MetCleaning v1.0.0 Instruction

Xiaotao Shen ([shenxt@sioc.ac.cn](mailto:shenxt@sioc.ac.cn)); Zhengjiang Zhu

November 24th, 2016

#### Introduction

##### MetCleaning package provides an integrated and automatic pipeline for data cleaning and statistical analysis of large scale mass spectrometry (MS) based-metabolomic data. It includes missing value (MV) filtering and imputation, zero value filtering, data normalization, data integration, data quality assessment, univariate statistical analysis, multivariate statistical analysis such as PCA and PLS-DA, potential marker selection and show. This document describes how to use the functions included in MetCleaning utilizing demo data.

D:\study\R\my git\MetCleaning\vignettes\workflow for instruction.tif

Figure 1. The workflow of *MetCleaning* andR functionsfor data cleaning and statistical analysis.

#### Installation of *MetCleaning*

##### **MetCleaning can be installed from Github.**

if(!require(devtools)) {  
 install.packages("devtools")  
}  
 devtools::install\_github("jaspershen/MetCleaning",ref="version1.0.5)

library("MetCleaning")

#### Help document of *MetCleaning*

help(package = "MetCleaning")

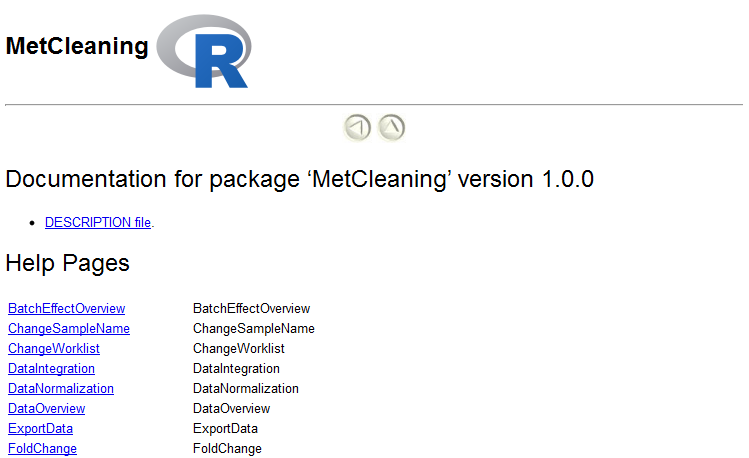


Figure 2. The help document of *MetCleaning*.

#### Organization of files

##### The metabolomic dataset from initial processing software (such as XCMS and MZmine) and sample information need to be placed in a folder：

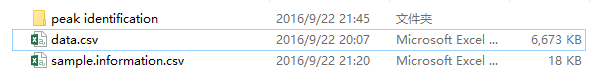


Figure 3. The data organization for *MetCleaning*.

##### 1. "data.csv" is the metabolomic dataset you want to process. Rows are features and columns are feature abundance of samples and tags of features. The tags of features must contain "name" (feature name), "mz" (mass to change ratio) and "rt" (retention time). Other tags of features are optional, for example "isotopes" and "adducts". The name of sample can contain ".", but cannot contain "-" and space. And the start of sample name cannot be number. **For example, "A210.a" and "A210a" are valid, and "210a" or "210-a" are invalid.**

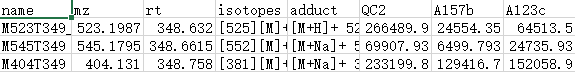


Figure 4. The metabolomic dataset for *MetCleaning*.

##### 2. "sample.information.csv" is sample information for metabolomic dataset. Column 1 is "sample.name" which is the names of subject and QC samples. Please confirm that the sample names in "sample.information.csv" and "data.csv" are completely same. Column 2 is "injection.order" which is the injection order of QC and subject samples. Column 3 is "class", which is used to distinguish "QC" and "Subject" samples. Column 4 is "batch" to provide acquisition batch information for samples. Column 5 is "group", which is used to label the group of subject sample, for example, "control" and "case". The "group" of QC samples is labeled as "QC".

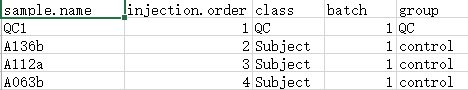


Figure 5. The sample information for *MetCleaning*.

##### 3. "peak identification" is optional. You can use the identification information of the metabolomic dataset to do metabolite identification according to mz and RT. Please create a folder named as "peak identification" and place identification information in it. **Please note that each name of identification information need to contain "ms2" like figure 6 shows**.

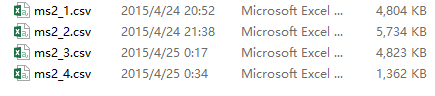


Figure 6. The identification information for *MetCleaning*.

#### Data cleaning

##### Use the demo data of MetCleaning as an example.

##### **1. Set the folder where you place files as work directory.**

##demo data of MetCleaning  
data(data, package = "MetCleaning")  
data(sample.information, package = "MetCleaning")  
data(new.group, package = "MetCleaning")

data(ms2\_1, package = "MetCleaning")  
data(ms2\_2, package = "MetCleaning")  
data(ms2\_3, package = "MetCleaning")  
  
##set demo work directory  
dir.create("Demo for MetCleaning")  
setwd("Demo for MetCleaning")  
path <- file.path(getwd(), "peak identification")  
dir.create(path)  
  
##write files

#metabolomic dataset, sample information and identification results  
write.csv(data, "data.csv", row.names = FALSE)  
write.csv(sample.information, "sample.information.csv", row.names=F)  
write.csv(new.group, "new.group.csv", row.names = F)  
write.csv(ms2\_1, file.path(path, "ms2\_1.csv"), row.names = F)  
write.csv(ms2\_2, file.path(path, "ms2\_2.csv"), row.names = F)  
write.csv(ms2\_3, file.path(path, "ms2\_3.csv"), row.names = F)

##### **2. Import data into R.**

met.data <- ImportData(data = "data.csv",  
 sample.information = "sample.information.csv",  
 polarity = "positive")  
##save data  
met.data.raw <- met.data  
save(met.data.raw, file = "met.data.raw") ##help for *ImportData* ?*ImportData*

###### "met.data" is a standard class in *MetCleaning*. It contains the metabolomic dataset, sample information. You can see the information of it through typing its name.

met.data

##### **3. Filter and impute missing values.** (If there are no missing values, this step can be skipped)

##### Use MZoverview function to investigate the missing values in metabolomic dataset.

MZoverview(MetFlowData = met.data, path = "MV overview", what = "mv") ##help for *MZoverview* ?*MZoverview*

##### A new folder named as "MV overview" is created. "MV overview" contains the missing value information of each batch:

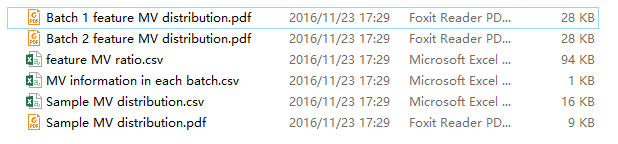


Figure 7. The MV information of metabolomic dataset.

##### "Batch x feature MV distribution.pdf" give the missing value distribution for features of each batch:

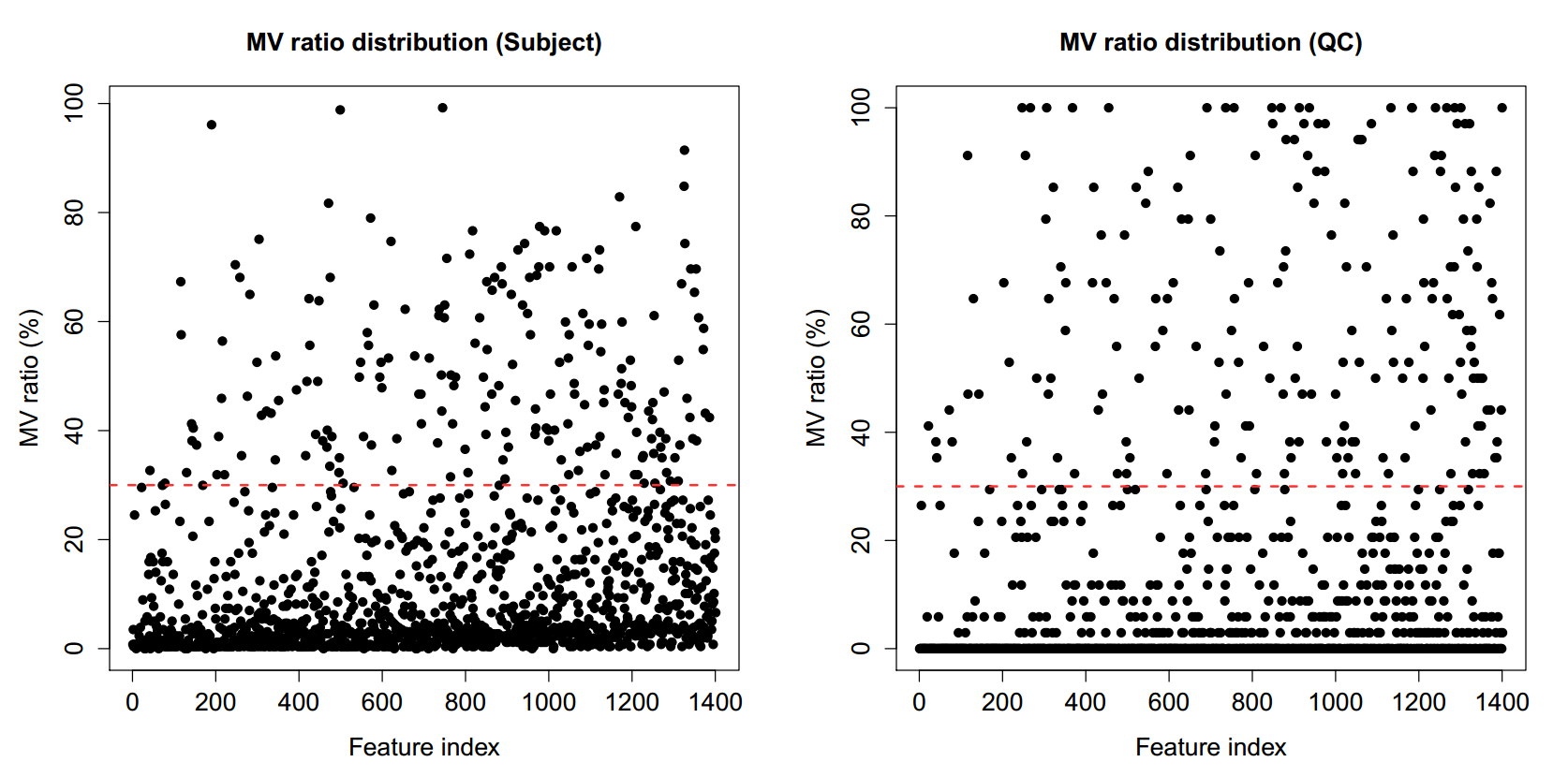


Figure 8. The distributions of MV ratios in features in subject and QC sample, respectively.

##### "Sample MV distribution.pdf" give the missing value distribution for samples:

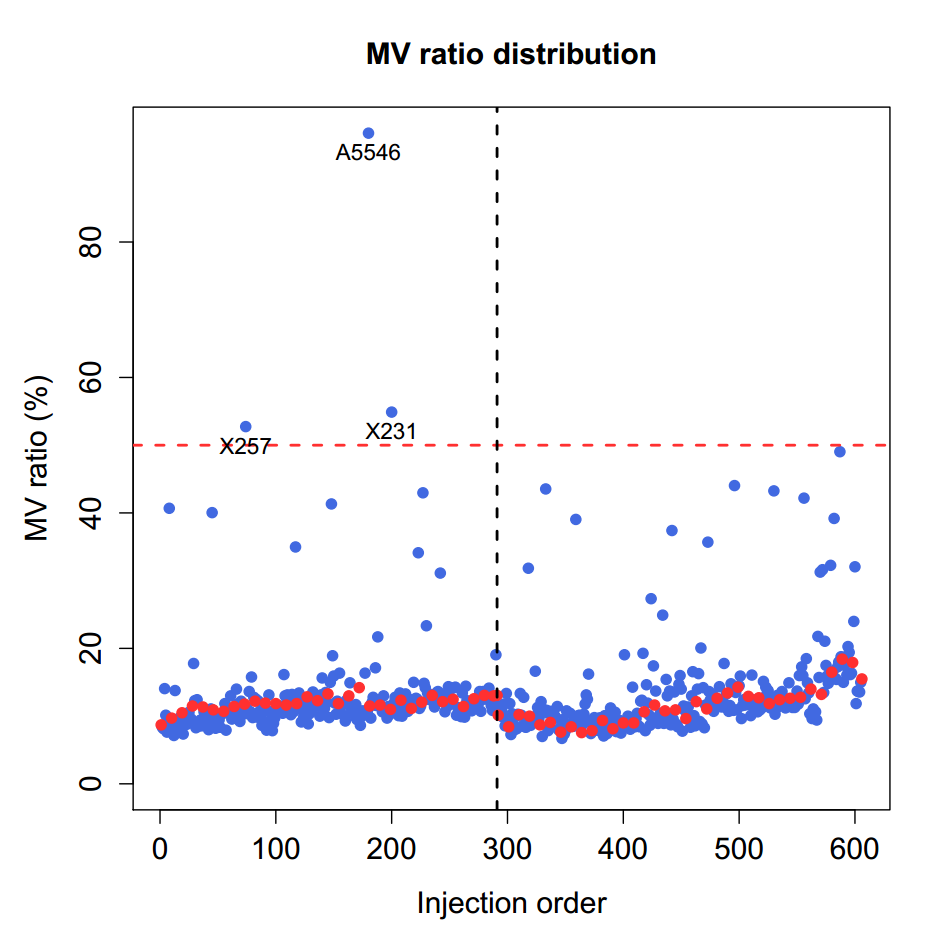


Figure 9. The distributions of MV ratios in samples. There are there samples are outliers.

##### Use MZfilter to filter missing values in metabolomic dataset.

met.data <- MZfilter(MetFlowData = met.data,  
 obs.per.cutoff = 0.5,  
 var.per.cutoff = 0.5, what = "mv", path = "MV filter")  
#save data  
met.data.mv.filter <- met.data  
save(met.data.mv.filter, file = "met.data.mv.filter") ##help for *MZfilter* ?*MZfilter*

##### MZfilter filter missing values according to the missing value ratios in each feature and sample. If the missing value ratio of one feature in subject or QC samples is larger than the cutoff value (default is 50%), this feature would be removed from the data. And if the missing value ratio of one sample is larger than the cutoff value (default is 50%), this sample also should be removed. The removing of sample is optional.

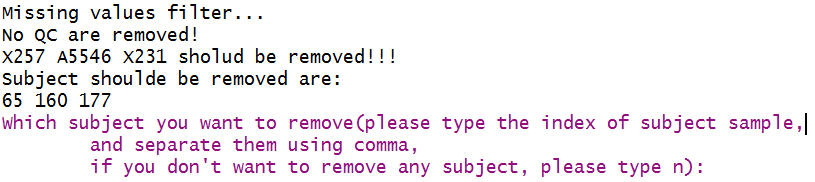


Figure 10. The running result of *MZfilter*. In this example, there are no QC samples are removed. And three subject sample should be removed. You can type the index of samples which you want to remove. And if you don’t want to remove any samples, you should type n and then clink Enter key.

##### Use MVimputation to impute missing values in metabolomic dataset.

met.data <- MVimputation(MetFlowData = met.data,  
 #MV imputation method  
 imputation.method = "knn",  
 #knn parameters  
 k = 10)  
#save met.data  
met.data.mv.imputation <- met.data  
save(met.data.mv.imputation, file = "met.data.mv.imputation") ##help for *MVimputation* ?*MVimputation*

##### **4. Filter zero values****.**

##### Use MZoverview function to investigate the zero values in metabolomics dataset.

MZoverview(MetFlowData = met.data, path="Zero overview", what="zero")

##### This is same with MV overview.

##### Use MZfilter function to filter zero values:

met.data <- MZfilter(MetFlowData = met.data,  
 obs.per.cutoff = 0.5,  
 var.per.cutoff = 0.5, what = "zero",  
 path = "Zero filter")  
#save data  
met.data.zero.filter <- met.data  
save(met.data.zero.filter, file = "met.data.zero.filter")

##### **5. Identify features according to identification results and HMDB database.**

#Using the MS2 identification information of QC  
met.data <- PeakIdentification(MetFlowData = met.data,  
 #parameters for matching  
 mz.tolerance = 30,  
 rt.tolerance = 180) ##help for *PeakIdentification* ?*PeakIdentification*  
#save data  
met.data.peak.iden <- met.data  
save(met.data.peak.iden, file = "met.data.peak.iden")  
  
#Using the HMDB database for identification according to mass  
met.data <- MassIdentification(MetFlowData = met.data,  
 mass.tolerance = 30,  
 polarity = "positive")  
#save data  
met.data.mass.iden <- met.data  
save(met.data.mass.iden, file = "met.data.mass.iden") ##help for *MassIdentification* ?*MassIdentification*

##### A new folder named "matching result" is created in "peak identification", it contains the identification result, and the identification results are also added in to tags.



Figure 11. The running result of peak identification.

##### "identification.information.txt" record some identification information.

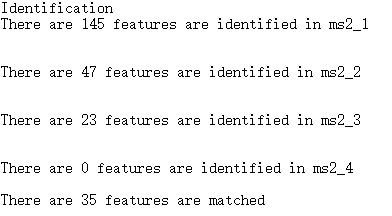


Figure 12. The information of peak identification.

##### MassIdentification utilize accurate mass to identify features in HMDB database. The identification results are added in tags. "HMDB.match.result" is the matching result, it contains the "HMDB.name", "HMDB.ID", "Formula", "Mass", "HMDB.adduct" and "mz.error". "HMDB.identification" is the identification result with the least m/z error.

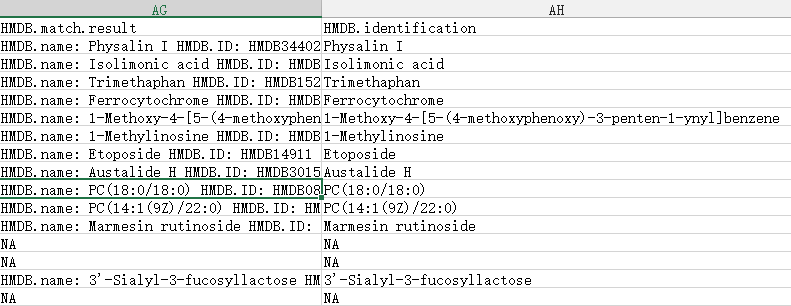


Figure 13. The HMDB identification results in metabolomic dataset.

##### **6. Detect QC sample outliers (PCA score plot).**

##### Use QCOutlierFilter function to filter QC sample outliers.

met.data <- QCOutlierFilter(MetFlowData = met.data,  
 CI = 0.95,  
 path = "QC outlier filter")  
met.data.qc.outlier.filter <- met.data save(met.data.qc.outlier.filter, file = "met.data.qc.outlier.filter") ##help for *QCOutlierFilter* ?*QCOutlierFilter*

##### A new folder named as "QC outlier finder" is created, it contains QC outliers for each batch. The QC samples which are outside the 95% confidence interval in PCA score plot are considered as outliers, and are labeled as red color.

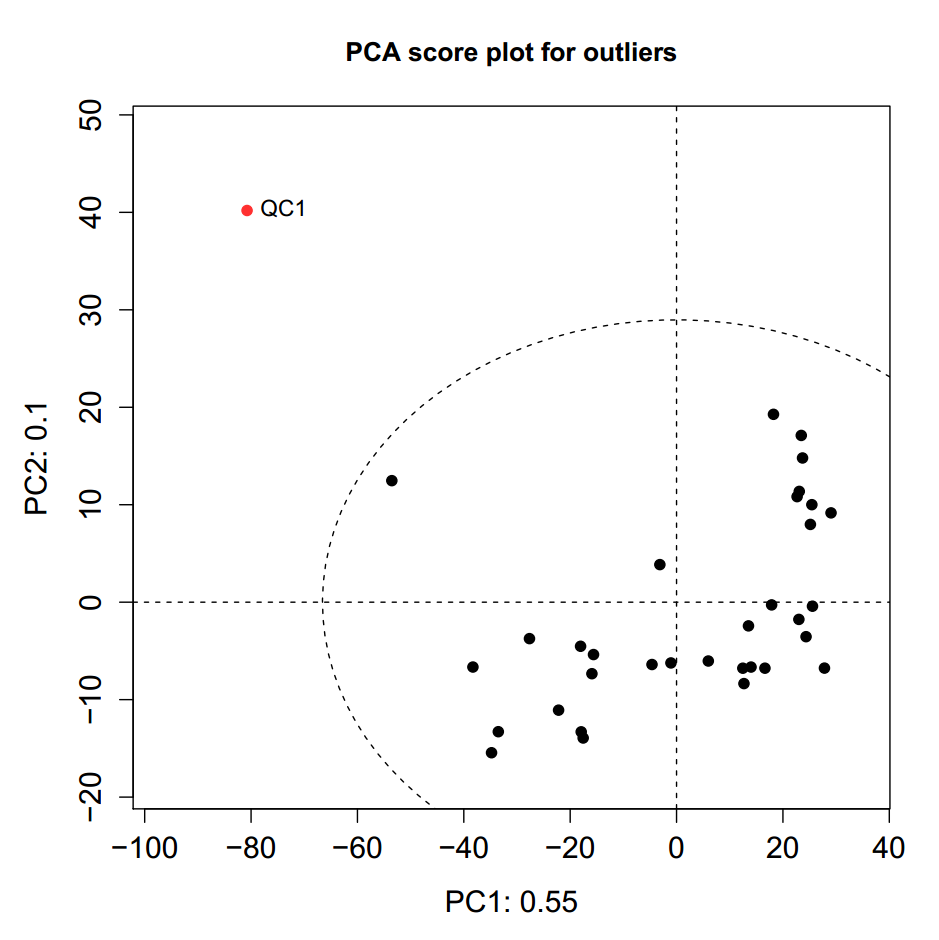


Figure 14. The PCA score plot of QC samples for each batch.

##### **7. Normalize metabolomic dataset.**

met.data <- DataNormalization(MetFlowData = met.data,  
 method = "svr",  
 threads = 2)  
#save data  
met.data.nor <- met.data  
save(met.data.nor, file = "met.data.nor") ##help for *DataNormalization* ?*DataNormalization*

##### A new folder named "Normalization result" is created, it contains the normalization for each batch:

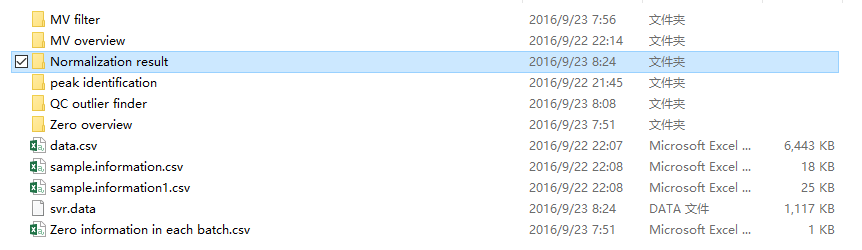


Figure 15. The running results of *DataNormalization*.

##### Data normalization for each batch:

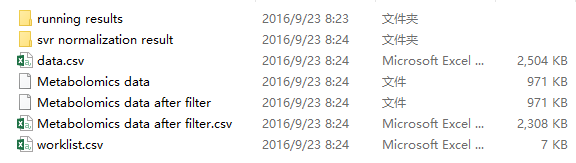


Figure 16. Data normalization results for each batch.

##### **8. Detect subject sample outliers.**

##### Use SubjectOutlierFilter function to filter outliers in subject samples.

met.data <- SubjectOutlierFilter(MetFlowData = met.data,  
 CI = 0.95,  
 path = "Subject outlier filter")  
met.data.subject.outlier.filter <- met.data save(met.data.subject.outlier.filter, file = "met.data.subject.outlier.filter") ##help for *SubjectOutlierFilter* ?*SubjectOutlierFilter*

##### A new folder named as "Subject outlier finder" is created, it contains outliers in each batch. The subject samples which are outside 95% confidence interval in PCA score plot are considered as outliers and labeled as red color.

##### **9. Integrate several batch datasets as an integrative dataset.** (If there is only one batch, this step can be skipped)

met.data <- DataIntegration(MetFlowData = met.data)  
#save data  
met.data.integration <- met.data  
save(met.data.integration, file = "met.data.integration")

##### **10. Investigate the batch effect.** (If there is only one batch, this step can be skipped)

BatchEffectOverview(MetFlowData.before = met.data.zero.filter,  
 MetFlowData.after = met.data.integration,  
 path = "Batch effect") ##help for *BatchEffectOverview* ?*BatchEffectOverview*

##### A folder named as "Batch effect" is created. "Before/After Batch effect in QC/Subject.pdf" utilize PCA score plot and QC total intensity distribution to show the batch effect in QC and subject samples both before and after cleaning.

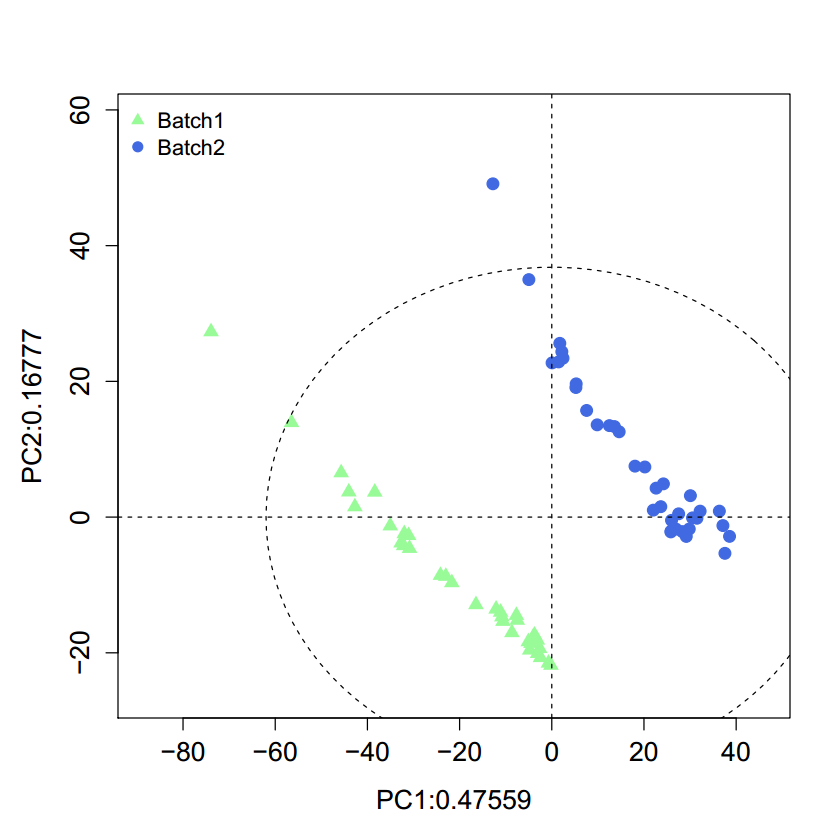
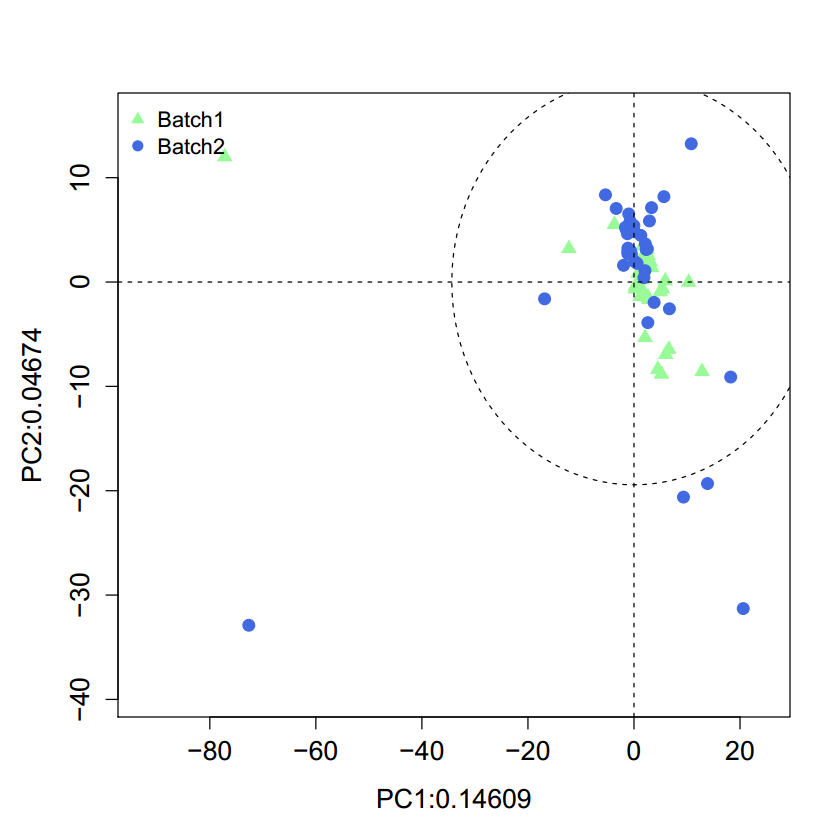
 

Figure 17. Batch effect in QC samples in both before and after data cleaning which utilizing PCA score plot.

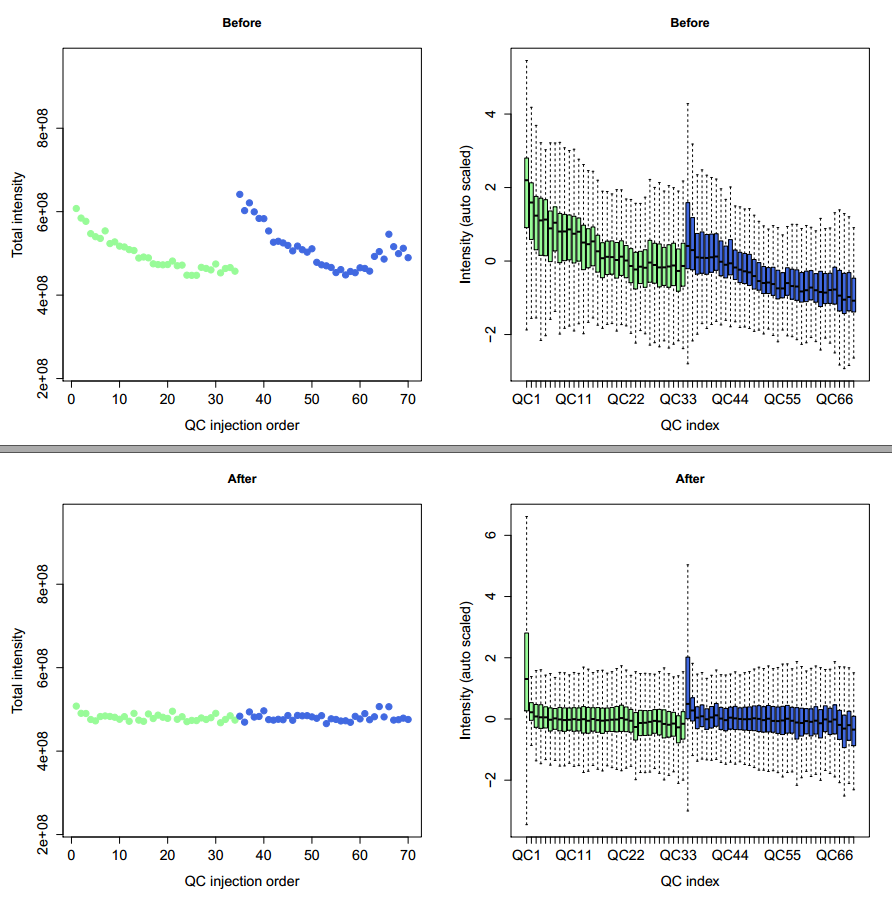


Figure 18. Batch effect in QC samples in both before and after data cleaning which utilizing the distributions of total intensity and z-score intensity for QC samples.

##### **11. Feature distribution in different batches.** (If there is only one batch, this step can be skipped)

MetabolitePlot(MetFlowData.before = met.data.zero.filter,  
 MetFlowData.after = met.data,  
 path = "metabolite plot") ##help for *MetabolitePlot* ?*MetabolitePlot*

##### A folder named as "metabolite plot" is created. Metabolite plot for each feature before and after data cleaning are provided.

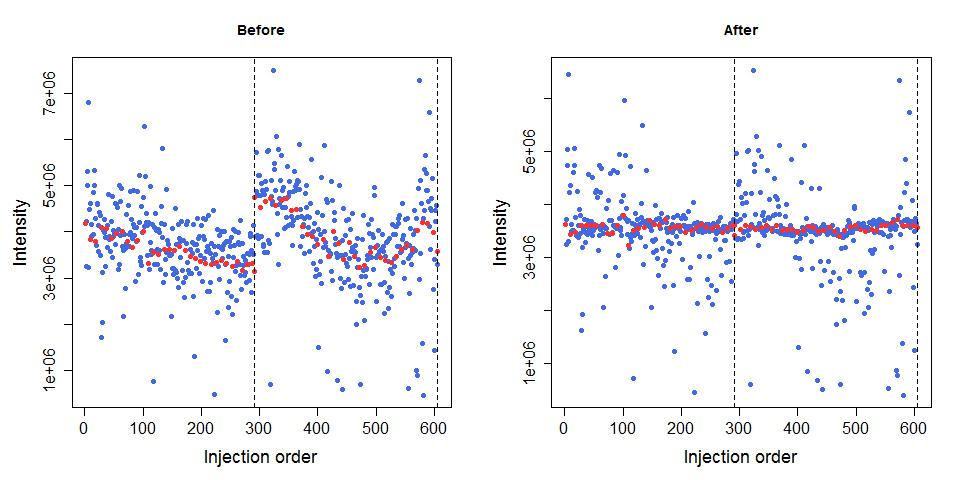


Figure 19. An example for one feature distribution in different batches.

##### **12. Investigate RSD distribution.**

RSDoverview(MetFlowData.before = met.data.zero.filter,  
 MetFlowData.after = met.data.integration,  
 path = "RSD overview") ##help for *RSDoverview* ?*RSDoverview*

##### A folder named as "RSD overview" is created. "RSD distribution in all batches.pdf" is the RSD distribution for each feature in all batches. "RSD distribution in different batch.pdf" is the RSD distribution for each feature in each batch.



Figure 20. The running results of *RSDoverview*.

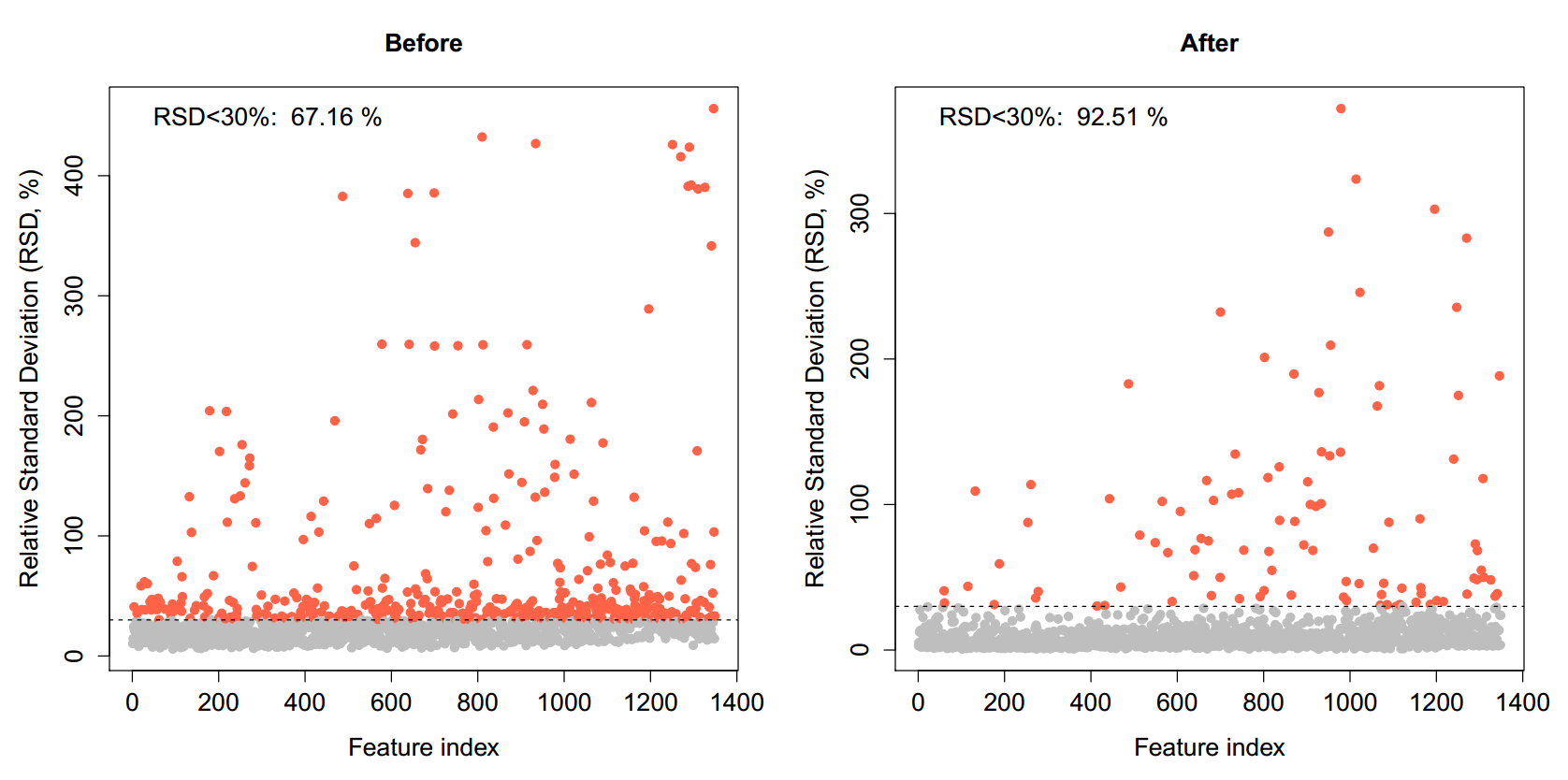
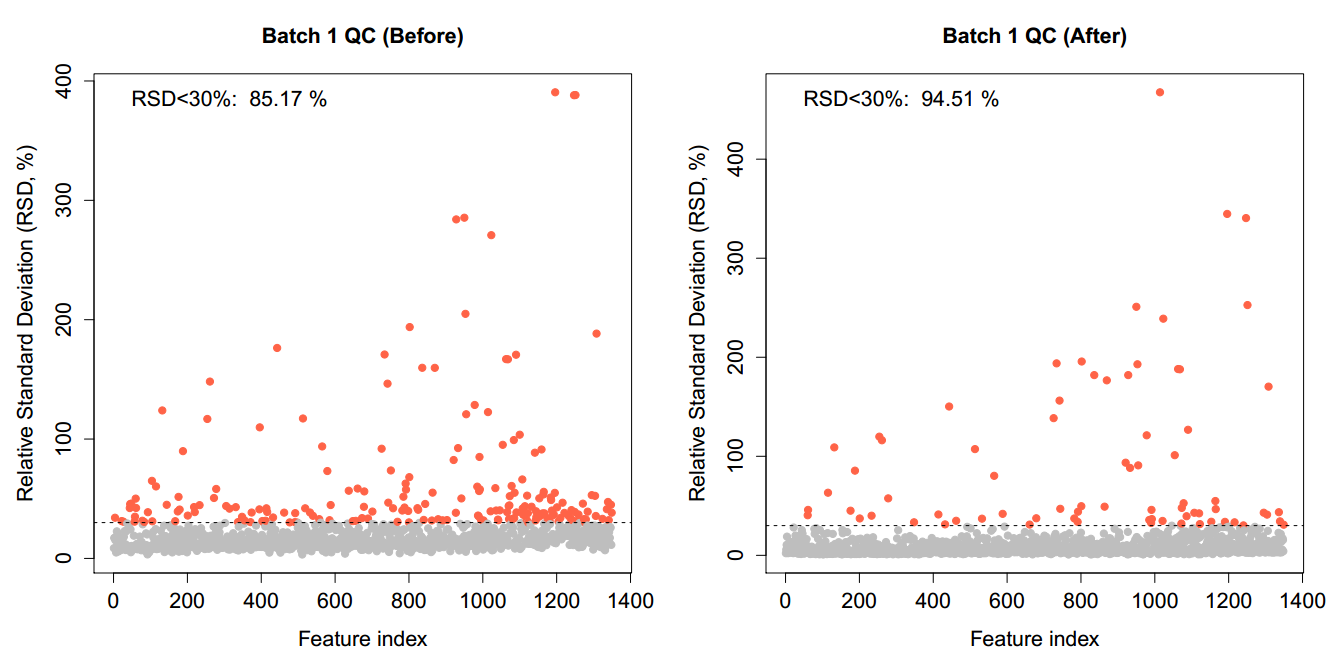
 

Figure 21. The distribution of feature RSDs in QC samples in each batch and all batches.

##### **13. Data overview.**

DataOverview(MetFlowData = met.data,  
 path = "Data overview") ##help for *DataOverview* ?*DataOverview*

##### "Data overview.txt" record some base information of data.

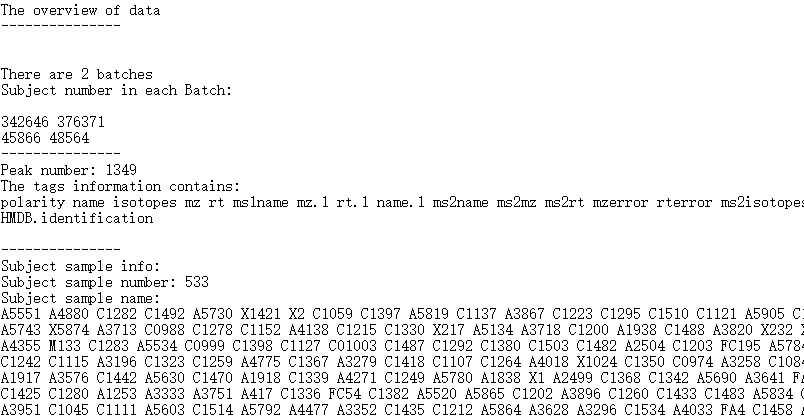


Figure 22. The information of metabolomic dataset.

##### "Data overview\_RT vs mz vs intensity.pdf" shows the distribution of features:

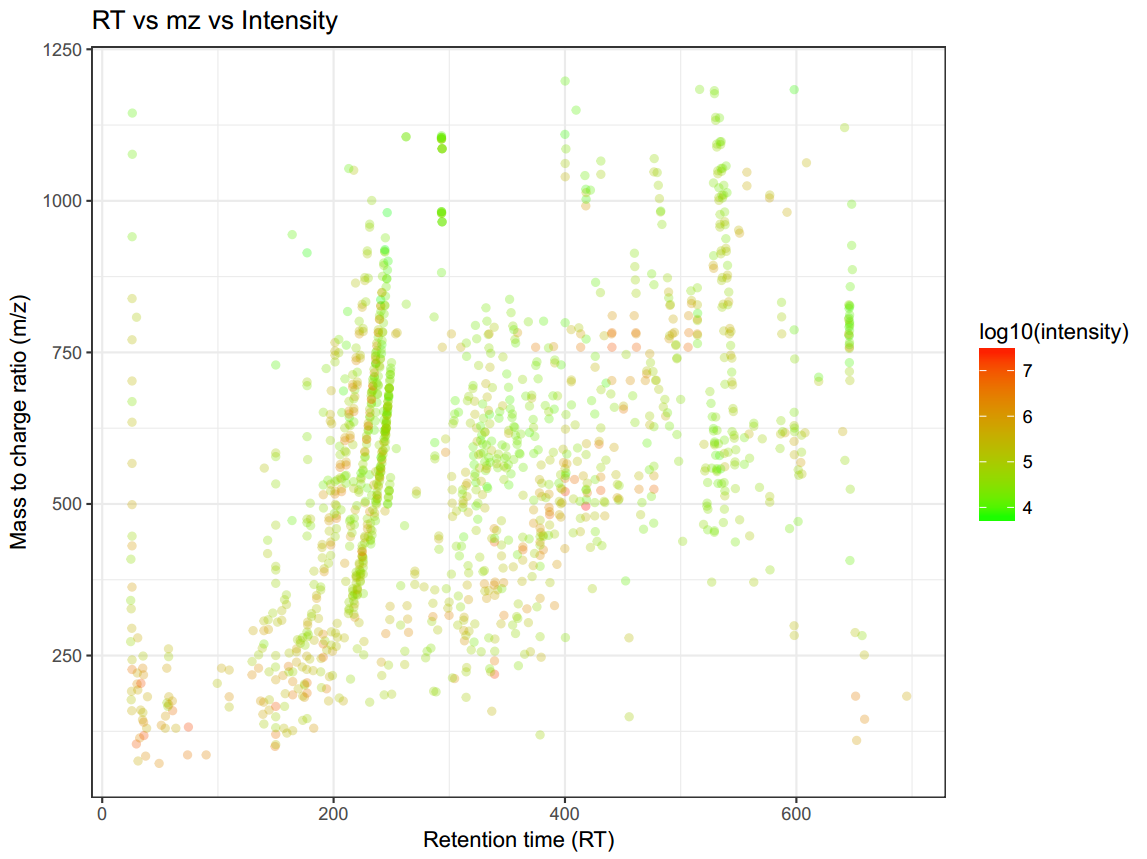


Figure 23. The distribution of features in m/z and RT.

##### **14. Output data as csv files.**

ExportData(MetFlowData = met.data,  
 data.name = "data\_after\_pre",  
 subject.info.name = "subject.info",  
 qc.info.name = "qc.info",  
 path = NULL) #save data  
met.data.after.pre <- met.data   
save(met.data.after.pre, file = "met.data.after.pre") ##help for *ExportData* ?*ExportData*

##### Data are outputted as "data\_after\_pre.csv", "subject.info.csv" and "qc.info.csv":

#### Statistical analysis

##### The metabolomic data after data cleaning can be used to do statistical analysis.

##### **1. Filter features who’s RSDs larger than the threshold.** RSDfilter filter the features who’s RSD in QC samples larger than the threshold.

met.data <- RSDfilter(MetFlowData = met.data.after.pre, rsd.cutoff =30) met.data.rsd.filter <- met.data ###save data save(met.data.rsd.filter, file = "met.data.rsd.filter") ##help for *RSDfilter* ?*RSDfilter*

##### **2. Give new group information.** ReChangeGroup can be used to change new group information in sample information. Change the group information in "sample.information.csv" and name it as "new.group.csv", then place it in data cleaning folder. The samples which you want to remove from the dataset, you can set they group information as NA.

met.data <- ReChangeGroup(MetFlowData = met.data, new.group = "new.group.csv") #save data met.data.new.group <- met.data save(met.data.new.group, file = "met.data.new.group") ##help for *ReChangeGroup* ?*ReChangeGroup*

##### **3. PCA analysis**

PCAanalysis(MetFlowData = met.data,  
 log.scale = FALSE,  
 QC = TRUE,  
 scale.method = "auto",  
 path = "PCA analysis", xlim1 = c(-40, 40), ylim1 = c(-40, 40)) ##help for *PCAanalysis* ?*PCAanalysis*

##### A new folder named as "PCA analysis" is created. "PCA score plot.pdf" is the PCA score plot.

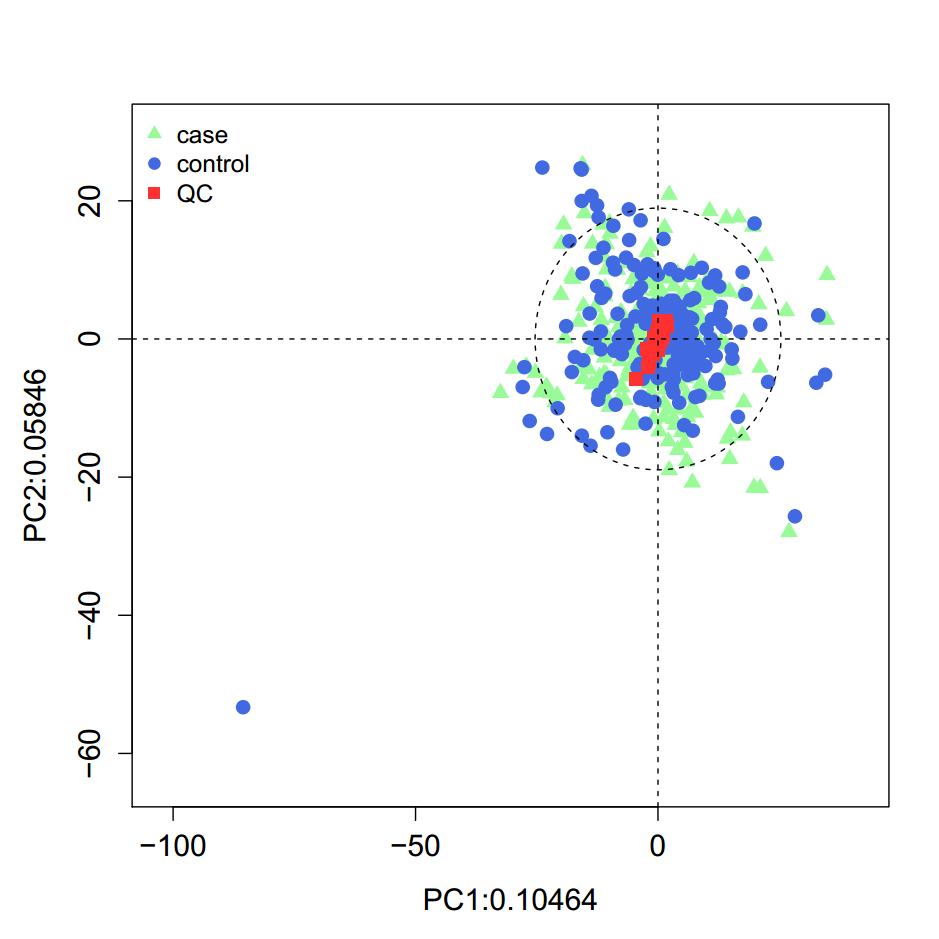


Figure 24. PCA score plot of *PCAanalysis*.

##### **4. PLS-DA analysis**

PLSanalysis(MetFlowData = met.data,  
 #used data  
 log.scale = FALSE,  
 scalemethod = "auto",  
 plsmethod = "plsr",  
 path = "PLS analysis", xlim1 = c(-40, 40), ylim1 = c(-40, 40))  
  
##Add vip value into tags  
 load(file.path("PLS analysis", "vip"))  
 vip <- apply(vip, 2, mean)  
 tags <- met.data[["tags"]]  
 tags <- data.frame(tags, vip)  
 met.data[["tags"]] <- tags  
##save data  
met.data.vip <- met.data  
save(met.data.vip, file = "met.data.vip") ##help for *PLSanalysis* ?*PLSanalysis*

##### Analysis results are placed in "PLS analysis" folder:

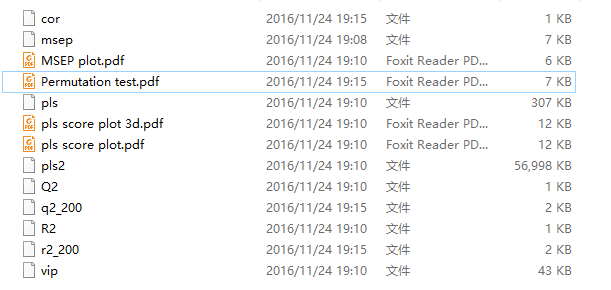


Figure 25. The running results of PLS analysis.

##### When you run PLSanalysis function, it give a message to show how many components you want to see:

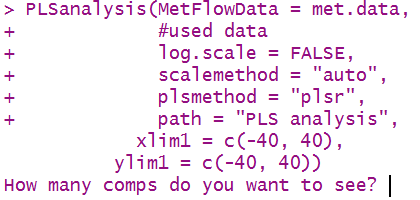


Figure 26. Choose the component number in PLS-DA analysis.

##### You can type 10, and type Enter key.

#### A MSEP plot of cross validation would show:

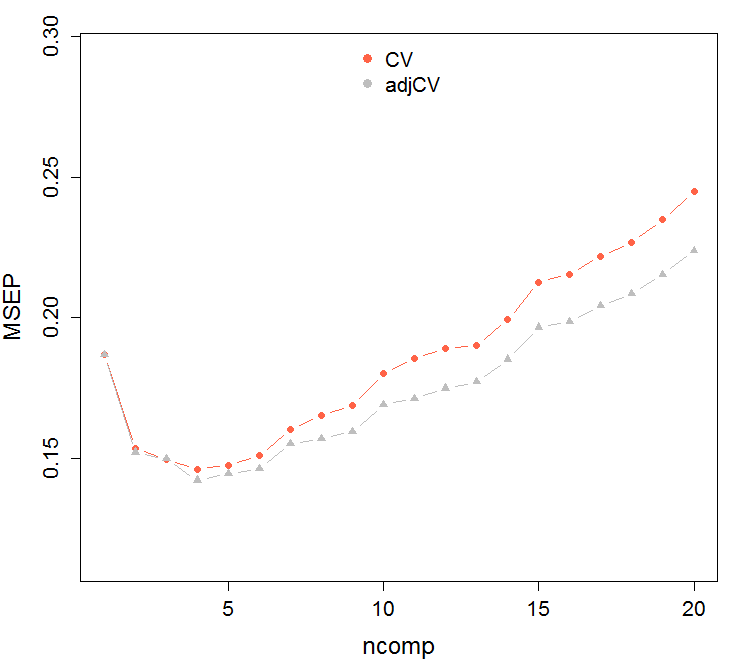


Figure 27. MSEP plot in PLS-DA analysis.

##### Then type the component number with the smallest MSEP. In this example, the appropriate component number is five.

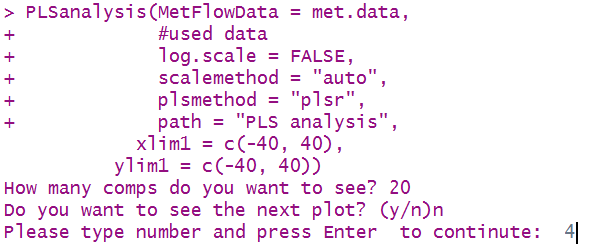


Figure 28. Choose the appropriate component number in PLS-DA analysis.

##### "pls score plot.pdf" is the PLS score plot and "Permutation test.pdf" is the permutation test.

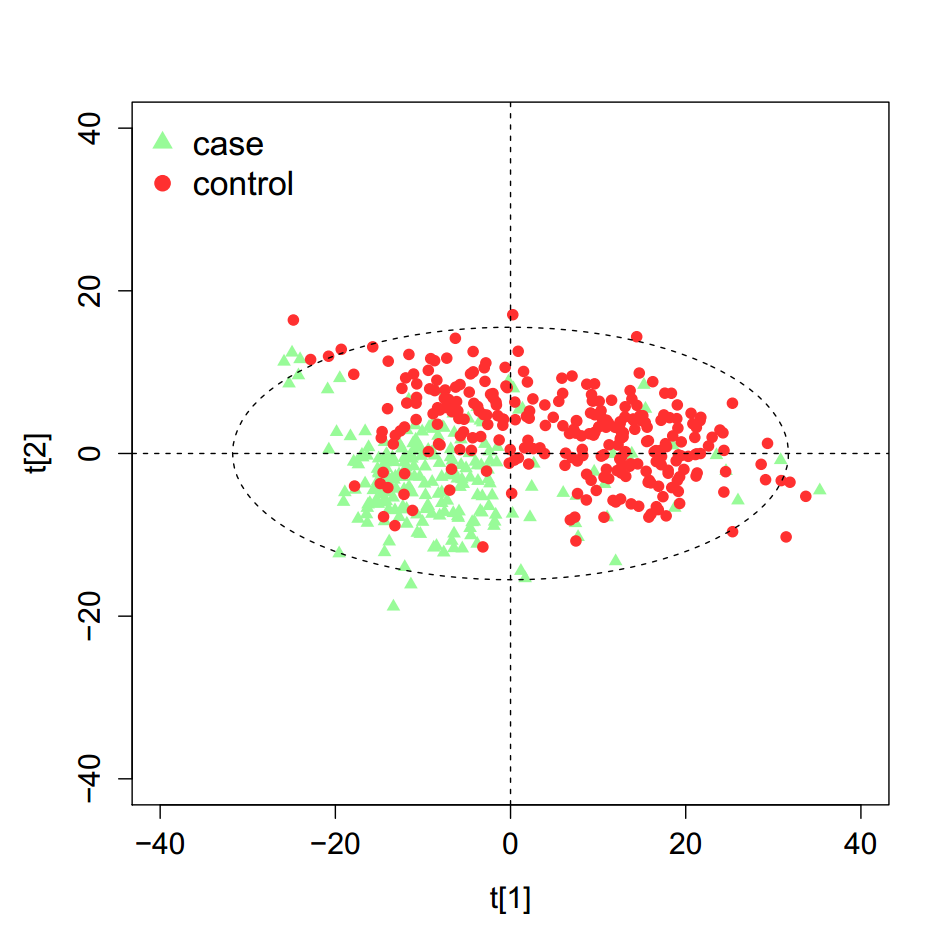
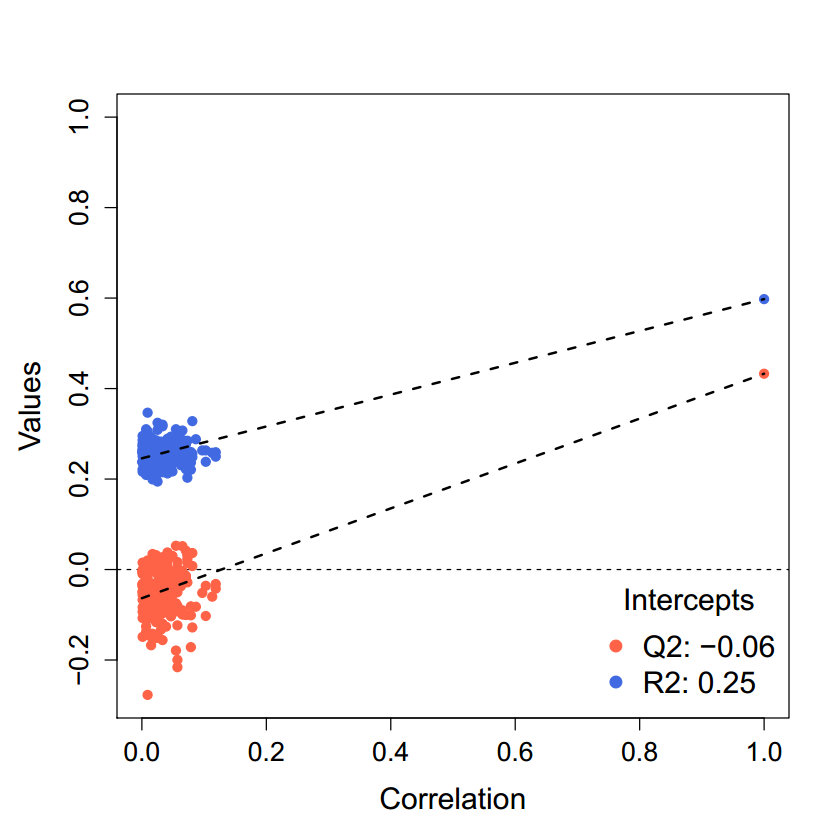
 

Figure 29. Score plot and Permutation test in PLS-DA analysis.

##### **5. HCA analysis**

HeatMap(MetFlowData = met.data,  
 log.scale = FALSE,  
 variable = "all",  
 Group = c("control", "case"),  
 scale.method = "auto",  
 path = "heat map") ##help for *HeatMap* ?*HeatMap*

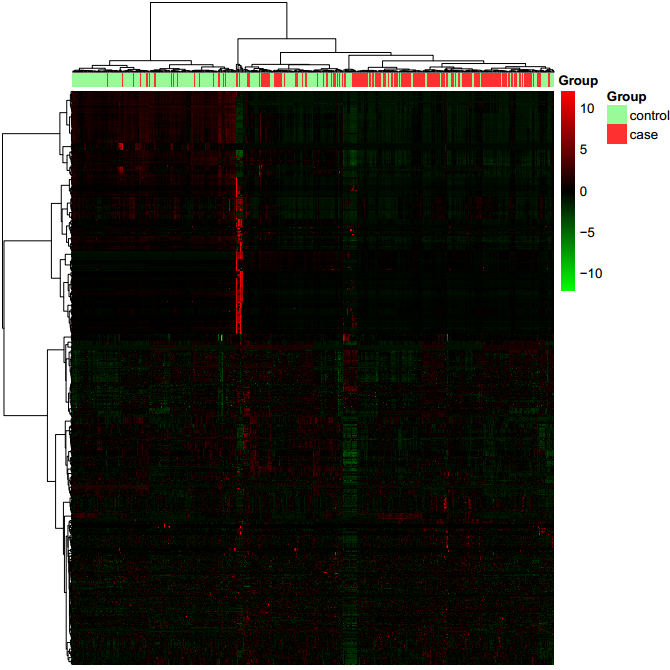


Figure 30. Heat map.

##### **6. Calculate the fold change.** (If there are more than two group, this step can be skipped)

met.data <- FoldChange(MetFlowData = met.data,  
 to = c("case", "control"),  
 ratio = "median")  
##save data  
met.data.fc <- met.data  
save(met.data.fc, file = "met.data.fc") ##help for *FoldChange* ?*FoldChange*

##### **7. Univariate test for two group data.**

met.data <- UnivariateTest(MetFlowData = met.data,  
 test.method = "t",  
 adjust.method = "fdr")  
#save data  
met.data.uni.test <- met.data  
save(met.data.uni.test, file = "met.data.uni.test") ##help for *UnivariateTest* ?*UnivariateTest*

##### **8. Select potential biomarkers.**

met.data <- MarkerSelection(MetFlowData = met.data,  
 p = "p",  
 foldchange.cutoff = c(4/3, 3/4),  
 p.cutoff = 0.05,  
 vip.cutoff = 0,  
 path = "marker selection")  
#save data  
met.data.marker <- met.data  
save(met.data.marker, file = "met.data.marker") ##help for *MarkerSelection* ?*MarkerSelection*

##### **9. Volcano plot**

VolcanoPlot(MetFlowData = met.data,  
 x = "foldchange",  
 y = "p",  
 z = "vip",  
 vip.cutoff = 0,  
 foldchange.cutoff = c(4/3, 3/4),  
 p.cutoff = 0.05,  
 path = "marker selection")

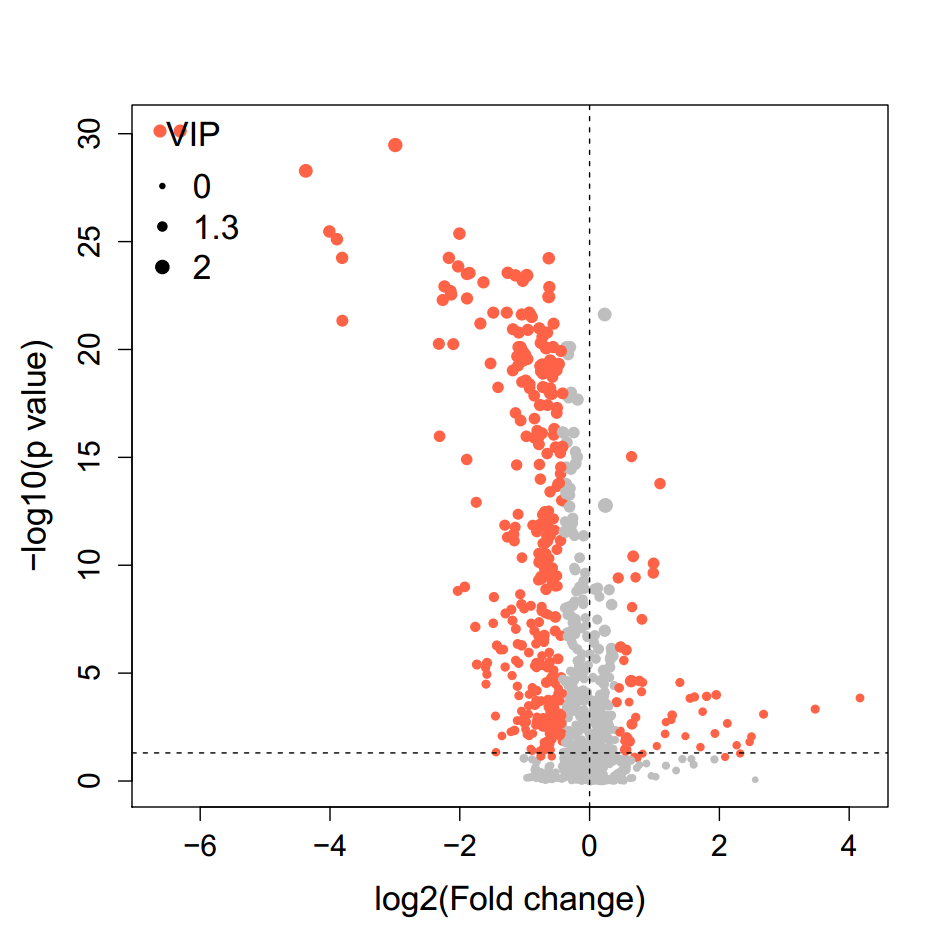


Figure 31. Volcano plot.

##### **10. Box plots of potential markers.**

MarkerShow(MetFlowData = met.data,  
 beeswarm = TRUE,  
 path = "marker selection")

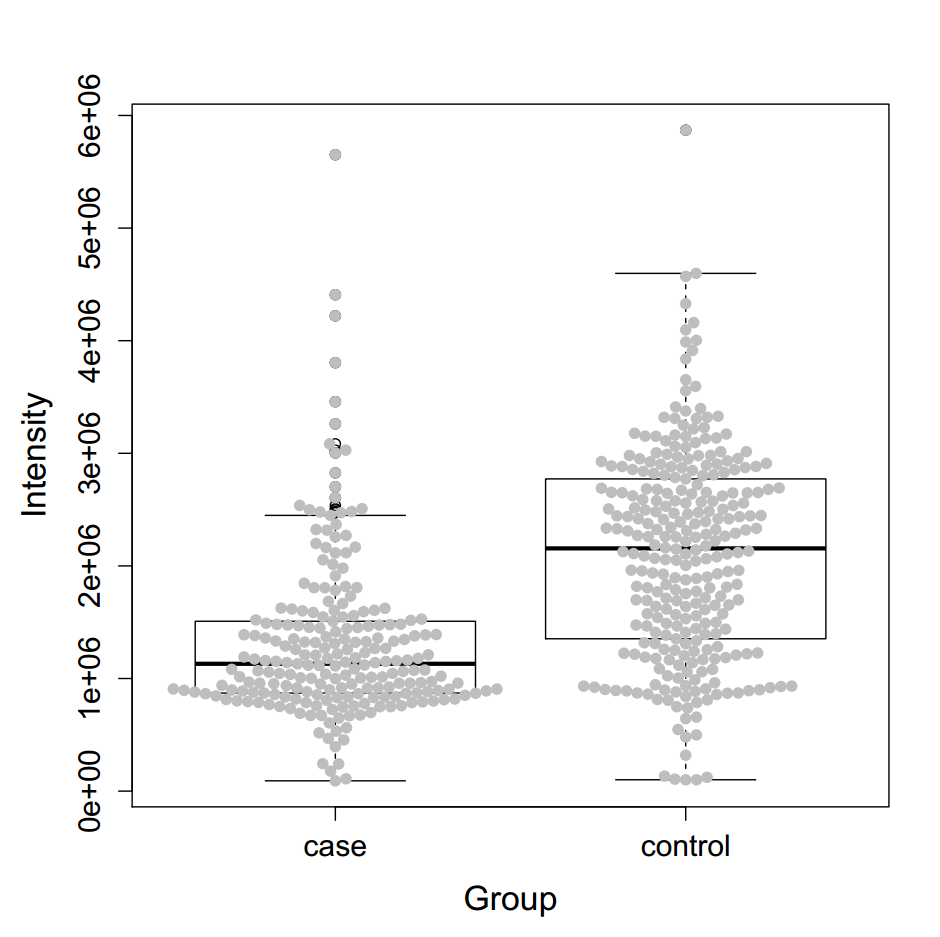


Figure 32. Boxplot of markers.

##### **11. Export data as csv files.**

ExportData(MetFlowData = met.data,  
 data.name = "data\_after\_Stat",  
 subject.info.name = "subject.info",  
 qc.info.name = "qc.info",  
 path = NULL) #save data  
met.data.after.stat <- met.data  
save(met.data.after.stat, file = "met.data.after.stat")

##### **MetCleaning contains two parts, data cleaning and statistical analysis. To simplify the usage, all the data cleaning procedures are integrated as a function, MetClean, and all the statistical analysis procedures are integrated as a function, MetStat. So, you can use MetClean to do all the data cleaning procedures and MetStat to do all the statistical analysis procedures which are described above.**

#### *MetClean* to do all cleaning procedures for metabolomic dataset.

MetClean(#ImportData para  
 data = "data.csv",  
 sample.information = "sample.information.csv",  
 polarity = "positive",  
 hasQC = "yes",  
 #MVFilter para  
 obs.mv.cutoff = 0.5,  
 var.mv.cutoff = 0.5,  
 #MVimputation  
 imputation.method = "knn",  
 k = 10,  
 #ZeroFilter para  
 obs.zero.cutoff = 0.5,  
 var.zero.cutoff = 0.5,  
 #DataNormalization  
 method = "svr",  
 threads = 2,  
 #PeakIdentification  
 hmdb.matching = TRUE,  
 mz.tolerance = 25,  
 rt.tolerance = 180,  
 #DataOverview para  
 feature.distribution = TRUE,  
 met.plot = TRUE,  
 path = NULL)

##### **All the parameters of MetStat can be found in previous functions described above.**

#### *MetStat* to do all statistical analysis procedures for metabolomics data.

MetStat(MetFlowData = met.data,  
 rsd.cutoff = 30,  
 #transformation para  
 log.scale = FALSE,  
 #PCA analysis para  
 QC = TRUE,  
 scale.method = "auto",  
 #PLS analysis para  
 plsmethod = "plsr",  
 #FoldChange para  
 fc = TRUE,  
 to = c("case", "control"),  
 ratio = "median",  
 #UnivariateTest para  
 test.method = "t",  
 adjust.method = "fdr",  
 #MarkerSelection para  
 foldchange = "foldchange",  
 p = "p",  
 vip = "vip",  
 foldchange.cutoff = c(4/3, 3/4),  
 p.cutoff = 0.05,  
 vip.cutoff = 0,  
 #VolcanoPlot para  
 x = "foldchange",  
 y = "p",  
 z = "vip",  
 col = c("black", "firebrick1"),  
 #heatmap para  
 variable = "all",  
 Group = c("control", "case"),  
 #MarkerShow para  
 path = NULL)

##### **All the parameters of MetStat can be found in previous functions described above.**