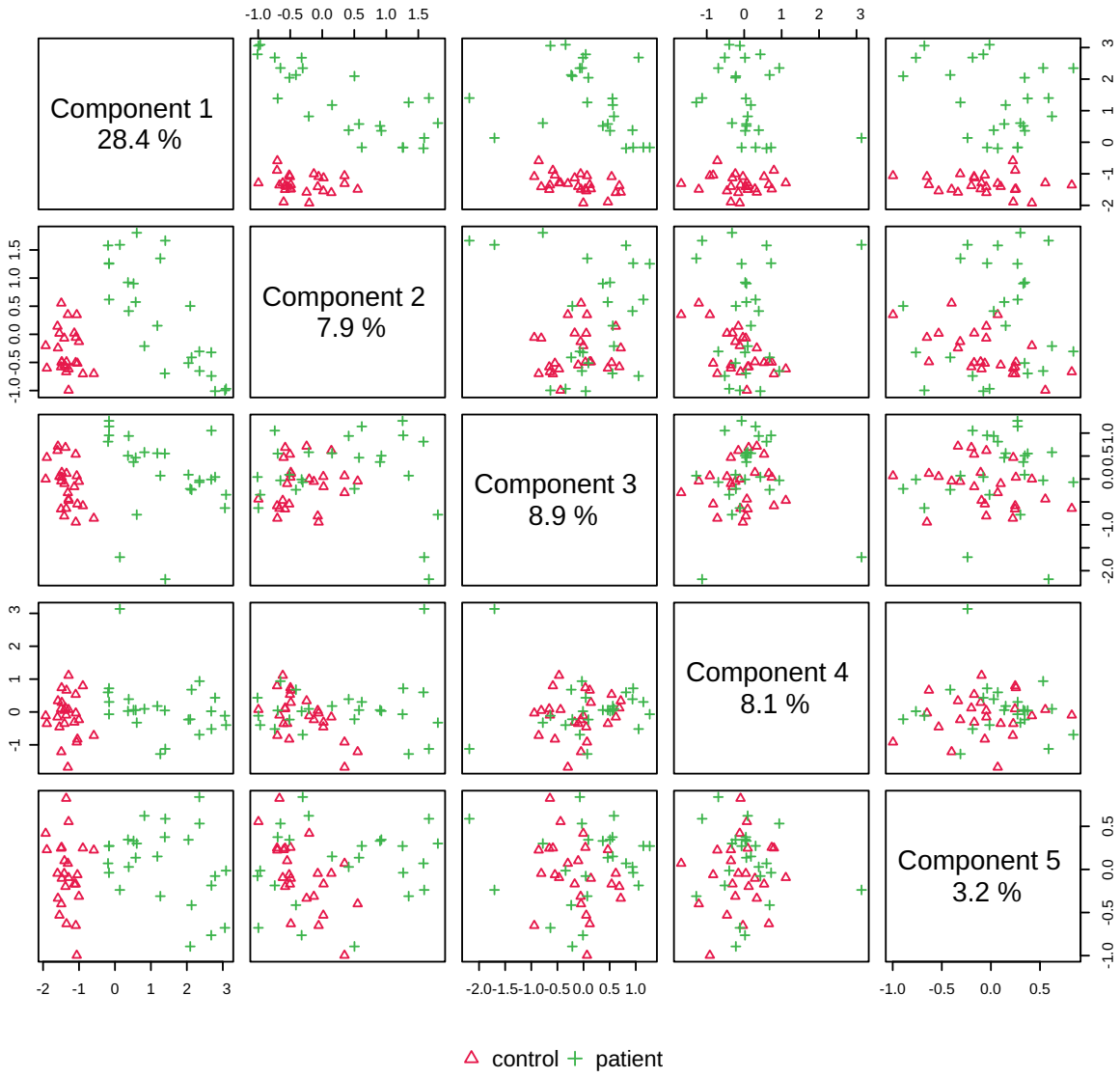

Figure 9: Pairwise scores plots between the selected components. The explained variance of each component is shown in the corresponding diagonal cell.

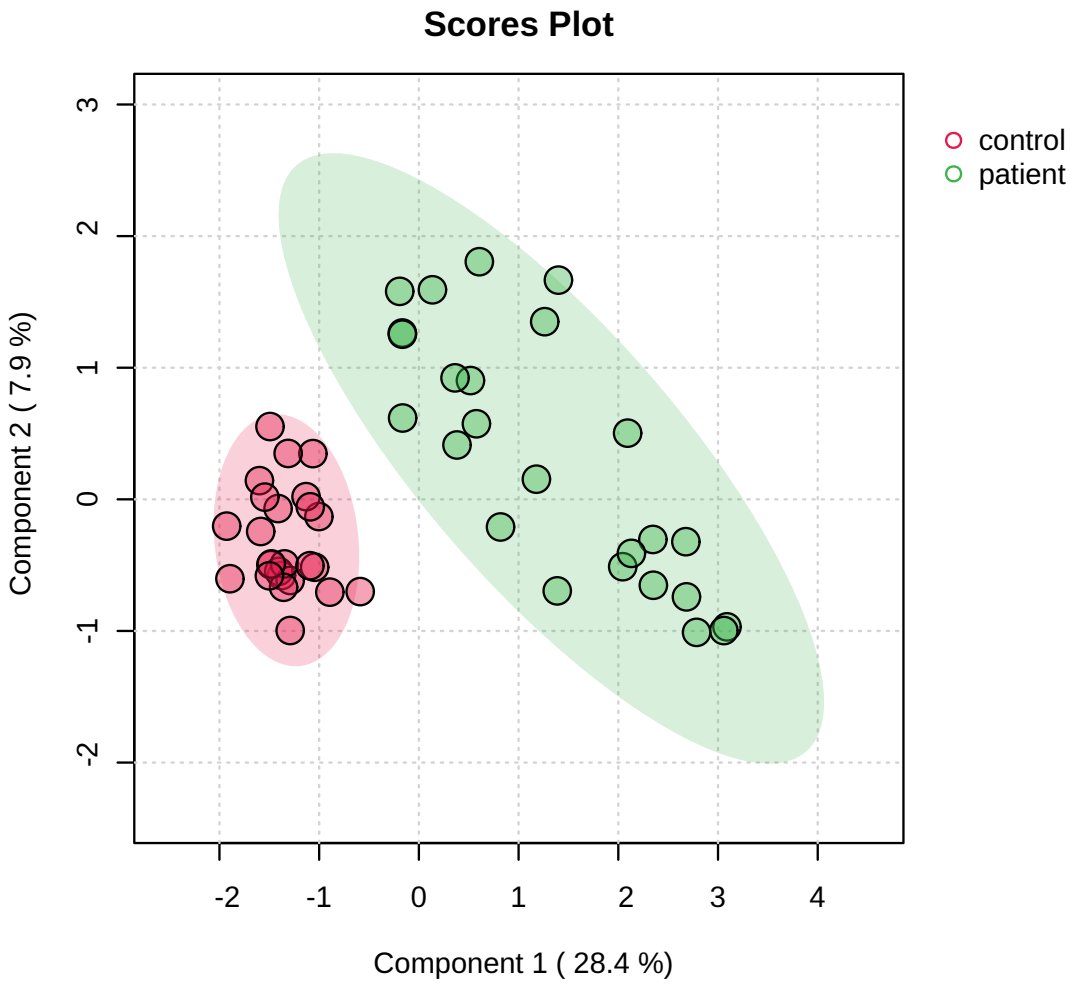

Figure 10: Scores plot between the selected PCs. The explained variances are shown in brackets.

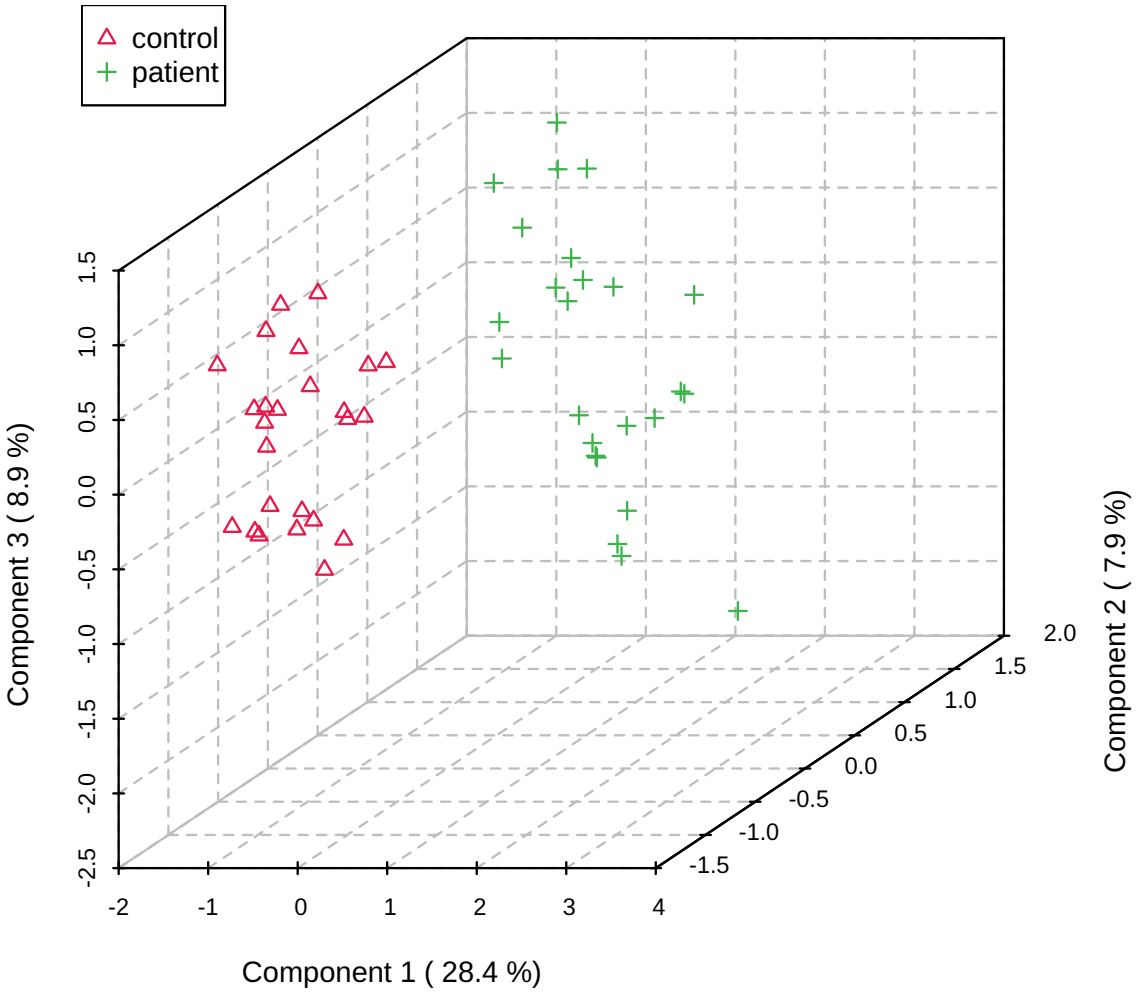

Figure 11: 3D scores plot between the selected PCs. The explained variances are shown in brackets.

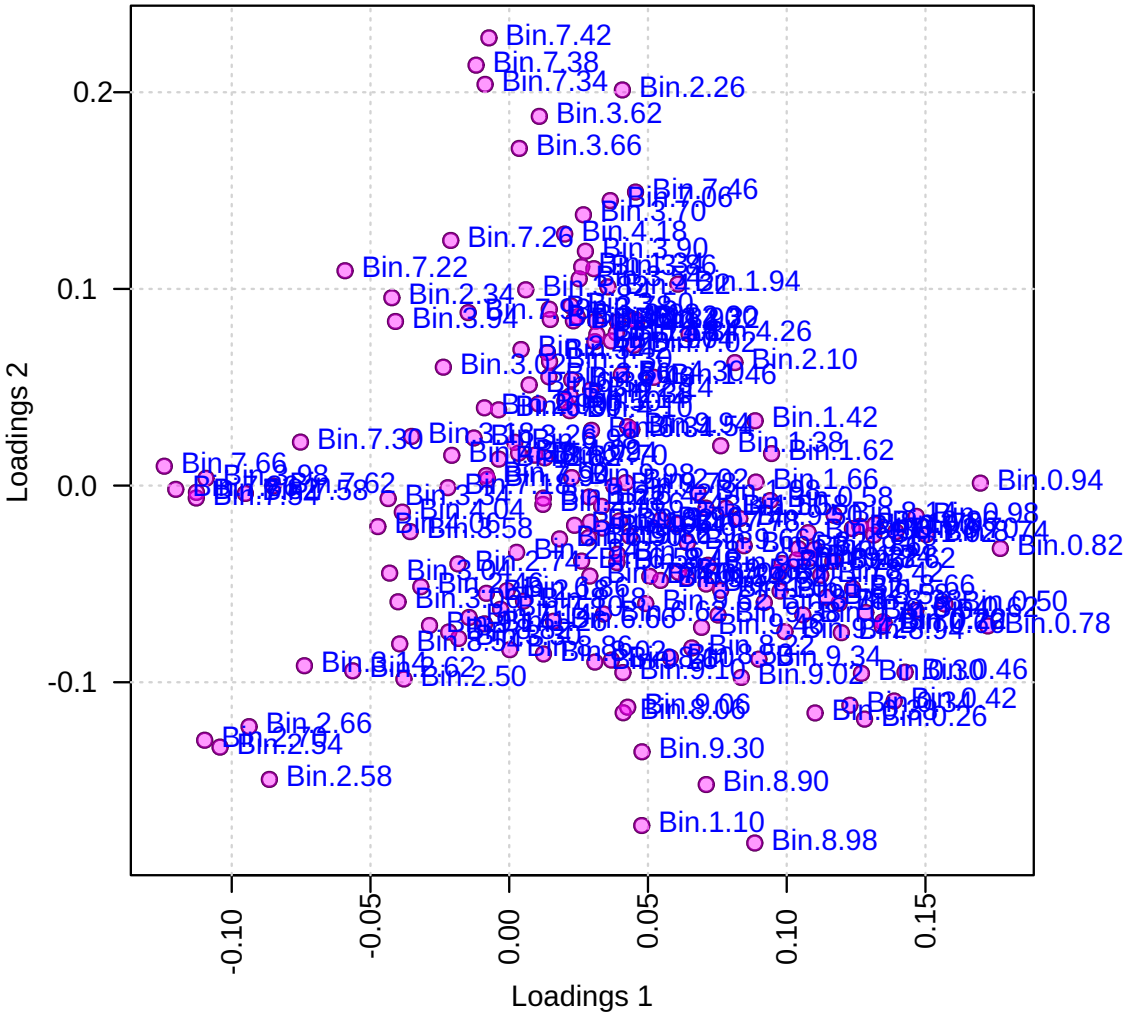

Figure 12: Loadings plot between the selected PCs.

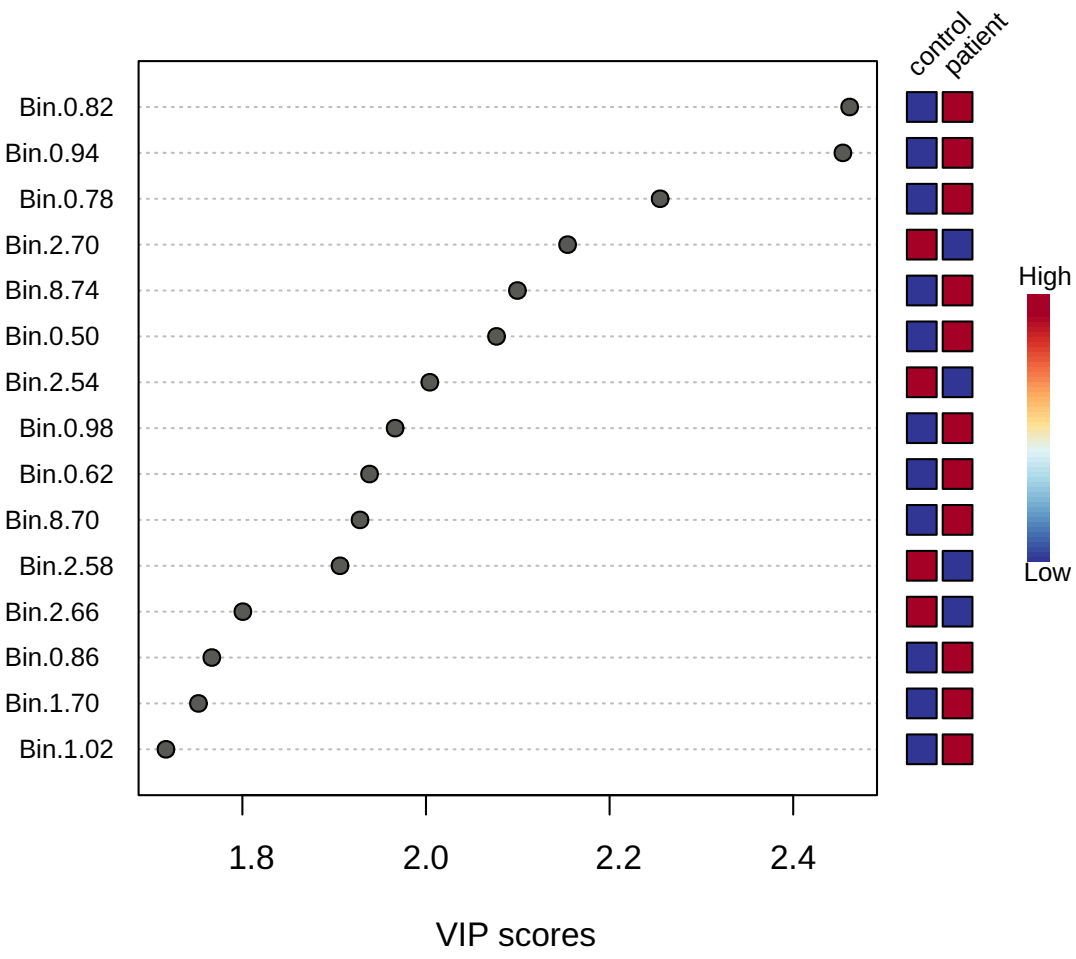

Figure 13: Important features identified by PLS-DA. The colored boxes on the right indicate the relative concentrations of the corresponding metabolite in each group under study.

2.5 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 16 shows clustering the results. Table 4 shows the members in each cluster from K-means analysis.

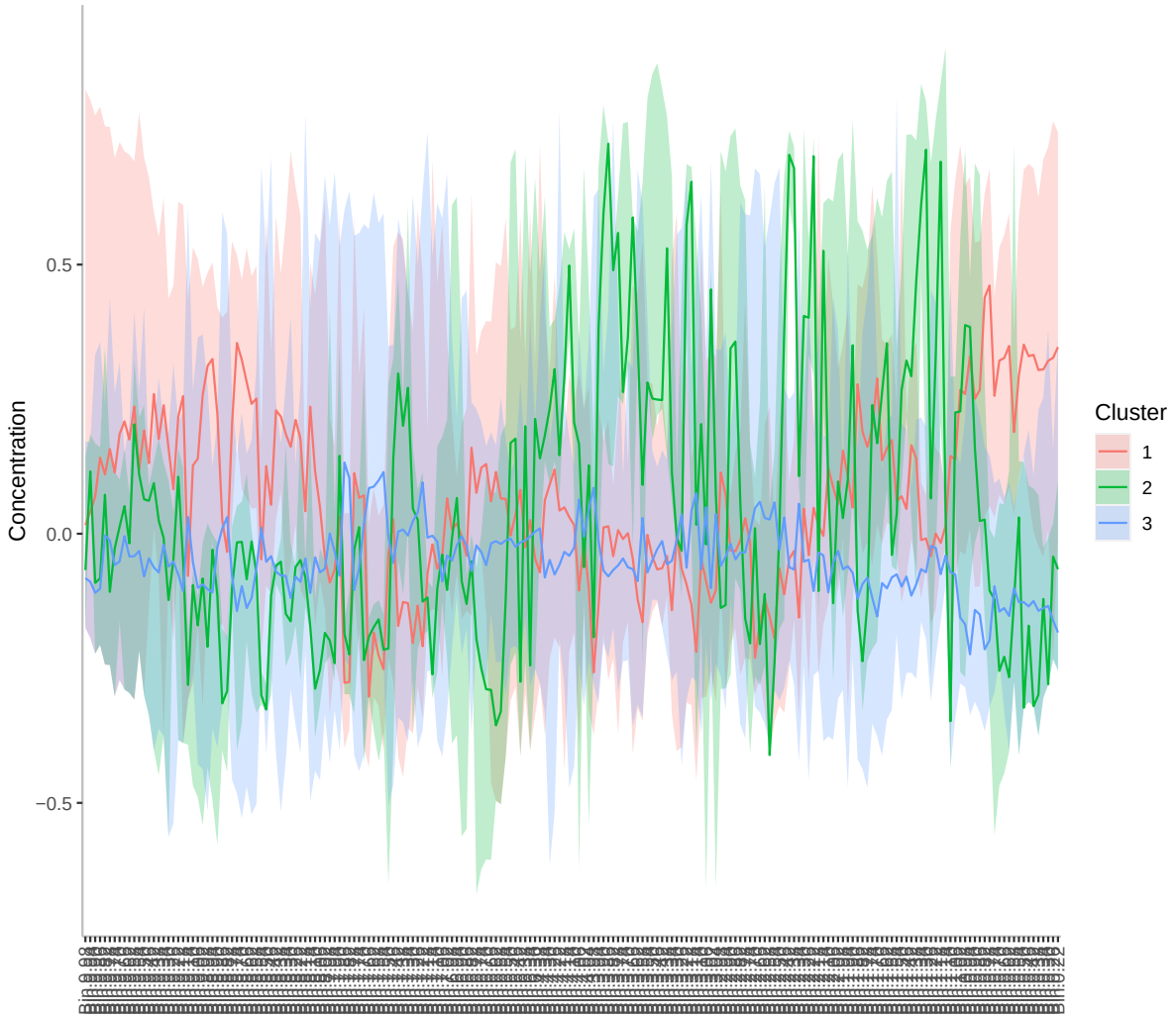

Figure 14: K-means cluster analysis. The x-axes are variable indices and y-axes are relative intensities. The blue lines represent median intensities of corresponding clusters

Table 4: Clustering result using K-means

	Samples in each cluster
Cluster(1)	P012 P014 P027 P034 P038 P041 P042 P064 P065 P085 P086 P089 P013b P100b
Cluster(2)	P002 P080 P113
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 P049 P056 P058 P060 P070 P092 P099

2.6 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[7]. 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 17 shows the SVM classification performance using recursive feature selection. Figure 18 shows the signicant features used by the best classifiers.

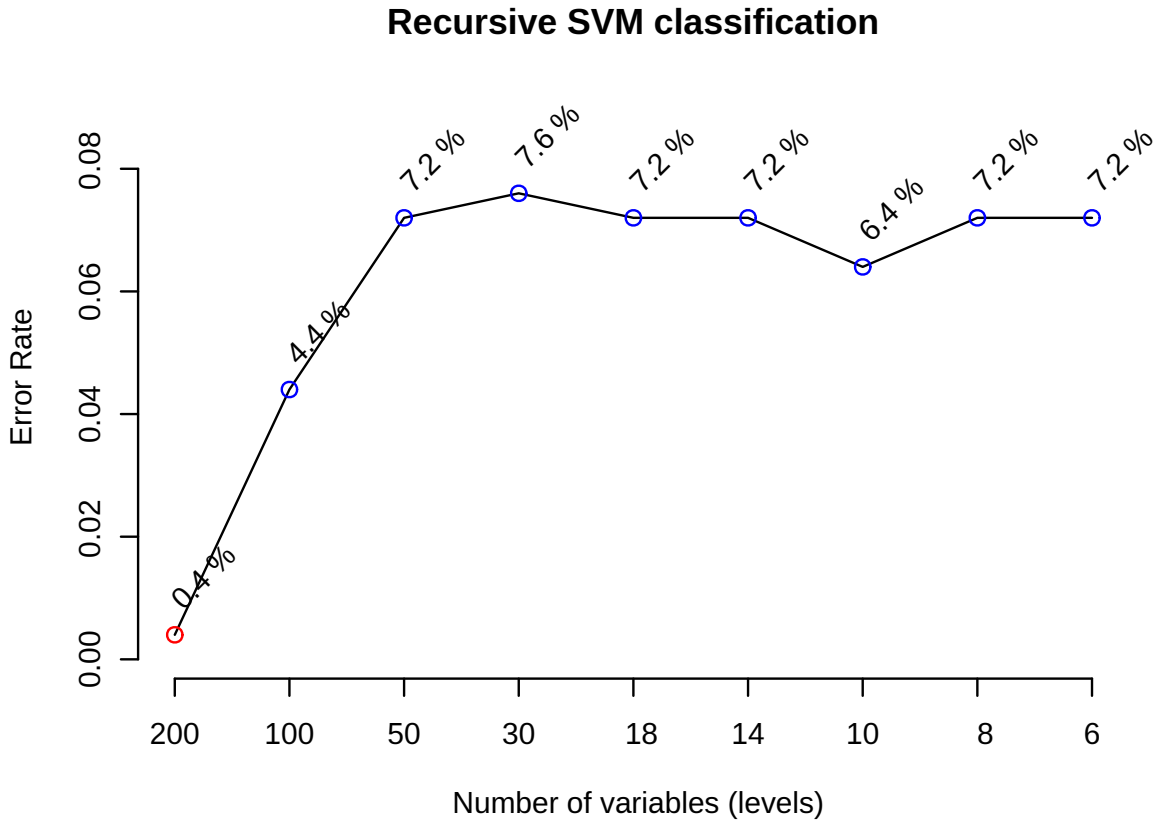

Figure 15: Recursive classification with SVM. The red circle indicates the best classifier.

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

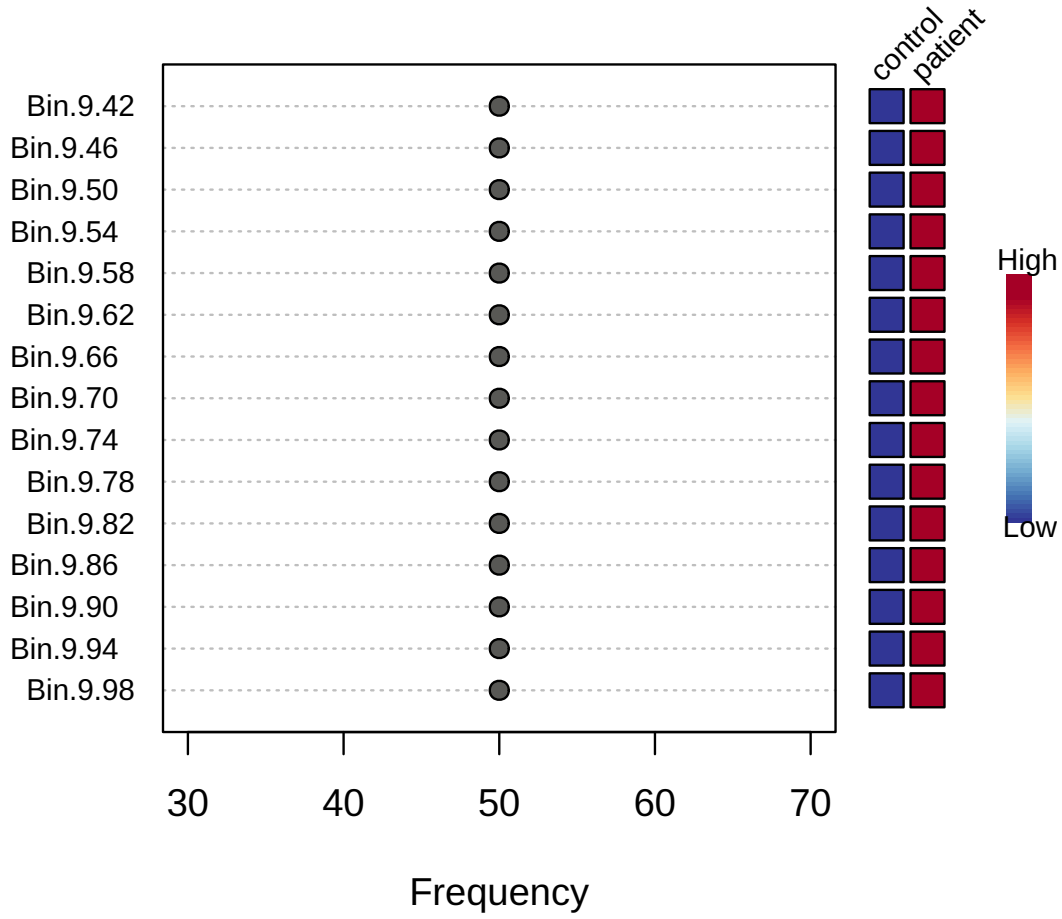

Figure 16: Significant features identified by R-SVM. Features are ranked by their frequencies of being selected in the classifer.

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<-PreparePrenormData(mSet)"
 [7] "mSet<-Normalization(mSet, \"MedianNorm\", \"SrNorm\", \"AutoNorm\", ratio=FALSE, ratioNum=20)"
 [8] "mSet<-PlotNormSummary(mSet, \"norm_0_\", \"png\", 72, width=NA)"
 [9] "mSet<-PlotSampleNormSummary(mSet, \"snorm_0_\", \"png\", 72, width=NA)"
[10] "mSet<-Normalization(mSet, \"MedianNorm\", \"CrNorm\", \"MeanCenter\", ratio=FALSE, ratioNum=20
[11] "mSet<-PlotNormSummary(mSet, \"norm_1_\", \"png\", 72, width=NA)"
[12] "mSet<-PlotSampleNormSummary(mSet, \"snorm_1_\", \"png\", 72, width=NA)"
[13] "mSet<-Normalization(mSet, \"MedianNorm\", \"SrNorm\", \"AutoNorm\", ratio=FALSE, ratioNum=20)"
[14] "mSet<-PlotNormSummary(mSet, \"norm_2_\", \"png\", 72, width=NA)"
[15] "mSet<-PlotSampleNormSummary(mSet, \"snorm_2_\", \"png\", 72, width=NA)"
[16] "mSet<-Normalization(mSet, \"MedianNorm\", \"SrNorm\", \"RangeNorm\", ratio=FALSE, ratioNum=20)
[17] "mSet<-PlotNormSummary(mSet, \"norm_3_\", \"png\", 72, width=NA)"
[18] "mSet<-PlotSampleNormSummary(mSet, \"snorm_3_\", \"png\", 72, width=NA)"
[19] "mSet<-Normalization(mSet, \"MedianNorm\", \"SrNorm\", \"MeanCenter\", ratio=FALSE, ratioNum=20
[20] "mSet<-PlotNormSummary(mSet, \"norm_4_\", \"png\", 72, width=NA)"
[21] "mSet<-PlotSampleNormSummary(mSet, \"snorm_4_\", \"png\", 72, width=NA)"
[22] "mSet<-Normalization(mSet, \"MedianNorm\", \"SrNorm\", \"RangeNorm\", ratio=FALSE, ratioNum=20)
[23] "mSet<-PlotNormSummary(mSet, \"norm_5_\", \"png\", 72, width=NA)"
[24] "mSet<-PlotSampleNormSummary(mSet, \"snorm_5_\", \"png\", 72, width=NA)"
[25] "mSet<-Normalization(mSet, \"MedianNorm\", \"SrNorm\", \"ParetoNorm\", ratio=FALSE, ratioNum=20
[26] "mSet<-PlotNormSummary(mSet, \"norm_6_\", \"png\", 72, width=NA)"
[27] "mSet<-PlotSampleNormSummary(mSet, \"snorm_6_\", \"png\", 72, width=NA)"
[28] "mSet<-Normalization(mSet, \"MedianNorm\", \"SrNorm\", \"RangeNorm\", ratio=FALSE, ratioNum=20)
[29] "mSet<-PlotNormSummary(mSet, \"norm_7_\", \"png\", 72, width=NA)"
[30] "mSet<-PlotSampleNormSummary(mSet, \"snorm_7_\", \"png\", 72, width=NA)"
[31] "mSet<-PCA.Anal(mSet)"
[32] "mSet<-PlotPCAPairSummary(mSet, \"pca_pair_0_\", \"png\", 72, width=NA, 5)"
[33] "mSet<-PlotPCAScree(mSet, \"pca_scree_0_\", \"png\", 72, width=NA, 5)"
[34] "mSet<-PlotPCA2DScore(mSet, \"pca_score2d_0_\", \"png\", 72, width=NA, 1,2,0.95,0,0, \"na\")"
[35] "mSet<-PlotPCALoading(mSet, \"pca_loading_0_\", \"png\", 72, width=NA, 1,2);"
[36] "mSet<-PlotPCABiplot(mSet, \"pca_biplot_0_\", \"png\", 72, width=NA, 1,2)"
[37] "mSet<-PlotPCA3DLoading(mSet, \"pca_loading3d_0_\", \"json\", 1,2,3)"
[38] "mSet<-Ttests.Anal(mSet, F, 0.05, FALSE, TRUE, \"fdr\", FALSE)"
[39] "mSet<-PlotTT(mSet, \"tt_0_\", \"png\", 72, width=NA)"
[40] "mSet<-Ttests.Anal(mSet, F, 0.05, FALSE, TRUE, \"fdr\", FALSE)"
[41] "mSet<-PlotTT(mSet, \"tt_1_\", \"png\", 72, width=NA)"
[42] "mSet<-UpdateLoadingCmpd(mSet, \"Bin.0.82\")"
[43] "mSet<-SetCmpdSummaryType(mSet, \"violin\")"
[44] "mSet<-PlotCmpdSummary(mSet, \"Bin.0.82\",\"NA\",\"NA\", 0, \"png\", 72)"
[45] "mSet<-PlotPCAPairSummary(mSet, \"pca_pair_1_\", \"png\", 72, width=NA, 5)"
[46] "mSet<-PlotCorrHeatMap(mSet, \"corr_1_\", \"png\", 72, width=NA, \"col\", \"pearson\", \"bwm\",
[47] "mSet<-RSVM.Anal(mSet, 10)"
[48] "mSet<-PlotRSVM.Classification(mSet, \"svm_cls_0_\", \"png\", 72, width=NA)"
[49] "mSet<-PlotRSVM.Cmpd(mSet, \"svm_imp_0_\", \"png\", 72, width=NA)"
[50] "mSet<-FC.Anal(mSet, 2.0, 0, FALSE)"
[51] "mSet<-PlotFC(mSet, \"fc_0_\", \"png\", 72, width=NA)"
[52] "mSet<-UpdateLoadingCmpd(mSet, \"Bin.8.70\")"
[53] "mSet<-SetCmpdSummaryType(mSet, \"violin\")"
[54] "mSet<-PlotCmpdSummary(mSet, \"Bin.8.70\",\"NA\",\"NA\", 1, \"png\", 72)"
[55] "mSet<-SanityCheckData(mSet)"
[56] "mSet<-SanityCheckData(mSet)"
```

```
[57] "mSet<-SanityCheckData(mSet)"
[58] "mSet<-GetGroupNames(mSet, \"\")"
[59] "mSet<-UpdateLoadingCmpd(mSet, \"Bin.0.82\")"
[60] "mSet<-SetCmpdSummaryType(mSet, \"violin\")"
[61] "mSet<-PlotCmpdSummary(mSet, \"Bin.0.82\",\"NA\",\"NA\", 2, \"png\", 72)"
[62] "mSet<-Ttests.Anal(mSet, F, 0.01, FALSE, TRUE, \"fdr\", FALSE)"
[63] "mSet<-PlotTT(mSet, \"tt_2_\", \"png\", 72, width=NA)"
[64] "mSet<-Ttests.Anal(mSet, F, 0.005, FALSE, TRUE, \"fdr\", FALSE)"
[65] "mSet<-PlotTT(mSet, \"tt_3_\", \"png\", 72, width=NA)"
[66] "mSet<-Ttests.Anal(mSet, F, 0.001, FALSE, TRUE, \"fdr\", FALSE)"
[67] "mSet<-PlotTT(mSet, \"tt_4_\", \"png\", 72, width=NA)"
[68] "mSet<-UpdateLoadingCmpd(mSet, \"Bin.0.82\")"
[69] "mSet<-SetCmpdSummaryType(mSet, \"violin\")"
[70] "mSet<-PlotCmpdSummary(mSet, \"Bin.0.82\",\"NA\",\"NA\", 3, \"png\", 72)"
[71] "mSet<-PlotCorrHeatMap(mSet, \"corr_2_\", \"png\", 72, width=NA, \"col\", \"pearson\", \"bwm\",
[72] "mSet<-PlotCorrHeatMap(mSet, \"corr_3_\", \"png\", 72, width=NA, \"col\", \"pearson\", \"bwm\",
[73] "mSet<-Kmeans.Anal(mSet, 3)"
[74] "mSet<-PlotKmeans(mSet, \"km_0_\", \"png\", 72, width=NA, \"default\", \"F\")"
[75] "mSet<-PlotClustPCA(mSet, \"km_pca_0_\", \"png\", 72, width=NA, \"default\", \"km\", \"F\")"
[76] "mSet<-PLSR.Anal(mSet, reg=TRUE)"
[77] "mSet<-PlotPLSPairSummary(mSet, \"pls_pair_0_\", \"png\", 72, width=NA, 5)"
[78] "mSet<-PlotPLS2DScore(mSet, \"pls_score2d_0_\", \"png\", 72, width=NA, 1,2,0.95,0,0, \"na\")"
[79] "mSet<-PlotPLS3DScoreImg(mSet, \"pls_score3d_0_\", \"png\", 72, width=NA, 1,2,3, 40)"
[80] "mSet<-PlotPLSLoading(mSet, \"pls_loading_0_\", \"png\", 72, width=NA, 1, 2);"
[81] "mSet<-PlotPLS3DLoading(mSet, \"pls_loading3d_0_\", \"json\", 1,2,3)"
[82] "mSet<-PlotPLS.Imp(mSet, \"pls_imp_0_\", \"png\", 72, width=NA, \"vip\", \"Comp. 1\", 15,FALSE)
[83] "mSet<-SaveTransformedData(mSet)"
[84] "mSet<-PreparePDFReport(mSet, \"guest12006273676888371586\")\n"
```

The report was generated on Wed Mar 20 04:12:33 2024 with R version 4.3.2 (2023-10-31), OS system: Linux, version: -Ubuntu SMP Tue Jan 9 15:25:40 UTC 2024 .