MS1-HRMS-Extractor.R

Admin

Tue Jan 21 13:29:30 2020

require("readxl")

## Loading required package: readxl

require("ggplot2")

## Loading required package: ggplot2

require("gridExtra")

## Loading required package: gridExtra

require("dplyr")

## Loading required package: dplyr

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:gridExtra':  
##   
## combine

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

require("reshape2")

## Loading required package: reshape2

require("grid")

## Loading required package: grid

require("tidyverse")

## Loading required package: tidyverse

## -- Attaching packages ------------------------------------------------------------------------------------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.2.1 v readr 1.1.1  
## v tibble 2.1.3 v purrr 0.2.5  
## v tidyr 0.8.2 v stringr 1.3.1  
## v ggplot2 3.2.1 v forcats 0.3.0

## -- Conflicts --------------------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::combine() masks gridExtra::combine()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

require("magrittr")

## Loading required package: magrittr

##   
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':  
##   
## set\_names

## The following object is masked from 'package:tidyr':  
##   
## extract

# When using this code data paths must be defined.  
# There are two options:  
# 1.) via Rstudio Session -> Set Working Directory -> To Source File Location  
# This allows working without changing data paths.  
# Store sequence .xlsx file in folder ~Sequences/  
# Store database .xlsx file in folder ~Database/  
# Store spectra files in folder ~Spectra/  
# 2.) if using data paths which are not directly placed in Source File location  
# define all paths individually

# Reading an .xlsx file that contains the information about target analytes and their species  
# Each row contains an information about only one species per compound. For instance, if the target analyte  
# forms three species during electrospray ionization - 3 rows are necessary.  
#   
# The first sheet contains the following columns:   
# Formula - molecular formula e.g. "C17H18F1N3O3"  
# Polarity - electrospray polarity: "POSITIVE"  
# Species - if Polarity is "POSITIVE", then  
# "[M+H]+", "[M+NH4]+", "[M+Na]+", "[M+K]+", "[M+CH3OH+H]+", "[2M+H]+" or "[M+2H]+".  
# Hierarchy - a number from 1-7, which corresponds to the order of species:  
# 1:[M+H]+, 2:[M+NH4]+, 3:[M+Na]+, 4:[M+K]+, 5:[M+CH3OH+H]+, 6:[2M+H]+, 7:[M+2H]+  
# mz1 - theoretical m/z value of the most abundant ion  
# mz2 - theoretical m/z value of second most abundant ion  
# Q2/Q1 - theoretical ratio between mz2/mz1  
# Class - compound class, e.g. "Quinolones"  
# Compound - name of the compound e.g. "Ciprofloxacin"  
#  
# For example,   
# Formula | Polarity | Species | Hierarchy | mz1 | mz2 | Q2/Q1 | Class | Compound |   
# 1st row: C17H18F1N3O3 | POSITIVE | [M+H]+ | 1 | 332.1404961 | 333.1438569 | 18.5589575 | Quinolone | Ciprofloxacin |   
#  
# The first row of the database should produce a data.frame, which is equal to this:  
# data.frame(Formula = "C17H18F1N3O3",  
# Polarity = "POSITIVE",  
# Species = "[M+H]+",  
# Hierarchy = 1,  
# mz1 = 332.1404961,  
# mz2 = 333.1438569,  
# Q2/Q1 = 18.5589575,  
# Class = "Quinolone",  
# Compound = "Ciprofloxacin",row.names = FALSE,stringsAsFactors = FALSE)

database <- read\_xlsx(path = "Database/Test\_database.xlsx",sheet = 1)  
database <- database %>% mutate\_at(c(colnames(database)[c(1:3,8:9)]),funs(factor(.)))

## Warning: funs() is soft deprecated as of dplyr 0.8.0  
## please use list() instead  
##   
## # Before:  
## funs(name = f(.))  
##   
## # After:   
## list(name = ~ f(.))  
## This warning is displayed once per session.

# Reading an .xlsx file that contains the information about sample sequence  
# File contains 3 columns, each row contains information about one sample:  
# File.name - filename that contains MS1 spectra in positive mode  
# Sample.name - sample name  
# Path - data path (note: all "\" must be substituted with "/")  
# For example,  
#   
# | File.name | Sample.name | Path |  
# | Name\_1-POS | Sample-1 | d:/data/datapath/ |  
#   
# If anlyzing only one sample us lines 78-80, if using more samples use line 77

#sequence <- read\_xlsx(path = "Sequences/Test\_sequence.xlsx", sheet = 1)  
sequence <- data.frame(File.name = "Test-spectra",  
 Sample.name = "This is a test",  
 Path = "Spectra/",stringsAsFactors = FALSE)

# Define file extension and seperator  
# For example, ".csv", ".acs", ".txt" etc.  
# Delimiter is ussually TAB or ",", hence:  
# file.sep = "\t", if using tab  
# file.sep = ",", if using comma  
# Other delimiters can also be defined

file.extension <- ".csv"  
file.sep <- "\t"

# Define path where the output files will be stored (note: all "\" must be substituted with "/")

out.path <- "Output/"

# Define maximum error values for Q1 and Q2 (ppm) and ratio between Q2/Q1 (%)  
# For example, if maximum mass error is 5 ppm and Q2/Q1 ratio error is 20%, then:  
# Q1.delta.thr <- 5  
# Q2.delta.thr <- 5  
# Q2Q1.rato.error.thr <- 20

Q1.delta.thr <- 5  
Q2.delta.thr <- 5  
Q2Q1.rato.error.thr <- 20

# RUN DATA PROCESSING  
# NOTE: All spectra files must contain at least two columns  
# The first column contains m/z values  
# The second column contains intesity values

for (i in 1:dim(sequence)[1]){  
 Index <-i  
 {  
 #Assigning sample names (file names from DataAnalysis .asc exports) and data path where files are located  
 #Add loop if more than 1 sample  
 sample.name.pos <- sequence$File.name[Index]  
 data.path <- sequence$Path[Index]  
 sample.name <- sequence$Sample.name[Index]  
 }  
   
   
   
 #Reading raw MS files  
 {  
 {  
 sample.pos <- read.csv(file = paste(data.path,sample.name.pos,file.extension, sep = ""),sep = file.sep)  
 sample.pos <- sample.pos[,1:2]  
 colnames(sample.pos) <- c("mz","intensity")  
 }  
   
 #Preparing databases and results table for search algortihm  
 {  
 results.pos <- subset(database, Polarity == "POSITIVE")[-2]  
 database.pos <- subset(database, Polarity == "POSITIVE")[-2]  
 add <- matrix(data = NA, nrow = as.numeric(dim(results.pos)[1]), ncol = 11,dimnames = list(NULL,c("Q1-measured, m/z","Q1-Intesity, cps","Q2-measured, m/z","Q2-Intensity,cps", "ratio-Q2/Q1, %", "Error Q1, ppm", "Error Q2, ppm", "Error Q2/Q1, %", "Compliant", "Suspicious","Only Q1 detected")))  
 results.pos <- cbind(results.pos, add)  
 remove(add)  
 detection.rate.pos <- results.pos[results.pos$Hierarchy == 1,7:8]  
 add <- matrix(data = NA, as.numeric(dim(detection.rate.pos)[1]), ncol = 7,dimnames = list(NULL,c(1,2,3,4,5,6,7)))  
 detection.rate.pos <- cbind(detection.rate.pos,add)  
 remove(add)  
 }  
   
 #Running search algortihm for postive mode  
 for (j in 1:as.numeric(dim(database.pos)[1])) {  
 x1 <- as.numeric(sample.pos[which(abs(database.pos$mz1[j]-sample.pos$mz)==min(abs(database.pos$mz1[j]-sample.pos$mz))),1])  
 i1 <- as.numeric(sample.pos[which(abs(database.pos$mz1[j]-sample.pos$mz)==min(abs(database.pos$mz1[j]-sample.pos$mz))),2])  
 x2 <- as.numeric(sample.pos[which(abs(database.pos$mz2[j]-sample.pos$mz)==min(abs(database.pos$mz2[j]-sample.pos$mz))),1])  
 i2 <- as.numeric(sample.pos[which(abs(database.pos$mz2[j]-sample.pos$mz)==min(abs(database.pos$mz2[j]-sample.pos$mz))),2])  
 delta1 <- abs(x1-as.numeric(database.pos$mz1[j]))/as.numeric(database.pos$mz1[j])\*1000000  
 delta2 <- abs(x2-as.numeric(database.pos$mz2[j]))/as.numeric(database.pos$mz2[j])\*1000000  
 ratio <- i2/i1\*100  
 error\_ratio <- abs(database.pos$`Q2/Q1`[j]-ratio)/database.pos$`Q2/Q1`[j]\*100  
 if ((delta1 < Q1.delta.thr) && (delta2 < Q2.delta.thr) && (error\_ratio < Q2Q1.rato.error.thr)) {compliant <- TRUE} else {compliant <- FALSE}  
 if ((delta1 < Q1.delta.thr) && (delta2 < Q2.delta.thr) && (error\_ratio > Q2Q1.rato.error.thr) && (error\_ratio < 100)) {suspicious <- TRUE} else {suspicious <- FALSE}  
 if ((delta1 <Q1.delta.thr) && (delta2 >Q2.delta.thr)) {only.q1 <- TRUE} else {only.q1 <- FALSE}  
 results.pos[j,9:19] <- c(x1,i1,x2,i2,ratio,delta1,delta2,error\_ratio,as.logical(compliant),as.logical(suspicious),as.logical(only.q1))  
 if (compliant == FALSE && suspicious == FALSE & only.q1 == FALSE) {results.pos[i,9:16] <- NA}  
 remove(x1,i1,x2,i2,ratio,delta1,delta2,error\_ratio,compliant,suspicious,only.q1)  
 }  
   
 #Reshaping results matrix  
 {  
 results.pos$Compliant <- as.logical(results.pos$Compliant)  
 results.pos$Suspicious <- as.logical(results.pos$Suspicious)  
 results.pos$`Only Q1 detected` <- as.logical(results.pos$`Only Q1 detected`)  
 results.pos.reshape<- reshape(results.pos[c(-(3:6),-9,-(11:13))], idvar=c("Compound","Class","Formula"),timevar = "Species", direction="wide")  
 results.pos.reshape2 <- results.pos  
 results.pos.reshape2 <- results.pos.reshape2 %>% arrange(Hierarchy)  
 }  
   
   
 #Write results to files (only compliant)  
  
 write.table(results.pos.reshape2, paste(out.path, sample.name.pos,".txt",sep = ""), sep="\t",row.names = FALSE)  
 }   
 }

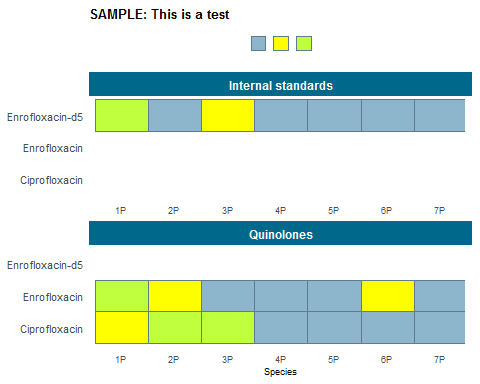
# Data visualization section  
# Can be used only for one sample at a time, hence cannot be run in sequence mode by loading an .xlsx file

# Visual data output

# Preparing data for plotting  
{  
holder.pos <- subset(results.pos.reshape,select = c(2:3,8:10,15:17,22:24,29:31,36:38,43:45,50:52))  
holder.pos[holder.pos$Compound == "Example",c(3,7,11)] <- c(TRUE,TRUE,TRUE)  
  
for (i in seq(3,dim(holder.pos)[2], by = 3)) {  
 holder.pos[(dim(holder.pos)[2]+1)] <- 0  
 for (j in 1:dim(holder.pos)[1]){  
 if (is.na(holder.pos[j,i]) == FALSE & is.na(holder.pos[j,i+1]) == FALSE & is.na(holder.pos[j,i+2]) == FALSE) {  
 if (holder.pos[j,i] == TRUE) {holder.pos[j,(dim(holder.pos)[2])] <- 3}  
 if (holder.pos[j,i+1] == TRUE) {holder.pos[j,(dim(holder.pos)[2])] <- 2}  
 if (holder.pos[j,i+2] == TRUE) {holder.pos[j,(dim(holder.pos)[2])] <- 1}   
 }  
 }  
}  
}  
{  
heatmap.detected.pos <- melt(data = holder.pos[c(1:2,24:30)],measure.vars = colnames(holder.pos[24:30]),id.vars = c("Class","Compound"),variable.name = "Species")  
levels(heatmap.detected.pos$Species) <- c("[M+H]+","[M+NH4]+","[M+Na]+","[M+K]+","[M+CH3OH+H]+","[2M+H]+","[M+2H]+")  
levels(heatmap.detected.pos$Species) <- c("1P","2P","3P","4P","5P","6P","7P")  
heatmap.detected <- heatmap.detected.pos  
heatmap.detected$value <- as.factor(heatmap.detected$value)  
factor(heatmap.detected$value, levels = c(0,1,2,3))  
colors <- c("lightskyblue3","palevioletred1","yellow1","olivedrab1")  
names(colors) <- c(0,1,2,3)  
}

# An overall heatmap for detected species  
# 1P:[M+H]+, 2P:[M+NH4]+, 3P:[M+Na]+, 4P:[M+K]+, 5P:[M+CH3OH+H]+, 6P:[2M+H]+, 7P:[M+2H]+

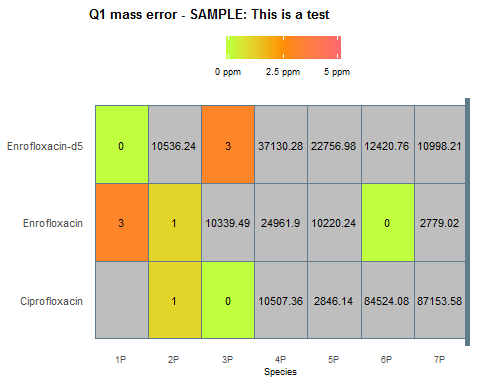
ggplot(heatmap.detected, aes(Species, Compound, colour = Class))+  
 geom\_tile(aes(fill = value), colour = "lightskyblue4", size = 0.1, stat = "identity", width = 1, height = 1)+  
 scale\_fill\_manual(values= colors,labels = c("Not detected","Only Q1 detected","Detected (ratio Q2/Q1 > 30%)", "Detected"))+  
 #scale\_fill\_gradientn(guide = "legend",colours = c("lightskyblue3","olivedrab1"),values = c(0,1))+  
 ggtitle(label = paste("SAMPLE:",sample.name))+  
 theme(panel.grid.major = element\_blank(),  
 panel.border = element\_blank(),  
 panel.background = element\_blank(),  
 axis.ticks = element\_blank(),  
 axis.text.x = element\_text(size = 7, vjust = 0.5),  
 axis.text.y = element\_text(size = 8),  
 legend.title = element\_blank(),  
 legend.position = "top",   
 legend.spacing.x = unit(0.1, 'cm'),  
 legend.text = if (i == 3) {element\_text(size = 7)} else {element\_blank()},  
 legend.key.size = unit(0.4, 'cm'),  
 axis.title.y = element\_blank(),  
 plot.title = element\_text(size = 10, face = "bold",hjust = 0),  
 axis.title.x = element\_text(size=7))+  
 geom\_vline(xintercept = 7.5, colour = "white", size = 2)+  
 theme(strip.background = element\_rect(fill = "deepskyblue4"), strip.text = element\_text(face="bold",colour = "white"))+  
 facet\_wrap(~Class,nrow = 2,scales = "free\_x")



# Q1 error plots

{  
holder.pos <- subset(results.pos.reshape,select = c(2:3,5,12,19,26,33,40,47))  
heatmap.q1.pos <- melt(data = holder.pos[c(1:2,3:9)],measure.vars = colnames(holder.pos[3:9]),id.vars = c("Class","Compound"),variable.name = "Species")  
levels(heatmap.q1.pos$Species) <- c("[M+H]+","[M+NH4]+","[M+Na]+","[M+K]+","[M+CH3OH+H]+","[2M+H]+","[M+2H]+")  
levels(heatmap.q1.pos$Species) <- c("1P","2P","3P","4P","5P","6P","7P")  
heatmap.q1 <- heatmap.q1.pos  
colnames(heatmap.q1)[4] <- "Error"  
}  
  
ggplot(heatmap.q1, aes(Species, Compound))+  
 geom\_tile(aes(fill = Error), colour = "lightskyblue4", size = 0.1, stat = "identity", width = 1, height = 1)+  
 scale\_fill\_gradientn(limits = c(0,Q1.delta.thr),colours= c("olivedrab1","darkorange","indianred1"),na.value = "grey",breaks=c(0,Q1.delta.thr/2,Q1.delta.thr),labels=c("0 ppm",paste(round(Q1.delta.thr/2,1)," ppm",sep=""),paste(round(Q1.delta.thr,1)," ppm",sep="")))+  
 ggtitle(label = paste("Q1 mass error - SAMPLE:",sample.name))+  
 geom\_text(aes(label=round(Error, digits = 2)),size = 3)+  
 theme(panel.grid.major = element\_blank(),  
 panel.border = element\_blank(),  
 panel.background = element\_blank(),  
 axis.ticks = element\_blank(),  
 axis.text.x = element\_text(size = 7, vjust = 0.5),  
 axis.text.y = element\_text(size = 8),  
 legend.title = element\_blank(),  
 legend.position = "top",   
 legend.spacing.x = unit(0.1, 'cm'),  
 legend.text = element\_text(size = 7),  
 axis.title.y = element\_blank(),  
 plot.title = element\_text(size = 10, face = "bold",hjust = 0),  
 axis.title.x = element\_text(size=7))+  
 geom\_vline(xintercept = 7.5, colour = "lightskyblue4", size = 2)

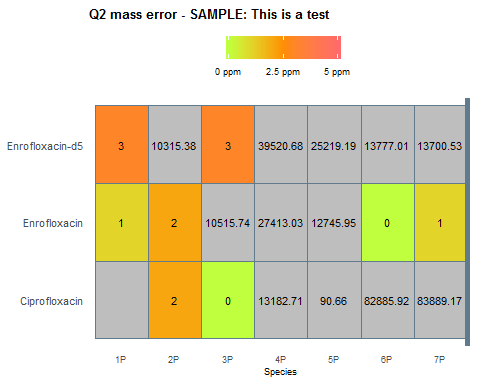
## Warning: Removed 1 rows containing missing values (geom\_text).



# Q2 error plots

{  
 holder.pos <- subset(results.pos.reshape,select = c(2:3,6,13,20,27,34,41,48))  
 heatmap.q2.pos <- melt(data = holder.pos[c(1:2,3:9)],measure.vars = colnames(holder.pos[3:9]),id.vars = c("Class","Compound"),variable.name = "Species")  
 levels(heatmap.q2.pos$Species) <- c("[M+H]+","[M+NH4]+","[M+Na]+","[M+K]+","[M+CH3OH+H]+","[2M+H]+","[M+2H]+")  
 levels(heatmap.q2.pos$Species) <- c("1P","2P","3P","4P","5P","6P","7P")  
 heatmap.q2 <- heatmap.q2.pos  
 colnames(heatmap.q2)[4] <- "Error"  
}  
ggplot(heatmap.q2, aes(Species, Compound))+  
 geom\_tile(aes(fill = Error), colour = "lightskyblue4", size = 0.1, stat = "identity", width = 1, height = 1)+  
 scale\_fill\_gradientn(limits = c(0,Q2.delta.thr),colours= c("olivedrab1","darkorange","indianred1"),na.value = "grey",breaks=c(0,Q2.delta.thr/2,Q2.delta.thr),labels=c("0 ppm",paste(round(Q2.delta.thr/2,1)," ppm",sep=""),paste(round(Q2.delta.thr,1)," ppm",sep="")))+  
 ggtitle(label = paste("Q2 mass error - SAMPLE:",sample.name))+  
 geom\_text(aes(label=round(Error, digits = 2)),size = 3)+  
 theme(panel.grid.major = element\_blank(),  
 panel.border = element\_blank(),  
 panel.background = element\_blank(),  
 axis.ticks = element\_blank(),  
 axis.text.x = element\_text(size = 7, vjust = 0.5),  
 axis.text.y = element\_text(size = 8),  
 legend.title = element\_blank(),  
 legend.position = "top",   
 legend.spacing.x = unit(0.1, 'cm'),  
 legend.text = element\_text(size = 7),  
 axis.title.y = element\_blank(),  
 plot.title = element\_text(size = 10, face = "bold",hjust = 0),  
 axis.title.x = element\_text(size=7))+  
 geom\_vline(xintercept = 7.5, colour = "lightskyblue4", size = 2)

## Warning: Removed 1 rows containing missing values (geom\_text).



# Q2/Q1 ratio error plots

{  
 holder.pos <- subset(results.pos.reshape,select = c(2:3,7,14,21,28,35,42,49))  
 heatmap.ratio.pos <- melt(data = holder.pos[c(1:2,3:9)],measure.vars = colnames(holder.pos[3:9]),id.vars = c("Class","Compound"),variable.name = "Species")  
 levels(heatmap.ratio.pos$Species) <- c("[M+H]+","[M+NH4]+","[M+Na]+","[M+K]+","[M+CH3OH+H]+","[2M+H]+","[M+2H]+")  
 levels(heatmap.ratio.pos$Species) <- c("1P","2P","3P","4P","5P","6P","7P")  
 heatmap.ratio <- heatmap.ratio.pos  
 colnames(heatmap.ratio)[4] <- "Error"  
 heatmap.ratio$Error[heatmap.ratio$Error > 100] <- NA  
}  
ggplot(heatmap.ratio, aes(Species, Compound))+  
 geom\_tile(aes(fill = Error), colour = "lightskyblue4", size = 0.1, stat = "identity", width = 1, height = 1)+  
 scale\_fill\_gradientn(limits = c(0,100),colours= c("olivedrab1","darkorange","indianred1"),na.value = "grey",breaks=c(0,Q2Q1.rato.error.thr,100),labels=c("0%",paste(round(Q2Q1.rato.error.thr,1)," %",sep=""),"100%"))+  
 ggtitle(label = paste("Q2/Q1 Error - SAMPLE:",sample.name))+  
 geom\_text(aes(label=round(Error, digits = 2)),size = 3)+  
 theme(panel.grid.major = element\_blank(),  
 panel.border = element\_blank(),  
 panel.background = element\_blank(),  
 axis.ticks = element\_blank(),  
 axis.text.x = element\_text(size = 7, vjust = 0.5),  
 axis.text.y = element\_text(size = 8),  
 legend.title = element\_blank(),  
 legend.position = "top",   
 legend.spacing.x = unit(0.1, 'cm'),  
 legend.text = element\_text(size = 7),  
 axis.title.y = element\_blank(),  
 plot.title = element\_text(size = 10, face = "bold",hjust = 0),  
 axis.title.x = element\_text(size=7))+  
 geom\_vline(xintercept = 7.5, colour = "lightskyblue4", size = 2)

## Warning: Removed 12 rows containing missing values (geom\_text).

