R codes of ‘practice’

Contents

1 File: auto_workflow.R

```r
library(usethis)
library(devtools)
library(progress)
## data reformat
library(pbapply)
library(data.table)
library(plyr)
library(dplyr)
library(MSnbase)
library(igraph)
## visualize
library(ggplot2)
library(ggraph)
library(ggsci)
library(stringr)
library(grid)
## suggest
library(ChemmineR)
library(ChemmineOB)
library(rsvg)
library(grImport2)
library(ggimage)
## extra
library(gt)
library(classyfireR)
library(aplot)
library(ggalluvial)
##
load_all("~/MCnebula/R")
load_all("~/extra/R")
initialize_mcnebula(".")
# collate_structure(exclude_element = c("P", "S", "B", "Si"), fc = NA)
collate_structure()
build_classes_tree_list()
# collate_ppcp(min_possess = 50, max_possess_pct = 0.1, filter_via_struc_score = NA)
collate_ppcp()
# generate_parent_nebula(min_tanimoto = 0.5)
generate_parent_nebula()
generate_child_nebulae()
```

```
visualize_parent_nebula()
visualize_child_nebulae()
```

2 File: barplot_horizental.R

```
 library(ggplot2)
 library(ggrepel)
 library(ggsci)
 library(Hmisc)
 args<-commandArgs(TRUE)
 setwd(args[1])
 file=args[2]
 savename=strsplit(file,split=".tsv")
 df <- read.csv(file=file,header= T,sep= "\t")
 df[,1] <- capitalize(df[,1])
 annonation <- df[which(df[,1]=="Annonation"),]
 df <- df[which(df[,1]!="Annonation"),]
 df[,2] <- as.numeric(df[,2])
 #order=df[order(df[,2],decreasing=F),]
 p <- ggplot(data=df, aes(x=reorder(df[,1],df[,2]), y=df[,2], fill=df[,2])) +
   geom_col(width = 0.7) +
   scale_fill_gradientn(colours = c("#FFBB78FF","#FF7F0EFF")) +
   coord_flip() +
   guides(fill="none") +
   labs(y=paste0("Number(", annonation[,2]), x="FC group", ")", title="Stat features") +
   geom_text(data=df, aes(x=df[,1], y=df[,2]+max(df[,2])*(1/30), label=df[,2]),
             hjust=0, color="black", family="Times", alpha=0.6, size=5, inherit.aes = FALSE ) +
theme(legend.position = "right",text=element_text(family="serif", size=20), plot.title = element_text(h
 ggsave(p,file=paste0(savename,".svg"), width=15, height=7)
```

3 File: blood_chemical.R

```
 library(ggplot2)
 library(ggrepel)
 library(ggsci)
 library(Hmisc)
 library(outliers)
 library(ggthemes)
 library(stringr)
grubbs<-function(x){
  x<-round(x,4)
  grubbs_outliers<-c()
  grubbs_p.value<-c()
  grubbs_g.value<-c()
  grubbs_g<-c()
  grubbs_minormax<-c()
  grubbs_pvalue<-c()
  grubbs_p<-0
  while(grubbs_p<0.05){
    grubbs_outliers<-c(grubbs_outliers,grubbs_minormax)
    grubbs_p.value<-c(grubbs_p.value,grubbs_pvalue)
    grubbs_g<-c(grubbs_g,grubbs_g.value)
    if(sum(x==grubbs_minormax)!=0)x<-x[-which(x==grubbs_minormax)]
    if(sd(x)==0) break
    grubbs_test<-grubbs.test(x,type=10,opposite=F,two.sided=F)
    grubbs_p<-grubbs_test$p.value
    grubbs_pvalue<-grubbs_test$p.value
    grubbs_g.value<-grubbs_test$statistic[1]
    grubbs_a<-strsplit(grubbs_test$alternative," ",fixed=T)
    grubbs_minormax<-as.numeric(unlist(grubbs_a)[3])
  }
  outliner_res<-data.frame(outliers=grubbs_outliers,gvalue=grubbs_g,pvalue=grubbs_p.value)
  return(outliner_res)
}
dixon<-function(x){
    dixon_p.value<-c()
    dixon_q.value<-c()
    dixon_q<-c()
    dixon_pvalue<-c()
    dixon_outliers<-c()
    dixon_minormax<-c()
    dixon_p<-0
    while(dixon_p<0.05){
      dixon_outliers<-c(dixon_outliers,dixon_minormax)
      dixon_p.value<-c(dixon_p.value,dixon_pvalue)
      dixon_q<-c(dixon_q,dixon_q.value)
      if(sum(x==dixon_minormax)!=0)x<-x[-which(x==dixon_minormax)]
      if(sd(x)==0) break
      dixon_test<-dixon.test(x,type=0,opposite=F,two.sided=F)
      dixon_p<-dixon_test$p.value
      dixon_pvalue<-dixon_test$p.value
      dixon_q.value<-dixon_test$statistic[1]
      dixon_a<-strsplit(dixon_test$alternative," ",fixed=T)
      dixon_minormax<-as.numeric(unlist(dixon_a)[3])
    }
    outliner_res<-data.frame(outliers=dixon_outliers,qvalue=dixon_q,pvalue=dixon_p.value)
    return(outliner_res)
  }
 args<-commandArgs(TRUE)
 setwd(args[1])
 file=args[2]
 savename=strsplit(file,split=".tsv")
 df <- read.csv(file=file,header= T,sep= "\t")
 class=unique(df$index)
 if(exists("re_stat")==TRUE){rm(re_stat)}
 if(exists("stat_exclude")==TRUE){rm(stat_exclude)}
 for(i in class){
 data <- df[which(df$index==i),]
 subgroup=unique(data$subgroup)
    for(k in subgroup){
    group=data[which(data$subgroup==k),]
    X=group$level
    #if(exists("x_filter")==TRUE){rm(x_filter)}
    out1=grubbs(round(X,2))
    out2=dixon(round(X,2))
    out3=NULL
    for(t in X){
        if( t>(mean(X)+2*sd(X)) || t<(mean(X)-2*sd(X))){out3=c(out3,t)}}
    out=c(out1$outliers, out2$outliers, out3)
    #num_x=length(x_filter)
    if(is.null(out)){re_group=group}else{re_group=group[which(group$level!=out),]}
    if(is.null(out)){exclude=NULL}else{exclude=group[which(group$level==out),]}
    if(exists("re_stat")==FALSE){re_stat=re_group}else{re_stat=rbind(re_stat,re_group)}
    if(exists("stat_exclude")==FALSE){stat_exclude=exclude}else{stat_exclude=rbind(stat_exclude,exclude
    }
 }
list=rbind(c("model","raw"),c("model","pro"),c("raw","pro"),c("model","control"))
t.test=array(c("compare1","compare2","p","index"),c(1,4))
for(chem in class){
    for(row in 1:nrow(list)){
    double=list[row,]
    escape=0
```

```r
        for(k in c("low","medium","high")){
        if(double[1]!="model"){compare2=paste0(double[1],"_",k)}else{compare2=double[1]}
        if(double[2]!="model"){compare1=paste0(double[2],"_",k)}else{compare1=double[2]}
        if(double[1]=="model" & double[2]=="control"){compare2="model"; compare1="control"; escape=1}
        X=re_stat[which(re_stat$subgroup==compare2 & re_stat$index==chem),] #x~compare2
        Y=re_stat[which(re_stat$subgroup==compare1 & re_stat$index==chem),] #y~compare1
        stat_p=t.test(X$level, Y$level, var.equal = T, paired = F)
        stat_p$p.value=round(stat_p$p.value,5)
        #hx=fivenum(X$level)[4]+(fivenum(X$level)[4]-fivenum(X$level)[2])*(1.5)
        #hy=fivenum(Y$level)[4]+(fivenum(Y$level)[4]-fivenum(Y$level)[2])*(1.5)
        t.test=rbind(t.test,c(compare1, compare2, stat_p$p.value, chem),
                    c(compare2, compare1, stat_p$p.value, chem))
        if(escape==1){break}
        }
    }
}
stat=data.frame(t.test[-1,])
colnames(stat)=t.test[1,]
write.table(stat, file = paste0("ttest",".tsv"), quote = FALSE, append = FALSE, sep = "\t", col.names =
write.table(re_stat, file = paste0("filter_chemical_index",".tsv"), quote = FALSE,
        append = FALSE, sep = "\t", col.names = TRUE, row.names = FALSE)
write.table(stat_exclude, file = paste0("exclude",".tsv"), quote = FALSE,
        append = FALSE, sep = "\t", col.names = TRUE, row.names = FALSE)
### loop
df=read.csv(file="filter_chemical_index.tsv",header= T,sep= "\t")
write.table(df, file = paste0("filter_chemical_index_",Sys.Date(),".tsv"), quote = FALSE,
        append = FALSE, sep = "\t", col.names = TRUE, row.names = FALSE)
class=unique(df$index)
 if(exists("re_stat")==TRUE){rm(re_stat)}
 if(exists("stat_exclude")==TRUE){rm(stat_exclude)}
 for(i in class){
 data <- df[which(df$index==i),]
 subgroup=unique(data$subgroup)
    for(k in subgroup){
    group=data[which(data$subgroup==k),]
    X=group$level
    #if(exists("x_filter")==TRUE){rm(x_filter)}
    out1=grubbs(round(X,2))
    out2=dixon(round(X,2))
    out3=NULL
    for(t in X){
        if( t>(mean(X)+2*sd(X)) || t<(mean(X)-2*sd(X))){out3=c(out3,t)}}
    out=c(out1$outliers, out2$outliers, out3)
    #num_x=length(x_filter)
    if(is.null(out)){re_group=group}else{re_group=group[which(group$level!=out),]}
    if(is.null(out)){exclude=NULL}else{exclude=group[which(group$level==out),]}
    if(exists("re_stat")==FALSE){re_stat=re_group}else{re_stat=rbind(re_stat,re_group)}
    if(exists("stat_exclude")==FALSE){stat_exclude=exclude}else{stat_exclude=rbind(stat_exclude,exclude
    }
 }
list=rbind(c("model","raw"),c("model","pro"),c("raw","pro"),c("model","control"))
t.test=array(c("compare1","compare2","p","index"),c(1,4))
for(chem in class){
    for(row in 1:nrow(list)){
    double=list[row,]
    escape=0
        for(k in c("low","medium","high")){
        if(double[1]!="model"){compare2=paste0(double[1],"_",k)}else{compare2=double[1]}
        if(double[2]!="model"){compare1=paste0(double[2],"_",k)}else{compare1=double[2]}
        if(double[1]=="model" & double[2]=="control"){compare2="model"; compare1="control"; escape=1}
        X=re_stat[which(re_stat$subgroup==compare2 & re_stat$index==chem),] #x~compare2
        Y=re_stat[which(re_stat$subgroup==compare1 & re_stat$index==chem),] #y~compare1
        stat_p=t.test(X$level, Y$level, var.equal = T, paired = F)
        stat_p$p.value=round(stat_p$p.value,5)
        #hx=fivenum(X$level)[4]+(fivenum(X$level)[4]-fivenum(X$level)[2])*(1.5)
        #hy=fivenum(Y$level)[4]+(fivenum(Y$level)[4]-fivenum(Y$level)[2])*(1.5)
        t.test=rbind(t.test,c(compare1, compare2, stat_p$p.value, chem),
                    c(compare2, compare1, stat_p$p.value, chem))
        if(escape==1){break}
        }
    }
}
stat=data.frame(t.test[-1,])
colnames(stat)=t.test[1,]
write.table(stat, file = paste0("ttest",".tsv"), quote = FALSE, append = FALSE, sep = "\t", col.names =
write.table(re_stat, file = paste0("re_filter_chemical_index",".tsv"), quote = FALSE,
        append = FALSE, sep = "\t", col.names = TRUE, row.names = FALSE)
write.table(stat_exclude, file = paste0("re_exclude",".tsv"), quote = FALSE,
        append = FALSE, sep = "\t", col.names = TRUE, row.names = FALSE)
### loop end here
re_stat$subgroup=capitalize(re_stat$subgroup)
stat$compare1=capitalize(stat$compare1)
```

```
t1=stat[which(stat$compare1!="Model" & stat$compare1!="Control"),]
    t1$anno=ifelse(t1$compare2=="model",
        ifelse(t1$p<0.05, ifelse(t1$p<0.01, "**", "*"), ""),
        ifelse(t1$p<0.05, ifelse(t1$p<0.01, "##", "#"), ""))
t2=stat[which(stat$compare1=="Model" & stat$compare2=="control"),]
    t2$anno=ifelse(t2$compare2=="control",
        ifelse(t2$p<0.05, ifelse(t2$p<0.01, "**", "*"), ""),
        ifelse(t2$p<0.05, ifelse(t2$p<0.01, "##", "#"), ""))
tt=rbind(t1[which(t1$anno!=""),], t2[which(t2$anno!=""),])
tt$h=NA
tt$h1=NA
tt$h2=NA
for(i in unique(re_stat$index)){
    for(j in unique(re_stat$subgroup)){
    calculate=re_stat[which(re_stat$index==i & re_stat$subgroup==j),]
    rule=re_stat[which(re_stat$index==i),]
    h=max(calculate$level) #fivenum(calculate$level)[4]+(fivenum(calculate$level)[4]-fivenum(calculate$
    h1=h+(max(rule$level)-min(rule$level))*(1/20)
    h2=h+(max(rule$level)-min(rule$level))*(3/20)
    #if(h > max(calculate$level)+(fivenum(calculate$level)[4]-fivenum(calculate$level)[2])){h=max(calcu
    test=tt[which(tt$index==i & tt$compare1==j),]
    if(nrow(test)!=0){tt[which(tt$index==i & tt$compare1==j),]$h=h;
            tt[which(tt$index==i & tt$compare1==j),]$h1=h1
            tt[which(tt$index==i & tt$compare1==j),]$h2=h2}
    }
}
for(i in class){
data=re_stat[which(re_stat$index==i),]
#delta=max(data$level)-min(data$level)
anno=tt[which(tt$index==i),]
savename=i
complement=anno[1,]
anno1=anno[c(which(anno$anno=="*"),which(anno$anno=="**")),]
anno2=anno[c(which(anno$anno=="#"),which(anno$anno=="##")),]
if(nrow(anno1)==0){anno1=complement; anno1$anno=""}
if(nrow(anno2)==0){anno2=complement; anno2$anno=""}
boxplot<-ggplot(data,aes(x=subgroup,y=level,fill=subgroup)) +
  stat_boxplot(geom="errorbar", width=0.4) +
  geom_boxplot(width=0.4) +
  #geom_point(aes(fill=subgroup), shape=21, color="black") +
  geom_jitter(aes(fill=subgroup, x=subgroup), shape=21, color="black", width=0.01, height=0, size=2) +
```

```r
  stat_summary(fun="mean",geom="point",shape=23,size=3,fill="grey") +
  geom_label(data=anno1, aes(x=compare1, label=anno, y=h1),
        hjust=0.5, color="black", family="Times",
        alpha=ifelse(anno1$anno[1]=="",0,0.7), label.size=ifelse(anno1$anno[1]=="",0,0.3),
        size=5, inherit.aes = FALSE ) +
  geom_label(data=anno2, aes(x=compare1, label=anno, y=h2),
        hjust=0.5, color="black", family="Times",
        alpha=ifelse(anno2$anno[1]=="",0,0.7), label.size=ifelse(anno2$anno[1]=="",0,0.3),
        size=4, inherit.aes = FALSE ) +
  scale_fill_manual(values = c("Control"="#ADB6B6FF","Drug"="#95CC5EFF","Model"="#7E6148FF",
                "Pro_low"="#FDAE6BFF","Pro_medium"="#FD8D3CFF","Pro_high"="#E6550DFF",
                "Raw_low"="#9ECAE1FF","Raw_medium"="#6BAED6FF","Raw_high"="#3182BDFF")) +
  scale_color_manual(values = c("Control"="#ADB6B6FF","Drug"="#95CC5EFF","Model"="#7E6148FF",
                "Pro_low"="#FDAE6BFF","Pro_medium"="#FD8D3CFF","Pro_high"="#E6550DFF",
                "Raw_low"="#9ECAE1FF","Raw_medium"="#6BAED6FF","Raw_high"="#3182BDFF")) +
  labs(y=paste0(unique(data$index),"(", unique(data$unit), ")"),
    x="Classification(Control vs model or Eucommia vs model: * ~ p < 0.05; ** ~ p < 0.01. \
    Pro- or raw- Eucommia in identical dosage compare with each other: # ~ p < 0.05; ## ~ p < 0.01)",
    title="Hematochemistry") +
  theme_minimal() +
  theme(legend.position = "right",text=element_text(family="serif", size=10), plot.title = element_text
    axis.text.x = element_blank(), plot.background = element_rect(fill ="white", color="white"))
ggsave(boxplot, file=paste0(i,".svg"), width=10, height=6)
}
 #### facet_grid all plot
 data=re_stat
 data$index_unit=paste0(data$index, "(", data$unit, ")")
 anno=tt
 ct=unique(data[,colnames(data) %in% c("index","index_unit")])
 anno=merge(anno, ct, by="index", all.x=TRUE, sort=TRUE)
 anno1=anno[c(which(anno$anno=="*"),which(anno$anno=="**")),]
 anno2=anno[c(which(anno$anno=="#"),which(anno$anno=="##")),]
 complement=anno[1,]
 if(nrow(anno1)==0){anno1=complement; anno1$anno=""}
 if(nrow(anno2)==0){anno2=complement; anno2$anno=""}
 boxplot<-ggplot(data,aes(x=subgroup,y=level,fill=subgroup)) +
  stat_boxplot(geom="errorbar", width=0.4) +
  geom_boxplot(width=0.4) +
  #geom_point(aes(fill=subgroup), shape=21, color="black") +
  geom_jitter(aes(fill=subgroup, x=subgroup), shape=21, color="black", width=0.01, height=0, size=2) +
  stat_summary(fun="mean",geom="point",shape=23,size=3,fill="grey") +
  geom_label(data=anno1, aes(x=compare1, label=anno, y=h1),
        hjust=0.5, color="black", family="Times",
        alpha=ifelse(anno1$anno[1]=="",0,0.7), label.size=ifelse(anno1$anno[1]=="",0,0.3),
        size=5, inherit.aes = FALSE ) +
  geom_label(data=anno2, aes(x=compare1, label=anno, y=h2),
        hjust=0.5, color="black", family="Times",
        alpha=ifelse(anno2$anno[1]=="",0,0.7), label.size=ifelse(anno2$anno[1]=="",0,0.3),
        size=4, inherit.aes = FALSE ) +
  scale_fill_manual(values = c("Control"="#ADB6B6FF","Drug"="#95CC5EFF","Model"="#7E6148FF",
              "Pro_low"="#FDAE6BFF","Pro_medium"="#FD8D3CFF","Pro_high"="#E6550DFF",
              "Raw_low"="#9ECAE1FF","Raw_medium"="#6BAED6FF","Raw_high"="#3182BDFF")) +
  scale_color_manual(values = c("Control"="#ADB6B6FF","Drug"="#95CC5EFF","Model"="#7E6148FF",
              "Pro_low"="#FDAE6BFF","Pro_medium"="#FD8D3CFF","Pro_high"="#E6550DFF",
              "Raw_low"="#9ECAE1FF","Raw_medium"="#6BAED6FF","Raw_high"="#3182BDFF")) +
  labs(y="",
    x="Classification(Control vs model or Eucommia vs model: * ~ p < 0.05; ** ~ p < 0.01. \
    Pro- or raw- Eucommia in identical dosage compare with each other: # ~ p < 0.05; ## ~ p < 0.01)",
    title="Hematochemistry") +
  theme_minimal() +
  facet_grid(index_unit~.,scales="free_y") +
  theme(legend.position = "right",text=element_text(family="serif", size=10), plot.title = element_text
    axis.text.x = element_blank(), plot.background = element_rect(fill ="white", color="white"))
ggsave(boxplot, file=paste0("chemical_facet",".svg"), width=8, height=20)
 ###################### BUN and CR
 data=re_stat[c(which(re_stat$index=="Urea"), which(re_stat$index=="CR")),]
 data$index_unit=paste0(data$index, "(", data$unit, ")")
 anno=tt[c(which(tt$index=="Urea"), which(tt$index=="CR")),]
 ct=unique(data[,colnames(data) %in% c("index","index_unit")])
 anno=merge(anno, ct, by="index", all.x=TRUE, sort=TRUE)
 anno1=anno[c(which(anno$anno=="*"),which(anno$anno=="**")),]
 anno2=anno[c(which(anno$anno=="#"),which(anno$anno=="##")),]
 complement=anno[1,]
 if(nrow(anno1)==0){anno1=complement; anno1$anno=""}
 if(nrow(anno2)==0){anno2=complement; anno2$anno=""}
 boxplot<-ggplot(data,aes(x=subgroup,y=level,fill=subgroup)) +
  stat_boxplot(geom="errorbar", width=0.4) +
  geom_boxplot(width=0.4) +
  #geom_point(aes(fill=subgroup), shape=21, color="black") +
  geom_jitter(aes(fill=subgroup, x=subgroup), shape=21, color="black", width=0.01, height=0, size=2) +
  stat_summary(fun="mean",geom="point",shape=23,size=3,fill="grey") +
  geom_label(data=anno1, aes(x=compare1, label=anno, y=h1),
        hjust=0.5, color="black", family="Times",
        alpha=ifelse(anno1$anno[1]=="",0,0.7), label.size=ifelse(anno1$anno[1]=="",0,0.3),
        size=5, inherit.aes = FALSE ) +
  geom_label(data=anno2, aes(x=compare1, label=anno, y=h2),
        hjust=0.5, color="black", family="Times",
        alpha=ifelse(anno2$anno[1]=="",0,0.7), label.size=ifelse(anno2$anno[1]=="",0,0.3),
        size=4, inherit.aes = FALSE ) +
  scale_fill_manual(values = c("Control"="#ADB6B6FF","Drug"="#95CC5EFF","Model"="#7E6148FF",
              "Pro_low"="#FDAE6BFF","Pro_medium"="#FD8D3CFF","Pro_high"="#E6550DFF",
              "Raw_low"="#9ECAE1FF","Raw_medium"="#6BAED6FF","Raw_high"="#3182BDFF")) +
  scale_color_manual(values = c("Control"="#ADB6B6FF","Drug"="#95CC5EFF","Model"="#7E6148FF",
              "Pro_low"="#FDAE6BFF","Pro_medium"="#FD8D3CFF","Pro_high"="#E6550DFF",
              "Raw_low"="#9ECAE1FF","Raw_medium"="#6BAED6FF","Raw_high"="#3182BDFF")) +
  labs(y="",
    x="Classification(Control vs model or Eucommia vs model: * ~ p < 0.05; ** ~ p < 0.01. \
    Pro- or raw- Eucommia in identical dosage compare with each other: # ~ p < 0.05; ## ~ p < 0.01)",
    title="Hematochemistry") +
  theme_minimal() +
  facet_grid(index_unit~.,scales="free_y") +
  theme(legend.position = "right",text=element_text(family="serif", size=10), plot.title = element_text
    axis.text.x = element_blank(), plot.background = element_rect(fill ="white", color="white"))
ggsave(boxplot, file=paste0("BUN_CR_facet",".svg"), width=10, height=8)
```

4 File: chemical_svg.R

```
library(grImport2)
```

5 File: cholic_acid.R

```
cat("Cholic acid formula: C24-H40-O5\n")
cat("Cholic acid-d4 formula: C24-H36-D4-O5\n")
cat("---Ion mode---\n")

cat("Cholic acid-d4 [M-H]-: C24-H35-D4-O5\n")
d4_neg <- 12*24 + 1.007825*35 + 2.014102*4 + 15.994915*5
cat(paste0("Cholic acid-d4 in negtive ion mode [M-H]-: ", d4_neg, "\n"))

cat("Cholic acid-d4 [M-2*H2O+H]+: C24-H33-D4-O3\n")
d4_pos <- 12*24 + 1.007825*33 + 2.014102*4 + 15.994915*3
cat(paste0("Cholic acid-d4 in positive ion mode [M-H2O+H]+: ", d4_pos, "\n"))
```

6 File: class_matrix.R

```r
library(tidyverse)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(reshape2)
library(hrbrthemes)
plimit=0.5
datapath1="stat_classification.tsv"
data <- read.csv(file=datapath1, header=T, sep="\t", quote = "")
df <- data[which(data$definition!="null"),]
datapath2="fingerid_first_score.tsv"
data2 <- read.csv(file=datapath2, header=T, sep="\t", quote = "")
df <- merge(df, data2[,colnames(data2) %in% c("id", "similarity")], by="id", all.x=T, sort=T)
df$similarity <- as.numeric(df$similarity)
instance <- c(2028, 274, 347, 495, 2268)
test <- df[df$id %in% instance, ]
## another: "specific_pp", "definition_pp"
facet <- c("specific_pp", "level_5_pp", "subclass_pp", "class_pp", "superclass_pp")
re_facet <- c("Most specific class", "Level 5", "Subclass", "Class", "Superclass")
colnames(test)[colnames(test) %in% facet]=re_facet
test <- melt(test, measure.vars=re_facet, variable.name="condition", value.name="expr")
p <- ggplot(test) +
  geom_col(aes(x=2, y=expr,
             fill=ifelse(expr>plimit,
                         ifelse(condition==re_facet[1], "PPCP (filter)", "PPCP"), "filter")),
           color="black") + # class
geom_col(aes(x=3, y=plimit, fill="PPCP threshold"), color="black") + # class expect
  geom_col(aes(x=1, y=0,
             fill=ifelse(similarity>0.4,
                         ifelse(condition==re_facet[1], "filter", "similarity"), "filter"))) + # simi
geom_col(aes(x=4, y=0, fill="similarity threshold")) + # similarity expect
  theme_ipsum() +
    scale_fill_manual(values=c("PPCP (filter)"="#ADB6B6FF", "PPCP"="#4DBBD5FF",
                               "PPCP threshold"="#0073C2FF")) +
labs(fill="Status") +
guides(fill=guide_legend(nrow=1)) +
theme(
      axis.text.x = element_blank(),
      axis.text.y = element_text(size = 6),
      panel.spacing = unit(0.1, "lines"),
      plot.background = element_rect(fill = "white", size=0),
      axis.title.x = element_blank(),
      axis.title.y = element_blank(),
      legend.position = "bottom",
      legend.key.height = unit(0.3, "cm"),
      legend.key.width = unit(0.3, "cm"),
      legend.text = element_text(size=8),
      legend.title = element_text(size=8),
      strip.text.x = element_text(size = 7, family="Times", face="bold", hjust=0.5),
      strip.text.y = element_blank(),
      text=element_text(family="Times"),
      panel.border = element_blank(),
      plot.margin =unit(c(0,0.2,0.2,0.2),"cm")
      ) +
xlim(0, 5) +
ylim(0, 1.5) +
facet_grid(id ~ condition)
 #ggsave(p, file="test.svg")
 ##########################
 ##########################
 eg <- cbind(data.frame(seq(6)), data.frame(c(0, 4, 3, 2, 1, 0)))
colnames(eg) <- c("x", "y")
p2 <- ggplot(eg, aes(x=x, y=y)) +
  geom_step(size=1) +
  theme_classic() +
  labs(y="Class priority") +
  theme(
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        panel.spacing = unit(0.1, "lines"),
        panel.border = element_blank(),
        plot.margin =unit(c(0.2,0.2,0,0.2),"cm"),
        #plot.background = element_rect(fill = "white"),
        axis.title.x = element_blank(),
        axis.title.y = element_text(hjust=0.5, size=7, family="Times", face="bold"),
        legend.position = "bottom",
        text=element_text(family="Times")
  )
  text <- df[df$id %in% instance, colnames(df) %in% c("id", "definition")]
  text$access <- "Access class"
  p3 <- ggplot() +
    geom_text(data=text, aes(x=5, y=5, label=str_wrap(definition, width=25)),
              size=2, fontface="bold", family="Times") +
xlim(0, 10) +
ylim(0, 10) +
facet_grid(id~access) +
theme(
      axis.text.x = element_blank(),
      axis.text.y = element_blank(),
      axis.ticks = element_blank(),
      panel.spacing = unit(0.1, "lines"),
      panel.border = element_blank(),
      plot.margin =unit(c(0,0.2,0,0),"cm"),
      #plot.background = element_rect(fill = "white"),
      axis.title.x = element_blank(),
      strip.text = element_text(size = 7, family="Times", face="bold", hjust=0.5),
      axis.title.y = element_blank(),
      legend.position = "bottom",
      text=element_text(family="Times")
)
svg("class_priority.svg", height=7, width=7)
#pdf("class_priority.svg", height=7, width=7)
grid.newpage()
pushViewport( viewport(layout = grid.layout(50, 60)) )
print( p2, vp=viewport(layout.pos.row=1:7, layout.pos.col=1:50 ))
print( p, vp=viewport(layout.pos.row=8:50, layout.pos.col=3:50 ))
print( p3, vp=viewport(layout.pos.row=8:46, layout.pos.col=51:60 ))
dev.off()
```

7 File: colorful_2.R

```
library(tidyverse)
data <- read.csv(file="sun.tsv",header=T,sep="\t")
data <- data %>%
  gather(key = "observation", value="value", -c(1,2))
nObsType <- nlevels(as.factor(data$observation))
data <- data %>%
  arrange(group, name)
data$id <- rep( seq(1, nrow(data)/nObsType), each=nObsType)
```

```
data=data[which(data$value!=0),]
data$group_comple<-ifelse(data$value>0,"pro","raw")
label_data <- data %>%
  group_by(id, name) %>%
  summarize(tot=sum(value))
number_of_bar <- nrow(label_data)
angle <- 90 - 360 * (label_data$id-0.5) /number_of_bar
label_data$hjust <- ifelse( angle < -90, 1, 0)
label_data$angle <- ifelse(angle < -90, angle+180, angle)
p <-
  ggplot(data) +
  geom_bar(aes(x=as.factor(id), y=value, fill=group_comple),
           colour="#000033",
           stat="identity", alpha=0.5
           ) +
theme_minimal() +
theme(
      text=element_text(family="serif"),
      legend.key.size = unit(35, "pt")
      ) +
labs(x = "", y = "")+
guides(
       fill = guide_legend(order = 1)
       ) +
geom_text(data=label_data, aes(x=id, y=0, label=name, hjust=hjust),
          color="black", alpha=1, fontface="bold", size=15, angle= 90,
          inherit.aes = FALSE,
          family="Times")
pdf("test.pdf",width=20,height=50)
p
dev.off()
```

8 File: colorful_3_for_volcano.R

```
            library(tidyverse)

            library(RColorBrewer)

            library(ggsci)

            data <- read.csv(file="pearson_filter_volcano_sun.tsv",header=T,sep="\t")
```

```r
data$group_comple<-ifelse(data$FC>0,"up","down")

label_data <- data %>% group_by(colnames(data), group) %>% summarize(tot=sum(FC))

label_data=label_data[order(label_data$group),]

label_data$num=rownames(label_data)

number_of_bar <- nrow(label_data)

angle <- 90 - 360 * (as.numeric(label_data$num)-0.5) /number_of_bar

label_data$hjust <- ifelse( angle < -90, 1, 0)

label_data$angle <- ifelse(angle < -90, angle+180, angle)

colourCount = length(unique(data$id))

getPalette = colorRampPalette(brewer.pal(12, "Paired"))

test1=label_data[nrow(label_data),]

p <-
 ggplot(data) +
 geom_bar(aes(x=as.factor(name), y=fold_change, fill=as.factor(id)),
          stat="identity", alpha=0.5
          ) +
 geom_hline(yintercept = 0, linetype = "twodash", size=0.5, alpha=0.5) +
 geom_text(data=label_data, aes(x=name, y=max(data$fold_change), label=name, hjust=hjust),
          color="black", alpha=0.6, fontface="bold", size=3, angle= label_data$angle,
          inherit.aes = FALSE,
          family="Times") +
 geom_text(data=test1, aes(x=name, y=-40, label="p-value < 0.05"),
          color="black", alpha=0.6, fontface="bold", size=3, hjust=0.5,
          inherit.aes = FALSE,
          family="Times") +
 geom_text(data=test1, aes(x=name, y=-50, label="r(pearson) >= 0.4"),
          color="black", alpha=0.6, fontface="bold", size=3, hjust=0.5,
          inherit.aes = FALSE,
          family="Times") +
 scale_fill_manual(values = getPalette(colourCount)) +
 theme_minimal() +
 ylim(-50,50) +
 theme(
       text=element_text(family="serif"),
           axis.text = element_blank(),
           axis.title.y = element_blank(),
           panel.grid = element_blank(),
           legend.position = c(0.5,0.1),
           legend.title= element_blank(),
           legend.key.width = unit(0.2, "cm"),
       legend.key.height = unit(0.2, "cm"),
       legend.text=element_text(size=6)
      ) +
 labs(x = "", y = "")+
 coord_polar() +
 guides(fill=guide_legend(nrow=11))
ggsave(p,file="test.pdf")
```

9 File: colorful_FC.R

```
library(tidyverse)
library(ggsci)
library(scales)
library(ggrepel)
pal= pal_d3("category20c")(20)
pal= data.frame(matrix(pal,5:4))
pal= c(pal[1,2:3],pal[2,3:1])
data <- read.csv(file="com_lignans_and_iridoids.tsv",header=T,sep="\t")
data <- data[!duplicated(data$id),]
data <- data[which(data$similarity >= 0.5), ]
data$id <- factor(data$id, levels=data[order(data$pro.raw, decreasing = T), colnames(data) %in% c("id")]
label_data <- data
label_data <- label_data[order(label_data$id), ]
label_data$sequ <- seq(nrow(label_data))
number_of_bar <- nrow(label_data)
angle <- 90 - 360 * (label_data$sequ-0.5) /number_of_bar
label_data$hjust <- ifelse( angle < -90, 1, 0)
label_data$angle <- ifelse(angle < -90, angle+180, angle)
p <- ggplot(data, aes(x=as.factor(id), y=log2(pro.raw), fill=log2(pro.raw))) +
  geom_bar(stat="identity", alpha=1) +
  ylim(-7,7) +
  coord_polar(start = 0) +
  labs(fill="Log2(FC)") +
  scale_fill_gradientn(colours = pal, breaks=c(-4,-2,0,2,4)) +
  geom_text(data=label_data, aes(x=id, y=ifelse(log2(pro.raw)>0, log2(pro.raw)+0.3, log2(pro.raw)-0.3),
                               label=ifelse(log2(pro.raw)>1 | log2(pro.raw)<(-1), as.character(id), "
          size=ifelse(log2(label_data$pro.raw)>0, 1.5, 1),
          color="black", fontface="bold",alpha=0.6, inherit.aes = FALSE, family="Times" ) +
annotate("text", x=data[which(data$pro.raw==max(data$pro.raw)), colnames(data) %in% c("id")], y=-5,
        label = "Lignans and iridoids\n(PPCP > 0.5;\nTanimoto similarity > 0.5)\nFC(peak area:\nEucommi
        family="Times", fontface="bold") +
#ggtitle("Lignans and iridoids(PP >= 0.9)\nFC(peak area:\nEucommia-pro/raw)") +
theme_minimal() +
theme(
      #plot.title = element_text(hjust = 0.5, vjust=-190, face="bold", size=12),
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      #panel.background = element_rect(fill="white"),
      axis.text = element_blank(),
      axis.title = element_blank(),
      panel.grid = element_blank(),
      legend.position = c(0.5,0.41),
      legend.title=element_text(face="bold", hjust= -0.5),
      #plot.margin = unit(rep(-1,4), "cm"),
      plot.margin = unit(c(-2, -2, -4, -2), "cm")    # Adjust the margin to make in sort labels are not
)
ggsave(p,file="fc.svg", width=8, height=8)
```

10 File: colorful_line_eucommia.R

```
### line
library(xcms)
path="/media/wizard/back/thermo_mzML_0518"
data <- read.csv(file="com_lignans_and_iridoids.tsv",header=T,sep="\t")
data <- data[!duplicated(data$id),]
data <- data[which(data$similarity >= 0.5), ]
# data <- data[which(data$id==1746|data$id==2081),]
metadata <- data[which(log2(data$pro.raw)>1 | log2(data$pro.raw)<(-1)), colnames(data) %in% c("id", "m
# metadata <- data[, colnames(data) %in% c("id", "m.z")]
dda_file=list.files(path = path, pattern = "*.mzML$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)
dir.create("/media/wizard/back/thermo_mzML_0518/EIC")
tolerance=0.005
for(filename in dda_file){
  dda_data <- readMSData(paste0(path,"/",filename), mode = "onDisk")
  dir.create(paste0(path,"/EIC/EIC_",filename))
  for(number in 1:nrow(metadata)){
    id <- metadata[number,colnames(metadata) %in% c("id")]
    mz <- metadata[number,colnames(metadata) %in% c("m.z")]
    mzrange <- c(as.numeric(mz)-tolerance,as.numeric(mz)+tolerance)
    ex_data <- chromatogram(dda_data, msLevel = 1L, mz = mzrange, aggregationFun = "max")
    ex_data_1 <- ex_data[1,1]
    if(number==1){
      write.table(rtime(ex_data_1),paste0(path, "/EIC/EIC_",filename,"/rt",".tsv"),col.names = FALSE,se
    write.table(intensity(ex_data_1),paste0(path, "/EIC/EIC_",filename,"/",id,"_intensity",".tsv"),col
    print(paste0(filename," >>> ",number,"/",nrow(metadata)))
  }
}
write.table(metadata, paste0(path, "/metadata.tsv"), col.names = T, row.names=F, sep="\t")
### run bash script colorful_line_eucommia_bash.sh
library(ggplot2)
library(ggrepel)
library(ggsci)
library(ggalt)
path="results/EIC_rt_during"
list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
                   full.names = FALSE, recursive = FALSE,
                   ignore.case = FALSE, include.dirs = FALSE)
data <- read.csv(file="com_lignans_and_iridoids.tsv",header=T,sep="\t")
data <- data[which(data$similarity >= 0.5), ]
draw_list <- data[which(log2(data$pro.raw)>1 | log2(data$pro.raw)<(-1)), colnames(data) %in% c("id")]
draw_list <- paste0(as.character(draw_list), ".tsv")
list <- list[list %in% draw_list]
for(file in list){
  savename <- strsplit(file, split=".tsv")
  data <- read.csv(file=paste0(path, "/", file),header=T,sep="\t")
  data <- data[which(data$group!=""),]
  data_anno_mz <- data[1,colnames(data) %in% c("mz")]
  data_anno_rt <- data[1,colnames(data) %in% c("center_rt")]
  tolerance = 0.005
  data_label <- data[which(data$label==1),]
```

```
   data_label$sample <- strsplit(data_label$sample, split=".mzML")
   anno_x_range <- c(0, max(data$rt))
   anno_y_range <- c(0, max(data$intensity))
   delta <- max(data$rt)-min(data$rt)
   if(file=="3938.tsv"){data <- data[which(data$rt<=12.08),] }
   p <- ggplot(data,aes(x=rt, y=intensity, group=sample, colour=color)) +
     geom_line() +
     geom_point(size=0.5, stroke=0) +
     #  geom_label_repel(data=data_label, aes(x=rt, y=intensity, label=sample),
     #                   color="black", alpha=0.5, fontface="bold", size=2, angle= 0, xlim=anno_x_rang
     #                   direction="both", segment.size = 0.2, segment.alpha = 0.3, force = 1,
     #                   #nudge_x = runif(1, min=delta*(1/18), max=delta*(1/10))*sample(c(1,-1),1),
     #                   nudge_y = max(data$intensity)*(1/10),
     #                   inherit.aes = FALSE, hjust = 0,
     #                   family="Times") +
     scale_color_manual(values = c("Blank"="#4A6990FF",
                                   "Non feature"="#B8B8B8FF",
                                   "Pro_Eucommia"="#E6550DFF",
                                   "Raw_Eucommia"="#3182BDFF")) +
#drug green
labs(color="Peak attribution", x="RT (min)", y="Intensity") +
annotate("text", x = min(data$rt), y = max(data$intensity)*(17/20),
         label = paste0("ID: ",strsplit(file, split=".tsv")),
         color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +
annotate("text", x = min(data$rt), y = max(data$intensity)*(16/20),
         label = paste0("Precursor m/z: ",data_anno_mz-tolerance," ~ ",data_anno_mz+tolerance),
         color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +
annotate("text", x = min(data$rt), y = max(data$intensity)*(15/20),
         label = paste0("RT (min): ",data_anno_rt),
         color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +
#theme_minimal() +
theme(text=element_text(family="Times"),
      legend.position = c(0.85,0.75),
      #axis.line = element_line(colour = "black", size=0.2),
      legend.background = element_rect(fill = "transparent", color = "transparent"),
      plot.margin = unit(c(3, 1, 3, 1), "cm"))
ggsave(p,file=paste0(path, "/", savename,".svg"),width=8,height=6.5)
print(paste0("Finish >>> ",file))
 }
 #########################
 #########################
```

```r
#########################
library(ggplot2)
library(ggrepel)
library(ggsci)
library(ggalt)
library(hrbrthemes)
path="results/EIC_rt_during"
list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)
data <- read.csv(file="com_lignans_and_iridoids.tsv",header=T,sep="\t")
data <- data[which(data$similarity >= 0.5), ]
draw_list <- data[which(log2(data$pro.raw)>1 | log2(data$pro.raw)<(-1)), colnames(data) %in% c("id")]
draw_list <- paste0(as.character(draw_list), ".tsv")
list <- list[list %in% draw_list]
n=0
for(file in list){
id <- strsplit(file, split=".tsv")
n=n+1
savename <- strsplit(file, split=".tsv")
data <- read.csv(file=paste0(path, "/", file),header=T,sep="\t")
data <- data[which(data$group!=""),]
data_anno_mz <- data[1,colnames(data) %in% c("mz")]
data_anno_rt <- data[1,colnames(data) %in% c("center_rt")]
tolerance = 0.005
data_label <- data[which(data$label==1),]
data_label$sample <- strsplit(data_label$sample, split=".mzML")
if(file=="3938.tsv"){data <- data[which(data$rt<=12.08),] }
if(file=="3380.tsv"){data <- data[which(data$rt>=12.55),] }
data$id <- id
if(n==1){sum_data=data[,c(1:6,9)]}else{sum_data<-rbind(sum_data, data[,c(1:6,9)])}
#text1=c("x"=min(data$rt), "y"=max(data$intensity)*(17/20), "label"=paste0("ID: ", id))
#
text2=c("x"=min(data$rt), "y"=max(data$intensity)*(17/20),
       "label"=paste0("Precursor m/z: ",data_anno_mz-tolerance," ~ ",data_anno_mz+tolerance))
text3=c("x"=min(data$rt), "y"=max(data$intensity)*(15/20), "label"=paste0("RT (min): ",data_anno_rt) )
text=data.frame(rbind(text2, text3))
text$id <- id
if(n==1){sum_text=text}else{sum_text<-rbind(sum_text, text)}
print(paste0("Finish >>> ",file))}
#######################################  plot
```

```r
 ########################################
 select <- c(279, 458, 574, 1107, 1445, 2227, 2529, 2664, 2824, 3380, 3918, 3938)
 sum_data <- sum_data[sum_data$id %in% select,]
 sum_data$id <- paste0("ID:", sum_data$id)
 sum_text <- sum_text[sum_text$id %in% select,]
 sum_text$id <- paste0("ID:", sum_text$id)
 p <- ggplot(sum_data, aes(x=rt, y=intensity, group=sample, colour=color)) +
  geom_line() +
  geom_point(size=0.5, stroke=0) +
  scale_color_manual(values = c("Blank"="#4A6990FF",
               "Non feature"="#B8B8B8FF",
               "Pro_Eucommia"="#E6550DFF",
               "Raw_Eucommia"="#3182BDFF")) +
  #drug green
  labs(color="Peak attribution", x="RT (min)", y="Intensity") +
  geom_text(data=sum_text, aes(x=as.numeric(x), y=as.numeric(y), label=label),
      hjust=0, fontface="bold", alpha=1, size=3.5, inherit.aes = FALSE, family="Times") +
  #theme_minimal() +
  scale_y_continuous(labels = scales::scientific) +
  facet_wrap(~as.character(id), scales="free", ncol=3) +
  guides(color=guide_legend(nrow=1)) +
  theme_ipsum() +
  theme(text=element_text(family="Times"),
    plot.background = element_rect(fill = "white", size=0),
    legend.position = "bottom",
    axis.title.y = element_text(face="bold", size=20, hjust=0.5, family="Times"),
    axis.title.x = element_text(face="bold", size=20, hjust=0.5, family="Times"),
    legend.key.height = unit(1, "cm"),
    legend.key.width = unit(1, "cm"),
    legend.text = element_text(size=15),
    legend.title = element_text(size=15, face="bold"),
    #axis.line = element_line(colour = "black", size=0.2),
    legend.background = element_rect(fill = "transparent", color = "transparent"),
    strip.text = element_text(size=15, face="bold", family="Times"),
    plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")
    #panel.spacing = unit(c(0,0,0,0),"cm")
    )
 ggsave(p,file=paste0("results", "/", "ms1_line.svg"), width=12, height=15)
 ######################################## plot
 ########################################
library(ggplot2)
library(ggforce)
library(ggrepel)
path="results/ms2_figures_label"
n=list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
             full.names = FALSE, recursive = FALSE,
             ignore.case = FALSE, include.dirs = FALSE)
select <- c(279, 458, 574, 1107, 1445, 2227, 2529, 2664, 2824, 3380, 3918, 3938)
# select <- c(1746, 2081)
n <- n[n %in% paste0(select, ".tsv")]
m=0
for(x in n){
  m=m+1
  file <- strsplit(x, split=".tsv")
  id <- as.character(file)
  savename=paste0(file,".svg")
  data <- read.csv(file=paste0(path, "/", file,".tsv"),header=T,sep="\t")
  data$id <- id
  if(m==1){
    sum_data=data[,c(1:3,10)]
  }else{
    sum_data<-rbind(sum_data, data[,c(1:3,10)])
  }
  text1=c("x"=max(data$mz), "y"=90, "label"=paste0("Precursor m/z: ",data[1,colnames(data) %in% c("prec
  text2=c("x"=max(data$mz), "y"=72,
          "label"=paste0("RT (min): ", data[1,colnames(data) %in% c("RT..min.")]) )
  text3=c("x"=max(data$mz), "y"=54, "label"=paste0("Tanimoto similarity: ", data[1,colnames(data) %in% c
  text=data.frame(rbind(text1, text2, text3))
  text$id <- id
  if(m==1){
    sum_text=text
  }else{
    sum_text<-rbind(sum_text, text)
  }
  print(file)
}
 #####################################
 #####################################
 data=sum_data
 data$id <- paste0("ID:", data$id)
 sum_text$id <- paste0("ID:", sum_text$id)
 p <- ggplot(data) +
  geom_segment(aes(x=mz, xend=mz, y=0, yend=rel.intensity), color=ifelse(data$rel.intensity>0,"black","
  geom_point(data=data[which(data$match>=1),], size=0.9, aes(x=mz, y=rel.intensity),
            color=ifelse(data[which(data$match>=1),]$rel.intensity>0,"black","red")) +
  geom_text(data=sum_text, aes(x=as.numeric(x)*3/5, y=as.numeric(y), label=label),
        hjust=0, fontface="bold", alpha=1, size=3.5, inherit.aes = FALSE, family="Times") +
  facet_wrap(~as.character(id), scales="free_x", ncol=3) +
  labs(x="m/z", y="Relative intensity") +
  theme(text=element_text(family="Times"),
  panel.background=element_rect(fill="white", size=0),
  panel.grid=element_line(color="grey85"),
  axis.title.y = element_text(face="bold", size=20, hjust=0.5, family="Times"),
  axis.title.x = element_text(face="bold", size=20, hjust=0.5, family="Times"),
  strip.text = element_text(size=15, face="bold", family="Times", hjust=0),
  strip.background  = element_rect(fill = "white", size = 0),
  plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")
  )
ggsave(p,file=paste0("results", "/", "ms2_segment.svg"),width=12,height=15)
###################################
###################################
library(ggplot2)
library(ggtree)
library(aplot)
library(tidyr)
library(rayshader)
library(ggsci)
library(reshape2)
 datapath="results/re_neg_RT.tsv"
 select <- c(279, 458, 574, 1107, 1445, 2227, 2529, 2664, 2824, 3380, 3918, 3938)
 data <- read.csv(file=datapath, header=T,sep="\t")
 grouppath="results/stat_classification.tsv"
 group <- read.csv(file=grouppath, header=T,sep="\t")
 col <- grep("row.ID|area", colnames(data), ignore.case=T)
 data <- data[data$row.ID %in% select, c(col)]
 colnames(data) <- c("id", "blank", "Pro-Eu1", "Pro-Eu2", "Pro-Eu3", "Raw-Eu1", "Raw-Eu2", "Raw-Eu3")
 rownames(data) <- paste0("ID: ",data$id)
 group_anno <- merge(data, group[, colnames(group) %in% c("id", "specific", "specific_pp")], by="id", a
 group_anno$id <- paste0("ID: ",group_anno$id)
 data <- data[, c(-1,-2)]
 data <- log2(data)
 ############### heatmap
 savename="results/features_heatmap.svg"
```

```r
 phr <- hclust(dist(data)) %>%
    ggtree(layout="rectangular", branch.length="none", size=1.2) +
    theme(
        plot.margin = unit(c(0, 0, 0, 0), "cm")
    )
phc <- hclust(dist(t(data))) %>%
    ggtree(layout="rectangular", branch.length="none", size=1.2) + layout_dendrogram() +
    theme(
        plot.margin = unit(c(0, 0, 0, 0), "cm")
    )
data$names <- rownames(data)
 p1 <- gather(data, 1:(ncol(data)-1), key="condition", value='expr')  ## keep the last col
 p1 <- merge(p1, group_anno[, colnames(group_anno) %in% c("id", "specific")], by.x="names", by.y="id", a
 anno <- ggplot(p1, aes(x="group", y=names)) +
  geom_tile(size=2, aes(fill=specific), color="black") +
  scale_fill_npg() +
  labs(x="", y="", fill="Classes") +
  #guides(fill="none") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle=90, face="bold"),
    axis.text.y = element_blank(),
    legend.title = element_text(face="bold"),
    #legend.text = element_text(size=20),
    legend.key.width = unit(0.5, "cm"),
        legend.key.height = unit(1, "cm"),
        text=element_text(family="Times", size=15, face="bold"),
        plot.margin = unit(c(0, 0, 0, 0), "cm")
    )
 pp <- ggplot(p1,aes(x=condition,y=names,fill=expr)) +
  geom_tile(size=1.5, color="black") +
  #theme_minimal()+
  scale_fill_gradientn(colors=c("#3182BDFF", "white", "#E6550DFF")) +
  #scale_color_gradientn(colors=c( "black", "#8F7700FF", "black")) +
  #scale_fill_gradient2(low="#3182BDFF", mid="black", high="#E6550DFF", midpoint=21) +
  scale_y_discrete(position="right") +
  guides(color="none") +
  labs(x="Sample name", y="", fill="Log2(peak area)") +
  theme_minimal()+
  theme(
    axis.text.x = element_text(angle=90),
    #axis.text.y = element_blank(),
    legend.title = element_text(face="bold"),
    #legend.text = element_text(size=20),
    legend.key.width = unit(0.5, "cm"),
        legend.key.height = unit(1, "cm"),
        text=element_text(family="Times", size=15, face="bold"),
        plot.margin = unit(c(0.5, 0, 0.5, 0.5), "cm")
    )
 pp_com <- pp %>%
  insert_right(anno, width=.15) %>%
  insert_left(phr, width=.1) %>%
  insert_top(phc, height=.1)
 ggsave(pp_com,file=savename, width=10, height=12)
 ###############

# extended data
 library(ggplot2)
 library(ggrepel)
 library(ggsci)
 library(ggalt)
 library(hrbrthemes)
 path="EIC_rt_during"
 list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
            full.names = FALSE, recursive = FALSE,
            ignore.case = FALSE, include.dirs = FALSE)
 data <- read.csv(file="com_lignans_and_iridoids.tsv",header=T,sep="\t")
 data <- data[which(data$similarity >= 0.5), ]
 n=0
 for(file in list){
 id <- strsplit(file, split=".tsv")
 n=n+1
 savename <- strsplit(file, split=".tsv")
 data <- read.csv(file=paste0(path, "/", file),header=T,sep="\t")
 data <- data[which(data$group!=""),]
 data_anno_mz <- data[1,colnames(data) %in% c("mz")]
 data_anno_rt <- data[1,colnames(data) %in% c("center_rt")]
 tolerance = 0.005
 data_label <- data[which(data$label==1),]
 data_label$sample <- strsplit(data_label$sample, split=".mzML")
 if(file=="3938.tsv"){data <- data[which(data$rt<=12.08),] }
 if(file=="3380.tsv"){data <- data[which(data$rt>=12.55),] }
```

```r
data$id <- id
if(n==1){sum_data=data[,c(1:6,9)]}else{sum_data<-rbind(sum_data, data[,c(1:6,9)])}
#text1=c("x"=min(data$rt), "y"=max(data$intensity)*(17/20), "label"=paste0("ID: ", id))
#
text2=c("x"=min(data$rt), "y"=max(data$intensity)*(17/20),
       "label"=paste0("Precursor m/z: ",data_anno_mz-tolerance," ~ ",data_anno_mz+tolerance))
text3=c("x"=min(data$rt), "y"=max(data$intensity)*(15/20), "label"=paste0("RT (min): ",data_anno_rt) )
text=data.frame(rbind(text2, text3))
text$id <- id
if(n==1){sum_text=text}else{sum_text<-rbind(sum_text, text)}
print(paste0("Finish >>> ",file))}
#######################################  plot
#######################################
select <- c(1746,2081)
sum_data <- sum_data[sum_data$id %in% select,]
sum_data$id <- paste0("ID:", sum_data$id)
sum_text <- sum_text[sum_text$id %in% select,]
sum_text$id <- paste0("ID:", sum_text$id)
p <- ggplot(sum_data, aes(x=rt, y=intensity, group=sample, colour=color)) +
 geom_line() +
 geom_point(size=0.5, stroke=0) +
 scale_color_manual(values = c("Blank"="#4A6990FF",
               "Non feature"="#B8B8B8FF",
               "Pro_Eucommia"="#E6550DFF",
               "Raw_Eucommia"="#3182BDFF")) +
 #drug green
 labs(color="Peak attribution", x="RT (min)", y="Intensity") +
 geom_text(data=sum_text, aes(x=as.numeric(x), y=as.numeric(y), label=label),
       hjust=0, fontface="bold", alpha=1, size=5, inherit.aes = FALSE, family="Times") +
 #theme_minimal() +
 scale_y_continuous(labels = scales::scientific) +
 facet_wrap(~as.character(id), scales="free", ncol=3) +
 guides(color=guide_legend(nrow=1)) +
 theme_ipsum() +
 theme(text=element_text(family="Times"),
   plot.background = element_rect(fill = "white", size=0),
   legend.position = "bottom",
   axis.title.y = element_text(face="bold", size=10, hjust=0.5, family="Times"),
   axis.title.x = element_text(face="bold", size=10, hjust=0.5, family="Times"),
   legend.key.height = unit(1, "cm"),
   legend.key.width = unit(1, "cm"),
   legend.text = element_text(size=15),
   legend.title = element_text(size=15, face="bold"),
   #axis.line = element_line(colour = "black", size=0.2),
   legend.background = element_rect(fill = "transparent", color = "transparent"),
   strip.text = element_text(size=15, face="bold", family="Times"),
   plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")
   #panel.spacing = unit(c(0,0,0,0),"cm")
   )
ggsave(p,file=paste0("extended_ms1_line.svg"), width=12, height=5)
```

11 File: colorful_line_fecal.R

```
### line

library(ggplot2)

library(ggrepel)

library(ggsci)

library(ggalt)

path="results/EIC_rt_during"

list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)

for(file in list){

savename <- strsplit(file, split=".tsv")

data <- read.csv(file=paste0(path, "/", file),header=T,sep="\t")

data <- data[which(data$group!=""),]

data_anno_mz <- data[1,colnames(data) %in% c("mz")]

data_anno_rt <- data[1,colnames(data) %in% c("center_rt")]

tolerance = 0.005
```

```
data_label <- data[which(data$label==1),]

anno_x_range <- c(0, max(data$rt))

anno_y_range <- c(0, max(data$intensity))

delta <- max(data$rt)-min(data$rt)



p <- ggplot(data,aes(x=rt, y=intensity, group=sample, colour=color)) +

 geom_line() +

 geom_point(size=0.5, stroke=0) +

 geom_label_repel(data=data_label, aes(x=rt, y=intensity, label=sample),
                    color="black", alpha=0.5, fontface="bold", size=2, angle= 0, xlim=anno_x_range, y
                    direction="both", segment.size = 0.2, segment.alpha = 0.3, force = 1,
                    nudge_x = runif(1, min=delta*(1/18), max=delta*(1/10)), nudge_y = max(data$intens
                    inherit.aes = FALSE, hjust = 0,
                    family="Times") +

 scale_color_manual(values = c("control"="#4A6990FF","drug"="#95CC5EFF","model"="#374E55FF",

               "Non feature"="#B8B8B8FF",

               "pro_low"="#FDAE6BFF","pro_medium"="#FD8D3CFF","pro_high"="#E6550DFF",

               "raw_low"="#9ECAE1FF","raw_medium"="#6BAED6FF","high_raw"="#3182BDFF")) +
 #drug green
 labs(color="Peak attribution", x="RT (min)", y="Intensity") +

 annotate("text", x = min(data$rt), y = max(data$intensity)*(16/20),
       label = paste0("Precursor m/z: ",data_anno_mz-tolerance," ~ ",data_anno_mz+tolerance),
           color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +

 annotate("text", x = min(data$rt), y = max(data$intensity)*(15/20),
       label = paste0("RT (min): ",data_anno_rt),
           color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +
 #theme_minimal() +

 theme(text=element_text(family="serif"),

   legend.position = c(0.85,0.75),

   #axis.line = element_line(colour = "black", size=0.2),

   legend.background = element_rect(fill = "transparent", color = "transparent"),

   plot.margin = unit(c(3, 1, 3, 1), "cm"))

ggsave(p,file=paste0(path, "/", savename,".svg"),width=8,height=6.5)

print(paste0("Finish >>> ",file))}
```

12 File: colorful.R

```
           library(tidyverse)

           library(RColorBrewer)

           data <- read.csv(file="Sun_0827-1.tsv",header=T,sep="\t")

           data <- data %>% gather(key = "observation", value="value", -c(1,2))

           nObsType <- nlevels(as.factor(data$observation))

           data <- data %>% arrange(group, name)

           data$id <- rep( seq(1, nrow(data)/nObsType), each=nObsType)

           label_data <- data %>% group_by(id, name) %>% summarize(tot=sum(value))

           number_of_bar <- nrow(label_data)

           angle <- 90 - 360 * (label_data$id-0.5) /number_of_bar

           label_data$hjust <- ifelse( angle < -90, 1, 0)
```

```
            label_data$angle <- ifelse(angle < -90, angle+180, angle)

            colourCount = length(unique(data$observation))

            getPalette = colorRampPalette(brewer.pal(12, "Paired"))

            p <-
             ggplot(data) +
             geom_bar(aes(x=as.factor(id), y=value, fill=observation),
                      stat="identity", alpha=0.5
                      ) +
             scale_fill_manual(values = getPalette(colourCount)) +
             ylim(-10,13) +
             theme_minimal() +
             theme(
                   text=element_text(family="serif"),
                       axis.text = element_blank(),
                       axis.title.y = element_blank(),
                       panel.grid = element_blank(),
                       legend.position = c(0.5,0.03),
                       legend.title= element_blank(),
                       legend.key.width = unit(0.2, "cm"),
                   legend.key.height = unit(0.2, "cm"),
                   legend.text=element_text(size=6)
                  ) +
            labs(x = "", y = "")+
            guides(fill=guide_legend(nrow=10))+
            coord_polar() +
            geom_text(data=label_data, aes(x=id, y=0, label=name, hjust=hjust),
                      color="black", alpha=1, fontface="bold", size=2, angle= label_data$angle,
                      inherit.aes = FALSE,
                      family="Times")

            pdf("test.pdf")
            p
            dev.off()
```

13 File: create_mcnebula.R

```
library(usethis)
library(devtools)
```

```r
library(progress)
## data reformat
library(pbapply)
library(data.table)
library(dplyr)
library(MSnbase)
library(igraph)
## visualize
library(ggplot2)
library(ggraph)
library(ggsci)
library(stringr)
library(grid)
## suggest
#library(ChemmineR)
library(ChemmineOB)
library(rsvg)
library(grImport2)
library(ggimage)
load_all("~/MCnebula/R")
usethis::create_package("MCnebula")
load_all()
check()
use_mit_license("Lichuang Huang")
use_package("data.table", type="Imports")
use_package("dplyr", type="Imports")
use_package("progress", type="Imports")
```

14 File: exact_mass.R

```r
isotopic_mass <- c("H"=1.007825,
                   "C"=12.0,
                   "N"=14.003074,
                   "O"=15.994915,
                   "F"=18.000938,
                   "P"=30.973762,
                   "S"=31.972071)
exact_mass <- function(formula, db=isotopic_mass){
  vector <- unlist(strsplit(formula, split=""))
  mass=0
  for(i in 1:length(vector)){
    if(vector[i] %in% names(db)){
      if(i == length(vector)){
        mass = mass + db[[vector[i]]]
        return(mass)
      }
      options (warn = -1)
      test <- as.numeric(vector[i+1])
      options (warn = 1)
      if(is.na(test) == T){
        mass = mass + db[[vector[i]]]
      }else{
        mass = mass + db[[vector[i]]] * test
      }
    }else{
      next
    }
  }
  return(mass)
}
```

15 File: auto_classy.R

```
auto_classy <-
  function(
           df,
           ...
           ){
    ## classyfireR
    list <- by_group_as_list(df, ".id")
    pbapply::pblapply(list, base_auto_classy,
                        ...)
  }
base_auto_classy <-
  function(
           df
           ){
    .id <- df[1,][[".id"]]
    lapply(df[["InChIKey"]], base2_classy,
                      .id = .id)
  }
```

```
base2_classy <-
  function(
           inchi,
           .id
           ){
    ch <- try(read_tsv(paste0(.id)), silent = T)
    if(class(ch) == "try-error"){
      ch <- classyfireR::get_classification(inchi)
    }else{
      return()
    }
    if(is.null(ch)){
      return()
    }else{
      ch <- classyfireR::classification(ch)
      write_tsv(ch, paste0(.id))
    }
  }
```

16 File: by_group_as_list.R

```
by_group_as_list <-
  function(
           df,
           colnames
           ){
    assign("envir_meta", environment(), envir = parent.env(environment()))
    vector <- unique(df[[colnames]])
    list <- lapply(vector, by_group_as_list_select,
                   colNames = colnames)
    names(list) <- vector
    return(list)
  }
by_group_as_list_select <-
  function(
           KEY,
           df = get("df", envir = get("envir_meta")),
           colNames
           ){
    df <- df[which(df[[colNames]] == KEY), ]
    return(df)
  }
```

17 File: collate_structure_table.R

```
collate_structure_table <-
  function(
           file = "method_pick_formula_excellent.structure.tsv",
           class = "../canopus_summary.tsv",
           cut_tanimoto = 0.4,
           delete_null = T,
           ...
           ){
    order = c(".id", "name", "molecularFormula", "adduct",
              "most specific class", "inchikey2D", "tanimotoSimilarity")
    df <- read_tsv(file)
    df <- dplyr::filter(df, tanimotoSimilarity >= cut_tanimoto) %>%
      dplyr::select(.id, name, tanimotoSimilarity, molecularFormula, inchikey2D) %>%
      dplyr::mutate(.id = as.character(.id))
    class <- read_tsv(class) %>%
      dplyr::select(name, `most specific class`, adduct) %>%
      dplyr::mutate(.id = stringr::str_extract(name, "(?<=_)[0-9]{1,4}$")) %>%
      dplyr::select(.id, `most specific class`, adduct)
    df <- merge(df, class, by = ".id", all.x = T, sort = T) %>%
      dplyr::select(all_of(order)) %>%
      dplyr::distinct(name, .keep_all = T)
    if(delete_null == T)
      df <- dplyr::filter(df, name != "null")
    write_tsv(df, "table_of_pdf.tsv")
    return(df)
  }
```

18 File: deal_with_msp_record.R

```
deal_with_msp_record <-
  function(
           string,
           id_prefix,
           cache,
           store,
           id = get("id", envir = cache),
           input = c(name = "NAME",
                     mass = "PRECURSORMZ",
                     adduct = "PRECURSORTYPE",
                     rt = "RETENTIONTIME"),
           other = c("NAME", "PRECURSORMZ", "PRECURSORTYPE",
                     "FORMULA", "Ontology", "INCHIKEY", "SMILES",
                     "RETENTIONTIME", "CCS", "IONMODE",
                     "INSTRUMENTTYPE","INSTRUMENT",
                     "COLLISIONENERGY", "Comment", "Num Peaks"),
           output = c(begin = "BEGIN IONS",
                     id = "FEATURE_ID=",
                     mass = "PEPMASS=",
                     charge = "CHARGE=",
                     rt = "RTINSECONDS=",
                     level = "MSLEVEL=",
                     end = "END IONS")
           ){
    ## ------------------------------------------------------------------------
    ## get name and value
    name = get_name(string)
    name = ifelse(is.na(name) == T, "", name)
    if(grepl("^[A-Z]", name) == T){
      value = get_value(string)
    }
    ## ------------------------------------------------------------------------
    cat = 0
    if(name == input[["name"]]){
      catapp(output[["begin"]], "\n")
      ## id update
      id = id + 1
      assign("id", id, envir = cache)
      ## output
      cat = 1
      p = output[["id"]]
      s = paste0(id_prefix, id)
      ## new var in envir: store
      info <- data.table::data.table(.id = s, name = value)
      assign(paste0(id), info, envir = store)
    ## ------------------------------------------------------------------------
    }else if(name == input[["mass"]]){
      cat = 1
      p = output[["mass"]]
      s = value
    ## ----------------------------------------------------------------------
    }else if(name == input[["adduct"]]){
      cat = 1
      p = output[["charge"]]
      s = ifelse(grepl("]-|]+", value) == F, "0",
                 ifelse(grepl("]-", value), "-1", "+1"))
      id <- get("id", envir = cache)
      info = get(paste0(id), envir = store)
      info[["charge"]] = s
      assign(paste0(id), info, envir = store)
    ## ----------------------------------------------------------------------
    }else if(name == input[["rt"]]){
      cat = 1
      p = output[["rt"]]
      s = value
    ## ----------------------------------------------------------------------
    }else if(name == "Num Peaks"){
      cat = 0
      id <- get("id", envir = cache)
      info = get(paste0(id), envir = store)
      catapp(output[["level"]], "1\n")
      catapp(info[["PRECURSORMZ"]], "\n")
      catapp(output[["end"]], "\n")
      catapp("\n")
      ## begin mass level 2
      catapp(output[["begin"]], "\n")
      catapp(output[["id"]], info[[".id"]], "\n")
      catapp(output[["mass"]], info[["PRECURSORMZ"]], "\n")
      catapp(output[["charge"]], info[["charge"]], "\n")
      catapp(output[["rt"]], info[["RETENTIONTIME"]], "\n")
      catapp(output[["level"]], "2\n")
    ## ----------------------------------------------------------------------
    }else if(grepl("^[0-9]", string)){
      cat = 2
      p = get_name(string, sep = "\t")
      s = get_value(string, sep = "\t")
    }else if(string == ""){
      cat = 1
      p = output[["end"]]
```

```
      s = "\n"
    }
    ## ---------------------------------------------------------------------
    if(cat == 1){
      catapp(p, s, "\n")
    }else if(cat == 2){
      catapp(p, s, "\n", sep = " ")
    }
    ## ---------------------------------------------------------------------
    ## data store
    if(name %in% other == T){
      id <- get("id", envir = cache)
      info = get(paste0(id), envir = store)
      info[[name]] = value
      assign(paste0(id), info, envir = store)
    }
    return()
    ## ---------------------------------------------------------------------
    ## output
  }
catapp <-
  function(
           ...,
           sep = "",
           mgf = get("mgf", envir = get("envir_meta"))
           ){
    cat(paste(..., sep = sep), file = mgf, append = T)
  }
get_value <-
  function(
           string,
           sep = ": "
           ){
    string <- unlist(strsplit(string, split = sep))
    return(string[2])
  }
get_name <-
  function(
           string,
           sep = ": "
           ){
```

```
    string <- unlist(strsplit(string, split = sep))
    return(string[1])
  }
```

19 File: inchi_curl.R

```
inchi_curl <-
  function(
           key,
           .id,
           type = "inchikey",
           get = "InChIkey",
           save = paste0(.id, ".csv")
           ){
    http = paste0("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/", type, "/")
    http_end = paste0("/property/", paste(get, collapse = ","), "/CSV > ")
    curl <- "curl -s --connect-timeout 20 --retry 100 --retry-delay 30 "
    curl_http <- paste0(curl, http)
    system(paste0(curl_http, key, http_end, save))
  }
int_inchi_curl <-
  function(
           seq,
           type = "inchikey",
           get = "InChIkey",
           ...
           ){
    init <- dload[seq, "init"]
    .id <- dload[seq, ".id"]
    save <- paste0(.id, ".csv")
    while(class(try(read.csv(save), silent = T))[1] == "try-error"){
      inchi_curl(init, .id,
                 get = get,
                 type = type,
                 ...)
    }
  }
```

20 File: list_merge_df.R

```
list_merge_df <-
  function(
           list,
           df,
           ...
           ){
    assign("envir_meta", environment(), parent.env(environment()))
    list <- lapply(list, merge,
                   y = get("df", envir = get("envir_meta")),
                   ...)
    return(list)
  }
```

21 File: msp_to_mgf.R

```
msp_to_mgf <-
  function(
           name,
           id_prefix,
           path = "~/Downloads/msp",
           write_meta_data = paste0(path, "/", name, ".meta.tsv")
           ){
    msp <- read_msp(paste0(path, "/", name))
    cache <- new.env()
    store <- new.env()
    assign("id", 0, envir = cache)
    mgf <- paste0(path, "/", name, ".mgf")
    assign("envir_meta", environment(), envir = parent.env(environment()))
    cat("", file = mgf)
    pbapply::pblapply(msp[[1]], deal_with_msp_record,
                      id_prefix = id_prefix,
                      cache = cache,
                      store = store)
    set <- ls(envir = store)
    meta_data <- lapply(set, get_envir_df,
                               envir = store)
    meta_data <- data.table::rbindlist(meta_data)
    if(is.null(write_meta_data) == F){
      write_tsv(meta_data, write_meta_data)
```

```r
    }
    return(meta_data)
  }
read_msp <-
  function(
           filepath
           ){
    msp <- data.table::fread(filepath, sep = NULL, header = F)
  }
get_envir_df <-
  function(
           var,
           envir
           ){
    df <- get(var, envir = envir)
    return(df)
  }
```

22 File: pretty_table.R

```r
pretty_table <-
  function(
           df,
           title = "compounds summary",
           subtitle = "LC-MS",
           footnote = "Compounds summary",
           filename = "tmp.html",
           path = ".",
           return_gt = T,
           font = "Times",
           shorter_name = T
           ){
    title = paste0("**", Hmisc::capitalize(title), "**")
    subtitle = paste0("**", Hmisc::capitalize(subtitle), "**")
    colnames(df) <- Hmisc::capitalize(colnames(df))
    t <- gt(df) %>%
      opt_table_font(font=list(font)) %>%
      tab_header(title = md(title),
                 subtitle = md(subtitle)) %>%
      opt_align_table_header(align = "left") %>%
      tab_footnote(footnote = footnote,
                   locations = cells_title(groups = c("title"))) %>%
      opt_table_lines(extent = c("none")) %>%
      cols_align(align = "left",
                 columns = everything()) %>%
      tab_style(style = cell_borders(sides = c("top", "bottom"),
                                     color = "black",
                                     weight = px(1.5),
                                     style = "solid"),
                locations = cells_column_labels()) %>%
      tab_style(style = cell_text(v_align="top"),
                locations = cells_column_labels(columns = everything())) %>%
      tab_style(style = cell_borders(sides = c("bottom"),
                                     color = "black",
                                     weight = px(1.5),
                                     style = "solid"),
                locations = cells_body(columns=everything(),
                                       rows=nrow(df))) %>%
      tab_style(style = cell_text(v_align="top"),
                locations = cells_body(columns = everything()))
    if(shorter_name == T){
      t <- t %>%
        cols_width(Name ~ px(300))
    }
    gtsave(t, filename, path)
    if(return_gt == T)
      return(t)
  }
```

23 File: read_tsv.R

```
read_tsv <- function(path){
  file <- data.table::fread(input=path, sep="\t", header=T, quote="", check.names=F)
  return(file)
}
write_tsv <-
  function(x, filename){
    write.table(x, file = filename, sep = "\t", col.names = T, row.names = F, quote = F)
  }
```

24 File: select_app.R

```
select_app <-
  function(
           df,
           col
           ){
    df <- dplyr::select(df, all_of(col))
    return(df)
  }
```

25 File: show_chem.R

```
## ---------------------------------------------------------------------
## a function to fast show df
show_meta <-
  function(
           x,
           df = meta
           ){
    prefix <- stringr::str_extract(df[1,]$".id", "^[a-z]{1,100}(?=[0-9])")
    df <- dplyr::filter(df, .id == paste0(prefix, x)) %>%
      data.table::data.table()
    return(df)
  }
## ---------------------------------------------------------------------
getk <-
  function(
           x,
           col = "INCHIKEY"
           ){
    df <- show_meta(x)
    return(df[[col]])
  }
## ---------------------------------------------------------------------
show_stru <-
  function(
           df,
           key = c("smiles", "SMILES")
           ){
    key <- key[key %in% colnames(df)][1]
    smiles <- df[[key]]
    molconvert_structure(smiles)
  }
## ---------------------------------------------------------------------
auto <-
  function(
           id
           ){
    id <- as.character(substitute(id))
    show_meta(id) %>%
      show_stru()
  }
```

26 File: vis_via_molconvert_nebulae.R

```
vis_via_molconvert_nebulae <-
  function(
           nebula_name
           ){
    df <- dplyr::filter(.MCn.nebula_index, name == nebula_name)
    stru <- dplyr::filter(.MCn.structure_set, .id %in% df$".id")
    vis_via_molconvert(stru$smiles, stru$".id")
    return("Done")
  }
vis_via_molconvert <-
  function(
           smiles_set,
           id_set,
           output = paste0(.MCn.output, "/", .MCn.results, "/tmp/structure")
           ){
    if(file.exists(output) == F){
      dir.create(paste0(.MCn.output, "/", .MCn.results))
      dir.create(output)
    }
    pbapply::pbmapply(molconvert_structure,
           smiles_set,
           id_set,
           MoreArgs = list(output = output)
    )
    return("Done")
  }
```

```r
## ---------------------------------------------------------------------
molconvert_structure <-
  function(
           smiles,
           id = "tmp",
           output = "."
           ){
    file = tempfile()
    system(paste0("molconvert mol \"", smiles, "\" -o ", file))
    system(paste0("obabel ", file, " -imol -osvg -O ", file, " > /dev/null 2>&1"))
    system(paste0("sed -i \'s/white/transparent/g\' ", file))
    system(paste0("cairosvg -f svg ", file, " -o ", output, "/", id, ".svg"))
    return()
  }
```

27 File: fecal_neg.R

```r
library(tidyverse)

path="~/operation/re_fecal_neg"
setwd(path)
mzmine <- read.csv(file="fecal_neg_mzmine.csv", header=T, sep=",", check.name=F)
#filter the blank
mzmine <- mzmine[,!(1:ncol(mzmine) %in% grep("Blank",colnames(mzmine)))]
metadata <- read.csv(file="metadata.tsv", header=T, sep="\t")
data_area <- mzmine[, grep("ID|m/z|retention|area", colnames(mzmine))]
rownames(data_area) <- data_area$"row ID"
df_area <- data.frame(
              t(
            data_area[,grep("area", colnames(data_area))]
              )
)
df_area$file <- rownames(df_area)
df_area <- separate(df_area, col="file", into=c("file", "peak", "area"), remove=T, sep=" ")
# df_area[1:10,(ncol(df_area)-5):ncol(df_area)]
df_area <- df_area[, !(colnames(df_area) %in% c("peak", "area"))]
df_area <- merge(df_area, metadata, all.x=T, by.x="file", by.y="file")
# finish reformat
# compare group as follow
element1 <- c("pro", "raw", "model", "control")
element2 <- c("high", "medium", "low", "multi")
```

```r
c_group <- data.frame(c(rep("model", 3), "pro", "multi"), c(element1[c(1,2,4)], "raw", "multi"))
colnames(c_group) <- c("n1", "n2")
# compare between multi-subgroup
for(i in 1:nrow(c_group)){
    for(j in element2){
        # grep group
        if(c_group[i,1]=="multi"){
            data <- df_area[grep("pro|raw|model|control|drug", df_area$subgroup), ]
        }else{
            data <- df_area[grep(paste0(c_group[i,1],"|",c_group[i,2]), df_area$subgroup), ]
        }
        # grep subgroup
        if(j!="multi"){
            data <- data[grep(paste0("model|control|", j), data$subgroup), ]
        }
        assign(paste0("list_", i, "_", j), data)
        # break the single comparation
        if(c_group[i,2]=="control"){
            break
        }
    }
}
# list all the compare group
team <- ls()[c(grep("list_", ls()))]
# single plot of pca
library(ggbiplot)
library(ggsci)
library(scales)
library(ggrepel)
dir.create("pca_plot")
for(i in team){
    df <- get(i)
    dfd <- df[, !(colnames(df) %in% c("file", "name", "subgroup", "group"))]
    dfm <- df[, (colnames(df) %in% c("file", "name", "subgroup", "group"))]
    #compute pca matrix
  dfd <- dfd[, apply(dfd, 2, var) != 0]
    pca <- prcomp(dfd, scale.=T)
    pca_anno <- as.data.frame(pca[5])
    p <- ggbiplot(pca, obs.scale = 1,
                  var.scale = 1,
                  groups = dfm$subgroup,
               ellipse = TRUE,
               circle = TRUE,
               varname.size=0,
               var.axes = F) +
        geom_label_repel(data=pca_anno,
                   aes(x=x.PC1, y=x.PC2, label=dfm$name),
                      color="black", alpha=0.5,
                  fontface="bold", size=2, angle= 0,
                      direction="both", segment.size = 0.2,
                  segment.alpha = 0.3,
                      inherit.aes = FALSE, hjust = 0,
                      family="Times") +
        scale_color_npg() +
        scale_fill_npg() +
        theme(legend.position = "right",text=element_text(family="Times"))
    ggsave(p, file=paste0("pca_plot/", i, ".svg"))
}
# pca_facet
origin_team <- team
team <- team[!team %in% "list_3_high"]
for(i in team){
    df <- get(i)
    dfd <- df[, !(colnames(df) %in% c("file", "name", "subgroup", "group"))]
    dfm <- df[, (colnames(df) %in% c("file", "name", "subgroup", "group"))]
  dfd <- dfd[, apply(dfd, 2, var) != 0]
    pca<- prcomp(dfd, scale.=T)
    pca_x=data.frame(pca$x)
    pca_x <- pca_x[, 1:3]
    # join the facet group
    n <- strsplit(i, split="_")[[1]][2]
    deep <- strsplit(i, split="_")[[1]][3]
    if(c_group[n,1]!="multi"){
        pca_x$facet_col <- paste0(c_group[n, 1], "_", c_group[n, 2])
    }else{
        pca_x$facet_col <- "multi"
    }
    pca_x$facet_row <- deep
    pca_x <- cbind(pca_x,dfm)
    # annotate PC value
    pca=summary(pca)
    summary=c(round(pca$importance[2,],4))[1:3]
    summary <- data.frame(summary)
    summary$annotate <- rownames(summary)
    if(c_group[n,1]!="multi"){
        summary$facet_col <- paste0(c_group[n,1], "_", c_group[n,2])
    }else{
        summary$facet_col <- "multi"
    }
    summary$facet_row <- deep
    if(i==team[1]){
        data_facet <- pca_x
        data_facet_anno <- summary
    }else{
        data_facet <- rbind(data_facet, pca_x)
        data_facet_anno <- rbind(data_facet_anno, summary)
    }
}

PC1 <- data_facet_anno[which(data_facet_anno$annotate=="PC1"),]
PC2 <- data_facet_anno[which(data_facet_anno$annotate=="PC2"),]
PC1$figure <- paste0("PC1(",PC1$summary*100,"%)")
PC2$figure <- paste0("PC2(",PC2$summary*100,"%)")

# calculate the annotate coord
anno_x=min(as.numeric(data_facet$PC1))*(20/20)
anno_y=max(as.numeric(data_facet$PC2))*(22/20)
# set the color
colors <- c("control"="grey",
  "model"="#374E55FF",
  "drug"="#00A087FF",
  "pro_low"="#FDAE6BFF",
  "pro_medium"="#FD8D3CFF",
  "pro_high"="#E6550DFF",
  "raw_low"="#9ECAE1FF",
  "raw_medium"="#6BAED6FF",
  "raw_high"="#3182BDFF")

p <- ggplot(data_facet, aes(x=as.numeric(PC1), y=as.numeric(PC2), fill=subgroup)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1) +
    stat_ellipse(aes(color=subgroup), level = 0.95) +
    #scale_color_npg() +
    #scale_fill_npg() +
    scale_color_manual(values = colors) +
    scale_fill_manual(values = colors) +
    guides(color= "none") +
    geom_text(data=PC1, aes(x=anno_x*1.4, y=anno_y*1.4, label=figure),
            hjust=0, color="black",
            fontface="bold",alpha=0.6,
            size=1.5, inherit.aes = FALSE,
            family="Times") +
    geom_text(data=PC2, aes(x=anno_x*1.4, y=anno_y*(18/22)*1.4, label=figure),
            hjust=0, color="black", fontface="bold",
            alpha=0.6, size=1.5, inherit.aes = FALSE,
            family="Times") +
    labs(y="PC2", x="PC1", fill="Group") +
    #scale_x_continuous(limits=c(-60 ,60)) +
    #scale_y_continuous(limits=c(-60 ,60)) +
    facet_grid(facet_row ~ facet_col) +
    theme(legend.position = "right",text=element_text(family="Times"))
ggsave(p, file=paste0("pca_plot/pca_facet.svg"),width=8,height=6.5)

# opls_da plot
team <- origin_team
escape <- grep(paste0("multi|",which(c_group$n1=="multi" | c_group$n2=="multi")), team)
team <- team[!team %in% team[c(escape)]]
# single plot
dir.create("opls_plot")
library(ropls)
library(scales)
for(i in team){
    df <- get(i)
    dfd <- df[, !(colnames(df) %in% c("file", "name", "subgroup", "group"))]
    dfm <- df[, (colnames(df) %in% c("file", "name", "subgroup", "group"))]
    oplsda <- opls(x = dfd, y = dfm[, "subgroup"], predI = 1, orthoI = NA)
    df <- cbind(oplsda@scoreMN[, 1], oplsda@orthoScoreMN[, 1])
    colnames(df) <- c("h1", paste0("o", 1))
    df <- as.data.frame(df)
    df <- cbind(df, dfm)
    # join the facet group
    n <- strsplit(i, split="_")[[1]][2]
    deep <- strsplit(i, split="_")[[1]][3]
    df$facet_col <- paste0(c_group[n, 1], "_", c_group[n, 2])
    df$facet_row <- deep
```

```
# x lab and y lab text
x_lab <- paste0("T score[1](", oplsda@modelDF[1, "R2X"] * 100, "%)")
y_lab <- paste0("Orthogonal T score[1](", oplsda@modelDF[2, "R2X"] * 100, "%)")
summary <- rbind(x_lab, y_lab)
summary <- data.frame(summary)
# join the facet group
summary$facet_col <- paste0(c_group[n, 1], "_", c_group[n, 2])
summary$facet_row <- deep
summary$annotate <- rownames(summary)
# vip
vip=data.frame(oplsda@vipVn)
vip=cbind(rownames(vip),vip)
colnames(vip)=c("id","vip")
# join the facet group
vip$facet_col <- paste0(c_group[n, 1], "_", c_group[n, 2])
vip$facet_row <- deep
vip$team <- paste0(i, "_", vip$id)
# gather the data into facet data.frame
if(i==team[1]){
    data_facet <- df
    data_facet_anno <- summary
    data_vip <- vip
}else{
    data_facet <- rbind(data_facet, df)
    data_facet_anno <- rbind(data_facet_anno, summary)
    data_vip <- rbind(data_vip, vip)
}
# ggplot plot the single plot
p <- ggplot(df, aes(x=h1, y=o1)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1,
           aes(fill=subgroup)) +
    stat_ellipse(aes(color=subgroup), level = 0.95) +
    geom_label_repel(data=df,
             aes(x=h1, y=o1, label=name),
             color="black", alpha=0.5, fontface="bold",
             size=2, angle= 0,
             direction="both", segment.size = 0.2,
             segment.alpha = 0.3,
             inherit.aes = FALSE, hjust = 0,
             family="Times") +
    scale_color_npg() +
        scale_fill_npg() +
        labs(x=x_lab,y=y_lab,title="OPLS-DA") +
        theme(plot.title = element_text(hjust = 0.5),
              text=element_text(family="serif"))

    ggsave(p, file=paste0("opls_plot/", i,".svg"), height=6, width=8)
}
# opls facet plot
df <- data_facet[which(data_facet$facet_col!="model_control"),]
df_anno <- data_facet_anno[which(data_facet_anno$facet_col!="model_control"),]
df_xlab <- df_anno[which(df_anno$annotate=="x_lab"),]
df_ylab <- df_anno[which(df_anno$annotate=="y_lab"),]
anno_x=min(as.numeric(df$h1))*(20/20)
anno_y=max(as.numeric(df$o1))*(22/20)
colors <- c("control"="grey",
  "model"="#374E55FF",
  "drug"="#00A087FF",
  "pro_low"="#FDAE6BFF",
  "pro_medium"="#FD8D3CFF",
  "pro_high"="#E6550DFF",
  "raw_low"="#9ECAE1FF",
  "raw_medium"="#6BAED6FF",
  "raw_high"="#3182BDFF")
p <- ggplot(df, aes(x=as.numeric(h1), y=as.numeric(o1), fill=subgroup)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1) +
    stat_ellipse(aes(color=subgroup), level = 0.95) +
    #scale_color_npg() +
    #scale_fill_npg() +
    scale_color_manual(values = colors) +
    scale_fill_manual(values = colors) +
    guides(color= "none") +
    geom_text(data=df_xlab,
             aes(x=anno_x, y=anno_y*1.4, label=summary),
         hjust=0, color="black", fontface="bold",alpha=0.6,
         size=2, inherit.aes = FALSE, family="Times") +
    geom_text(data=df_ylab,
             aes(x=anno_x, y=anno_y*(18/22)*1.4, label=summary),
         hjust=0, color="black", fontface="bold",alpha=0.6,
         size=2, inherit.aes = FALSE, family="Times") +
    labs(y="Orthogonal T score[1]", x="T score[1]", fill="Group") +
    #scale_x_continuous(limits=c(-60 ,60)) +
```

```r
    #scale_y_continuous(limits=c(-60 ,60)) +
    facet_grid(facet_row ~ facet_col) +
    theme(legend.position = "right",text=element_text(family="Times"))

ggsave(p,file="opls_plot/opls_facet.svg",width=8,height=6.5)

# volcano_plot
team <- origin_team
escape <- grep(paste0("multi|",which(c_group$n1=="multi" | c_group$n2=="multi")), team)
team <- team[!team %in% team[c(escape)]]
# single plot
dir.create("volcano")
for(i in team){
    df <- get(i)
    dfd <- df[, !(colnames(df) %in% c("file", "name", "subgroup", "group"))]
    dfm <- df[, (colnames(df) %in% c("file", "name", "subgroup", "group"))]
    compare <- unique(dfm$group)
    xn <- which(dfm$group==compare[1])
    yn <- which(dfm$group==compare[2])
    fc_name <- paste0(compare[1], "_d_", compare[2])
    for(j in colnames(dfd)){
        # t.test calculate p.value
        x <- dfd[c(xn), colnames(dfd) %in% j]
        y <- dfd[c(yn), colnames(dfd) %in% j]
        stat=t.test(x, y, var.equal = T, paired = F)$p.value
        assign(paste0("p.value_", j), stat)
        # mean and fc
        fc=mean(x)/mean(y)
        assign(paste0("fc_", j), fc)
    }
    # gather the fc list
    fc_list <- ls()[c(grep("fc_X", ls()))]
    fc <- data.frame(fc_list)
    fc$fc <- NA
    for(k in 1:nrow(fc)){
        fc$fc[k] <- get(fc$fc_list[k])
    }
    fc <- separate(fc, col="fc_list", sep="fc_",
               into=c("m", "id"), remove=T)[, 2:3]
    # gather the p.value list
    p.value_list <- ls()[c(grep("p.value_X", ls()))]
    p.value <- data.frame(p.value_list)
    p.value$p.value <- NA
    for(k in 1:nrow(p.value)){
        p.value$p.value[k] <- get(p.value$p.value_list[k])
    }
    p.value <- separate(p.value, col="p.value_list", sep="p.value_",
                into=c("m", "id"), remove=T)[, 2:3]
    # merge
    fc_p <- merge(fc, p.value, by="id", all.x=T, sort=T)
    fc_p$fc <- log2(fc_p$fc)
    fc_p$change <- factor(ifelse(fc_p$p.value < 0.05 & abs(fc_p$fc) >= 1,
                      ifelse(fc_p$fc >= 1,"up","down"),"stable"),
                      levels = c("up","down","stable"))
    #plot volcano
    data <- fc_p
    title <-  paste0(compare[1], "/", compare[2])
    p <- ggplot(data,aes(x=fc, y=-log10(p.value),color = change)) +
        geom_point(alpha=0.8, stroke=0, size=3) +
        scale_color_manual(values = c("down"="#4DBBD5FF",
                          "stable"="#8491B4FF",
                          "up"="#DC0000FF")) +
        ylim(1,max(-log10(data$p.value))) +
        geom_hline(yintercept = -log10(0.05), linetype=4, size=0.8) +
        geom_vline(xintercept = c(-1,1), linetype=4, size=0.8) +
        labs(x = "log2(FC)", y="-log10(p-value)", title=title) +
        geom_text_repel(data = data[data$p.value<0.01 & abs(data$fc) >= 2,],
                aes(label = substring(id, 2)),
                size = 3,family="Times") +
        theme(text=element_text(family="Times"),
                #axis.line = element_line(colour = "black", size=0.2),
                #plot.margin = unit(c(3, 1, 3, 1), "cm")
                plot.title = element_text(hjust = 0.5))
    ggsave(p, file=paste0("volcano/", i, ".svg"),width=8,height=6.5)

    # for facet plot
    deep <- strsplit(i, split="_")[[1]][3]
    fc_p$facet_col <- paste0(compare[1], "/", compare[2])
    fc_p$facet_row <- deep
    fc_p$team <- paste0(i, "_", fc_p$id)
    if(i==team[1]){
        data_facet <- fc_p
```

```r
    }else{
        data_facet <- rbind(data_facet, fc_p)
    }
}
data <- data_facet[!(1:nrow(data_facet) %in% grep("control", data_facet$facet_col)),]
p <- ggplot(data,aes(x=fc, y=-log10(p.value),color = change)) +
    geom_point(alpha=0.8, stroke=0, size=1.5) +
    scale_color_manual(values = c("down"="#4DBBD5FF",
                                  "stable"="#8491B4FF",
                                  "up"="#DC0000FF")) +
    ylim(1,max(-log10(data$p.value))) +
    geom_hline(yintercept = -log10(0.05),linetype=4,size=0.8) +
    geom_vline(xintercept = c(-1,1),linetype=4,size=0.8) +
    labs(x = "log2(FC)", y = "-log10(p-value)") +
    geom_text_repel(data = data[data$p.value<0.01 & abs(data$fc) >= 2,],
            aes(label = substring(id, 2)),
            size = 3,family="Times") +
    theme(text=element_text(family="Times"),
          plot.title = element_text(hjust = 0.5)) +
    facet_grid(facet_row ~ facet_col)
ggsave(p, file=paste0("volcano/volcano_facet.svg"),width=8,height=6.5)

# plot vip-p plot
# merge the vip dataset and p.value dataset
dir.create("vip")
df <- merge(data, data_vip, all.x=T, by="team", sort=T)
p <- ggplot(df, aes(x=p.value, y=vip, color=fc)) +
    geom_point(alpha=0.8, size=1.5, stroke=0) +
    xlim(0,0.5) +
    #scale_color_gradientn(limits = c(-5, +5),
    #              breaks = c(-3, 0, +3),
    #              colours = c("#E6550DFF", "#79AF97FF", "#3182BDFF")) +
    scale_color_viridis_c() +
    labs(y="VIP", x="p-value", color="log2(FC)") +
    geom_hline(yintercept = 1,linetype=4,size=0.8) +
    geom_vline(xintercept = 0.05,linetype=4,size=0.8) +
    facet_grid(facet_row.x ~ facet_col.x) +
    theme(legend.position = "right",
          text=element_text(family="Times"),
          plot.title = element_text(hjust = 0.5))
```

```r
ggsave(p,file="vip/vip_p_facet.svg")
# filter the data according to vip, p.value, fc
# FDR revise
library(fdrtool)
df <- df[which(is.na(df$p.value)==F),] %>%
  mutate(id=substring(id.x, 2))
df$q.value <- fdrtool(df$p.value, statistic='pvalue', plot=F)$qval
write.table(df, file="discrepancy_raw.tsv", col.names=T, row.names=F, sep="\t")
# p.value filter
data <- df[which(df$fc>1 & df$vip>1 & df$p.value<0.05),]
data_u <- data[!duplicated(data$id.x),]
write.table(data_u, file="discrepancy_fc1_vip1_p005.tsv", col.names=T, row.names=F, sep="\t")
# get sirius idenfication results
stru_path="0070_results/"
structure <- read.csv(file=paste0(stru_path, "fingerid_first_score.tsv"), sep="\t", header=T)
# merge the structure according to qdata
data1 <- merge(data_u, structure, by="id", all.x=T, sort=T)
write.table(data1, file="discrepancy_pdata_structure.tsv", sep="\t", col.names=T, row.names=F)

# q.value filter
qdata <- df[which(df$fc>1 & df$vip>1 & df$q.value<0.05),]
qdata_u <- qdata[!duplicated(qdata$id.x),]
# merge the structure according to qdata
data0 <- merge(qdata_u, structure, by="id", all.x=T, sort=T)
write.table(data0, file="discrepancy_qdata_structure.tsv", sep="\t", col.names=T, row.names=F)
# pathway enrichment analysis
library(FELLA)
# data0 <- read.csv(file="discrepancy_qdata_structure.tsv", sep="\t", header=T)
# data1 <- read.csv(file="discrepancy_pdata_structure.tsv", sep="\t", header=T)
# according to q.value
qdata_h <- data0[grep("high",data0$team), ]
qdata_h_simi <- qdata_h[which(qdata_h$similarity>=0.5), ]
qdata_h_simi <- qdata_h_simi[!duplicated(qdata_h_simi$smiles),]
# according to p.value
pdata_h <- data1[grep("high",data1$team), ]
pdata_h_simi <- pdata_h[which(pdata_h$similarity>=0.5), ]
pdata_h_simi <- pdata_h_simi[!duplicated(pdata_h_simi$smiles),]
# this table is summarized by fc, vip, p, dosage of high, idenfication of high tanimoto similarity, and
write.table(pdata_h_simi, file="p_cluster_pathway.tsv", sep="\t", col.names=T, row.names=F)
```

28 File: fecal_pos.R

```
library(tidyverse)

path="~/operation/re_fecal_pos"
setwd(path)

mzmine <- read.csv(file="fecal_pos_mzmine.csv", header=T, sep=",", check.name=F)
metadata <- read.csv(file="metadata.tsv", header=T, sep="\t")
data_area <- mzmine[, grep("ID|m/z|retention|area", colnames(mzmine))]
rownames(data_area) <- data_area$"row ID"
df_area <- data.frame(
              t(
            data_area[,grep("area", colnames(data_area))]
              )
)
df_area$file <- rownames(df_area)
df_area <- separate(df_area, col="file", into=c("file", "peak", "area"), remove=T, sep=" ")
# df_area[1:10,(ncol(df_area)-5):ncol(df_area)]
df_area <- df_area[, !(colnames(df_area) %in% c("peak", "area"))]
df_area <- merge(df_area, metadata, all.x=T, by.x="file", by.y="file")
# finish reformat
# compare group as follow
element1 <- c("pro", "raw", "model", "control")
element2 <- c("high", "medium", "low", "multi")
c_group <- data.frame(c(rep("model", 3), "pro", "multi"), c(element1[c(1,2,4)], "raw", "multi"))
colnames(c_group) <- c("n1", "n2")
# compare between multi-subgroup
for(i in 1:nrow(c_group)){
    for(j in element2){
        # grep group
        if(c_group[i,1]=="multi"){
            data <- df_area[grep("pro|raw|model|control|drug", df_area$subgroup), ]
        }else{
            data <- df_area[grep(paste0(c_group[i,1],"|",c_group[i,2]), df_area$subgroup), ]
        }
        # grep subgroup
        if(j!="multi"){
            data <- data[grep(paste0("model|control|", j), data$subgroup), ]
        }
        assign(paste0("list_", i, "_", j), data)
        # break the single comparation
        if(c_group[i,2]=="control"){
            break
        }
    }
}
# list all the compare group
team <- ls()[c(grep("list_", ls()))]
# single plot of pca
library(ggbiplot)
library(ggsci)
library(scales)
library(ggrepel)
dir.create("pca_plot")
for(i in team){
    df <- get(i)
    dfd <- df[, !(colnames(df) %in% c("file", "name", "subgroup", "group"))]
    dfm <- df[, (colnames(df) %in% c("file", "name", "subgroup", "group"))]
    #compute pca matrix
    pca <- prcomp(dfd, scale. = TRUE)
    pca_anno <- as.data.frame(pca[5])
    p <- ggbiplot(pca, obs.scale = 1,
              var.scale = 1,
              groups = dfm$subgroup,
              ellipse = TRUE,
              circle = TRUE,
              varname.size=0,
              var.axes = F) +
        geom_label_repel(data=pca_anno,
                   aes(x=x.PC1, y=x.PC2, label=dfm$name),
                      color="black", alpha=0.5,
                  fontface="bold", size=2, angle= 0,
                      direction="both", segment.size = 0.2,
                  segment.alpha = 0.3,
                      inherit.aes = FALSE, hjust = 0,
                      family="Times") +
        scale_color_npg() +
        scale_fill_npg() +
        theme(legend.position = "right",text=element_text(family="Times"))
    ggsave(p, file=paste0("pca_plot/", i, ".svg"))
}
```

```r
# pca_facet
origin_team <- team
team <- team[!team %in% "list_3_high"]
for(i in team){
    df <- get(i)
    dfd <- df[, !(colnames(df) %in% c("file", "name", "subgroup", "group"))]
    dfm <- df[, (colnames(df) %in% c("file", "name", "subgroup", "group"))]
    pca<- prcomp(dfd, scale. = TRUE)
    pca_x=data.frame(pca$x)
    pca_x <- pca_x[, 1:3]
    # join the facet group
    n <- strsplit(i, split="_")[[1]][2]
    deep <- strsplit(i, split="_")[[1]][3]
    if(c_group[n,1]!="multi"){
        pca_x$facet_col <- paste0(c_group[n, 1], "_", c_group[n, 2])
    }else{
        pca_x$facet_col <- "multi"
    }
    pca_x$facet_row <- deep
    pca_x <- cbind(pca_x,dfm)
    # annotate PC value
    pca=summary(pca)
    summary=c(round(pca$importance[2,],4))[1:3]
    summary <- data.frame(summary)
    summary$annotate <- rownames(summary)
    if(c_group[n,1]!="multi"){
        summary$facet_col <- paste0(c_group[n,1], "_", c_group[n,2])
    }else{
        summary$facet_col <- "multi"
    }
    summary$facet_row <- deep
    if(i==team[1]){
        data_facet <- pca_x
        data_facet_anno <- summary
    }else{
        data_facet <- rbind(data_facet, pca_x)
        data_facet_anno <- rbind(data_facet_anno, summary)
    }
}

PC1 <- data_facet_anno[which(data_facet_anno$annotate=="PC1"),]
```

```r
PC2 <- data_facet_anno[which(data_facet_anno$annotate=="PC2"),]
PC1$figure <- paste0("PC1(",PC1$summary*100,"%)")
PC2$figure <- paste0("PC2(",PC2$summary*100,"%)")

# calculate the annotate coord
anno_x=min(as.numeric(data_facet$PC1))*(20/20)
anno_y=max(as.numeric(data_facet$PC2))*(22/20)
# set the color
colors <- c("control"="grey",
  "model"="#374E55FF",
  "drug"="#00A087FF",
  "pro_low"="#FDAE6BFF",
  "pro_medium"="#FD8D3CFF",
  "pro_high"="#E6550DFF",
  "raw_low"="#9ECAE1FF",
  "raw_medium"="#6BAED6FF",
  "raw_high"="#3182BDFF")

p <- ggplot(data_facet, aes(x=as.numeric(PC1), y=as.numeric(PC2), fill=subgroup)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1) +
    stat_ellipse(aes(color=subgroup), level = 0.95) +
    #scale_color_npg() +
    #scale_fill_npg() +
    scale_color_manual(values = colors) +
    scale_fill_manual(values = colors) +
    guides(color= "none") +
    geom_text(data=PC1, aes(x=anno_x*1.4, y=anno_y*1.4, label=figure),
            hjust=0, color="black",
            fontface="bold",alpha=0.6,
            size=1.5, inherit.aes = FALSE,
            family="Times") +
    geom_text(data=PC2, aes(x=anno_x*1.4, y=anno_y*(18/22)*1.4, label=figure),
            hjust=0, color="black", fontface="bold",
            alpha=0.6, size=1.5, inherit.aes = FALSE,
            family="Times") +
    labs(y="PC2", x="PC1", fill="Group") +
    #scale_x_continuous(limits=c(-60 ,60)) +
    #scale_y_continuous(limits=c(-60 ,60)) +
    facet_grid(facet_row ~ facet_col) +
    theme(legend.position = "right",text=element_text(family="Times"))
ggsave(p, file=paste0("pca_plot/pca_facet.svg"),width=8,height=6.5)
```

```
# opls_da plot
team <- origin_team
escape <- grep(paste0("multi|",which(c_group$n1=="multi" | c_group$n2=="multi")), team)
team <- team[!team %in% team[c(escape)]]
# single plot
dir.create("opls_plot")
library(ropls)
library(scales)
for(i in team){
    df <- get(i)
    dfd <- df[, !(colnames(df) %in% c("file", "name", "subgroup", "group"))]
    dfm <- df[, (colnames(df) %in% c("file", "name", "subgroup", "group"))]
    oplsda <- opls(x = dfd, y = dfm[, "subgroup"], predI = 1, orthoI = NA)
    df <- cbind(oplsda@scoreMN[, 1], oplsda@orthoScoreMN[, 1])
    colnames(df) <- c("h1", paste0("o", 1))
    df <- as.data.frame(df)
    df <- cbind(df, dfm)
    # join the facet group
    n <- strsplit(i, split="_")[[1]][2]
    deep <- strsplit(i, split="_")[[1]][3]
    df$facet_col <- paste0(c_group[n, 1], "_", c_group[n, 2])
    df$facet_row <- deep
    # x lab and y lab text
    x_lab <- paste0("T score[1](", oplsda@modelDF[1, "R2X"] * 100, "%)")
    y_lab <- paste0("Orthogonal T score[1](", oplsda@modelDF[2, "R2X"] * 100, "%)")
    summary <- rbind(x_lab, y_lab)
    summary <- data.frame(summary)
    # join the facet group
    summary$facet_col <- paste0(c_group[n, 1], "_", c_group[n, 2])
    summary$facet_row <- deep
    summary$annotate <- rownames(summary)
    # vip
    vip=data.frame(oplsda@vipVn)
    vip=cbind(rownames(vip),vip)
    colnames(vip)=c("id","vip")
    # join the facet group
    vip$facet_col <- paste0(c_group[n, 1], "_", c_group[n, 2])
    vip$facet_row <- deep
    vip$team <- paste0(i, "_", vip$id)
    # gather the data into facet data.frame
    if(i==team[1]){
        data_facet <- df
        data_facet_anno <- summary
        data_vip <- vip
    }else{
        data_facet <- rbind(data_facet, df)
        data_facet_anno <- rbind(data_facet_anno, summary)
        data_vip <- rbind(data_vip, vip)
    }
    # ggplot plot the single plot
    p <- ggplot(df, aes(x=h1, y=o1)) +
        geom_point(alpha=0.8, size=3, shape=21, stroke=0.1,
               aes(fill=subgroup)) +
        stat_ellipse(aes(color=subgroup), level = 0.95) +
        geom_label_repel(data=df,
                 aes(x=h1, y=o1, label=name),
                 color="black", alpha=0.5, fontface="bold",
                 size=2, angle= 0,
                 direction="both", segment.size = 0.2,
                 segment.alpha = 0.3,
                 inherit.aes = FALSE, hjust = 0,
                 family="Times") +
        scale_color_npg() +
        scale_fill_npg() +
        labs(x=x_lab,y=y_lab,title="OPLS-DA") +
        theme(plot.title = element_text(hjust = 0.5),
              text=element_text(family="serif"))

    ggsave(p, file=paste0("opls_plot/", i,".svg"), height=6, width=8)
}
# opls facet plot
df <- data_facet[which(data_facet$facet_col!="model_control"),]
df_anno <- data_facet_anno[which(data_facet_anno$facet_col!="model_control"),]
df_xlab <- df_anno[which(df_anno$annotate=="x_lab"),]
df_ylab <- df_anno[which(df_anno$annotate=="y_lab"),]
anno_x=min(as.numeric(df$h1))*(20/20)
anno_y=max(as.numeric(df$o1))*(22/20)
colors <- c("control"="grey",
  "model"="#374E55FF",
  "drug"="#00A087FF",
  "pro_low"="#FDAE6BFF",
  "pro_medium"="#FD8D3CFF",
  "pro_high"="#E6550DFF",
  "raw_low"="#9ECAE1FF",
  "raw_medium"="#6BAED6FF",
  "raw_high"="#3182BDFF")
p <- ggplot(df, aes(x=as.numeric(h1), y=as.numeric(o1), fill=subgroup)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1) +
    stat_ellipse(aes(color=subgroup), level = 0.95) +
    #scale_color_npg() +
    #scale_fill_npg() +
    scale_color_manual(values = colors) +
    scale_fill_manual(values = colors) +
    guides(color= "none") +
    geom_text(data=df_xlab,
             aes(x=anno_x, y=anno_y*1.4, label=summary),
         hjust=0, color="black", fontface="bold",alpha=0.6,
         size=2, inherit.aes = FALSE, family="Times") +
    geom_text(data=df_ylab,
             aes(x=anno_x, y=anno_y*(18/22)*1.4, label=summary),
         hjust=0, color="black", fontface="bold",alpha=0.6,
         size=2, inherit.aes = FALSE, family="Times") +
    labs(y="Orthogonal T score[1]", x="T score[1]", fill="Group") +
    #scale_x_continuous(limits=c(-60 ,60)) +
    #scale_y_continuous(limits=c(-60 ,60)) +
    facet_grid(facet_row ~ facet_col) +
    theme(legend.position = "right",text=element_text(family="Times"))

ggsave(p,file="opls_plot/opls_facet.svg",width=8,height=6.5)

# volcano_plot
team <- origin_team
escape <- grep(paste0("multi|",which(c_group$n1=="multi" | c_group$n2=="multi")), team)
team <- team[!team %in% team[c(escape)]]
# single plot
dir.create("volcano")
for(i in team){
    df <- get(i)
    dfd <- df[, !(colnames(df) %in% c("file", "name", "subgroup", "group"))]
    dfm <- df[, (colnames(df) %in% c("file", "name", "subgroup", "group"))]
    compare <- unique(dfm$group)
    xn <- which(dfm$group==compare[1])
    yn <- which(dfm$group==compare[2])
    fc_name <- paste0(compare[1], "_d_", compare[2])
    for(j in colnames(dfd)){
        # t.test calculate p.value
        x <- dfd[c(xn), colnames(dfd) %in% j]
        y <- dfd[c(yn), colnames(dfd) %in% j]
        stat=t.test(x, y, var.equal = T, paired = F)$p.value
        assign(paste0("p.value_", j), stat)
        # mean and fc
        fc=mean(x)/mean(y)
        assign(paste0("fc_", j), fc)
    }
    # gather the fc list
    fc_list <- ls()[c(grep("fc_X", ls()))]
    fc <- data.frame(fc_list)
    fc$fc <- NA
    for(k in 1:nrow(fc)){
        fc$fc[k] <- get(fc$fc_list[k])
    }
    fc <- separate(fc, col="fc_list", sep="fc_",
              into=c("m", "id"), remove=T)[, 2:3]
    # gather the p.value list
    p.value_list <- ls()[c(grep("p.value_X", ls()))]
    p.value <- data.frame(p.value_list)
    p.value$p.value <- NA
    for(k in 1:nrow(p.value)){
        p.value$p.value[k] <- get(p.value$p.value_list[k])
    }
    p.value <- separate(p.value, col="p.value_list", sep="p.value_",
               into=c("m", "id"), remove=T)[, 2:3]
    # merge
    fc_p <- merge(fc, p.value, by="id", all.x=T, sort=T)
    fc_p$fc <- log2(fc_p$fc)
    fc_p$change <- factor(ifelse(fc_p$p.value < 0.05 & abs(fc_p$fc) >= 1,
                      ifelse(fc_p$fc >= 1,"up","down"),"stable"),
                      levels = c("up","down","stable"))
    #plot volcano
    data <- fc_p
    title <-  paste0(compare[1], "/", compare[2])
    p <- ggplot(data,aes(x=fc, y=-log10(p.value),color = change)) +
        geom_point(alpha=0.8, stroke=0, size=3) +
        scale_color_manual(values = c("down"="#4DBBD5FF",
```

```r
                            "stable"="#8491B4FF",
                            "up"="#DC0000FF")) +
        ylim(1,max(-log10(data$p.value))) +
        geom_hline(yintercept = -log10(0.05), linetype=4, size=0.8) +
        geom_vline(xintercept = c(-1,1), linetype=4, size=0.8) +
        labs(x = "log2(FC)", y="-log10(p-value)", title=title) +
        geom_text_repel(data = data[data$p.value<0.01 & abs(data$fc) >= 2,],
                aes(label = strsplit(id, split="X")[[1]][2]),
                size = 3,family="Times") +
        theme(text=element_text(family="Times"),
                #axis.line = element_line(colour = "black", size=0.2),
                #plot.margin = unit(c(3, 1, 3, 1), "cm")
                plot.title = element_text(hjust = 0.5))
    ggsave(p, file=paste0("volcano/", i, ".svg"),width=8,height=6.5)

    # for facet plot
    deep <- strsplit(i, split="_")[[1]][3]
    fc_p$facet_col <- paste0(compare[1], "/", compare[2])
    fc_p$facet_row <- deep
    fc_p$team <- paste0(i, "_", fc_p$id)
    if(i==team[1]){
        data_facet <- fc_p
    }else{
        data_facet <- rbind(data_facet, fc_p)
    }
}
data <- data_facet[!(1:nrow(data_facet) %in% grep("control", data_facet$facet_col)),]
p <- ggplot(data,aes(x=fc, y=-log10(p.value),color = change)) +
    geom_point(alpha=0.8, stroke=0, size=1.5) +
    scale_color_manual(values = c("down"="#4DBBD5FF",
                        "stable"="#8491B4FF",
                        "up"="#DC0000FF")) +
    ylim(1,max(-log10(data$p.value))) +
    geom_hline(yintercept = -log10(0.05),linetype=4,size=0.8) +
    geom_vline(xintercept = c(-1,1),linetype=4,size=0.8) +
    labs(x = "log2(FC)", y = "-log10(p-value)") +
    geom_text_repel(data = data[data$p.value<0.01 & abs(data$fc) >= 2,],
            aes(label = strsplit(id, split="X")[[1]][2]),
            size = 3,family="Times") +
    theme(text=element_text(family="Times"),
          plot.title = element_text(hjust = 0.5)) +
    facet_grid(facet_row ~ facet_col)
ggsave(p, file=paste0("volcano/volcano_facet.svg"),width=8,height=6.5)

# plot vip-p plot
# merge the vip dataset and p.value dataset
dir.create("vip")
df <- merge(data, data_vip, all.x=T, by="team", sort=T)
p <- ggplot(df, aes(x=p.value, y=vip, color=fc)) +
    geom_point(alpha=0.8, size=1.5, stroke=0) +
    xlim(0,0.5) +
    scale_color_gradientn(limits = c(-5, +5),
                 breaks = c(-3, 0, +3),
                 colours = c("#E6550DFF", "#79AF97FF", "#3182BDFF")) +
#   scale_color_viridis_c() +
    labs(y="VIP", x="p-value", color="log2(FC)") +
    geom_hline(yintercept = 1,linetype=4,size=0.8) +
    geom_vline(xintercept = 0.05,linetype=4,size=0.8) +
    facet_grid(facet_row.x ~ facet_col.x) +
    theme(legend.position = "right",
          text=element_text(family="Times"),
          plot.title = element_text(hjust = 0.5))

ggsave(p,file="vip/vip_p_facet.svg")
# filter the data according to vip, p.value, fc
# FDR revise
library(fdrtool)
df$q.value <- fdrtool(df$p.value, statistic='pvalue', plot=F)$qval

# p.value filter
data <- df[which(df$fc>1 & df$vip>1 & df$p.value<0.01),]
data_u <- data[!duplicated(data$id.x),]
data_u <- separate(data_u, col="id.x", into=c("X", "id"), sep="X", remove=T)
# get sirius idenfication results
structure <- read.csv(file="fingerid_first_score.tsv", sep="\t", header=T)
# merge the structure according to qdata
data1 <- merge(data_u, structure, by="id", all.x=T, sort=T)
write.table(data1, file="data_structure.tsv", sep="\t", col.names=T, row.names=F)

# q.value filter
qdata <- df[which(df$fc>1 & df$vip>1 & df$q.value<0.05),]
qdata_u <- qdata[!duplicated(qdata$id.x),]
qdata_u <- separate(qdata_u, col="id.x", into=c("X", "id"), sep="X", remove=T)
# merge the structure according to qdata
data0 <- merge(qdata_u, structure, by="id", all.x=T, sort=T)
write.table(data0, file="qdata_structure.tsv", sep="\t", col.names=T, row.names=F)
# pathway enrichment analysis
library(FELLA)
```

29 File: format_geen.medical.R

```
# format_geen.medical <-
  # function(
  #          file = "~/Downloads/pubmed.xlsx",
  #          get_full.text = T,
  #          save_path = paste0("~/Documents/", Sys.time())
  #          ){
  #   metadata <- readxl::read_xlsx(file)
  #   colnames(metadata) <- c("title", "IF", "author.1", "freq.author.1",
  #                           "author.corr", "freq.author.corr", "email.author.corr",
  #                           "journal", "freq.journal", "publish.year",
  #                           "pmid", "url", "affi.author.1", "freq.affi.author.1")
  #   if(get_full.text){
  #     save_path <- gsub(":", "-", save_path)
  #     ## create dir
  #     if(!file.exists(save_path))
  #       dir.create(save_path)
  #   }
  #   return(metadata)
  # }
```

30 File: ggplot2_heatmap.R

```
###  ggplot2_heatmap.R
library(ggplot2)
library(ggtree)
library(aplot)
library(tidyr)
library(rayshader)
library(ggsci)
# instance: 2655 3058 2631 4719 3591 2629 2835 3378 3188 2528 1121 525 196 1683 3983 2428 34 4322 2943
##########################################################################
```

```
##########################################################################
file="0703_all/ftalign.tsv"
data <- read.csv(file=file,header=T,sep="\t",row.names=1, check.name=F)
name=data.frame(strsplit(colnames(data), split="initial_8_neg_"))[2,]
colnames(data)=name
rownames(data)=name
instance <- c(sample(colnames(data),27), c("347", "495", "2268"))
data <- data[colnames(data) %in% instance, colnames(data) %in% instance]
##########################################################################
##########################################################################
savename="instance.svg"
phr <- hclust(dist(data)) %>% ggtree(layout="rectangular", branch.length="none")
phc <- hclust(dist(t(data))) %>% ggtree(layout="rectangular", branch.length="none") + layout_dendrogram
data$names <- rownames(data)
p1 <- gather(data, 1:(ncol(data)-1), key="condition", value='expr')  ## keep the last col
pp <- ggplot(p1,aes(x=condition,y=names,fill=expr)) +
  geom_tile(size=0.5, color="black") +
  #theme_minimal()+
  scale_fill_viridis_c() +
  #scale_fill_npg() +
  scale_y_discrete(position="right") +
  labs(x="Feature ID", y="Feature ID", fill="FTAS") +
  theme(
        axis.text.x = element_text(angle=90),
        axis.text = element_text(size=20),
        axis.title = element_text(size=20, face="bold"),
        #axis.text = element_blank(),
        legend.title = element_text(size=20, face="bold"),
        legend.text = element_text(size=20),
        legend.key.width = unit(2, "cm"),
        legend.key.height = unit(4, "cm"),
        text=element_text(family="serif")
        #axis.title.y = element_text(size = 14),
        #plot.title = element_text(hjust = 1,vjust=-40,size=14)
  )
  pp_com <- pp %>%
    insert_left(phr, width=.1) %>%
    insert_top(phc, height=.1)
  ggsave(pp_com,file=savename, width=16, height=15)
  data <- data[, c(ncol(data), 1:(ncol(data)-1))]
  write.table(data, file="instance_data.tsv", col.names=T, row.names=F, sep="\t")
```

```
##############################################################################
##############################################################################
###  normalized data
file="norm_instance_data.tsv"
data <- read.csv(file=file,header=T,sep="\t",row.names=1, check.name=F)
###########################################################################
###########################################################################
savename="norm_instance.svg"
phr <- hclust(dist(data)) %>% ggtree(layout="rectangular", branch.length="none")
phc <- hclust(dist(t(data))) %>% ggtree(layout="rectangular", branch.length="none") + layout_dendrogra
data$names <- rownames(data)
p1 <- gather(data, 1:(ncol(data)-1), key="condition", value='expr')  ## keep the last col
pp <- ggplot(p1,aes(x=condition,y=names,fill=expr)) +
  geom_tile(size=0.5, color="black") +
  #theme_minimal()+
  scale_fill_viridis_c() +
  #scale_fill_npg() +
  scale_y_discrete(position="right") +
  labs(x="Feature ID", y="Feature ID", fill="NFTAS") +
  theme(
        axis.text.x = element_text(angle=90),
        axis.text = element_text(size=20),
        axis.title = element_text(size=20, face="bold"),
        #axis.text = element_blank(),
        legend.title = element_text(size=20, face="bold"),
        legend.text = element_text(size=20),
        legend.key.width = unit(2, "cm"),
        legend.key.height = unit(4, "cm"),
        text=element_text(family="serif")
        #axis.title.y = element_text(size = 14),
        #plot.title = element_text(hjust = 1,vjust=-40,size=14)
  )
  pp_com <- pp %>%
    insert_left(phr, width=.1) %>%
    insert_top(phc, height=.1)
  ggsave(pp_com,file=savename, width=16, height=15)
  #plot_gg(pp_com, multicore = TRUE, width = 20 ,height=20, scale=250) # 加载图形
  #render_depth(focallength=100,focus=0.72)
  library(ggupset)
  library(reshape2)
  ### instance: 627 880 362 835 26 289 39 482 871 213 433 636 609 295 132 245 15 162 740 599 585 412
```

```
    file="norm_instance_data.tsv"
    data <- read.csv(file=file,header=T,sep="\t",row.names=1, check.name=F)
    data$names <- rownames(data)
    data <- melt(data, measure.vars=rownames(data), variable.name="condition", value.name="expr")
    instance <- sample(1:nrow(data), 50)
    data <- data[instance, ]
    data$upset_x=paste0(data$names, "_", data$condition)
    p <- ggplot(data, aes(x=factor(upset_x), y=expr, fill=expr)) +
      geom_col() +
      #scale_x_mergelist(sep = "_") +
      axis_combmatrix(sep = "_") +
      scale_fill_viridis_c() +
      #coord_flip() +
      labs(x="Feature link", y="NFTAS") +
      theme(
            #axis.text.x = element_text(angle=90),
            axis.text.y = element_text(size=20, angle=90),
            axis.title.x = element_text(size=30, face="bold", angle=180),
            axis.title.y = element_text(size=30, face="bold"),
            #axis.text = element_blank(),
            legend.title = element_text(size=20, face="bold"),
            legend.text = element_text(size=20),
            legend.position = "none",
            #legend.key.width = unit(2, "cm"),
            #legend.key.height = unit(4, "cm"),
            text=element_text(family="serif")
            #axis.title.y = element_text(size = 14),
            #plot.title = element_text(hjust = 1,vjust=-40,size=14)
            ) +
theme_combmatrix(combmatrix.panel.point.size = 5,
                 #combmatrix.panel.margin = unit(c(0.5,0.5),"pt"),
                 combmatrix.panel.line.size = 2,
                 combmatrix.label.height = unit(500, "pt"),
                 combmatrix.label.text = element_text(family="serif", angle=145, size=12, face="bold",
                 combmatrix.label.make_space = F)
ggsave(p, file="test1.pdf", width=20, height=20)
######################
######################
library(ggplot2)
library(ggtree)
library(aplot)
```

```r
library(tidyr)
library(rayshader)
library(ggsci)
file="0703_all/ftalign.tsv"
savename="small_heatmap.svg"
data <- read.csv(file=file,header=T,sep="\t",row.names=1, check.name=F)
name=data.frame(strsplit(colnames(data), split="initial_8_neg_"))[2,]
colnames(data)=name
rownames(data)=name
data <- data[colnames(data) %in% c("347", "495", "2268"), colnames(data) %in% c("347", "495", "2268")]
phr <- hclust(dist(data)) %>% ggtree(layout="rectangular", branch.length="none")
phc <- hclust(dist(t(data))) %>% ggtree(layout="rectangular", branch.length="none") + layout_dendrogram
data$names <- rownames(data)
p1 <- gather(data, 1:(ncol(data)-1), key="condition", value='expr')  ## keep the last col
pp <- ggplot(p1,aes(x=condition,y=names,fill=expr)) +
  geom_tile(color="black", size=2) +
  #theme_minimal()+
  scale_fill_viridis_c() +
  #scale_fill_npg() +
  scale_y_discrete(position="right") +
  labs(x="Feature ID", y="Feature ID", fill="Normalized\nftalign\nsimilarity\n") +
  theme_minimal() +
  theme(
        axis.text = element_blank(),
        axis.title = element_blank(),
        #axis.text = element_blank(),
        legend.title = element_blank(),
        legend.position = "none",
        axis.ticks = element_blank(),
        panel.grid = element_blank()
        #legend.text = element_text(size=20),
        #legend.key.width = unit(2, "cm"),
        #legend.key.height = unit(4, "cm"),
        #text=element_text(family="serif")
        #axis.title.y = element_text(size = 14),
        #plot.title = element_text(hjust = 1,vjust=-40,size=14)
  )
  pp_com <- pp %>%
    insert_left(phr, width=.1) %>%
    insert_top(phc, height=.1)
  ggsave(pp_com,file=savename, width=5, height=5)
```

31 File: ggraph_facet_network.R

```r
library(tidyverse)
library(gridtext)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(gridExtra)
#### Eucommia analyses
path="network_facet_0.50"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
                        full.names = FALSE, recursive = FALSE,
                        ignore.case = FALSE, include.dirs = FALSE)
# facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
# facet_group <- facet_group[order(facet_group$classification),] ### lead
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes2$classification[grep("null", nodes2$classification)] <- "Undifined"
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
# edge <- list()
##### color palette
pal1= pal_simpsons()(16)
pal1= pal1[c(14,2,4,6,7,8,13,1,15,16)]
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
plot <- list()
for(i in 1:length(edge_list)){
```

```
edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
network_nodes <- as_tbl_graph(edges) %>%
  activate(nodes) %>% #as_tibble()
  mutate(deg = centrality_degree(mode='in')) %>%
    merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
    as_tibble()
  network_edges <- as_tbl_graph(edges) %>%
    activate(edges) %>%
    as_tibble()
  network <- tbl_graph(nodes = network_nodes, edges = network_edges)
  #########
  ######### candidate: graphopt kk fr mds
  layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
  plot[[i]] <-
    ggraph(network, layout = layout) +
    geom_edge_fan(aes(edge_width=ftalign_similarity), color="black", show.legend=F) +
    geom_node_point(aes(size = as.numeric(similarity), fill=classification), shape=21) +
    # geom_node_text(aes(label=name),size=3) +
    scale_color_manual(values=palette) +
    scale_fill_manual(values=palette) +
    scale_edge_width(range=c(0.1,0.7)) +
    facet_edges(~str_wrap(facet, width=30)) +
    guides(size="none", fill="none") +
    theme_grey() +
    theme(
          text=element_text(family="serif"),
          axis.ticks = element_blank(),
          axis.text = element_blank(),
          axis.title = element_blank(),
          panel.background = element_rect(fill="white"),
          #axis.line = element_blank(),
          panel.grid = element_blank(),
          legend.position = "none",
          strip.text = element_text(size=15, face="bold")
    )
    # ggsave(plot[[i]], file="test.svg", width=5, height=5)
```

```
      cat(i, edge_list[i], "\n")
}
##
svg("test_child_nebula.svg", width=18*1.6, height=22*1.5)
n=length(edge_list)
s=n^(1/2); if(round(s)!=s){s=round(s); ss=s+1}else{ss=s}
grid.newpage()
pushViewport(viewport(layout = grid.layout(ss, s)))
r_ss=1
for(i in 1:n){
  c_s=i%%s
  if(c_s==0){c_s=s}
  print( plot[[i]], vp=viewport(layout.pos.row=r_ss, layout.pos.col=c_s ))
  cat("push view port of ",i,"\n")
  if(c_s==s){ r_ss=r_ss+1}
}
dev.off()
#############################
#############################
#############################
#############################
#############################
#############################
#############################
#############################
## network zoomed
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
library(gridSVG)
library(rsvg)
library(gridExtra)
path="network_facet_0.50"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
                         full.names = FALSE, recursive = FALSE,
                         ignore.case = FALSE, include.dirs = FALSE)
# facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
# facet_group <- facet_group[order(facet_group$classification),] ### lead
# edge <- list()
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
nodes$similarity <- as.numeric(nodes$similarity)
nodes$classification <- gsub("null", "Undefined", nodes$classification)
##### color palette
pal1= pal_simpsons()(16)
pal1= pal1[c(14,2,4,6,7,8,13,1,15,16)]
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
select1="Iridoids and derivatives.tsv"
select2="Lignans, neolignans and related compounds.tsv"
select_list=c(34,39)
plot <- list()
stru_path="structure_2d/smiles_draw"
structure_list <- list.files(path = stru_path, pattern = "*.mol.svg.cairo.svg$", all.files = FALSE,
                             full.names = FALSE, recursive = FALSE,
                             ignore.case = FALSE, include.dirs = FALSE)
matrix <- data.frame(structure_list) %>% mutate(catch="T")
canopus_set <- read.csv(file="canopus_pp_filter.tsv",header=T,sep="\t")
canopus_set <- t(canopus_set)
colnames(canopus_set)=as.character(canopus_set[1,])
canopus_set <- canopus_set[-1,]
######################################################################
metadata_path="../canopus_neg.tsv"
metadata <- read.csv(file=metadata_path, header=T, sep="\t", quote = "")
metadata <- metadata[,c(2,3,4)]
```

```
metadata$class <- paste0("C",metadata$absoluteIndex)
######################################################################
######################### start plot
#########################
for(i in select_list){
  edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
  edges_id <- unique(c(edges$source, edges$target))
  cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
  ###########################################
  edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
  edges <- edges[!duplicated(edges[,1:2]), ]
  network_nodes <- as_tbl_graph(edges) %>%
    activate(nodes) %>% #as_tibble()
    mutate(deg = centrality_degree(mode='in')) %>%
      merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
      as_tibble()
    network_edges <- as_tbl_graph(edges) %>%
      activate(edges) %>%
      as_tibble()
    network <- tbl_graph(nodes = network_nodes, edges = network_edges)
    #########
    ######### candidate: graphopt kk fr mds
    layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
    layout_n <- create_layout(network, layout = layout)
    ##########################
    ##########################
    ### figure elements
    ### structure
    elements <- layout_n[,colnames(layout_n) %in% c("name", "x", "y", "similarity", "classification")]
    elements$link_structure <- paste0(elements$name,".mol.svg.cairo.svg")
    grid_elements <- merge(elements, matrix, all.x=T, by.x="link_structure", by.y="structure_list", sor
    ### grobify
    structure_p<-list()
    list_stru <- grid_elements[which(grid_elements$catch=="T"), colnames(grid_elements) %in% c("link_st
    prefix=c()
    for(j in list_stru){
      id<-strsplit(j, split=".mol.svg.cairo.svg")[[1]]
      #structure_p[[as.numeric(id)]] = grobify(readPicture(paste0(path,"/",j)))
      assign(paste0("grob_",id), grobify(readPicture(paste0(stru_path,"/",j))))
      cat(id," >>> ", "structure_p\n")
    }
    aes_stru <- mutate(grid_elements[which(grid_elements$catch=="T"),], grob=paste0("grob_",name), id=na
    ##########################
    ##########################
    ##########################
    ### ring_bar plot
    ring_data <- canopus_set[,colnames(canopus_set) %in% c(elements$name)]
    list_palette = data.frame(cbind(sort(unique(elements$classification)), palette[1:length(unique(eleme
    colnames(list_palette) <- c("classification", "color")
    elements_palette <- merge(elements, list_palette, all.x=T, by="classification", sort=T)
    for(j in 1:ncol(ring_data)){
      id <- colnames(ring_data)[j]
      fill_in=elements_palette[which(elements_palette$name==id),]$color
      fill_border="black"
      df <- data.frame(ring_data[,colnames(ring_data) %in% c(id)])
      df <- mutate(df, class=rownames(df))
      colnames(df)=c("value","class")
      df$num <- seq(nrow(df))
      df <- merge(df, metadata, all.x=T, by="class", sort=T)
      df <- df[order(df$num),]
      df$fill <- paste0(df$class, ": ", df$name)
      df$fill <- factor(df$fill, levels=df[order(df$absoluteIndex), colnames(df) %in% c("fill")])
      p <- ggplot(df, aes(x=num, y=value)) +
        geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = -5, ymax = 0), f
        geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = 0, ymax = 1.1), 
        geom_col(alpha=1, aes(fill=fill), color="white", size=0.02) +
        ylim(-5,1.3) +
        coord_polar() +
        labs(fill="") +
        scale_fill_manual(values=palette) +
        #annotate("text", x=df[nrow(df), colnames(df) %in% "num"], y=-1,
        #        label=paste0("ID: ", id), hjust=0, family="Times", fontface="bold", size=0.7, alpha=0.5
        #scale_fill_gsea() +
        #geom_text(data=df, aes(x=class, y=value+0.1, label=ifelse(value>0.7, class, "")),
        #        color="black", fontface="bold",alpha=0.6, size=0.3, inherit.aes = FALSE ) +
        theme_minimal() +
        theme(
              text=element_text(family="Times"),
              axis.ticks = element_blank(),
              #plot.background = element_rect(fill = "transparent"),
              axis.text = element_blank(),
              axis.title = element_blank(),
              panel.grid = element_blank(),
              panel.grid.major =element_blank(),
              panel.grid.minor = element_blank(),
              legend.position = "none",
              legend.title=element_text(face="bold", hjust= -0.5),
              panel.border = element_blank(),
              plot.margin =unit(c(0,0,0,0),"cm"),
              panel.spacing =unit(c(0,0,0,0),"cm") # Adjust the margin to make in sort labels are not t
        )
        assign(paste0("ring_", id), p)
    }
    ##########################
    ##########################
    plot[[i]] <-
      ggraph(layout_n) +
      geom_edge_fan(aes(edge_width=ftalign_similarity),
                    color="lightblue", show.legend=F) +
geom_node_point(aes(fill=str_wrap(classification, width=25)), size=1, shape=21) +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1,0.7)) +
facet_edges(~facet) +
labs(fill="Access classes", size="Tanimoto\nsimilarity") +
guides(fill=guide_legend(override.aes=list(size=5))) +
theme_grey() +
theme(
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      legend.title= element_text(face="bold"),
      #panel.background = element_rect(fill="white"),
      legend.key.height = unit(0.6, "cm"),
      axis.line = element_blank(),
      #panel.grid = element_blank(),
      #legend.position = "none",
      strip.text = element_text(size=15, face="bold"),
      #plot.margin =unit(c(0,0,0,0),"cm"),
      #panel.spacing =unit(c(0,0,0,0),"cm")
)
assign("find_id", plot[[i]])
```

```
## size and posion adjust
min=min(elements$similarity)
delta=max(elements$similarity)-min
step=1/delta
if(i==39 | i==34){p_size=0.85; p_dist=0.105}else{p_size=0.65; p_dist=0.09}
## draw subview into the network
for(k in 1:ncol(ring_data)){
  id=colnames(ring_data)[k]
  aes_ring=elements[which(elements$name==id),]
  size=(aes_ring$similarity-min)*step+p_size
  ## ppcp datacet
  plot[[i]] <- plot[[i]] + geom_subview(x=aes_ring$x-p_dist, y=aes_ring$y-p_dist,
                                        subview=get(paste0("ring_", id)),
                                        width=size, height=size )
  ## structure
  kk <- which(aes_stru$id==id)
  size=aes_stru[kk,]$similarity
  plot[[i]] <- plot[[i]] + geom_subview(x=aes_stru[kk,]$x, y=aes_stru[kk,]$y,
                                        subview=arrangeGrob( get(aes_stru[kk,]$grob) ),
                                        width=size*6/5, height=size*6/5)
}
#############################################
ggsave(plot[[i]], file=paste0("tt_zoom_",i,".svg"), width=7, height=5.5)
cat(i, edge_list[i], "\n")
}

p <- find_id + geom_node_text(aes(label=name),size=3)
#############################################
#############################################
#############################################
### ladder network
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
```

```r
library(gridSVG)
library(ggpubr)
path="network_facet_ladder1"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
                        full.names = FALSE, recursive = FALSE,
                        ignore.case = FALSE, include.dirs = FALSE)
# facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
# facet_group <- facet_group[order(facet_group$classification),] ### lead
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
# edge <- list()
##### color palette
pal0= pal_npg()(10)
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal0, pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
select1="Iridoids and derivatives.tsv"
select2="Lignans, neolignans and related compounds.tsv"
select_list=c(34,39)
plot <- list()
######################### start plot
#########################
#for(i in select_list){
i=39
edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
network_nodes <- as_tbl_graph(edges) %>%
  activate(nodes) %>% #as_tibble()
  mutate(deg = centrality_degree(mode='in')) %>%
    merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
    as_tibble()
  network_edges <- as_tbl_graph(edges) %>%
    activate(edges) %>%
    as_tibble()
  ##########################################
  num_edges <- data.frame(matrix(c(0.2, 0.3, 0.4, 0.5, 0.6, 0.7)))
  colnames(num_edges) <- "tlimit"
  num_edges$num=NA
  for(tlimit in c(0.2, 0.3, 0.4, 0.5, 0.6, 0.7)){
    network_edges_filter <- network_edges[which(network_edges$ftalign_similarity > tlimit),]
    list <- unique(c(network_edges_filter$from, network_edges_filter$to))
    network_nodes$rownames=rownames(network_nodes)
    num_edges[which(num_edges$tlimit==tlimit),]$num <- nrow(network_edges_filter)
    network <- tbl_graph(nodes = network_nodes, edges = network_edges_filter) %>%
      activate(nodes) %>% #as_tibble()
      mutate(deg = centrality_degree(mode='in'))
      #########
      ######### candidate: graphopt kk fr mds
      layout=ifelse(tlimit==0.2, "fr", "fr")
      layout_n <- create_layout(network, layout = layout)
      ##########################################
      plot[[i]] <-
        ggraph(layout_n) +
        geom_edge_fan(aes(edge_width=ftalign_similarity), label_alpha=0.3, color="#4DBBD5FF", show.lege
        #geom_node_point(aes(fill=str_wrap(classification, width=25), size=similarity), shape=21) +
        geom_node_point(aes(fill=deg, size=as.numeric(similarity)), alpha=ifelse(layout_n$rownames %in%
        #scale_fill_gradientn(labels=c("low", "", "", "", "high"), colors=c("#1B1919FF", "#4DBBD5FF"))
        scale_fill_gradientn(colors=c("#1B1919FF", "#4DBBD5FF")) +
        scale_edge_width(range=c(0.1,0.7)) +
        #facet_edges(~facet) +
        guides(alpha="none") +
        labs(fill="Centrality\ndegree", size="Tanimoto\nsimilarity") +
        #guides(size="none", fill="none") +
        theme_grey() +
        theme(
              text=element_text(family="serif", size=20),
              axis.ticks = element_blank(),
              axis.text = element_blank(),
              axis.title = element_blank(),
              legend.title= element_text(face="bold", size=15),
              #panel.background = element_rect(fill="white"),
              legend.key.height = unit(0.3, "cm"),
              legend.key.width = unit(0.5, "cm"),
              axis.line = element_blank(),
              #panel.grid = element_blank(),
              #legend.position = "none",
              #strip.text = element_text(size=15, face="bold")
              plot.margin =unit(c(0,0,0,0),"cm"),
              panel.spacing =unit(c(0,0,0,0),"cm")
        )
        if(tlimit==0.7 & i==39){
          plot[[i]] <- plot[[i]] + scale_fill_gradientn(labels = c("low", "", "", "high"),
                                                        colors=c("#1B1919FF", "#4DBBD5FF"))
        }
        assign(paste0("plot_", tlimit), plot[[i]])
        #ggsave(plot[[i]], file=paste0("tlimit_", tlimit, ".svg"), width=7, height=5)
        cat(i, edge_list[i], "\n")
  }
  #######################################################################
  p_step1 <- ggplot(num_edges, aes(x=as.factor(tlimit), y=num)) +
    geom_col() +
    labs(x="NFTAS (PPCP   0.5)", y="Edges")+
    theme_classic() +
    theme(
          text=element_text(family="Times", size=20),
          axis.title = element_text(face="bold"),
          axis.title.y = element_text(hjust=1),
          plot.margin =unit(c(0.1,0.1,0.1,0.1),"cm"),
          panel.spacing =unit(c(0,0,0,0),"cm")
          ) +
      coord_cartesian(ylim = c(0,1500))
    p_step2 <- ggplot(num_edges, aes(x=as.factor(tlimit), y=num)) +
      geom_col() +
      labs(x=NULL, y="number")+
      theme_classic() +
      ggtitle("Lignans, neolignans and related compounds") +
      theme(
            text=element_text(family="Times", size=20),
            axis.title.y = element_text(hjust=0, face="bold"),
            plot.margin =unit(c(0.1,0.1,0.1,0.1),"cm"),
            plot.title = element_text(hjust=0.5, size=25, face="bold"),
            panel.spacing =unit(c(0,0,0,0),"cm"),
            axis.text.x = element_blank(),
            axis.ticks.x = element_blank()
            ) +
      coord_cartesian(ylim = c(6500,7000))
    p_step <- ggarrange(p_step2, p_step1, heights=c(2/5, 3/5),ncol = 1, nrow = 2, common.legend = TRUE,
    # p2 <- gg.gap(plot = p_step,
    #           segments = c(1000, 3400),
    #           fontfamily = "serif",
    #           #tick_width = 10,
    #rel_heights = c(0.25, 0, 0.1),# 设置分隔为的三个部分的宽度
    #           ylim = c(0, 3800)
    #           ) +
    #       theme(
    #    axis.title = element_text(face="bold"),
    #    plot.margin =unit(c(0,0,0,0),"cm"),
    #         panel.spacing =unit(c(0,0,0,0),"cm"),
    #         text=element_text(family="serif")
    #        )
    ########################################################################
    #png("test.png", width=1000, height=1000)
    #tiff("step_network.tiff", width=1200*7/5, height=400*7/5, compression="lzw")
    svg("step_network.svg", width=24, height=8)
    #pdf("step_network.pdf", width=24, height=8)
    grid.newpage()
    n=0
    adjust=3.8
    posi_just=3
    pushViewport( viewport(layout = grid.layout(40, 60+adjust+posi_just)) )
    print( p_step, vp=viewport(layout.pos.row=1:20, layout.pos.col=1:(60+adjust) ))
    for(tlimit in c(0.2, 0.3, 0.4, 0.5, 0.6, 0.7)){
      n=n+1
      print( get( paste0("plot_", tlimit) ),
            vp=viewport(layout.pos.row=21:39,
                        layout.pos.col=( ((n-1)*10+1+posi_just):(n*10+adjust+posi_just) )
            )
            ) }
    grid.text("Morphology", x=0.02, y=0.25, rot=90,
              gp = gpar( fontface = "bold", fontsize = 20, fontfamily = "Times", fontangle=90) )
```

```
dev.off()
#}
############################
############################
############################
############################
############################
############################
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
library(gridSVG)
library(ggpubr)
library(reshape2)
# facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
# facet_group <- facet_group[order(facet_group$classification),] ### lead
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
##############################
##############################
nums <- data.frame(matrix(c(0.3, 0.5, 0.7, 0.9, 0.95, 0.99)))
colnames(nums) <- "pp"
nums$e_num=NA
nums$n_num=NA
##############################
##############################
for(pp in c(0.3,0.5,0.7,0.9,0.95,0.99)){
  path=paste0("network_facet_", pp)
  edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
                          full.names = FALSE, recursive = FALSE,
                          ignore.case = FALSE, include.dirs = FALSE)
```

```r
    name="Lignans, neolignans and related compounds.tsv"
    edges <- read.csv(file=paste0(path, "/", edge_list[which(edge_list==name)]), header=T,sep="\t")
    edges_id <- unique(c(edges$source, edges$target))
    cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
    ###########################################
    edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
    edges <- edges[!duplicated(edges[,1:2]), ]
    network_nodes <- as_tbl_graph(edges) %>%
      activate(nodes) %>% #as_tibble()
      mutate(deg = centrality_degree(mode='in')) %>%
        merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
        as_tibble()
      network_edges <- as_tbl_graph(edges) %>%
        activate(edges) %>%
        as_tibble()
      list <- unique(c(network_edges[which(network_edges$ftalign_similarity!=1),]$from,
                       network_edges[which(network_edges$ftalign_similarity!=1),]$to))
      network_nodes$rownames=rownames(network_nodes)
      ##############################
      nums[which(nums$pp==pp), ]$e_num= nrow(network_edges)
      nums[which(nums$pp==pp), ]$n_num= nrow(network_nodes)
      ##############################
      ##############################
      ##############################
      network <- tbl_graph( nodes = network_nodes, edges = network_edges ) %>%
        activate(nodes) %>% #as_tibble()
        mutate(deg = centrality_degree(mode='in'))
        #########
        ######### candidate: graphopt kk fr mds
        layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
        layout_n <- create_layout(network, layout = layout)
        p <-
          ggraph(layout_n) +
          geom_edge_fan(aes(edge_width=ftalign_similarity), color="#4DBBD5FF", show.legend=F) +
          #geom_node_point(aes(fill=str_wrap(classification, width=25), size=similarity), shape=21) +
          geom_node_point(aes(fill=deg, size=as.numeric(similarity)),
                          alpha=ifelse(layout_n$rownames %in% list, 1, 0.1), shape=21) +
    scale_edge_width(range=c(0.1,0.7)) +
    scale_fill_gradient(low="#1B1919FF", high="#DC0000FF") +
    guides(alpha="none") +
    labs(fill="Centrality\ndegree", size="Tanimoto\nsimilarity") +
    theme_grey() +
    theme(
          text=element_text(family="Times", size=20),
          axis.ticks = element_blank(),
          axis.text = element_blank(),
          axis.title = element_blank(),
          legend.title= element_text(face="bold", size=15),
          #panel.background = element_rect(fill="white"),
          legend.key.height = unit(0.3, "cm"),
          legend.key.width = unit(0.5, "cm"),
          axis.line = element_blank(),
          #panel.grid = element_blank(),
          #legend.position = "none",
          #strip.text = element_text(size=15, face="bold")
          plot.margin =unit(c(0,0,0,0),"cm"),
          panel.spacing =unit(c(0,0,0,0),"cm")
    )
    if(pp==0.99){
      p <- p + scale_fill_gradient(low="#1B1919FF", high="#DC0000FF", labels = c("low", "", "", "high
    }
    assign(paste0("plot_", pp), p)
  }
  nums1 <- melt(nums, measure.vars=c("e_num", "n_num"), variable.name="var", value.name="expr")
  p_step <- ggplot(nums1, aes(x=as.factor(pp), y=expr, fill=var)) +
    geom_col(position=position_dodge(width = 0.9), width = 0.8) +
    labs(x="", y="")+
    theme_classic() +
    scale_fill_uchicago("dark", labels=c("nodes", "edges")) +
    labs(x="PPCP (NFTAS   0.4)", y="Number", fill="Type") +
    ggtitle("Lignans, neolignans and related compounds") +
    theme(
          text=element_text(family="Times", size=20),
          axis.title = element_text(face="bold"),
          axis.title.y = element_text(hjust=0.5),
          legend.title = element_text(size=15, face="bold"),
          plot.title = element_text(hjust=0.5, size=25, face="bold"),
          legend.key.height = unit(0.3, "cm"),
          legend.key.width = unit(0.5, "cm"),
          legend.position = c(1.02, 0.5),
          plot.margin =unit(c(0.1,0.1,0.1,0.1),"cm"),
          panel.spacing =unit(c(0,0,0,0),"cm")
          ) +
    coord_cartesian(ylim = c(0,600))
  ##############################
  ##############################
  svg("step2_network.svg", width=24, height=8)
  grid.newpage()
  n=0
  adjust=3.8
  posi_just=3
  pushViewport( viewport(layout = grid.layout(40, 60+adjust+posi_just)) )
  print( p_step, vp=viewport(layout.pos.row=1:20, layout.pos.col=1:(60+adjust) ))
  for(pp in c(0.3, 0.5, 0.7, 0.9, 0.95, 0.99)){
    n=n+1
    print( get( paste0("plot_", pp) ),
          vp=viewport(layout.pos.row=21:39,
                      layout.pos.col=( ((n-1)*10+1+posi_just):(n*10+adjust+posi_just) )
          )
          ) }
  grid.text("Morphology", x=0.02, y=0.25, rot=90,
            gp = gpar( fontface = "bold", fontsize = 20, fontfamily = "Times", fontangle=90) )
  dev.off()
  #############################
  #############################
  # EX
  metadata_path="../canopus_neg.tsv"
  metadata <- read.csv(file=metadata_path, header=T, sep="\t", quote = "")
  metadata <- metadata[,c(2,3,4)]
  metadata$class <- paste0("C",metadata$absoluteIndex)
  id <- 527
  fill_in=elements_palette[which(elements_palette$name==id),]$color
  fill_border="black"
  df <- data.frame(ring_data[,colnames(ring_data) %in% c(id)])
  df <- mutate(df, class=rownames(df))
  colnames(df)=c("value","class")
  df$num <- seq(nrow(df))
  df <- merge(df, metadata, all.x=T, by="class", sort=T)
  df <- df[order(df$num),]
  df$fill <- paste0(df$class, ": ", df$name)
  df$fill <- factor(df$fill, levels=df[order(df$absoluteIndex), c("fill")])
  label_data <- df
  number_of_bar <- nrow(label_data)
```

```
    label_data$id <- seq(1,nrow(label_data))
    angle <-  90 - 360 * (label_data$id-0.5) /number_of_bar
    label_data$hjust<-ifelse( angle < -90, 1, 0)
    label_data$angle<-ifelse(as.numeric(angle) < (-90), angle+180, angle)
    rm(angle)
    p <- ggplot(df, aes(x=num, y=value)) +
      geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = -5, ymax = 0), fil
      geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = 0, ymax = 1.1), fi
      geom_col(alpha=1, aes(fill=fill), color="white", size=0.1) +
      ylim(-5,1.3) +
      coord_polar() +
      labs(fill="") +
      scale_fill_manual(values=palette) +
      #scale_fill_gsea() +
      geom_text(data=label_data, aes(x=num, y=0.5, label=class, angle=angle),
                color="white", fontface="bold",alpha=0.8, size=1.5,inherit.aes = FALSE ) +
      theme_minimal() +
      theme(
            text=element_text(family="Times", size=8),
            axis.ticks = element_blank(),
            #plot.background = element_rect(fill = "transparent"),
            axis.text = element_blank(),
            axis.title = element_blank(),
            panel.grid = element_blank(),
            panel.grid.major =element_blank(),
            panel.grid.minor = element_blank(),
            legend.key.width = unit(0.5,"cm"),
            legend.key.height = unit(0.5,"cm"),
            #legend.position = "none",
            legend.title=element_text(face="bold", hjust= -0.5),
            panel.border = element_blank(),
            plot.margin =unit(c(0,0,0,0),"cm"),
            panel.spacing =unit(c(0,0,0,0),"cm") # Adjust the margin to make in sort labels are not tru
      )
      complement <- readPicture(paste0(stru_path,"/", id,".mol.svg.cairo.svg"))
      svg("annotate.svg", width=14, height=7)
      grid.newpage()
      pushViewport(viewport(layout = grid.layout(50, 50)))
      #print( pal_grey, vp=viewport(layout.pos.row=1:50, layout.pos.col=1:20 ))
      print( p, vp=viewport(layout.pos.row=1:50, layout.pos.col=1:50 ))
      grid.picture(complement, x=0.18, y=0.5, width=0.5, height=0.5)
```

```r
dev.off()
# ggsave(pal_grey, file="bg.svg")

## compare with gnps, so add noise
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(stringr)
#### Eucommia analyses
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
nodes <- nodes[which(nodes$superclass!=""), ]
edges <- read.csv(file="source_target_tree_0.4.tsv", header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
############################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
###########  only show pp > 0.9
network_nodes <- as_tbl_graph(edges) %>%
  activate(nodes) %>% #as_tibble()
  mutate(deg = centrality_degree(mode='in')) %>%
    merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
    as_tibble()
  network_edges <- as_tbl_graph(edges) %>%
    activate(edges) %>%
    as_tibble()
  network <- tbl_graph(nodes = network_nodes, edges = network_edges)
  layout_n <- create_layout(network, layout = "mds")
  #### molecular network
  p <- ggraph(layout_n) +
    geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
    geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(superclass, width=25)), shap
    #geom_node_text(aes(filter= deg>12,label=name),size=1) +
    scale_color_manual(values=palette) +
    scale_fill_manual(values=palette) +
    scale_edge_width(range=c(0.1, 0.7)) +
    #facet_nodes(~classification) +
    #guides(size="none") +
    labs(fill="Superclass", size="Tanimoto similarity") +
    theme_grey() +
    theme(
          text=element_text(family="Times"),
          axis.ticks = element_blank(),
          axis.text = element_blank(),
          axis.title = element_blank(),
          panel.background = element_rect(fill="white"),
          #axis.line = element_blank(),
          legend.key.width = unit(1, "cm"),
          legend.key.height = unit(1.8, "cm"),
          legend.title = element_text(size=20, face="bold", hjust=0.2),
          legend.text = element_text(size=20),
          legend.background = element_rect(fill="transparent"),
          #legend.position = c(0.6,0.25),
          panel.grid = element_blank(),
          strip.text = element_text(size=20, face="bold")
    )
    # ggsave(p, file="parent_network.tiff", width=20, height=22)
    ggsave(p, file="parent_network.svg", width=20, height=16)
```

32 File: ggraph_gnps_merge.R

```
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
#### Eucommia analyses
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
```

```
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
#lai; palette= unique(c(pal1[c(6:8,9:10,1:4,11:16)], pal2, pal3, pal4))
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
nodes <- nodes[which(nodes$superclass!=""),]
# from mcnebula
edges <- read.csv(file="source_target_tree_0.4.tsv", header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
allid <- unique(c(edges$source, edges$target))
nrow(edges[edges$source!=edges$target,])
# from gnps
edges <- read.csv(file="gnps07_merge.tsv", header=T,sep="\t")
colnames(edges)[1:5] <- c("source", "target",  "delta_m.z", "MEH", "ftalign_similarity")
edges <- edges[which(edges$ftalign_similarity > 0.6808 & edges$source!=edges$target),]
edges_id <- unique(c(edges$source, edges$target))
# edges id which not find in nodes
cont_edges_id <- edges_id[!(edges_id %in% allid)]
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nrow(edges)
# insert the connectionless nodes
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- allid[!(allid %in% edges_id)]
cont_list <- data.frame(cont_edges_id)
colnames(cont_list) <- c("source")
cont_list <- mutate(cont_list, target=source, delta_m.z=0, MEH="N", ftalign_similarity=1)
edges <- rbind(edges[, 1:5], cont_list)
# write.table(edges[, c(1,2,5)], file="filter_net_0.6808", sep="\t", quote=F, row.names=F, col.names=F
###########  only show pp > 0.9
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
```

```
 network_edges <- as_tbl_graph(edges) %>%
    activate(edges) %>%
    as_tibble()
 network <- tbl_graph(nodes = network_nodes, edges = network_edges)
 layout_n <- create_layout(network, layout = 'stress')
 #### molecular network
 p <- ggraph(layout_n) +
   geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
   geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(superclass, width=25)), shape=21) +
   #geom_node_text(aes(filter= deg>12,label=name),size=1) +
   scale_color_manual(values=palette) +
   scale_fill_manual(values=palette) +
   scale_edge_width(range=c(0.1, 0.7)) +
   #facet_nodes(~classification) +
   labs(fill="Superclass", size="Tanimoto similarity") +
   guides(fill = guide_legend(override.aes = list(size=5))) +
   theme_grey() +
   theme(
         text=element_text(family="Times"),
         axis.ticks = element_blank(),
         axis.text = element_blank(),
         axis.title = element_blank(),
         panel.background = element_rect(fill="white"),
         #axis.line = element_blank(),
         legend.key.width = unit(1.5, "cm"),
         legend.key.height = unit(1.8, "cm"),
         legend.title = element_text(size=20, face="bold", hjust=0.2),
         legend.text = element_text(size=20),
         legend.background = element_rect(fill="transparent"),
         #legend.position = c(0.6,0.25),
         panel.grid = element_blank(),
         strip.text = element_text(size=20, face="bold")
   )
   # ggsave(p, file="parent_network.tiff", width=20, height=22)
   ggsave(p, file="gnps_network_merge07.svg", width=19, height=16)
## facet plot
 library(tidyverse)
 library(igraph)
 library(ggraph)
 library(tidygraph)
 library(ggsci)
```

```r
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(gridExtra)
library(stringr)
#### Eucommia analyses
path="gnps_network_facet_0.5"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)
# facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
# facet_group <- facet_group[order(facet_group$classification),] ### lead
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes2$classification[grep("null", nodes2$classification)] <- "Undifined"
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
# edge <- list()
##### color palette
pal1= pal_simpsons()(16)
pal1= pal1[c(14,2,4,6,7,8,13,1,15,16)]
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
plot <- list()
for(i in 1:length(edge_list)){
edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
#########
######### candidate: graphopt kk fr mds
layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
plot[[i]] <-
ggraph(network, layout = layout) +
geom_edge_fan(aes(edge_width=ftalign_similarity), color="black", show.legend=F) +
geom_node_point(aes(size = as.numeric(similarity), fill=classification), shape=21) +
# geom_node_text(aes(label=name),size=3) +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1,0.7)) +
facet_edges(~str_wrap(facet, width=30)) +
guides(size="none", fill="none") +
theme_grey() +
theme(
      text=element_text(family="serif"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      panel.background = element_rect(fill="white"),
      #axis.line = element_blank(),
      panel.grid = element_blank(),
      legend.position = "none",
      strip.text = element_text(size=15, face="bold")
     )
# ggsave(plot[[i]], file="test.svg", width=5, height=5)
cat(i, edge_list[i], "\n") }
# theme_graph(base_family = "Times",
#       foreground = "#8491B4FF",
#       strip_text_size = 20,
#       fg_text_colour = "white",
#       plot_margin = margin(10, 10, 10, 10))
##############################
```

```
##############################
#png("test.png", width=1000, height=1000)
#tiff("merge_grey.tiff", width=2000, height=2200, compression="lzw")
svg("gnps_child_nebula.svg", width=18*1.6, height=22*1.5)
n=length(edge_list)
s=n^(1/2); if(round(s)!=s){s=round(s); ss=s+1}else{ss=s}
grid.newpage()
pushViewport(viewport(layout = grid.layout(ss, s)))
r_ss=1
for(i in 1:n){
c_s=i%%s
if(c_s==0){c_s=s}
print( plot[[i]], vp=viewport(layout.pos.row=r_ss, layout.pos.col=c_s ))
cat("push view port of ",i,"\n")
if(c_s==s){ r_ss=r_ss+1}
}
dev.off()
## zoom the specific class
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
library(gridSVG)
library(rsvg)
library(gridExtra)
path="gnps_network_facet_0.5"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)
# facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
# facet_group <- facet_group[order(facet_group$classification),] ### lead
# edge <- list()
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
```

```r
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
nodes$similarity <- as.numeric(nodes$similarity)
nodes$classification <- gsub("null", "Undefined", nodes$classification)
##### color palette
pal1= pal_simpsons()(16)
pal1= pal1[c(14,2,4,6,7,8,13,1,15,16)]
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
select1="Iridoids and derivatives.tsv"
select2="Lignans, neolignans and related compounds.tsv"
select_list=c(34,39)
plot <- list()
stru_path="structure_2d/smiles_draw"
structure_list <- list.files(path = stru_path, pattern = "*.mol.svg.cairo.svg$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)
matrix <- data.frame(structure_list) %>% mutate(catch="T")
canopus_set <- read.csv(file="canopus_pp_filter.tsv",header=T,sep="\t")
canopus_set <- t(canopus_set)
colnames(canopus_set)=as.character(canopus_set[1,])
canopus_set <- canopus_set[-1,]
######################################################################
metadata_path="../canopus_neg.tsv"
metadata <- read.csv(file=metadata_path, header=T, sep="\t", quote = "")
metadata <- metadata[,c(2,3,4)]
metadata$class <- paste0("C",metadata$absoluteIndex)
######################################################################
######################### start plot
#########################
for(i in select_list){
edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
```

```
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
#########
######### candidate: graphopt kk fr mds
layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
layout_n <- create_layout(network, layout = layout)
##########################
##########################
### figure elements
### structure
elements <- layout_n[,colnames(layout_n) %in% c("name", "x", "y", "similarity", "classification")]
elements$link_structure <- paste0(elements$name,".mol.svg.cairo.svg")
grid_elements <- merge(elements, matrix, all.x=T, by.x="link_structure", by.y="structure_list", sort=T
### grobify
structure_p<-list()
list_stru <- grid_elements[which(grid_elements$catch=="T"), colnames(grid_elements) %in% c("link_struc
#grid.picture(readPicture(paste0(stru_path,"/",j)))
for(j in list_stru){
  id<-strsplit(j, split=".mol.svg.cairo.svg")[[1]]
  #structure_p[[as.numeric(id)]] = grobify(readPicture(paste0(path,"/",j)))
  assign(paste0("grob_",id), grobify(readPicture(paste0(stru_path,"/",j))))
  cat(id," >>> ", "structure_p\n")
}
aes_stru <- mutate(grid_elements[which(grid_elements$catch=="T"),], grob=paste0("grob_",name))
##########################
##########################
##########################
### ring_bar plot
ring_data <- canopus_set[,colnames(canopus_set) %in% c(elements$name)]
list_palette = data.frame(cbind(sort(unique(elements$classification)), palette[1:length(unique(element
colnames(list_palette) <- c("classification", "color")
```

```
elements_palette <- merge(elements, list_palette, all.x=T, by="classification", sort=T)
for(j in 1:ncol(ring_data)){
id <- colnames(ring_data)[j]
fill_in=elements_palette[which(elements_palette$name==id),]$color
fill_border="black"
df <- data.frame(ring_data[,colnames(ring_data) %in% c(id)])
df <- mutate(df, class=rownames(df))
colnames(df)=c("value","class")
df$num <- seq(nrow(df))
df <- merge(df, metadata, all.x=T, by="class", sort=T)
df <- df[order(df$num),]
df$fill <- paste0(df$class, ": ", df$name)
df$fill <- factor(df$fill, levels=df[order(df$absoluteIndex), colnames(df) %in% c("fill")])
p <- ggplot(df, aes(x=num, y=value)) +
     geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = -5, ymax = 0), fil
     geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = 0, ymax = 1.1), fi
     geom_col(alpha=1, aes(fill=fill), color="white", size=0.02) +
         ylim(-5,1.3) +
     coord_polar() +
     labs(fill="") +
     scale_fill_manual(values=palette) +
     #annotate("text", x=df[nrow(df), colnames(df) %in% "num"], y=-1,
     #         label=paste0("ID: ", id), hjust=0, family="Times", fontface="bold", size=0.7, alpha=0.5
     #scale_fill_gsea() +
     #geom_text(data=df, aes(x=class, y=value+0.1, label=ifelse(value>0.7, class, "")),
   #       color="black", fontface="bold",alpha=0.6, size=0.3, inherit.aes = FALSE ) +
     theme_minimal() +
     theme(
       text=element_text(family="Times"),
       axis.ticks = element_blank(),
           #plot.background = element_rect(fill = "transparent"),
       axis.text = element_blank(),
       axis.title = element_blank(),
       panel.grid = element_blank(),
       panel.grid.major =element_blank(),
       panel.grid.minor = element_blank(),
       legend.position = "none",
       legend.title=element_text(face="bold", hjust= -0.5),
       panel.border = element_blank(),
       plot.margin =unit(c(0,0,0,0),"cm"),
           panel.spacing =unit(c(0,0,0,0),"cm") # Adjust the margin to make in sort labels are not tru
```

```
    )
assign(paste0("ring_", id), p) }
##########################
##########################
plot[[i]] <-
ggraph(layout_n) +
geom_edge_fan(aes(edge_width=ftalign_similarity, label=delta_m.z), label_size=0.7, label_alpha=0.3, col
geom_node_point(aes(fill=str_wrap(classification, width=25)), size=1, shape=21) +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1,0.7)) +
facet_edges(~facet) +
labs(fill="Access classes", size="Tanimoto\nsimilarity") +
#guides(size="none", fill="none") +
theme_grey() +
theme(
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      legend.title= element_text(face="bold"),
      #panel.background = element_rect(fill="white"),
      legend.key.height = unit(0.6, "cm"),
      axis.line = element_blank(),
      #panel.grid = element_blank(),
      #legend.position = "none",
      strip.text = element_text(size=15, face="bold"),
      #plot.margin =unit(c(0,0,0,0),"cm"),
      #panel.spacing =unit(c(0,0,0,0),"cm")
     )
assign("find_id", plot[[i]])
#plot[[i]] <- plot[[i]] + geom_subview(x=aes_ring$x-(size/4.7)*1.05, y=aes_ring$y-(size/4.7)*1.05,
       #           subview=get(paste0("ring_", id)), width=size*4.5*1.05, height=size*4.5*1.05 )
#ggsave(plot[[i]], file="test.svg", width=7, height=5)
min=min(elements$similarity)
delta=max(elements$similarity)-min
step=1/delta
if(i==39 | i==34){p_size=0.85; p_dist=0.105}else{p_size=0.65; p_dist=0.09}
for(k in 1:ncol(ring_data)){
id=colnames(ring_data)[k]
aes_ring=elements[which(elements$name==id),]
```

```r
size=(aes_ring$similarity-min)*step+p_size
plot[[i]] <- plot[[i]] + geom_subview(x=aes_ring$x-p_dist, y=aes_ring$y-p_dist,
                   subview=get(paste0("ring_", id)), width=size, height=size )  }
#############################################
#############################################
for(k in 1:nrow(aes_stru)){
size=aes_stru[k,]$similarity
plot[[i]] <- plot[[i]] + geom_subview(x=aes_stru[k,]$x, y=aes_stru[k,]$y,
                   subview=arrangeGrob( get(aes_stru[k,]$grob) ), width=size*6/5, height=size*6/5) }
# plot[[i]] <- plot[[i]] + facet_zoom(xlim = c(2, 4), ylim=c(5,7))
ggsave(plot[[i]], file=paste0("gnps_zoom_",i,".svg"), width=7, height=5)
cat(i, edge_list[i], "\n")
}
p <- find_id + geom_node_text(aes(label=name),size=3)
##############################################################################################
## add noise (other class)
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
#### Eucommia analyses
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
#lai; palette= unique(c(pal1[c(6:8,9:10,1:4,11:16)], pal2, pal3, pal4))
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
nodes <- nodes[which(nodes$superclass!=""),]
# from mcnebula
edges <- read.csv(file="source_target_tree_0.4.tsv", header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
############################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
allid <- unique(c(edges$source, edges$target))
```

```r
nrow(edges[edges$source!=edges$target,])
# from gnps
edges <- read.csv(file="gnps04.tsv", header=T,sep="\t")
colnames(edges)[1:5] <- c("source", "target",  "delta_m.z", "MEH", "ftalign_similarity")
edges <- edges[which(edges$ftalign_similarity > 0.6808 & edges$source!=edges$target),]
edges_id <- unique(c(edges$source, edges$target))
# edges id which not find in nodes
cont_edges_id <- edges_id[!(edges_id %in% allid)]
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nrow(edges)
# insert the connectionless nodes
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- allid[!(allid %in% edges_id)]
cont_list <- data.frame(cont_edges_id)
colnames(cont_list) <- c("source")
cont_list <- mutate(cont_list, target=source, delta_m.z=0, MEH="N", ftalign_similarity=1)
edges <- rbind(edges[, 1:5], cont_list)
# add lignans and iridoids
path="gnps_network_facet_0.5"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
            full.names = FALSE, recursive = FALSE,
            ignore.case = FALSE, include.dirs = FALSE)
allid <- c()
for(i in c(34,39,41)){
  ed <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
  ed_id <- unique(c(ed$source, ed$target))
  allid <- c(allid, ed_id)
  assign(paste0("from_", i), ed_id)
}
allid <- unique(allid)
edges_id <- unique(c(edges$source, edges$target))
# edges id which not find in nodes
cont_edges_id <- edges_id[!(edges_id %in% allid)]
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nodes <- mutate(nodes, from_class=
                ifelse(nodes$id %in% from_34, "Iridoids",
                       ifelse(nodes$id %in% from_39, "Lignans",
                              ifelse(nodes$id %in% from_41, "Long-chain fatty acids", NA)
                       )
```

```
                        ))

# write.table(edges[, c(1,2,5)], file="filter_net_0.6808", sep="\t", quote=F, row.names=F, col.names=F
###########  only show pp > 0.9
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
layout_n <- create_layout(network, layout = 'fr')
#### molecular network
p <- ggraph(layout_n) +
  geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
  geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(from_class, width=25)), shape=21) +
  #geom_node_text(aes(filter= deg>12,label=name),size=1) +
  scale_color_manual(values=palette) +
  scale_fill_manual(values=palette) +
  scale_edge_width(range=c(0.1, 0.7)) +
  #facet_nodes(~classification) +
  labs(fill="From class", size="Tanimoto similarity") +
  guides(fill=guide_legend(override.aes=list(size=5))) +
  theme_grey() +
  theme(
        text=element_text(family="Times"),
        axis.ticks = element_blank(),
        axis.text = element_blank(),
        axis.title = element_blank(),
        panel.background = element_rect(fill="white"),
        #axis.line = element_blank(),
        legend.key.width = unit(1, "cm"),
        legend.key.height = unit(1, "cm"),
        legend.title = element_text(size=15, face="bold", hjust=0.2),
        legend.text = element_text(size=15),
        legend.background = element_rect(fill="transparent"),
        #legend.position = c(0.6,0.25),
        panel.grid = element_blank(),
        strip.text = element_text(size=20, face="bold")
```

```r
  )
  # ggsave(p, file="parent_network.tiff", width=20, height=22)
  ggsave(p, file="gnps_add_noise_34_39_41.svg", width=12, height=10)
  #ggsave(p, file="test_gnps_add_noise_34_39.svg", width=12, height=10)
```

33 File: ggraph_gnps.R

```r
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
#### Eucommia analyses
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
#lai; palette= unique(c(pal1[c(6:8,9:10,1:4,11:16)], pal2, pal3, pal4))
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
nodes <- nodes[which(nodes$superclass!=""),]
# from mcnebula
edges <- read.csv(file="source_target_tree_0.4.tsv", header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
allid <- unique(c(edges$source, edges$target))
nrow(edges[edges$source!=edges$target,])
# from gnps
edges <- read.csv(file="gnps04.tsv", header=T,sep="\t")
colnames(edges)[1:5] <- c("source", "target",  "delta_m.z", "MEH", "ftalign_similarity")
edges <- edges[which(edges$ftalign_similarity > 0.6808 & edges$source!=edges$target),]
edges_id <- unique(c(edges$source, edges$target))
# edges id which not find in nodes
cont_edges_id <- edges_id[!(edges_id %in% allid)]
```

```r
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nrow(edges)
# insert the connectionless nodes
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- allid[!(allid %in% edges_id)]
cont_list <- data.frame(cont_edges_id)
colnames(cont_list) <- c("source")
cont_list <- mutate(cont_list, target=source, delta_m.z=0, MEH="N", ftalign_similarity=1)
edges <- rbind(edges[, 1:5], cont_list)
# write.table(edges[, c(1,2,5)], file="filter_net_0.6808", sep="\t", quote=F, row.names=F, col.names=F
###########  only show pp > 0.9
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
layout_n <- create_layout(network, layout = 'mds')
#### molecular network
p <- ggraph(layout_n) +
  geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
  geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(superclass, width=25)), shape=21) +
  #geom_node_text(aes(filter= deg>12,label=name),size=1) +
  scale_color_manual(values=palette) +
  scale_fill_manual(values=palette) +
  scale_edge_width(range=c(0.1, 0.7)) +
  #facet_nodes(~classification) +
  labs(fill="Superclass", size="Tanimoto similarity") +
  guides(fill=guide_legend(override.aes=list(size=5))) +
  theme_grey() +
  theme(
        text=element_text(family="Times"),
        axis.ticks = element_blank(),
        axis.text = element_blank(),
        axis.title = element_blank(),
        panel.background = element_rect(fill="white"),
        #axis.line = element_blank(),
         legend.key.width = unit(1.5, "cm"),
         legend.key.height = unit(1.8, "cm"),
         legend.title = element_text(size=20, face="bold", hjust=0.2),
         legend.text = element_text(size=20),
         legend.background = element_rect(fill="transparent"),
         #legend.position = c(0.6,0.25),
         panel.grid = element_blank(),
         strip.text = element_text(size=20, face="bold")
   )
   # ggsave(p, file="parent_network.tiff", width=20, height=22)
   ggsave(p, file="gnps_network.svg", width=19, height=16)
## facet plot
 library(tidyverse)
 library(igraph)
 library(ggraph)
 library(tidygraph)
 library(ggsci)
 library(scales)
 library(grid)
 library(stringr)
 library(ggimage)
 library(gridExtra)
 library(stringr)
 #### Eucommia analyses
 path="gnps_network_facet_0.5"
 edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)
 # facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
 # facet_group <- facet_group[order(facet_group$classification),] ### lead
 nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
 nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
 nodes2$classification <- nodes2$definition
 nodes2$classification[grep("null", nodes2$classification)] <- "Undifined"
 nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
 nodes <- nodes[which(nodes$superclass!=""), ]
 # edge <- list()
 ##### color palette
 pal1= pal_simpsons()(16)
 pal1= pal1[c(14,2,4,6,7,8,13,1,15,16)]
 pal2= pal_d3("category20")(20)
```

```r
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
plot <- list()
for(i in 1:length(edge_list)){
edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
#########
######### candidate: graphopt kk fr mds
layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
plot[[i]] <-
ggraph(network, layout = layout) +
geom_edge_fan(aes(edge_width=ftalign_similarity), color="black", show.legend=F) +
geom_node_point(aes(size = as.numeric(similarity), fill=classification), shape=21) +
# geom_node_text(aes(label=name),size=3) +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1,0.7)) +
facet_edges(~str_wrap(facet, width=30)) +
guides(size="none", fill="none") +
theme_grey() +
theme(
      text=element_text(family="serif"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      panel.background = element_rect(fill="white"),
      #axis.line = element_blank(),
      panel.grid = element_blank(),
      legend.position = "none",
      strip.text = element_text(size=15, face="bold")
     )
# ggsave(plot[[i]], file="test.svg", width=5, height=5)
cat(i, edge_list[i], "\n") }
# theme_graph(base_family = "Times",
#        foreground = "#8491B4FF",
#        strip_text_size = 20,
#        fg_text_colour = "white",
#        plot_margin = margin(10, 10, 10, 10))
#############################
#############################
#png("test.png", width=1000, height=1000)
#tiff("merge_grey.tiff", width=2000, height=2200, compression="lzw")
svg("gnps_child_nebula.svg", width=18*1.6, height=22*1.5)
n=length(edge_list)
s=n^(1/2); if(round(s)!=s){s=round(s); ss=s+1}else{ss=s}
grid.newpage()
pushViewport(viewport(layout = grid.layout(ss, s)))
r_ss=1
for(i in 1:n){
c_s=i%%s
if(c_s==0){c_s=s}
print( plot[[i]], vp=viewport(layout.pos.row=r_ss, layout.pos.col=c_s ))
cat("push view port of ",i,"\n")
if(c_s==s){ r_ss=r_ss+1}
}
dev.off()
## zoom the specific class
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
```

```r
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
library(gridSVG)
library(rsvg)
library(gridExtra)
path="gnps_network_facet_0.5"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)
# facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
# facet_group <- facet_group[order(facet_group$classification),] ### lead
# edge <- list()
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
nodes$similarity <- as.numeric(nodes$similarity)
nodes$classification <- gsub("null", "Undefined", nodes$classification)
##### color palette
pal1= pal_simpsons()(16)
pal1= pal1[c(14,2,4,6,7,8,13,1,15,16)]
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
select1="Iridoids and derivatives.tsv"
select2="Lignans, neolignans and related compounds.tsv"
select_list=c(34,39)
plot <- list()
stru_path="structure_2d/smiles_draw"
structure_list <- list.files(path = stru_path, pattern = "*.mol.svg.cairo.svg$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)
```

```r
matrix <- data.frame(structure_list) %>% mutate(catch="T")
canopus_set <- read.csv(file="canopus_pp_filter.tsv",header=T,sep="\t")
canopus_set <- t(canopus_set)
colnames(canopus_set)=as.character(canopus_set[1,])
canopus_set <- canopus_set[-1,]
######################################################################
metadata_path="../canopus_neg.tsv"
metadata <- read.csv(file=metadata_path, header=T, sep="\t", quote = "")
metadata <- metadata[,c(2,3,4)]
metadata$class <- paste0("C",metadata$absoluteIndex)
######################################################################
######################### start plot
#########################
for(i in select_list){
edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
#########
######### candidate: graphopt kk fr mds
layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
layout_n <- create_layout(network, layout = layout)
##########################
##########################
### figure elements
### structure
elements <- layout_n[,colnames(layout_n) %in% c("name", "x", "y", "similarity", "classification")]
elements$link_structure <- paste0(elements$name,".mol.svg.cairo.svg")
grid_elements <- merge(elements, matrix, all.x=T, by.x="link_structure", by.y="structure_list", sort=T
### grobify
```

```
structure_p<-list()
list_stru <- grid_elements[which(grid_elements$catch=="T"), colnames(grid_elements) %in% c("link_struc
#grid.picture(readPicture(paste0(stru_path,"/",j)))
for(j in list_stru){
  id<-strsplit(j, split=".mol.svg.cairo.svg")[[1]]
  #structure_p[[as.numeric(id)]] = grobify(readPicture(paste0(path,"/",j)))
  assign(paste0("grob_",id), grobify(readPicture(paste0(stru_path,"/",j))))
  cat(id," >>> ", "structure_p\n")
}
aes_stru <- mutate(grid_elements[which(grid_elements$catch=="T"),], grob=paste0("grob_",name))
##########################
##########################
##########################
### ring_bar plot
ring_data <- canopus_set[,colnames(canopus_set) %in% c(elements$name)]
list_palette = data.frame(cbind(sort(unique(elements$classification)), palette[1:length(unique(element
colnames(list_palette) <- c("classification", "color")
elements_palette <- merge(elements, list_palette, all.x=T, by="classification", sort=T)
for(j in 1:ncol(ring_data)){
id <- colnames(ring_data)[j]
fill_in=elements_palette[which(elements_palette$name==id),]$color
fill_border="black"
df <- data.frame(ring_data[,colnames(ring_data) %in% c(id)])
df <- mutate(df, class=rownames(df))
colnames(df)=c("value","class")
df$num <- seq(nrow(df))
df <- merge(df, metadata, all.x=T, by="class", sort=T)
df <- df[order(df$num),]
df$fill <- paste0(df$class, ": ", df$name)
df$fill <- factor(df$fill, levels=df[order(df$absoluteIndex), colnames(df) %in% c("fill")])
p <- ggplot(df, aes(x=num, y=value)) +
     geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = -5, ymax = 0), fil
     geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = 0, ymax = 1.1), fi
     geom_col(alpha=1, aes(fill=fill), color="white", size=0.02) +
         ylim(-5,1.3) +
     coord_polar() +
     labs(fill="") +
     scale_fill_manual(values=palette) +
     #annotate("text", x=df[nrow(df), colnames(df) %in% "num"], y=-1,
     #         label=paste0("ID: ", id), hjust=0, family="Times", fontface="bold", size=0.7, alpha=0.5
     #scale_fill_gsea() +
     #geom_text(data=df, aes(x=class, y=value+0.1, label=ifelse(value>0.7, class, "")),
   #       color="black", fontface="bold",alpha=0.6, size=0.3, inherit.aes = FALSE ) +
     theme_minimal() +
     theme(
       text=element_text(family="Times"),
       axis.ticks = element_blank(),
           #plot.background = element_rect(fill = "transparent"),
       axis.text = element_blank(),
       axis.title = element_blank(),
       panel.grid = element_blank(),
       panel.grid.major =element_blank(),
       panel.grid.minor = element_blank(),
       legend.position = "none",
       legend.title=element_text(face="bold", hjust= -0.5),
       panel.border = element_blank(),
       plot.margin =unit(c(0,0,0,0),"cm"),
           panel.spacing =unit(c(0,0,0,0),"cm") # Adjust the margin to make in sort labels are not tru
     )
assign(paste0("ring_", id), p) }
##########################
##########################
plot[[i]] <-
ggraph(layout_n) +
geom_edge_fan(aes(edge_width=ftalign_similarity, label=delta_m.z), label_size=0.7, label_alpha=0.3, col
geom_node_point(aes(fill=str_wrap(classification, width=25)), size=1, shape=21) +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1,0.7)) +
facet_edges(~facet) +
labs(fill="Access classes", size="Tanimoto\nsimilarity") +
#guides(size="none", fill="none") +
theme_grey() +
theme(
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      legend.title= element_text(face="bold"),
      #panel.background = element_rect(fill="white"),
      legend.key.height = unit(0.6, "cm"),
      axis.line = element_blank(),
       #panel.grid = element_blank(),
       #legend.position = "none",
       strip.text = element_text(size=15, face="bold"),
       #plot.margin =unit(c(0,0,0,0),"cm"),
       #panel.spacing =unit(c(0,0,0,0),"cm")
      )
assign("find_id", plot[[i]])
#plot[[i]] <- plot[[i]] + geom_subview(x=aes_ring$x-(size/4.7)*1.05, y=aes_ring$y-(size/4.7)*1.05,
       #           subview=get(paste0("ring_", id)), width=size*4.5*1.05, height=size*4.5*1.05 )
#ggsave(plot[[i]], file="test.svg", width=7, height=5)
min=min(elements$similarity)
delta=max(elements$similarity)-min
step=1/delta
if(i==39 | i==34){p_size=0.85; p_dist=0.105}else{p_size=0.65; p_dist=0.09}
for(k in 1:ncol(ring_data)){
id=colnames(ring_data)[k]
aes_ring=elements[which(elements$name==id),]
size=(aes_ring$similarity-min)*step+p_size
plot[[i]] <- plot[[i]] + geom_subview(x=aes_ring$x-p_dist, y=aes_ring$y-p_dist,
                   subview=get(paste0("ring_", id)), width=size, height=size )  }
#############################################
#############################################
for(k in 1:nrow(aes_stru)){
size=aes_stru[k,]$similarity
plot[[i]] <- plot[[i]] + geom_subview(x=aes_stru[k,]$x, y=aes_stru[k,]$y,
                   subview=arrangeGrob( get(aes_stru[k,]$grob) ), width=size*6/5, height=size*6/5) }
# plot[[i]] <- plot[[i]] + facet_zoom(xlim = c(2, 4), ylim=c(5,7))
ggsave(plot[[i]], file=paste0("gnps_zoom_",i,".svg"), width=7, height=5)
cat(i, edge_list[i], "\n")
}
p <- find_id + geom_node_text(aes(label=name),size=3)
##############################################################################################
## add noise (other class)
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
#### Eucommia analyses
pal1= pal_simpsons()(16)
```

```r
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
#lai; palette= unique(c(pal1[c(6:8,9:10,1:4,11:16)], pal2, pal3, pal4))
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
nodes <- nodes[which(nodes$superclass!=""),]
# from mcnebula
edges <- read.csv(file="source_target_tree_0.4.tsv", header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
allid <- unique(c(edges$source, edges$target))
nrow(edges[edges$source!=edges$target,])
# from gnps
edges <- read.csv(file="gnps04.tsv", header=T,sep="\t")
colnames(edges)[1:5] <- c("source", "target",  "delta_m.z", "MEH", "ftalign_similarity")
edges <- edges[which(edges$ftalign_similarity > 0.6808 & edges$source!=edges$target),]
edges_id <- unique(c(edges$source, edges$target))
# edges id which not find in nodes
cont_edges_id <- edges_id[!(edges_id %in% allid)]
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nrow(edges)
# insert the connectionless nodes
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- allid[!(allid %in% edges_id)]
cont_list <- data.frame(cont_edges_id)
colnames(cont_list) <- c("source")
cont_list <- mutate(cont_list, target=source, delta_m.z=0, MEH="N", ftalign_similarity=1)
edges <- rbind(edges[, 1:5], cont_list)
# add lignans and iridoids
path="gnps_network_facet_0.5"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)
allid <- c()
```

```r
for(i in c(34,39,41)){
  ed <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
  ed_id <- unique(c(ed$source, ed$target))
  allid <- c(allid, ed_id)
  assign(paste0("from_", i), ed_id)
}
allid <- unique(allid)
edges_id <- unique(c(edges$source, edges$target))
# edges id which not find in nodes
cont_edges_id <- edges_id[!(edges_id %in% allid)]
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nodes <- mutate(nodes, from_class=
                ifelse(nodes$id %in% from_34, "Iridoids",
                       ifelse(nodes$id %in% from_39, "Lignans",
                              ifelse(nodes$id %in% from_41, "Long-chain fatty acids", NA)
                       )
                       ))

# write.table(edges[, c(1,2,5)], file="filter_net_0.6808", sep="\t", quote=F, row.names=F, col.names=F
###########  only show pp > 0.9
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
layout_n <- create_layout(network, layout = 'fr')
#### molecular network
p <- ggraph(layout_n) +
  geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
  geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(from_class, width=25)), shape=21) +
  geom_node_text(aes(filter=name==1746|name==2081,label=name),size=1, color="white") +
  scale_color_manual(values=palette) +
  scale_fill_manual(values=palette) +
  scale_edge_width(range=c(0.1, 0.7)) +
  #facet_nodes(~classification) +
  labs(fill="From class", size="Tanimoto similarity") +
   theme_grey() +
   theme(
         text=element_text(family="Times"),
         axis.ticks = element_blank(),
         axis.text = element_blank(),
         axis.title = element_blank(),
         panel.background = element_rect(fill="white"),
         #axis.line = element_blank(),
         legend.key.width = unit(1, "cm"),
         legend.key.height = unit(1, "cm"),
         legend.title = element_text(size=15, face="bold", hjust=0.2),
         legend.text = element_text(size=15),
         legend.background = element_rect(fill="transparent"),
         #legend.position = c(0.6,0.25),
         panel.grid = element_blank(),
         strip.text = element_text(size=20, face="bold")
   )
   # ggsave(p, file="parent_network.tiff", width=20, height=22)
   ggsave(p, file="tt_gnps_add_noise_34_39_41.svg", width=12, height=10)
   #ggsave(p, file="test_gnps_add_noise_34_39.svg", width=12, height=10)

## scale change
 library(tidyverse)
 library(igraph)
 library(ggraph)
 library(tidygraph)
 library(ggsci)
 library(scales)
 #### Eucommia analyses
 pal1= pal_simpsons()(16)
 pal2= pal_d3("category20")(20)
 pal3= pal_futurama()(12)
 pal4= pal_uchicago("light")(9)
 palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
 #lai; palette= unique(c(pal1[c(6:8,9:10,1:4,11:16)], pal2, pal3, pal4))
 nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
 nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
 nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
 nodes <- nodes[which(nodes$superclass!=""),]
 # from mcnebula
 edges <- read.csv(file="source_target_tree_0.1.tsv", header=T,sep="\t")
 edges <- edges[which(edges$ftalign_similarity>=0.3),]

 edges_id <- unique(c(edges$source, edges$target))
 cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
 ###########################################
 edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
 edges <- edges[!duplicated(edges[,1:2]), ]
 allid <- unique(c(edges$source, edges$target))
 nrow(edges[edges$source!=edges$target,])
 # from gnps
 edges <- read.csv(file="gnps04.tsv", header=T,sep="\t")
 colnames(edges)[1:5] <- c("source", "target",  "delta_m.z", "MEH", "ftalign_similarity")
 edges <- edges[which(edges$ftalign_similarity > 0.5074 & edges$source!=edges$target),]
 edges_id <- unique(c(edges$source, edges$target))
 # edges id which not find in nodes
 cont_edges_id <- edges_id[!(edges_id %in% allid)]
 edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
 edges <- edges[!duplicated(edges[,1:2]), ]
 nrow(edges)
 # insert the connectionless nodes
 edges_id <- unique(c(edges$source, edges$target))
 cont_edges_id <- allid[!(allid %in% edges_id)]
 cont_list <- data.frame(cont_edges_id)
 colnames(cont_list) <- c("source")
 cont_list <- mutate(cont_list, target=source, delta_m.z=0, MEH="N", ftalign_similarity=1)
 edges <- rbind(edges[, 1:5], cont_list)
 # add lignans and iridoids
 path="gnps_network_facet_0.5"
 edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
             full.names = FALSE, recursive = FALSE,
             ignore.case = FALSE, include.dirs = FALSE)
 allid <- c()
 for(i in c(34,39,41)){
   ed <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
   ed_id <- unique(c(ed$source, ed$target))
   allid <- c(allid, ed_id)
   assign(paste0("from_", i), ed_id)
 }
 allid <- unique(allid)
 edges_id <- unique(c(edges$source, edges$target))
 # edges id which not find in nodes
```

```
cont_edges_id <- edges_id[!(edges_id %in% allid)]
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nodes <- mutate(nodes, from_class=
                ifelse(nodes$id %in% from_34, "Iridoids",
                       ifelse(nodes$id %in% from_39, "Lignans",
                              ifelse(nodes$id %in% from_41, "Long-chain fatty acids", NA
                              )
                       )
                       ))

# write.table(edges[, c(1,2,5)], file="filter_net_0.6808", sep="\t", quote=F, row.names=F, col.names=F
###########  only show pp > 0.9
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
layout_n <- create_layout(network, layout = 'fr')
#### molecular network
p <- ggraph(layout_n) +
  geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
  geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(from_class, width=25)), shape=21) +
  scale_color_manual(values=palette) +
  scale_fill_manual(values=palette) +
  scale_edge_width(range=c(0.1, 0.7)) +
  #facet_nodes(~classification) +
  labs(fill="From class", size="Tanimoto similarity") +
  guides(fill=guide_legend(override.aes=list(size=5))) +
  theme_grey() +
  theme(
        text=element_text(family="Times"),
        axis.ticks = element_blank(),
        axis.text = element_blank(),
        axis.title = element_blank(),
        panel.background = element_rect(fill="white"),
        #axis.line = element_blank(),
        legend.key.width = unit(1, "cm"),
        legend.key.height = unit(1, "cm"),
        legend.title = element_text(size=15, face="bold", hjust=0.2),
        legend.text = element_text(size=15),
        legend.background = element_rect(fill="transparent"),
        #legend.position = c(0.6,0.25),
        panel.grid = element_blank(),
        strip.text = element_text(size=20, face="bold")
  )
  # ggsave(p, file="parent_network.tiff", width=20, height=22)
  ggsave(p, file="tt05_gnps_add_noise_34_39_41.svg", width=12, height=10)
```

34 File: ggraph_network.R

```
library(igraph)
library(dplyr)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(stringr)
#### Eucommia analysis
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
nodes <- nodes[which(nodes$superclass!=""), ]
edges <- read.csv(file="source_target_tree_0.4.tsv", header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
```

```r
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
layout_n <- create_layout(network, layout = "mds")
#### molecular network
p <- ggraph(layout_n) +
geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(superclass, width=25)), shape=21) +
#geom_node_text(aes(filter= deg>12,label=name),size=1) +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1, 0.7)) +
#guides(size="none") +
guides(fill = guide_legend(override.aes = list(size=5))) +
labs(fill="Superclass", size="Tanimoto similarity") +
theme_grey() +
theme(
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      panel.background = element_rect(fill="white"),
      legend.key.width = unit(1, "cm"),
      legend.key.height = unit(1.8, "cm"),
      legend.title = element_text(size=20, face="bold", hjust=0.2),
      legend.text = element_text(size=20),
      legend.background = element_rect(fill="transparent"),
      panel.grid = element_blank(),
      strip.text = element_text(size=20, face="bold")
     )
ggsave(p, file="parent_network.svg", width=20, height=16)



library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(stringr)
#### Eucommia analyses
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
nodes <- nodes[which(nodes$superclass!=""), ]
edges <- read.csv(file="source_target_tree_0.4.tsv", header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
###########################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
## select the specific class
path="network_facet_0.50"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
            full.names = FALSE, recursive = FALSE,
            ignore.case = FALSE, include.dirs = FALSE)
allid <- c()
for(i in c(34,39,41)){
  ed <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
  ed_id <- unique(c(ed$source, ed$target))
  allid <- c(allid, ed_id)
  assign(paste0("from_", i), ed_id)
}
allid <- unique(allid)
edges_id <- unique(c(edges$source, edges$target))
# edges id which not find in nodes
cont_edges_id <- edges_id[!(edges_id %in% allid)]
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nodes <- mutate(nodes, from_class=
                ifelse(nodes$id %in% from_34, "Iridoids",
                        ifelse(nodes$id %in% from_39, "Lignans",
                               ifelse(nodes$id %in% from_41, "Long-chain fatty acids", NA)
                               )
                        ))
###########  only show pp > 0.9
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
layout_n <- create_layout(network, layout = "fr")
#### molecular network
p <- ggraph(layout_n) +
geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(from_class, width=25)), shape=21) +
geom_node_text(aes(filter= name==1746|name==2081, label=name),size=1, color="white") +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1, 0.7)) +
#facet_nodes(~classification) +
#guides(size="none") +
guides(fill=guide_legend(override.aes=list(size=10))) +
labs(fill="From class", size="Tanimoto similarity") +
theme_grey() +
theme(
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      panel.background = element_rect(fill="white"),
      #axis.line = element_blank(),
      legend.key.width = unit(1, "cm"),
      legend.key.height = unit(1, "cm"),
      legend.title = element_text(size=15, face="bold", hjust=0.2),
      legend.text = element_text(size=15),
      legend.background = element_rect(fill="transparent"),
      #legend.position = c(0.6,0.25),
       panel.grid = element_blank(),
       strip.text = element_text(size=20, face="bold")
      )
 # ggsave(p, file="parent_network.tiff", width=20, height=22)
 ggsave(p, file="tt_mcnebula_add_noise_34_39_41.svg", width=12, height=10)
 # ggsave(p, file="test_mcnebula_add_noise_34_39.svg", width=12, height=10)

## scale change
library(tidyverse)
 library(igraph)
 library(ggraph)
 library(tidygraph)
 library(ggsci)
 library(scales)
 library(stringr)
 #### Eucommia analyses
 pal1= pal_simpsons()(16)
 pal2= pal_d3("category20")(20)
 pal3= pal_futurama()(12)
 pal4= pal_uchicago("light")(9)
 palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
 nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
 nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
 nodes2$classification <- nodes2$definition
 nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
 nodes <- nodes[which(nodes$superclass!=""), ]
 edges <- read.csv(file="source_target_tree_0.1.tsv", header=T,sep="\t")
 edges <- edges[which(edges$ftalign_similarity>=0.3),]
 edges_id <- unique(c(edges$source, edges$target))
 cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
 ###########################################
 edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
 edges <- edges[!duplicated(edges[,1:2]), ]
 nrow(edges[edges$source!=edges$target,])


 ## select the specific class
 path="network_facet_0.50"
 edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
             full.names = FALSE, recursive = FALSE,
             ignore.case = FALSE, include.dirs = FALSE)
```

```r
allid <- c()
for(i in c(34,39,41)){
  ed <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
  ed_id <- unique(c(ed$source, ed$target))
  allid <- c(allid, ed_id)
  assign(paste0("from_", i), ed_id)
}
allid <- unique(allid)
edges_id <- unique(c(edges$source, edges$target))
# edges id which not find in nodes
cont_edges_id <- edges_id[!(edges_id %in% allid)]
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
nodes <- mutate(nodes, from_class=
                ifelse(nodes$id %in% from_34, "Iridoids",
                       ifelse(nodes$id %in% from_39, "Lignans",
                              ifelse(nodes$id %in% from_41, "Long-chain fatty acids", NA
                              )
                              )
                       ))
###########  only show pp > 0.9
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
layout_n <- create_layout(network, layout = "fr")
#### molecular network
p <- ggraph(layout_n) +
geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(from_class, width=25)), shape=21) +
geom_node_text(aes(filter= name==1746|name==2081, label=name),size=1, color="white") +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1, 0.7)) +
#facet_nodes(~classification) +
#guides(size="none") +
guides(fill=guide_legend(override.aes=list(size=5))) +
labs(fill="From class", size="Tanimoto similarity") +
theme_grey() +
theme(
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      panel.background = element_rect(fill="white"),
      #axis.line = element_blank(),
      legend.key.width = unit(1, "cm"),
      legend.key.height = unit(1, "cm"),
      legend.title = element_text(size=15, face="bold", hjust=0.2),
      legend.text = element_text(size=15),
      legend.background = element_rect(fill="transparent"),
      #legend.position = c(0.6,0.25),
      panel.grid = element_blank(),
      strip.text = element_text(size=20, face="bold")
     )
# ggsave(p, file="parent_network.tiff", width=20, height=22)
ggsave(p, file="tt03_mcnebula_add_noise_34_39_41.svg", width=12, height=10)
```

35 File: ggraph_zoom.R

```
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
library(gridSVG)
library(ggpubr)
library(gridExtra)
path="network_facet_0.3"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
                        full.names = FALSE, recursive = FALSE,
                        ignore.case = FALSE, include.dirs = FALSE)
# facet_group <- read.csv(file="../for_violin.tsv", header=T,sep="\t")
# facet_group <- facet_group[order(facet_group$classification),] ### lead
nodes1 <- read.csv(file="../com_compound.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
 nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
 nodes <- nodes[which(nodes$superclass!=""), ]

################################
################################
stru_path="structure_2d/smiles_draw"
structure_list <- list.files(path = stru_path, pattern = "*.mol.svg.cairo.svg$", all.files = FALSE,
                             full.names = FALSE, recursive = FALSE,
                             ignore.case = FALSE, include.dirs = FALSE)
matrix <- data.frame(structure_list) %>%
  mutate(catch="T")
################################
################################
# edge <- list()
##### color palette
pal0= pal_npg()(10)
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
pal10= pal_igv("default")(51)
palette= unique(c(pal0, pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
select1="Iridoids and derivatives.tsv"
select2="Lignans, neolignans and related compounds.tsv"
select_list=c(34,39)
plot <- list()
i=34
############################################
edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
group_select2 <- c(3938, 2529, 1445)
group_select1 <- c(279, 458, 574, 1107, 1445, 2227, 2529, 2664, 2824)
```

```
if(i==34){
  select=group_select1
}else{
  select=group_select2
} ############### filter
colnames(edges)[colnames(edges) %in% "facet"] <- "facet_c"
# select <- c(3938, 2529, 1445) ##### lignans
edges <- edges[(edges$source %in% select | edges$target %in% select), ]
#edges2 <- edges[edges$target %in% select, ]
#edges <- rbind(edges1, edges2)
edges <- edges[!duplicated(edges[,1:2]),]
edges_set <- edges
########################################
tlimit <- ifelse(i==34, 0.2, 0.1)
#for(tlimit in c(0.2, 0.25, 0.3, 0.4)){
edges <- edges_set[which(edges_set$ftalign_similarity > tlimit),] %>%
  mutate(group=ifelse(source %in% select, source, target), group_sub=ifelse(source %in% select, target,
edges <- merge(edges, nodes[,colnames(nodes) %in% c("id", "norm_delta")], by.x="group", by.y="id", all.
colnames(edges)[which(colnames(edges)=="norm_delta")] <- "norm_delta_group"
edges <- merge(edges, nodes[,colnames(nodes) %in% c("id", "norm_delta", "similarity")], by.x="group_sub
colnames(edges)[colnames(edges) %in% c("norm_delta", "similarity")] <- c("norm_delta_group_sub", "sub_s
edges <- edges[which(edges$norm_delta_group * edges$norm_delta_group_sub < 0),]
edges <- edges[, c(3:4, 1:2, 5:ncol(edges))]
s_limit <- ifelse(i==34, 0.5, 0.5)
edges <- edges[which(edges$sub_similarity > s_limit), ]
edges <- edges[order(edges$group, edges$norm_delta_group_sub), ]
edges <- edges[which(edges$ftalign_similarity > tlimit), ]
write.table(edges, file=paste0("structure_2d/candidate/", i, "_class.tsv"), sep="\t", col.names=T, row.
########################################
network_nodes <- as_tbl_graph(edges) %>%
  activate(nodes) %>% #as_tibble()
  mutate(deg = centrality_degree(mode='in')) %>%
    merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
    as_tibble()
  network_edges <- as_tbl_graph(edges) %>%
    activate(edges) %>%
    as_tibble()
  ########################################
  ###################
  network_edges_filter <- network_edges[which(network_edges$ftalign_similarity > tlimit),]
  list <- unique(c(network_edges_filter$from, network_edges_filter$to))
```

```r
network_nodes$rownames=rownames(network_nodes)
network <- tbl_graph(nodes = network_nodes, edges = network_edges_filter) %>%
  activate(nodes) %>% #as_tibble()
  mutate(deg = centrality_degree(mode='in'))
  #########################################
  layout_n <- create_layout(network, layout = "circle")
  select <- unique(edges$group)
  outer_circle <- layout_n %>%
    filter(!(name %in% select)) %>%
    mutate(
           x = cos((row_number() - 1) / ( nrow(network_nodes) - length(select) ) * 2 * pi),
           y = sin((row_number() - 1) / ( nrow(network_nodes) - length(select) ) * 2 * pi)
    )
    #############
    angles <- seq(360, 0, -360/length(select) )
    angles <- angles[-1]
    radius <- rep(0.5, length(select))
    centers <- tibble(
                      x = radius * cos(angles / 180 * pi),
                      y = radius * sin(angles / 180 * pi)
    )
    inner_circle <- bind_cols(centers, select(filter(layout_n, name %in% select), -x, -y))
    #############
    layout_n[] <- bind_rows(outer_circle, inner_circle) %>%
      arrange(.ggraph.index)
    #########################################
    elements <- layout_n[,colnames(layout_n) %in% c("name", "x", "y", "similarity")]
    elements$link_structure <- paste0(elements$name,".mol.svg.cairo.svg")
    grid_elements <- merge(elements, matrix, all.x=T, by.x="link_structure", by.y="structure_list", s
    ### grobify
    structure_p<-list()
    list_stru <- grid_elements[which(grid_elements$catch=="T"), colnames(grid_elements) %in% c("link_
    # grid.picture(readPicture(paste0(stru_path,"/",j)))
    for(j in list_stru){
      id<-strsplit(j, split=".mol.svg.cairo.svg")[[1]]
      #structure_p[[as.numeric(id)]] = grobify(readPicture(paste0(path,"/",j)))
      assign(paste0("grob_",id), grobify(readPicture(paste0(stru_path,"/",j))))
      cat(id," >>> ", "structure_p\n")
    }
    aes_stru <- mutate(grid_elements[which(grid_elements$catch=="T"),], grob=paste0("grob_",name))
    #########################################
```

```
      ratio <- (nrow(network_nodes) - length(select))/length(select)
      plot[[i]] <-
        ggraph(layout_n) +
        #geom_edge_fan(aes(edge_width=ftalign_similarity), label_alpha=0.3, color="#4DBBD5FF", show.leg
        geom_edge_diagonal(aes(edge_width=ftalign_similarity, edge_color = as.factor(group) ),
                          label_alpha=0.3, show.legend=T, alpha=0.6) +
#color="lightblue",
#geom_node_point(aes(fill=str_wrap(classification, width=25), size=similarity), shape=21) +
geom_node_point(aes( fill=norm_delta*log2(10) ), size=ifelse( layout_n$name %in% select, 40, 40*4.5/rat
               stroke=0,
               alpha=0.6, shape=21) +
geom_node_text(aes(label=paste0("ID: ", name, "\n", "simi: ", round(similarity,2) )),
               size=ifelse(layout_n$name %in% select, 5, 5*4.5/ratio), color="white", alpha=0.4) +
#scale_color_manual(values=palette) +
# blue green grey yellow red
scale_fill_gradientn(colors=c("#197EC0FF", "#B8B8B8FF", "#EFC000FF")) +
scale_color_npg() +
scale_edge_colour_manual(values=pal1) +
scale_edge_width(range=c(0.1,4)) +
facet_edges(~facet_c) +
guides(alpha="none", guide_legend(override.aes = list(alpha = 0.6))) +
labs(fill="Log2(delta area)", edge_width="Normalized\nftalign similarity", edge_colour="From ID") +
theme_grey() +
theme(
      text=element_text(family="Times", size=ifelse(i==34, 10, 15) ),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      legend.title= element_text(face="bold"),
      #panel.background = element_rect(fill="white"),
      legend.key.height = unit(0.5, "cm"),
      legend.key.width = unit(0.2, "cm"),
      axis.line = element_blank(),
      #panel.grid = element_blank(),
      #legend.position = "none",
      strip.text = element_text(size=ifelse(i==34, 15, 24), face="bold"),
      plot.margin =unit(c(0,0,0,0),"cm"),
      panel.spacing =unit(c(0,0,0,0),"cm")
)
#assign(paste0("plot_", tlimit), plot[[i]])
#ggsave(plot_0.25, file=paste0("tlimit_", 0.25, ".svg"), width=7, height=5)
```

```r
#if(tlimit==0.7 & i==28){plot[[i]] <- plot[[i]] + scale_fill_gradient(labels = c("low", "", "", "high")
#if(tlimit==0.7 & i==26){plot[[i]] <- plot[[i]] + scale_fill_gradient(labels = c("low", "", "", "","hig
n = ifelse(i==34, 3, 1.7)
for(k in 1:nrow(aes_stru)){
  size=aes_stru[k,]$similarity
  id <- aes_stru[k,]$name
  plot[[i]] <- plot[[i]] + geom_subview( x=aes_stru[k,]$x, y=aes_stru[k,]$y,
                                       subview=arrangeGrob( get(aes_stru[k,]$grob) ),
                                       width=ifelse(id %in% select, size*n/5, size*n/ratio),
                                       height=ifelse(id %in% select, size*n/5, size*n/ratio) ) }
assign(paste0("plot_", tlimit), plot[[i]])
#cat(i, edge_list[i], "\n")  }
ggsave(plot_0.2, file=paste0("iridoids_", 0.2, ".svg"), width=14, height=12)
#ggsave(plot_0.25, file=paste0("tlimit_", 0.25, ".svg"), width=6.5, height=5)
ggsave(plot_0.1, file=paste0("lignans_", 0.1, ".svg"), width=9, height=7.5)
################################
################################
######## import data
path <- "structure_2d/candidate"
structure_list <- list.files(path = path, pattern = "*.mol.svg.cairo.svg$", all.files = FALSE,
                             full.names = FALSE, recursive = FALSE,
                             ignore.case = FALSE, include.dirs = FALSE)
matrix <- data.frame(structure_list) %>% mutate(catch="T")
matrix <- mutate(matrix, idd=strsplit(structure_list, split=".mol.svg.cairo.svg")) %>%
  separate(c("idd"), c("id", "candidate"), sep="_can_", remove=F)
for(id in unique(matrix$id)){
  df <- matrix[which(matrix$id==id), ]
  df <- df[order(as.numeric(df$candidate)),]
  for(i in 1:nrow(df)){
    # for(i in 20:29){
    assign(paste0("stru_", df[i, colnames(df) %in% c("idd")]), readPicture(paste0(path,"/",df[i, colname
    cat(i, "\n")
  }
}
################################
################################
dir.create(paste0(path,"/merge"))
for(id in unique(matrix$id)){
  df <- matrix[which(matrix$id==id), ]
  df <- df[order(as.numeric(df$candidate)),]
  svg(paste0(path, "/merge/", id, "_merge.svg"), height=1.2, width=12)
  n=0
  for(i in 1:nrow(df)){
    n=n+1
    grid.picture( get(paste0("stru_", df[i, colnames(df) %in% c("idd")])) , x=n/11, y=0.5, height=1, wi
    cat("Info: the grid picture number is ", i, " of ID: ", id, "\n")
  }
  dev.off()
}
```

36 File: inchi_curl_syno.R

```
mutate_inchi_curl_syno <-
  function(
           key_set,
           .id_set,
           dir = "inchi_pub",
           ...
           ){
    if(file.exists(dir) == F)
      dir.create(dir)
    origin <- getwd()
    setwd(dir)
    list <- mapply(data.table, key_set, .id_set, SIMPLIFY = F)
    pbapply::pblapply(list, function(df, ...){
                        inchi_curl_syno(df[[1]], df[[2]], ...)}, ..., cl = 20)
    setwd(origin)
  }
inchi_curl_syno <-
  function(
           key,
           .id,
           type = "inchikey",
           get = "synonyms",
           save = paste0(.id, ".xml")
           ){
    http = paste0("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/", type, "/")
    http_end = paste0("/", paste(get, collapse = ","), "/XML > ")
    curl <- "curl -s --connect-timeout 20 --retry 100 --retry-delay 30 "
    curl_http <- paste0(curl, http)
    system(paste0(curl_http, key, http_end, save))
  }
```

37 File: inchi_curl.R

```r
inchikey_get_formula <-
  function(
           inchikey
           ){
    df <- data.table::data.table(sp.id = as.character(1:length(inchikey)), inchikey = inchikey)
    ## ------------------------------------
    args <- list(df$inchikey, df$sp.id, get = c("MolecularFormula", "ExactMass", "MonoisotopicMass"))
    do.call(mutate_inchi_curl, args)
    from_csv <- gather_inchi_curl()
    system("rm -r inchi_pub")
    ## ------------------------------------
    df <- merge(df, from_csv, by = "sp.id", all.x = T)
    return(df)
  }
gather_inchi_curl <-
  function(
           path = "inchi_pub"
           ){
    file_set <- list.files(path = path, pattern = "csv$", full.names = T)
    list <- lapply(file_set, mutate_fread)
    names(list) <- file_set %>%
      stringr::str_extract("(?<=/)[0-9]{1,100}")
    df <- data.table::rbindlist(list, idcol = T, fill = T) %>%
      dplyr::rename(sp.id = .id)
    return(df)
  }
mutate_fread <-
  function(
           path
           ){
    check <- try(df <- fread(path, fill = T), silent = T)
    if(class(check)[1] == "try-error")
      df <- fread(path, fill = T, skip = 3)
    if("V1" %in% colnames(df)){
      print(path)
      return()
    }
    return(df)
  }
mutate_inchi_curl <-
```

```
  function(
           key_set,
           .id_set,
           dir = "inchi_pub",
           ...
           ){
    if(file.exists(dir) == F)
      dir.create(dir)
    origin <- getwd()
    setwd(dir)
    list <- mapply(data.table, key_set, .id_set, SIMPLIFY = F)
    pbapply::pblapply(list, function(df, ...){
                        inchi_curl(df[[1]], df[[2]], ...)}, ..., cl = 8)
    setwd(origin)
  }
inchi_curl <-
  function(
           key,
           .id,
           type = "inchikey",
           get = "InChIkey",
           save = paste0(.id, ".csv")
           ){
    http = paste0("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/", type, "/")
    http_end = paste0("/property/", paste(get, collapse = ","), "/CSV > ")
    curl <- "curl -s --connect-timeout 20 --retry 100 --retry-delay 30 "
    curl_http <- paste0(curl, http)
    system(paste0(curl_http, key, http_end, save))
  }
int_inchi_curl <-
  function(
           seq,
           type = "inchikey",
           get = "InChIkey",
           ...
           ){
    init <- dload[seq, "init"]
    .id <- dload[seq, ".id"]
    save <- paste0(.id, ".csv")
    while(class(try(read.csv(save), silent = T))[1] == "try-error"){
      inchi_curl(init, .id,
                 get = get,
                 type = type,
                 ...)
    }
  }
```

38 File: inchikey2d_search.R

```
inchikey2d_search <-
  function(
           inchikey2d,
           db,
           col = "inchikey2D"
           ){
    check <- try(df <- db[which(db[[col]] == inchikey2d), ], silent = T)
    if(class(check)[1] == "try-error")
      return()
    return(df)
  }
```

39 File: instance_3d_xcms.R

```
library(xcms)
library(MSnbase)
setwd("/media/wizard/back/thermo_mzML_0518")
savepath="/home/wizard/operation/back/"
dataname="EU-Pro2.mzML"
com_data <- readMSData(file = dataname, mode = "onDisk")
  ####### ms1 in peaks
  data <- filterRt(com_data, c(600, 780))
  mzrange <- c(380, 390)  # build the eic model
  ex_data <- chromatogram(data, msLevel = 1L, mz = mzrange, aggregationFun = "max")
  ex_data_1 <- ex_data[1,1]
  rt=data.frame(rtime(ex_data_1))
  int=data.frame(intensity(ex_data_1))
  rt_int=cbind(rt,int)
  colnames(rt_int)=c("rt", "int")
  rt_int$rt=rt_int$rt/60
  ########
  n=3
```

```
time=vector(mode="list", length=n)
time[[1]]=c(10,11)
time[[2]]=c(11,12)
time[[3]]=c(12,13)
peak=vector(mode="list", length=n)
scan=vector(mode="list", length=n)
for(i in 1:n){
  peak[[i]]=rt_int[which(rt_int$rt > time[[i]][1] & rt_int$rt < time[[i]][2]),]
  scan[[i]]=rt_int[which(rt_int$int==max(peak[[i]]$int)),] }
# rownames(scan1)
####### ms1 in scans
mz=mz(data)
inn=intensity(data)
mz_get=vector(mode="list", length=n)
int_get=vector(mode="list", length=n)
scan_set=vector(mode="list", length=n)
a=pi/3 ##### 45
filter=50000
xlim=c(10.6,12.7)
mzfilter=c(360, 420)
ratio = ((xlim[2])/(mzfilter[2]))  #((xlim[2]-xlim[1])/(mzfilter[2]-mzfilter[1]))
ratio_x_y = max(rt_int$int)/(xlim[2]-xlim[1])
lift=(max(rt_int$int))*(1/20)
height_lift=lift*22
base=(mzfilter[2]-mzfilter[1])*ratio *ratio_x_y*sin(a)
#zoom=0.95
for(i in 1:n){
mz_get[[i]]=mz[which(names(mz)==rownames(scan[[i]]))]
int_get[[i]]=inn[which(names(inn)==rownames(scan[[i]]))]
scan_set[[i]]=data.frame(mz_get[[i]], int_get[[i]]);   colnames(scan_set[[i]])=c("mz", "int")
scan_set[[i]]=scan_set[[i]][which(scan_set[[i]]$int >= filter &
                     scan_set[[i]]$mz >=mzfilter[1] &
                     scan_set[[i]]$mz <= mzfilter[2] ),]
scan_set[[i]]$x <- xlim[1] + ((scan_set[[i]]$mz-mzfilter[1])*ratio*cos(a) +(scan[[i]]$rt-xlim[1]))
scan_set[[i]]$xend <- scan_set[[i]]$x
scan_set[[i]]$s_x <- xlim[1] + (scan_set[[i]]$mz-mzfilter[1])*ratio*cos(a)
scan_set[[i]]$s_xend <- scan_set[[i]]$s_x
scan_set[[i]]$y <- ((scan_set[[i]]$mz-mzfilter[1])*ratio *ratio_x_y*sin(a) + scan_set[[i]]$int) + li
scan_set[[i]]$yend <- ((scan_set[[i]]$mz-mzfilter[1])*ratio *ratio_x_y*sin(a)) + lift
scan_set[[i]]$group=i
scan_set[[i]][nrow(scan_set[[i]])+1,] <- c(0, 0,
                           scan[[i]]$rt,  ## x
                           scan[[i]]$rt+(mzfilter[2]-mzfilter[1])*ratio*cos(a), ## xend
                           NA,
                           NA,
                           lift, ## y
                           base+lift, ## yend
                           0)  ## group
  scan_set[[i]] <- scan_set[[i]][order(scan_set[[i]]$mz),] }
 segment=scan_set[[1]]
 for(i in 2:n){segment <- rbind(segment, scan_set[[i]])}
 segment_bottom <- segment[which(segment$mz==0), ]
 segment <- segment[which(segment$mz!=0), ]
 segment_outlier=data.frame(rbind(
           c(xlim[1], xlim[1]+(mzfilter[2]-mzfilter[1])*ratio*cos(a), 0, base ), # 1
           c(xlim[2], xlim[2]+(mzfilter[2]-mzfilter[1])*ratio*cos(a), 0, base ), # 2
           c(xlim[1], xlim[2], 0, 0), # 3
           #c(xlim[1], xlim[2], lift*21, lift*21), #top
           c(xlim[1], xlim[1], 0, height_lift), # 4
           c(xlim[2], xlim[2], 0, height_lift), # 5
           c(xlim[1]+(mzfilter[2]-mzfilter[1])*ratio*cos(a), xlim[1]+(mzfilter[2]-mzfilter[1])*ratio*co
                   base, base+height_lift), # 6
           # c(xlim[2]+(xlim[2]-xlim[1])*cos(a), xlim[2]+(xlim[2]-xlim[1])*cos(a), lift*21, lift*21*2)
           c(xlim[1], xlim[1]+(mzfilter[2]-mzfilter[1])*ratio*cos(a), height_lift, base+height_lift),
           c(xlim[1], xlim[2], height_lift, height_lift)  # 8
           ))
 segment_outlier$group="0"
 colnames(segment_outlier) <- c("x", "xend", "y", "yend", "group")
# segment_top=data.frame(rbind(c(xlim[1], xlim[2], base, base)))
# segment_top$group="0"
# colnames(segment_top) <- c("x", "xend", "y", "yend", "group")
 delta=(mzfilter[2]-mzfilter[1])*ratio*cos(a)
 point <- data.frame(rbind(c(xlim[1],0),
             c(xlim[1]+(mzfilter[2]-mzfilter[1])*ratio*cos(a), base),
             c(xlim[1]+(mzfilter[2]-mzfilter[1])*ratio*cos(a), base + height_lift),
             c(xlim[1], height_lift),
             c(xlim[2],0),
             c(xlim[2], height_lift),
             c(xlim[2]+(mzfilter[2]-mzfilter[1])*ratio*cos(a), base)
             ))
 point$group="0"
 colnames(point)=c("x","y", "group")
```

```
  grey <- point[1:4,]
  white <- point[c(1,5,6,4),]
  grey_bottom <- point[c(1,5,7,2),]
  grey_bottom$y1 <- grey_bottom$y -lift*7
  grey_bottom$y2 <- grey_bottom$y -lift*6
  grey_bottom$y3 <- grey_bottom$y -lift*5
  grey_bottom$y4 <- grey_bottom$y -lift*4
  grey_bottom$y5 <- grey_bottom$y -lift*3
 library(ggalt)
 library(tidyverse)
 library(ggsci)
   data=raw_data=rt_int
   data=data[which(data$rt>=xlim[1] & data$rt<=xlim[2]),]
   data=data.frame(spline(data$rt, data$int, n=10000))
  ### beauty
  peak_to=0.15
  scan_tot=scan[[1]]
  for(i in 2:n){scan_tot <- rbind(scan_tot, scan[[i]])}
  for(i in 1:n){
    scan_tot <- rbind(scan_tot,
               c(scan_tot[i,1]-peak_to*4/5, scan_tot[i,2]*1/5),
               c(scan_tot[i,1]-peak_to, scan_tot[i,2]*1/10),
               c(scan_tot[i,1]+peak_to*4/5, scan_tot[i,2]*1/5),
               c(scan_tot[i,1]+peak_to, scan_tot[i,2]*1/10) )
    if(i!=n){scan_tot <- rbind(scan_tot,
               c(scan_tot[i,1]+(scan_tot[i+1,1]-scan_tot[i,1])*1/3, 0),
               c(scan_tot[i,1]+(scan_tot[i+1,1]-scan_tot[i,1])*1/2, 0),
               c(scan_tot[i,1]+(scan_tot[i+1,1]-scan_tot[i,1])*2/3, 0)
                )
      }else{scan_tot <- rbind(scan_tot,
               c(scan_tot[i,1]+peak_to*6/4, 0))}
      }
  data=rbind(scan_tot, c(xlim[1],0), c(xlim[2],0))
  data=data.frame(spline(data$rt, data$int, n=10000))
  peak_to=peak_to *1.2
   colnames(data)=c("rt", "int")
   group_set=vector(mode="list", length=n)
   for(i in 1:n){
   group_set[[i]] <- data[which(data$rt > as.numeric(scan[[i]][1]-peak_to) & data$rt < as.numeric(scan[
            mutate(group=i) }
   set=group_set[[1]]
```

```
for(i in 2:n){set <- rbind(set, group_set[[i]])}
data=merge(data, set[,c(1,3)], by="rt", all.x=TRUE, sort=TRUE)
data[which(is.na(data$group)==T),]$group=0
color_set <- c("0"="black", "3"="#E64B35FF", "2"="#4DBBD5FF", "1"="#00A087FF")
###
p <- ggplot(data, aes(x=rt, y=int, fill=as.character(group), color=as.character(group))) +
  geom_polygon(data=grey, aes(x=x, y=y), alpha=0.25, fill="grey") +
  geom_segment(data=segment, aes(x=s_x, y=y-lift, xend=s_xend, yend=yend-lift), color="#BDBDBDFF", s
 # geom_polygon(data=grey_bottom, aes(x=x, y=y1), alpha=0.15, fill="skyblue", color="#BDBDBDFF") +
  geom_polygon(data=grey_bottom, aes(x=x, y=y2), alpha=0.15, fill="skyblue", color="#BDBDBDFF") +
  geom_polygon(data=grey_bottom, aes(x=x, y=y3), alpha=0.15, fill="skyblue", color="#BDBDBDFF") +
  geom_polygon(data=grey_bottom, aes(x=x, y=y4), alpha=0.15, fill="skyblue", color="#BDBDBDFF") +
  geom_polygon(data=grey_bottom, aes(x=x, y=y5), alpha=0.15, fill="skyblue", color="#BDBDBDFF") +
  geom_polygon(data=grey_bottom, aes(x=x, y=y), alpha=0.15, fill="skyblue", color="#BDBDBDFF") +
  #geom_area(data=point[!duplicated(point$x),], aes(x=x, y=y), fill=grey, alpha=0.5) +
  geom_segment(data=segment_bottom, aes(x=x, y=y, xend=xend, yend=yend, color=as.character(group)), s
  annotate("rect", xmin=xlim[1]+(mzfilter[2]-mzfilter[1])*ratio*cos(a), xmax=xlim[2]+(mzfilter[2]-mz
             ymin=base, ymax=base+height_lift,
         alpha=1, color="#3C5488FF", fill="white", size=2) +
  geom_point(data=point, aes(x=x, y=y), size=1.35) +
  geom_segment(data=segment_outlier, aes(x=x, y=y, xend=xend, yend=yend, color=as.character(group)),
  geom_segment(data=segment, aes(x=x, y=y, xend=xend, yend=yend, color=as.character(group)), size=4,
  geom_line() +
  geom_polygon(data=white, aes(x=x, y=y), alpha=0.4, fill="white") +
  geom_area(alpha=1) +
  annotate("text", x=scan[[1]]$rt+delta, y=height_lift*8/10+base, label="LC-MS data set", family="Ti
  annotate("text", x=scan[[1]]$rt, y=height_lift*8/10, label="The Sample", family="Times", size=8, h
  annotate("text", x=scan[[median(1:n)]]$rt, y=height_lift*1/10*(-1), label="Retention time", family
  annotate("text", x=xlim[1]-peak_to/2, y=lift*7*(-1), label="Samples", family="Times", size=8, hjus
  annotate("text", x=xlim[2]+delta/2+peak_to/2, y=base/2, label="m/z", family="Times", size=8, hjust
#  geom_segment(data=segment_top, aes(x=x, y=y, xend=xend, yend=yend, color=as.character(group)), si
  #geom_xspline(spline_shape = -0.5, size=5) +
  #geom_smooth(method = "loess", formula=y~poly(x), se=F) +
  #stat_smooth(method = "loess", geom="area", se=F, formula=y~poly(x)) +
  #xlim(xlim[1],xlim[2]*22/20) +
  scale_fill_manual(values=color_set) +
  scale_color_manual(values=color_set) +
  theme_classic() +
  labs(x="", y="") +
  theme(
        text=element_text(family="serif"),
          axis.ticks = element_blank(),
          axis.text = element_blank(),
          axis.title.y = element_blank(),
          axis.line = element_blank(),
          panel.grid = element_blank(),
          legend.position = "none"
         )
 ggsave(p, file=paste0(savepath, "test.svg"), width=12, height=7)
 ######################################################################################################
 ######################################################################################################
 ######################################################################################################
 ## card
library(ggalt)
library(tidyverse)
library(ggsci)
library(grid)
setwd("/media/wizard/back/thermo_mzML_0518")
savepath="/home/wizard/operation/back/"
dataname="EU-Pro2.mzML"
com_data <- readMSData(file = dataname, mode = "onDisk")
  ####### ms1 in peaks
  data <- filterRt(com_data, c(600, 780))
  mzrange <- c(380, 390)  # build the eic model
  ex_data <- chromatogram(data, msLevel = 1L, mz = mzrange, aggregationFun = "max")
  ex_data_1 <- ex_data[1,1]
  rt=data.frame(rtime(ex_data_1))
  int=data.frame(intensity(ex_data_1))
  rt_int=cbind(rt,int)
  colnames(rt_int)=c("rt", "int")
  rt_int$rt=rt_int$rt/60
  ########
  n=3
  time=vector(mode="list", length=n)
  time[[1]]=c(10,11)
  time[[2]]=c(11,12)
  time[[3]]=c(12,13)
  peak=vector(mode="list", length=n)
  scan=vector(mode="list", length=n)
  for(i in 1:n){
    peak[[i]]=rt_int[which(rt_int$rt > time[[i]][1] & rt_int$rt < time[[i]][2]),]
    scan[[i]]=rt_int[which(rt_int$int==max(peak[[i]]$int)),] }
  # rownames(scan1)
  ####### ms1 in scans
  xlim=c(10.6,12.7)
  data=raw_data=rt_int
  data=data[which(data$rt>=xlim[1] & data$rt<=xlim[2]),]
  data=data.frame(spline(data$rt, data$int, n=10000))
 ### beauty
 peak_to=0.15
 scan_tot=scan[[1]]
 for(i in 2:n){scan_tot <- rbind(scan_tot, scan[[i]])}
 for(i in 1:n){
   scan_tot <- rbind(scan_tot,
              c(scan_tot[i,1]-peak_to*4/5, scan_tot[i,2]*1/5),
              c(scan_tot[i,1]-peak_to, scan_tot[i,2]*1/10),
              c(scan_tot[i,1]+peak_to*4/5, scan_tot[i,2]*1/5),
              c(scan_tot[i,1]+peak_to, scan_tot[i,2]*1/10) )
   if(i!=n){scan_tot <- rbind(scan_tot,
              c(scan_tot[i,1]+(scan_tot[i+1,1]-scan_tot[i,1])*1/3, 0),
              c(scan_tot[i,1]+(scan_tot[i+1,1]-scan_tot[i,1])*1/2, 0),
              c(scan_tot[i,1]+(scan_tot[i+1,1]-scan_tot[i,1])*2/3, 0)
               )
       }else{scan_tot <- rbind(scan_tot,
              c(scan_tot[i,1]+peak_to*6/4, 0))}
       }
 data=rbind(scan_tot, c(xlim[1],0), c(xlim[2],0))
 data=data.frame(spline(data$rt, data$int, n=10000))
 peak_to=peak_to *1.2
  colnames(data)=c("rt", "int")
  group_set=vector(mode="list", length=n)
  for(i in 1:n){
  group_set[[i]] <- data[which(data$rt > as.numeric(scan[[i]][1]-peak_to) & data$rt < as.numeric(scan[
           mutate(group=i) }
  set=group_set[[1]]
  for(i in 2:n){set <- rbind(set, group_set[[i]])}
  data=merge(data, set[,c(1,3)], by="rt", all.x=TRUE, sort=TRUE)
  data[which(is.na(data$group)==T),]$group=0
  color_set <- c("0"="black", "3"="#E64B35FF", "2"="#4DBBD5FF", "1"="#00A087FF")
 ####################################################
  segment <- data.frame(c(0, 10, 0, 0),
                c(0, 10, 10, 10),
                c(0, 0, 0, 10),
                 c(10, 10, 0, 10)
                )
  colnames(segment) <- c("x", "xend", "y", "yend")
 ####################################################
setwd("/home/wizard/operation/back/0703_all/results")
 p1_1 <- ggplot(data[which(data$rt > 10.68 & data$rt < 11.12), ],
           aes(x=rt, y=int, fill=as.character(group), color=as.character(group))) +
   geom_area() +
   scale_fill_manual(values=color_set) +
   scale_color_manual(values=color_set) +
   #geom_segment(data=segment, aes(x=x, y=y, xend=xend, yend=yend, color=as.character(group)), size=2,
   labs(x="") +
   xlim(10.6,12.6) +
   theme_minimal() +
   theme(
      text=element_text(family="serif"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title.y = element_blank(),
      axis.line = element_blank(),
      panel.grid = element_blank(),
      legend.position = "none"
     )
 p1_2 <- ggplot(data[which(data$rt > 11.32 & data$rt < 11.8), ],
           aes(x=rt, y=int, fill=as.character(group), color=as.character(group))) +
   geom_area() +
   scale_fill_manual(values=color_set) +
   scale_color_manual(values=color_set) +
   #geom_segment(data=segment, aes(x=x, y=y, xend=xend, yend=yend, color=as.character(group)), size=2,
   labs(x="") +
   xlim(10.6,12.6) +
   theme_minimal() +
   theme(
      text=element_text(family="serif"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title.y = element_blank(),
      axis.line = element_blank(),
      panel.grid = element_blank(),
      legend.position = "none"
     )
```

```
p1_3 <- ggplot(data[which(data$rt > 12.05 & data$rt < 12.55), ],
            aes(x=rt, y=int, fill=as.character(group), color=as.character(group))) +
  geom_area() +
  scale_fill_manual(values=color_set) +
  scale_color_manual(values=color_set) +
  #geom_segment(data=segment, aes(x=x, y=y, xend=xend, yend=yend, color=as.character(group)), size=2,
  labs(x="") +
  xlim(10.6,12.6) +
  theme_minimal() +
  theme(
     text=element_text(family="serif"),
     axis.ticks = element_blank(),
     axis.text = element_blank(),
     axis.title.y = element_blank(),
     axis.line = element_blank(),
     panel.grid = element_blank(),
     legend.position = "none"
    )
p2 <- ggplot(segment) +
  #geom_segment(aes(x=x, y=y, xend=xend, yend=yend), color="#BDBDBDFF", size=4, alpha=0.8) +
  annotate("rect", xmin=0, xmax=10, ymin=0 , ymax=10, color="black", fill="white", size=3, alpha=0.5)
  theme_minimal() +
  theme(
     text=element_text(family="serif"),
     axis.ticks = element_blank(),
     axis.text = element_blank(),
     axis.title.y = element_blank(),
     axis.line = element_blank(),
     panel.grid = element_blank(),
     legend.position = "none"
    )
p3 <- ggplot(segment) +
  #geom_segment(aes(x=x, y=y, xend=xend, yend=yend), color="#BDBDBDFF", size=4, alpha=0.8) +
  annotate("rect", xmin=0, xmax=10, ymin=0 , ymax=10, color="black", fill="white", size=3, alpha=0) +
  theme_minimal() +
  theme(
     text=element_text(family="serif"),
     axis.ticks = element_blank(),
     axis.text = element_blank(),
     axis.title.y = element_blank(),
     axis.line = element_blank(),
      panel.grid = element_blank(),
      legend.position = "none"
     )
svg("2d_ms.svg", width=12, height=8)
grid.newpage()
pushViewport( viewport(layout = grid.layout(100, 100) ))
print( p2, vp=viewport(layout.pos.row=12:92, layout.pos.col=16:98 ))
print( p1_1, vp=viewport(layout.pos.row=48:93, layout.pos.col=16:98 ))
print( p3, vp=viewport(layout.pos.row=12:92, layout.pos.col=16:98 ))
############
print( p2, vp=viewport(layout.pos.row=8:88, layout.pos.col=14:96 ))
print( p1_2, vp=viewport(layout.pos.row=44:89, layout.pos.col=16:98 ))
print( p3, vp=viewport(layout.pos.row=8:88, layout.pos.col=14:96 ))
############
print( p2, vp=viewport(layout.pos.row=4:84, layout.pos.col=12:94 ))
print( p1_3, vp=viewport(layout.pos.row=24:85, layout.pos.col=12:94 ))
print( p3, vp=viewport(layout.pos.row=4:84, layout.pos.col=12:94 ))
############
print( p3, vp=viewport(layout.pos.row=4:84, layout.pos.col=12:94 ))
# grid.text("Morphology", x=0.02, y=0.25, rot=90, gp = gpar( fontface = "bold", fontsize = 20, fontfam
dev.off()
##############################################################################################
##############################################################################################
path="ms2_figures"
id_1=495
id_2=347
id_3=2268
for(i in 1:3){
id=get(paste0("id_", i))
msms <- read.csv(file=paste0(path, "/", id, ".tsv"), header=T, sep="\t")
set <- msms[which(msms$rel.intensity > 0),]
pal=pal_npg()(10)
width=max(set$mz)/150
p <- ggplot(set, aes(x=mz, y=rel.intensity)) +
  geom_col(width=width, fill=pal[i]) +
   #geom_segment(data=segment, aes(x=x, y=y, xend=xend, yend=yend, color=as.character(group)), size=2,
   labs(x="") +
   theme_minimal() +
   theme(
      text=element_text(family="serif"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title.y = element_blank(),
      axis.line = element_blank(),
      panel.grid = element_blank(),
      legend.position = "none"
     )
 assign(paste0("p1_", i), p)    }
p2 <- ggplot(segment) +
   #geom_segment(aes(x=x, y=y, xend=xend, yend=yend), color="#BDBDBDFF", size=4, alpha=0.8) +
   annotate("rect", xmin=0, xmax=10, ymin=0 , ymax=10, color="black", fill="white", size=3, alpha=0.5)
   theme_minimal() +
   theme(
      text=element_text(family="serif"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title.y = element_blank(),
      axis.line = element_blank(),
      panel.grid = element_blank(),
      legend.position = "none"
     )
svg("msms.svg", width=12, height=8)
grid.newpage()
pushViewport( viewport(layout = grid.layout(100, 100) ))
print( p2, vp=viewport(layout.pos.row=12:92, layout.pos.col=16:98 ))
print( p1_3, vp=viewport(layout.pos.row=48:93, layout.pos.col=16:98 ))
print( p3, vp=viewport(layout.pos.row=12:92, layout.pos.col=16:98 ))
############
print( p2, vp=viewport(layout.pos.row=8:88, layout.pos.col=14:96 ))
print( p1_2, vp=viewport(layout.pos.row=44:89, layout.pos.col=16:98 ))
print( p3, vp=viewport(layout.pos.row=8:88, layout.pos.col=14:96 ))
############
print( p2, vp=viewport(layout.pos.row=4:84, layout.pos.col=12:94 ))
print( p1_1, vp=viewport(layout.pos.row=24:85, layout.pos.col=12:94 ))
print( p3, vp=viewport(layout.pos.row=4:84, layout.pos.col=12:94 ))
############
print( p3, vp=viewport(layout.pos.row=4:84, layout.pos.col=12:94 ))
# grid.text("Morphology", x=0.02, y=0.25, rot=90, gp = gpar( fontface = "bold", fontsize = 20, fontfam
dev.off()
##########################################################################################
##########################################################################################
path="ms2_figures_label"
```

```
id_1=495
id_2=347
id_3=2268
for(i in 1:3){
id=get(paste0("id_", i))
source <- read.csv(file=paste0(path, "/", id, ".tsv"), header=T, sep="\t")
p <- ggplot(source, aes(x=mz, y=rel.intensity)) +
 geom_bar(stat="identity",width=max(source$mz)/150,fill=ifelse(source$rel.intensity>0,"black","red")) +
 geom_point(size=1.3,color=ifelse(source$rel.intensity>0,"black","red"),alpha=ifelse(source$match==0,0
 #xlim(0,max(source$mz, na.rm=T)) +
 theme(text=element_text(family="serif"),
   panel.background=element_rect(fill="transparent",color="white"),
   panel.grid=element_line(color="grey85"),
   panel.border = element_rect(fill=NA, color="black", size=3, linetype="solid"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      axis.line = element_blank() )
   #plot.margin = unit(c(3, 1, 3, 1), "cm"))
 assign(paste0("p1_", i), p)
 ggsave(p, file=paste0("ms2_", id, ".svg"), width=6, height=4) }
####################################################################################################
####################################################################################################
library(ggforce)
library(ggalt)
library(tidyverse)
library(ggsci)
library(grid)
p <- ggplot() +
 geom_ellipse(aes(x0 = 0, y0 = 0, a = 7, b = 3, angle = 0), size=5, color="white", fill="#8491B4FF") +
 coord_fixed() +
 theme_minimal() +
 theme(text=element_text(family="serif"),
   #panel.background=element_rect(fill="transparent",color="white"),
   panel.grid=element_blank(),
   plot.margin =unit(c(0,0,0,0),"cm"),
   panel.spacing =unit(c(0,0,0,0),"cm"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      axis.line = element_blank() )
```

```
svg("database.svg", width=5, height=7)
grid.newpage()
pushViewport( viewport(layout = grid.layout(100, 100) ))
print( p, vp=viewport(layout.pos.row=60:90, layout.pos.col=5:95 ) )
print( p, vp=viewport(layout.pos.row=50:80, layout.pos.col=5:95 ) )
print( p, vp=viewport(layout.pos.row=40:70, layout.pos.col=5:95 ) )
print( p, vp=viewport(layout.pos.row=30:60, layout.pos.col=5:95 ) )
print( p, vp=viewport(layout.pos.row=20:50, layout.pos.col=5:95 ) )
dev.off()
#######################################################################################################
#######################################################################################################
library(ggforce)
library(ggalt)
library(tidyverse)
library(ggsci)
library(grid)
data="/media/wizard/back/0703_all/490_initial_8_neg_495/fingerprints/C17H24O10_[M-H]-.fpt"
fp <- read.csv(file=data, header=T, sep="\t")
fp <- cbind(seq(nrow(fp)),fp)
colnames(fp) <- c("num", "pp")
fp$group <- "g1"
eg_fp <- fp[1:15,]
p <- ggplot(eg_fp, aes(x=as.factor(num), y=group, fill=pp)) +
  geom_tile(color="black", size=3) +
  scale_fill_gradient(low="white", high="#3C5488FF") +
  coord_fixed() +
   theme_minimal() +
   theme(text=element_text(family="serif"),
   panel.grid=element_blank(),
   plot.margin =unit(c(0,0,0,0),"cm"),
   panel.spacing =unit(c(0,0,0,0),"cm"),
   legend.position = "none",
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      axis.line = element_blank() )
ggsave(p, file="495.fp.svg", width=8, height=1)
#######################################################################################################
#######################################################################################################
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
library(sunburstR)
library(RColorBrewer)
library(shiny)
metadata_path="../canopus_neg.tsv"
metadata <- read.csv(file=metadata_path, header=T, sep="\t", quote = "")
##### see in network
edges <- metadata[, colnames(metadata) %in% c("id", "parentId")]
network <- as_tbl_graph(edges) %>%
    activate(nodes) %>% #as_tibble()
    mutate(deg = centrality_degree(mode='in'))
layout=ifelse(nrow(metadata)>1000, "unrooted", "fr")
layout_n <- create_layout(network, layout = layout)
##############################################################
##############################################################
p <-
ggraph(layout_n) +
geom_edge_fan(edge_width=0.5, color="lightblue", show.legend=F) +
#geom_node_point(aes(fill=str_wrap(classification, width=25), size=similarity), shape=21) +
geom_node_point(shape=21, aes(fill=deg, size=deg)) +
scale_edge_width(range=c(0.1,0.7)) +
scale_fill_gradient(low="#1B1919FF", high="#DC0000FF") +
guides(alpha="none") +
#labs(fill="Centrality\ndegree", size="Tanimoto\nsimilarity") +
theme_grey() +
theme(
      text=element_text(family="serif", size=20),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      legend.title= element_text(face="bold", size=15),
      #panel.background = element_rect(fill="white"),
      legend.key.height = unit(0.3, "cm"),
      legend.key.width = unit(0.5, "cm"),
      axis.line = element_blank(),
      #panel.grid = element_blank(),
      #legend.position = "none",
      #strip.text = element_text(size=15, face="bold")
      plot.margin =unit(c(0,0,0,0),"cm"),
      panel.spacing =unit(c(0,0,0,0),"cm")
     )
ggsave(p, file="class_network.svg", width=7, height=6)
#############################################################
#############################################################
pal0= pal_npg()(10)
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
getPalette = colorRampPalette(pal0)
#####
# data=data[which(data$value > 0.01),]
# data$color <- seq(nrow(data))
# plot <- plot_ly( data=data,
# labels = ~id,
# parents = ~parentid,
# values = ~value,
# color = ~color,
# colors = c(palette),
 #colors = colorRamp(c("red", "blue")),
 #colors = c(`4` = "red", `5` = "black", `6` = "blue", `8` = "green"),
# type = "sunburst",
# insidetextorientation="radial"
# ) %>%
# layout( margin = list(l = 0, r = 0, b = 0, t = 0))
# https://stackoverflow.com/questions/12926779/how-to-make-a-sunburst-plot-in-r-or-pythonlibrary(sunbu
metadata_parent_path="../canopus_parent_index.tsv"
metadata_parent <- read.csv(file=metadata_parent_path, header=T, sep="\t", quote = "")
data_path="/media/wizard/back/0703_all/490_initial_8_neg_495/canopus/C17H24O10_[M-H]-.fpt"
```

```
 data <- read.csv(file=data_path, header=F, sep="\t")
 guide_data <- cbind(metadata_parent[,c(1)], data)
 dataset <- cbind(metadata_parent[,c(2)], data)
 colnames(dataset) <- c("id", "value")
 dataset_s <- dataset[which(dataset$value>0.01), ]
 dataset_s <- dataset_s[order(dataset_s$id),]
 #pal<-palette[1:nrow(dataset_s)]
 pal_do <- getPalette(nrow(dataset_s))
 #sunburst(dataset_s, colors=list(range = RColorBrewer::brewer.pal(9, "Set3")))
 sund2b(dataset_s, colors=list(range = RColorBrewer::brewer.pal(9, "Set3")))
 ######################################################################################################
 ######################################################################################################
 metadata_parent_path="../canopus_parent_index.tsv"
 metadata_parent <- read.csv(file=metadata_parent_path, header=T, sep="\t", quote = "")
 data <- metadata_parent[, 2:3]
 #data$num <- 1
 sund2b(data, colors=palette)
# sunburst(data, colors=palette)
```

40 File: json_tree.R

```
library(tidyverse)

library(igraph)

library(ggraph)

library(tidygraph)

library(ggsci)

library(scales)

library(grid)

pal1= pal_jco()(10)

pal2= pal_jama()(7)

pal3= pal_uchicago("dark")(9)
```

```r
pal4= pal_igv("default")(51)

palette= unique(c(pal1, pal2, pal3, pal4))

#########################

setwd("json_tree")

for(i in c(495, 347, 2268)){

id=i

nodes <- read.csv(file=paste0("nodes_",id,".tsv"),header=T,sep="\t")

edges <- read.csv(file=paste0("edges_",id,".tsv"),header=T,sep="\t")

network_nodes <- as_tbl_graph(edges) %>%

   activate(nodes) %>% #as_tibble()

   mutate(deg = centrality_degree(mode='in'), shape = ifelse(name==0, 16, 3)) %>%

   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=TRUE) %>%

   as_tibble()

network_edges <- as_tbl_graph(edges) %>%

   activate(edges) %>%

   as_tibble()

network <- tbl_graph(nodes = network_nodes, edges = network_edges)

###### network

## layout="dendrogram"

layout="tree"

p <- ggraph(network, layout = layout) +
geom_edge_elbow(edge_width=1, color="black", show.legend=F, strength = 1) +

geom_node_point(size = 4, shape=3, color="red") +

#geom_node_label(aes(label=label, fill=label), color="white", size=7,
        #fill="transparent",
       # label.padding = unit(0.5, "lines"),
       # label.r = unit(0.25, "lines"),
       # label.size = 2,
       # fontface="bold",
       # nudge_y = 0.1,
       # family="Times",
       # repel = T) +

scale_color_manual(values=palette) +

scale_fill_manual(values=palette) +

# scale_edge_width(range=c(0.1,0.7)) +

#facet_edges(~facet) +

#guides(size="none", fill="none") +

theme_grey() +

theme(
      text=element_text(family="serif"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      panel.background = element_rect(fill="white"),
      #axis.line = element_blank(),
      panel.grid = element_blank(),
      legend.position = "none",
      strip.text = element_text(size=20, face="bold")
     )

ggsave(p, file=paste0("tree_",id,".svg"), width=4, height=8)  }
```

41 File: kegg_get.R

```r
library(KEGGREST)
library(org.Mm.eg.db)
library(clusterProfiler)
#BiocManager::install("org.Hs.eg.db")
#org <- keggList('organism')
### human <- hsa
### mice <- mmu
### rats <- rno
# save
# load
species="mmu"
pathway <- keggLink("pathway", species)
pathway <- unique(pathway)
# capture.output(mmu_test,file="test")
pathway.database <- vector(mode = "list", length = length(pathway))
for(i in 1:length(pathway)){
    cat(i," >>> ",pathway[[i]],"\n")
    pathway.database[[i]] <- keggGet(dbentries = pathway[[i]]) }
############ compound message
compound.id <- keggList('compound')
compound.id <- names(compound.id)
compound.database <- vector(mode = 'list', length = length(compound.id))
for(i in 1:length(compound.id)){
    cat(i," >>> ",compound.id[i],"\n")
    compound.database[[i]] <- keggGet(dbentries = compound.id[i]) }
#### search <- keggGet(entries = object)
####
## export DBLINKS
sink("~/operation/re_fecal_neg/kegg/dblink.list")
for(i in 1:length(compound.database)){
  cat("BEGIN_COMPOUND\n")
  cat(compound.database[[i]][[1]]$ENTRY, "\n")
  cat("DBLINKS\n")
  for(j in 1:length(compound.database[[i]][[1]]$DBLINKS)){
    cat(compound.database[[i]][[1]]$DBLINKS[j], "\n")
  }
  cat("NAME\n")
  for(j in 1:length(compound.database[[i]][[1]]$NAME)){
    cat(compound.database[[i]][[1]]$NAME[j], "\n")
  }
  cat("END_COMPOUND\n")
  cat("\n")
}
sink()

#####################################################
## kegg api
png <- keggGet("mmu01100", "image")
library(png)
writePNG(png, "test.png")
#####################################################
query=keggGet(c("mmu:01100"))
```

42 File: lcms_xcms.R

```
library(xcms)
library(RColorBrewer)
library(magrittr)
library(pheatmap)
library(SummarizedExperiment)
# data name list
path <- "."
dda_file=list.files(path = path, pattern = "*.mzML$", all.files = FALSE,
                    full.names = FALSE, recursive = FALSE,
                    ignore.case = FALSE, include.dirs = FALSE)
# metadata
pd <- data.frame(sample_name = dda_file,
                 sample_group = rep("test", length(dda_file)),
                 stringsAsFactors = FALSE)
# xcms read the data
raw_data <- readMSData(files = dda_file, pdata = new("NAnnotatedDataFrame", pd),
                       mode = "onDisk")
```

43 File: linear_regression.R

```
library(ggsci)
library(tidyverse)
library(dplyr)
library(stringr)
library(ggforce)
library(ggpmisc)

setwd("linear_regression")

files=list.files(path = ".", pattern = "*.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)

for(file in files){

savename=strsplit(file,split=".tsv")

data <- read.csv(file=file,header=T,sep="\t")

p <- ggplot(data, aes(x=chem, y=id_content)) +

  geom_smooth(method = "lm", se=FALSE, color = "black", alpha=0.3, size=0.4, formula = y ~ x) +

  geom_point(shape=21, stroke=0, size=4, aes(fill=subgroup)) +

  labs(fill="Group") +

  coord_cartesian(clip = "off") +

  annotate("text", x = min(data$chem), y = max(data$id_content)*(16/20),
        label = paste0("Linear correlation: ",as.character(savename)),
            color="black",size = 4, fontface="bold", family="Times", hjust = 0, alpha=0.8) +

  annotate("text", x = min(data$chem), y = max(data$id_content)*(15/20),
        label = paste0("r(pearson): ",as.character(round(data[1,colnames(data) %in% c("r.pearson.")],4)
            color="black",size = 4, fontface="bold", family="Times", hjust = 0, alpha=0.8) +

  annotate("text", x = min(data$chem), y = max(data$id_content)*(14/20),
        label = paste0("p-value: ",as.character(round(data[1,colnames(data) %in% c("p.value")],4))),
            color="black",size = 4, fontface="bold", family="Times", hjust = 0, alpha=0.8) +

  scale_size_continuous( trans="exp", range=c(2, 5)) +

  scale_fill_manual(values = c("control"="#4A6990FF","drug"="#95CC5EFF","model"="black",
                "pro_low"="#FDAE6BFF","pro_medium"="#FD8D3CFF","pro_high"="#E6550DFF",

                "raw_low"="#9ECAE1FF","raw_medium"="#6BAED6FF","raw_high"="#3182BDFF")) +

  theme(text=element_text(family="serif"))

ggsave(p,file=paste0(savename,".svg"),width=8,height=6.5)

print(file)

}
```

44 File: magick_svg.R

```
library(magick)

setwd("results/structure_2d")

img <- image_read_svg("3918.svg", width = 3000, height = 3000)

image_display(img)

image_write(img,path="test.svg", format="svg")
```

45 File: MCnebula_child_nebula.R

```
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
library(gridSVG)
library(rsvg)
library(gridExtra)
## Many steps of this script may not be easy to implement, such as having to draw the compound structur
```

```r
path="network_facet_0.50"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
                        full.names = FALSE, recursive = FALSE,
                        ignore.case = FALSE, include.dirs = FALSE)
nodes1 <- read.csv(file="fingerid_first_score.tsv", quote="", header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv", quote="", header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
nodes$similarity <- as.numeric(nodes$similarity)
nodes$classification <- gsub("null", "Undefined", nodes$classification)
##### color palette
pal1= pal_simpsons()(16)
pal1= pal1[c(14,2,4,6,7,8,13,1,15,16)]
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
## an instance to draw lignans and iridoids of E. ulmoides
## select1="Iridoids and derivatives.tsv"
## select2="Lignans, neolignans and related compounds.tsv"
select_list=c(34,39)
plot <- list()
## Unfortunately, the structure of the compound must be drawn previously and converted into a Cairo SVG
stru_path="structure_2d/smiles_draw"
structure_list <- list.files(path = stru_path, pattern = "*.mol.svg.cairo.svg$", all.files = FALSE,
                             full.names = FALSE, recursive = FALSE,
                             ignore.case = FALSE, include.dirs = FALSE)
matrix <- data.frame(structure_list) %>% mutate(catch="T")
canopus_set <- read.csv(file="canopus_pp_filter.tsv",header=T,sep="\t")
canopus_set <- t(canopus_set)
colnames(canopus_set)=as.character(canopus_set[1,])
canopus_set <- canopus_set[-1,]
######################################################################
metadata_path="../canopus_neg.tsv"
metadata <- read.csv(file=metadata_path, header=T, sep="\t", quote = "")
metadata <- metadata[,c(2,3,4)]
metadata$class <- paste0("C",metadata$absoluteIndex)
######################### start plot
for(i in select_list){
  edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
  edges_id <- unique(c(edges$source, edges$target))
  cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
  ###########################################
  edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
  edges <- edges[!duplicated(edges[,1:2]), ]
  network_nodes <- as_tbl_graph(edges) %>%
    activate(nodes) %>% #as_tibble()
    mutate(deg = centrality_degree(mode='in')) %>%
      merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
      as_tibble()
    network_edges <- as_tbl_graph(edges) %>%
      activate(edges) %>%
      as_tibble()
    network <- tbl_graph(nodes = network_nodes, edges = network_edges)
    #########
    ######### candidate: graphopt kk fr mds
    layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
    layout_n <- create_layout(network, layout = layout)
    ##########################
    ##########################
    ### structure
    elements <- layout_n[,colnames(layout_n) %in% c("name", "x", "y", "similarity", "classification")]
    elements$link_structure <- paste0(elements$name,".mol.svg.cairo.svg")
    grid_elements <- merge(elements, matrix, all.x=T, by.x="link_structure", by.y="structure_list", sor
    ### grobify
    structure_p<-list()
    list_stru <- grid_elements[which(grid_elements$catch=="T"), colnames(grid_elements) %in% c("link_st
    prefix=c()
    for(j in list_stru){
      id<-strsplit(j, split=".mol.svg.cairo.svg")[[1]]
      #structure_p[[as.numeric(id)]] = grobify(readPicture(paste0(path,"/",j)))
      assign(paste0("grob_",id), grobify(readPicture(paste0(stru_path,"/",j))))
      cat(id," >>> ", "structure_p\n")
    }
    aes_stru <- mutate(grid_elements[which(grid_elements$catch=="T"),], grob=paste0("grob_",name), id=na
    ##########################
```

```
##########################
##########################
### ring_bar plot
ring_data <- canopus_set[,colnames(canopus_set) %in% c(elements$name)]
list_palette = data.frame(cbind(sort(unique(elements$classification)), palette[1:length(unique(eleme
colnames(list_palette) <- c("classification", "color")
elements_palette <- merge(elements, list_palette, all.x=T, by="classification", sort=T)
for(j in 1:ncol(ring_data)){
  id <- colnames(ring_data)[j]
  fill_in=elements_palette[which(elements_palette$name==id),]$color
  fill_border="black"
  df <- data.frame(ring_data[,colnames(ring_data) %in% c(id)])
  df <- mutate(df, class=rownames(df))
  colnames(df)=c("value","class")
  df$num <- seq(nrow(df))
  df <- merge(df, metadata, all.x=T, by="class", sort=T)
  df <- df[order(df$num),]
  df$fill <- paste0(df$class, ": ", df$name)
  df$fill <- factor(df$fill, levels=df[order(df$absoluteIndex), colnames(df) %in% c("fill")])
  p <- ggplot(df, aes(x=num, y=value)) +
    geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = -5, ymax = 0), f
    geom_ribbon(aes(x=ifelse(num==1, 0, ifelse(num==nrow(df), num+1, num)), ymin = 0, ymax = 1.1), 
    geom_col(alpha=1, aes(fill=fill), color="white", size=0.02) +
    ylim(-5,1.3) +
    coord_polar() +
    labs(fill="") +
    scale_fill_manual(values=palette) +
    theme_minimal() +
    theme(
          text=element_text(family="Times"),
          axis.ticks = element_blank(),
          axis.text = element_blank(),
          axis.title = element_blank(),
          panel.grid = element_blank(),
          panel.grid.major =element_blank(),
          panel.grid.minor = element_blank(),
          legend.position = "none",
          legend.title=element_text(face="bold", hjust= -0.5),
          panel.border = element_blank(),
          plot.margin =unit(c(0,0,0,0),"cm"),
          panel.spacing =unit(c(0,0,0,0),"cm")
```

```
        )
        assign(paste0("ring_", id), p)
    }
    ##########################
    ##########################
    plot[[i]] <-
      ggraph(layout_n) +
      geom_edge_fan(aes(edge_width=ftalign_similarity),
                    color="lightblue", show.legend=F) +
geom_node_point(aes(fill=str_wrap(classification, width=25)), size=1, shape=21) +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1,0.7)) +
facet_edges(~facet) +
labs(fill="Access classes", size="Tanimoto\nsimilarity") +
guides(fill=guide_legend(override.aes=list(size=5))) +
theme_grey() +
theme(
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      legend.title= element_text(face="bold"),
      legend.key.height = unit(0.6, "cm"),
      axis.line = element_blank(),
      strip.text = element_text(size=15, face="bold"),
)
assign("find_id", plot[[i]])
## As we see it, there must be some bug in the ggimage package that causes the position to be misaligne
## size and posion adjust
min=min(elements$similarity)
delta=max(elements$similarity)-min
step=1/delta
if(i==39 | i==34){p_size=0.85; p_dist=0.105}else{p_size=0.65; p_dist=0.09}
## draw subview into the network
for(k in 1:ncol(ring_data)){
  id=colnames(ring_data)[k]
  aes_ring=elements[which(elements$name==id),]
  size=(aes_ring$similarity-min)*step+p_size
  ## ppcp datacet
  plot[[i]] <- plot[[i]] + geom_subview(x=aes_ring$x-p_dist, y=aes_ring$y-p_dist,
                                        subview=get(paste0("ring_", id)),
                                        width=size, height=size )
  ## structure
  kk <- which(aes_stru$id==id)
  size=aes_stru[kk,]$similarity
  plot[[i]] <- plot[[i]] + geom_subview(x=aes_stru[kk,]$x, y=aes_stru[kk,]$y,
                                        subview=arrangeGrob( get(aes_stru[kk,]$grob) ),
                                        width=size*6/5, height=size*6/5)
}
#############################################
ggsave(plot[[i]], file=paste0("tt_zoom_",i,".svg"), width=7, height=5.5)
cat(i, edge_list[i], "\n")
}
```

46 File: MCnebula_multi_nebulae.R

```
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(grid)
library(stringr)
library(ggimage)
library(gridExtra)
#### Eucommia analyses
path="network_facet_0.50"
edge_list <- list.files(path = path, pattern = "*.tsv$", all.files = FALSE,
                        full.names = FALSE, recursive = FALSE,
                        ignore.case = FALSE, include.dirs = FALSE)
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes2$classification[grep("null", nodes2$classification)] <- "Undifined"
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=T)
nodes <- nodes[which(nodes$superclass!=""), ]
##### color palette
pal1= pal_simpsons()(16)
pal1= pal1[c(14,2,4,6,7,8,13,1,15,16)]
pal2= pal_d3("category20")(20)
```

```
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
pal5= pal_jco()(10)
pal6= pal_startrek()(7)
pal7= pal_ucscgb()(26)
pal8= pal_locuszoom()(7)
pal9= pal_rickandmorty()(12)
palette= unique(c(pal1, pal2, pal3, pal4, pal5, pal6, pal7, pal8, pal9))
plot <- list()
for(i in 1:length(edge_list)){
  edges <- read.csv(file=paste0(path, "/", edge_list[i]), header=T,sep="\t")
  edges_id <- unique(c(edges$source, edges$target))
  cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
  ##########################################
  edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
  edges <- edges[!duplicated(edges[,1:2]), ]
  network_nodes <- as_tbl_graph(edges) %>%
    activate(nodes) %>% #as_tibble()
    mutate(deg = centrality_degree(mode='in')) %>%
      merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
      as_tibble()
    network_edges <- as_tbl_graph(edges) %>%
      activate(edges) %>%
      as_tibble()
    network <- tbl_graph(nodes = network_nodes, edges = network_edges)
    ######### candidate: graphopt kk fr mds
    layout=ifelse(nrow(network_nodes)>1000, "mds", "fr")
    plot[[i]] <-
      ggraph(network, layout = layout) +
      geom_edge_fan(aes(edge_width=ftalign_similarity), color="black", show.legend=F) +
      geom_node_point(aes(size = as.numeric(similarity), fill=classification), shape=21) +
      # geom_node_text(aes(label=name),size=3) +
      scale_color_manual(values=palette) +
      scale_fill_manual(values=palette) +
      scale_edge_width(range=c(0.1,0.7)) +
      facet_edges(~str_wrap(facet, width=30)) +
      guides(size="none", fill="none") +
      theme_grey() +
      theme(
            text=element_text(family="serif"),
            axis.ticks = element_blank(),
            axis.text = element_blank(),
            axis.title = element_blank(),
            panel.background = element_rect(fill="white"),
            #axis.line = element_blank(),
            panel.grid = element_blank(),
            legend.position = "none",
            strip.text = element_text(size=15, face="bold")
      )
      # ggsave(plot[[i]], file="test.svg", width=5, height=5)
      cat(i, edge_list[i], "\n")
}
## grid
svg("child_nebula.svg", width=18*1.6, height=22*1.5)
n=length(edge_list)
s=n^(1/2); if(round(s)!=s){s=round(s); ss=s+1}else{ss=s}
grid.newpage()
pushViewport(viewport(layout = grid.layout(ss, s)))
r_ss=1
for(i in 1:n){
  c_s=i%%s
  if(c_s==0){c_s=s}
  print( plot[[i]], vp=viewport(layout.pos.row=r_ss, layout.pos.col=c_s ))
  cat("push view port of ",i,"\n")
  if(c_s==s){ r_ss=r_ss+1}
}
dev.off()
```

47 File: MCnebula_parent_nebula.R

```
library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
library(ggsci)
library(scales)
library(stringr)
#### Eucommia analysis
pal1= pal_simpsons()(16)
pal2= pal_d3("category20")(20)
pal3= pal_futurama()(12)
pal4= pal_uchicago("light")(9)
```

```r
palette= unique(c(pal1[c(4, 2:3, 1, 5:8,10,11, 13:16)], pal2, pal3, pal4))
nodes1 <- read.csv(file="fingerid_first_score.tsv",header=T,sep="\t")
nodes2 <- read.csv(file="stat_classification.tsv",header=T,sep="\t")
nodes2$classification <- nodes2$definition
nodes <- merge(nodes1, nodes2, by="id", all.x=T, sort=F)
nodes <- nodes[which(nodes$superclass!=""), ]
edges <- read.csv(file="source_target_tree_0.4.tsv", header=T,sep="\t")
edges_id <- unique(c(edges$source, edges$target))
cont_edges_id <- edges_id[!(edges_id %in% nodes$id)]
############################################
edges <- edges[!(edges$source %in% cont_edges_id | edges$target %in% cont_edges_id), ]
edges <- edges[!duplicated(edges[,1:2]), ]
network_nodes <- as_tbl_graph(edges) %>%
   activate(nodes) %>% #as_tibble()
   mutate(deg = centrality_degree(mode='in')) %>%
   merge(nodes, by.x="name", by.y="id", all.x=TRUE, sort=F) %>%
   as_tibble()
network_edges <- as_tbl_graph(edges) %>%
   activate(edges) %>%
   as_tibble()
network <- tbl_graph(nodes = network_nodes, edges = network_edges)
layout_n <- create_layout(network, layout = "mds")
#### molecular network
p <- ggraph(layout_n) +
geom_edge_fan(aes(edge_width=ftalign_similarity), color="lightblue", show.legend=F) +
geom_node_point(aes(size = as.numeric(similarity), fill=str_wrap(superclass, width=25)), shape=21) +
#geom_node_text(aes(filter= deg>12,label=name),size=1) +
scale_color_manual(values=palette) +
scale_fill_manual(values=palette) +
scale_edge_width(range=c(0.1, 0.7)) +
#facet_nodes(~classification) +
#guides(size="none") +
guides(fill = guide_legend(override.aes = list(size=5))) +
labs(fill="Superclass", size="Tanimoto similarity") +
theme_grey() +
theme(
      text=element_text(family="Times"),
      axis.ticks = element_blank(),
      axis.text = element_blank(),
      axis.title = element_blank(),
      panel.background = element_rect(fill="white"),
      #axis.line = element_blank(),
      legend.key.width = unit(1, "cm"),
      legend.key.height = unit(1.8, "cm"),
      legend.title = element_text(size=20, face="bold", hjust=0.2),
      legend.text = element_text(size=20),
      legend.background = element_rect(fill="transparent"),
      #legend.position = c(0.6,0.25),
      panel.grid = element_blank(),
      strip.text = element_text(size=20, face="bold")
     )
# ggsave(p, file="parent_network.tiff", width=20, height=22)
ggsave(p, file="parent_network.svg", width=20, height=16)
```

48 File: opls_da.R

```
library(ropls)
library(ggbiplot)
library(ggsci)
library(scales)
library(ggrepel)

args<-commandArgs(TRUE)

setwd(args[1])

datas=list.files(path = ".", pattern = "*pca.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)

metadatas=list.files(path = ".", pattern = "^metadata*", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)

n=length(datas)

for(i in 1:n){

savename=strsplit(datas[i],split="_pca.tsv")
data <- read.table(file=datas[i],header= T,row.names= 1,sep= "\t",check.names=F)
class <- read.table(file=metadatas[i],header= T,row.names= 1,sep= "\t",check.names=F)
```

```
oplsda <- opls(x = data, y = class[, "group"], predI = 1, orthoI = NA)
df <- cbind(oplsda@scoreMN[, 1], oplsda@orthoScoreMN[, 1])
colnames(df) <- c("h1", paste0("o", 1))
x_lab <- paste0("T score[1](", oplsda@modelDF[1, "R2X"] * 100, "%)")
y_lab <- paste0("Orthogonal T score[1](", oplsda@modelDF[2, "R2X"] * 100, "%)")
df <- as.data.frame(df)
vip=data.frame(oplsda@vipVn)
vip=cbind(rownames(vip),vip)
colnames(vip)=c("id","vip")
p <- ggplot(df, aes(x=h1, y=o1)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1,aes(fill=class$subgroup)) +
    stat_ellipse(aes(color=class$group), level = 0.95) +
    geom_label_repel(data=df, aes(x=h1, y=o1, label=class$name),
                     color="black", alpha=0.5, fontface="bold", size=2, angle= 0,
                     direction="both", segment.size = 0.2, segment.alpha = 0.3,
                     inherit.aes = FALSE, hjust = 0,
                     family="Times") +
    scale_color_npg() +
    scale_fill_npg() +
    labs(x=x_lab,y=y_lab,title="OPLS-DA") +
    theme(plot.title = element_text(hjust = 0.5),text=element_text(family="serif"))

ggsave(p,file=paste0(savename,"_oplsda.svg"), height=6, width=10)

df=cbind(rownames(df),df)

colnames(df)=c("sample","h1","o1")

df=rbind(df,c("summary",paste0("t1(", oplsda@modelDF[1, "R2X"] * 100, "%)"),paste0("ot1(", oplsda@modelI

write.table(vip, file = paste0(savename,".vip"), quote = FALSE, append = FALSE, sep = "\t", row.names =

write.table(df, file = paste0(savename,".ropls"), quote = FALSE, append = FALSE, sep = "\t", row.names =
}
```

49 File: opls_facet_ggplot.R

```
library(ggplot2)
library(ggsci)

args<-commandArgs(TRUE)
```

```
setwd(args[1])

df <- read.csv(file= "opls_facet.tsv",header= T,sep= "\t")

PC1 <- df[which(df$PC1=="PC1"),]

PC2 <- df[which(df$PC1=="PC2"),]

df <- df[which(df$sample!="anno"),]

anno_x=min(as.numeric(df$PC1))*(20/20)

anno_y=max(as.numeric(df$PC2))*(22/20)

p <- ggplot(df, aes(x=as.numeric(PC1), y=as.numeric(PC2), fill=subgroup)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1) +
    stat_ellipse(aes(color=subgroup), level = 0.8) +
    #scale_color_npg() +
    #scale_fill_npg() +
    scale_color_manual(values = c("control"="grey","model"="#374E55FF",

                "pro_low"="#FDAE6BFF","pro_medium"="#FD8D3CFF","pro_high"="#E6550DFF",

                "raw_low"="#9ECAE1FF","raw_medium"="#6BAED6FF","high_raw"="#3182BDFF")) +

    scale_fill_manual(values = c("control"="grey","model"="#374E55FF",

                "pro_low"="#FDAE6BFF","pro_medium"="#FD8D3CFF","pro_high"="#E6550DFF",

                "raw_low"="#9ECAE1FF","raw_medium"="#6BAED6FF","raw_high"="#3182BDFF")) +

    guides(color= "none") +

    geom_text(data=PC1, aes(x=anno_x, y=anno_y, label=figure),
            hjust=0, color="black", fontface="bold",alpha=0.6, size=2, inherit.aes = FALSE, family="Tim

    geom_text(data=PC2, aes(x=anno_x, y=anno_y*(18/22), label=figure),
            hjust=0, color="black", fontface="bold",alpha=0.6, size=2, inherit.aes = FALSE, family="Tim

    labs(y="Orthogonal T score[1]", x="T score[1]", fill="Group") +
    #scale_x_continuous(limits=c(-60 ,60)) +

    #scale_y_continuous(limits=c(-60 ,60)) +

    facet_grid(g2_deep ~ g1_from) +

    theme(legend.position = "right",text=element_text(family="serif"))

ggsave(p,file="opls_facet.svg",width=8,height=6.5)
```

50 File: pca_facet_ggplot.R

```
library(ggplot2)
library(ggsci)

args<-commandArgs(TRUE)

setwd(args[1])

df <- read.csv(file= "pca_facet.tsv",header= T,sep= "\t")

PC1 <- df[which(df$PC1=="PC1"),]

PC1$figure <- paste0("PC1(",PC1$figure*100,"%)")

PC2 <- df[which(df$PC1=="PC2"),]

PC2$figure <- paste0("PC2(",PC2$figure*100,"%)")

df <- df[which(df$sample!="anno"),c(1:6)]

anno_x=min(as.numeric(df$PC1))*(20/20)

anno_y=max(as.numeric(df$PC2))*(22/20)

p <- ggplot(df, aes(x=as.numeric(PC1), y=as.numeric(PC2), fill=subgroup)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1) +
    stat_ellipse(aes(color=subgroup), level = 0.7) +
    #scale_color_npg() +
    #scale_fill_npg() +
    scale_color_manual(values = c("control"="grey","model"="#374E55FF",

                "pro_low"="#FDAE6BFF","pro_medium"="#FD8D3CFF","pro_high"="#E6550DFF",

                "raw_low"="#9ECAE1FF","raw_medium"="#6BAED6FF","high_raw"="#3182BDFF")) +

    scale_fill_manual(values = c("control"="grey","model"="#374E55FF",

                "pro_low"="#FDAE6BFF","pro_medium"="#FD8D3CFF","pro_high"="#E6550DFF",

                "raw_low"="#9ECAE1FF","raw_medium"="#6BAED6FF","raw_high"="#3182BDFF")) +

    guides(color= "none") +

    geom_text(data=PC1, aes(x=anno_x, y=anno_y, label=figure),
            hjust=0, color="black", fontface="bold",alpha=0.6, size=1.5, inherit.aes = FALSE, family="T

    geom_text(data=PC2, aes(x=anno_x, y=anno_y*(18/22), label=figure),
            hjust=0, color="black", fontface="bold",alpha=0.6, size=1.5, inherit.aes = FALSE, family="T

    labs(y="PC2", x="PC1", fill="Group") +

    #scale_x_continuous(limits=c(-60 ,60)) +

    #scale_y_continuous(limits=c(-60 ,60)) +

    facet_grid(g2_deep ~ g1_from) +

    theme(legend.position = "right",text=element_text(family="serif"))

ggsave(p,file="pca_facet.svg",width=8,height=6.5)
```

51 File: pca_ggbiplot.R

```
library(ggbiplot)
library(ggsci)
library(scales)
library(ggrepel)

args<-commandArgs(TRUE)
```

```
setwd(args[1])

datas=list.files(path = ".", pattern = "*pca.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)

metadatas=list.files(path = ".", pattern = "^metadata*", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)
n=length(datas)

for(i in 1:n){
savename=strsplit(datas[i],split=".tsv")
source<- read.table(file= datas[i],header= T,row.names= 1,sep= "\t")
class<- read.table(file= metadatas[i],header= T,row.names= 1,sep= "\t")
pca<- prcomp(source, scale. = TRUE)
pca_anno <- as.data.frame(pca[5])

output <- ggbiplot(pca, obs.scale = 1, var.scale = 1, groups = class$subgroup, ellipse = TRUE, circle =
            varname.size=0, var.axes = F) +
        geom_label_repel(data=pca_anno, aes(x=x.PC1, y=x.PC2, label=class$name),
                      color="black", alpha=0.5, fontface="bold", size=2, angle= 0,
                      direction="both", segment.size = 0.2, segment.alpha = 0.3,
                      inherit.aes = FALSE, hjust = 0,
                      family="Times") +
        scale_color_npg() +
        scale_fill_npg() +
        theme(legend.position = "right",text=element_text(family="serif"))

ggsave(output,file=paste0(savename,".svg"))

}
```

52 File: pca_prcomp.R

```
args<-commandArgs(TRUE)

setwd(args[1])

datas=list.files(path = ".", pattern = "*pca.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)

for(data in datas){

savename=strsplit(data,split="_pca.tsv")

data <- read.table(file=data,header= T,row.names= 1,sep= "\t",check.names=F)

pca<- prcomp(data, scale. = TRUE)

pca_x=data.frame(pca$x)

rownames=rownames(pca$x)

pca_x=cbind(rownames,pca_x)

pca=summary(pca)

summary=c("summary",round(pca$importance[2,],4))

pca_x=rbind(pca_x,summary)

write.table(pca_x, file = paste0(savename,".prcomp"), quote = FALSE, append = FALSE, sep = "\t", col.na

}
```

53 File: pearson_correlation.R

```
 library(outliers)

grubbs<-function(x){
  x<-round(x,4)
  grubbs_outliers<-c()
  grubbs_p.value<-c()
  grubbs_g.value<-c()
  grubbs_g<-c()
  grubbs_minormax<-c()
  grubbs_pvalue<-c()
  grubbs_p<-0
  while(grubbs_p<0.05){
    grubbs_outliers<-c(grubbs_outliers,grubbs_minormax)
    grubbs_p.value<-c(grubbs_p.value,grubbs_pvalue)
    grubbs_g<-c(grubbs_g,grubbs_g.value)
    if(sum(x==grubbs_minormax)!=0)x<-x[-which(x==grubbs_minormax)]
    if(sd(x)==0) break
    grubbs_test<-grubbs.test(x,type=10,opposite=F,two.sided=F)
    grubbs_p<-grubbs_test$p.value
    grubbs_pvalue<-grubbs_test$p.value
    grubbs_g.value<-grubbs_test$statistic[1]
    grubbs_a<-strsplit(grubbs_test$alternative," ",fixed=T)
    grubbs_minormax<-as.numeric(unlist(grubbs_a)[3])
  }
  outliner_res<-data.frame(outliers=grubbs_outliers,gvalue=grubbs_g,pvalue=grubbs_p.value)
  return(outliner_res)

}

dixon<-function(x){
    dixon_p.value<-c()
    dixon_q.value<-c()
    dixon_q<-c()
    dixon_pvalue<-c()
    dixon_outliers<-c()
    dixon_minormax<-c()
    dixon_p<-0
    while(dixon_p<0.05){
      dixon_outliers<-c(dixon_outliers,dixon_minormax)
      dixon_p.value<-c(dixon_p.value,dixon_pvalue)
      dixon_q<-c(dixon_q,dixon_q.value)
      if(sum(x==dixon_minormax)!=0)x<-x[-which(x==dixon_minormax)]
      if(sd(x)==0) break
      dixon_test<-dixon.test(x,type=0,opposite=F,two.sided=F)
      dixon_p<-dixon_test$p.value
      dixon_pvalue<-dixon_test$p.value
      dixon_q.value<-dixon_test$statistic[1]
      dixon_a<-strsplit(dixon_test$alternative," ",fixed=T)
      dixon_minormax<-as.numeric(unlist(dixon_a)[3])
    }
    outliner_res<-data.frame(outliers=dixon_outliers,qvalue=dixon_q,pvalue=dixon_p.value)
    return(outliner_res)
  }
```

```r
savepath="correlation"

dir.create(savepath)

data <- read.csv(file="fecal_pos_correlation.tsv",header=T,sep="\t",check.names=F)

group=unique(data$subgroup)

data_set=array(NA,dim=c(nrow(data)*3,ncol(data)*3))

for(i in colnames(data)){
    test<-try(as.numeric(i)); if(is.na(test)==F){begin=i;break}}

for(i in 1:ncol(data)){if(colnames(data)[i]==begin){begin=i}}

#for(i in 3:(begin-1)){data=data[which(is.na(data[,i])==F),]}

for(i in 3:ncol(data)){

    for(j in group){

    X=data[which(data$subgroup==j),i]

    n=try(out<-dixon(round(X,3)))

    if(class(n)=="try-error"){n=try(out<-grubbs(round(X,3)))}

    if(class(n)=="try-error"){out=NULL}

    if(is.null(out$outliers)){x_filter=X}else{x_filter=X[which(X!=c(out$outliers))]}

    if(exists("team_filter")==TRUE){
            team_filter=c(team_filter,x_filter)
        }else{
            team_filter=c(x_filter)}
    }

    data_set[1:length(team_filter),i]=team_filter

    rm(team_filter)}
```

```r
sink(paste0(savepath,"/","pearson_p_value"),append = FALSE, split = FALSE)

print("|factor|id|r|p_value|")

name_set=colnames(data)

for(i in 3:(begin-1)){

    for(j in begin:ncol(data)){

    stat=cor.test(data_set[,i],data_set[,j])

    print(paste0("|",name_set[i],"|",name_set[j],"|",as.vector(stat$estimate),"|",stat$p.value,"|"))}}

sink()
```

54 File: phase1_propagation.R

```r
## library(igraph)
## library(ggraph)
## library(tidygraph)
## library(ggsci)
## library(scales)
library(tidyverse)
library(grid)
library(stringr)
library(ggimage)
library(grImport2)
library(gridSVG)
library(ggpubr)
library(gridExtra)
library(reshape2)
library(gt)
library(Hmisc)
library(stringr)
## obabel, cairosvg, molconvert, these should be installed previously.
parse_structure <- function(smiles, id_n){
  system(paste0("molconvert mol \"", smiles, "\" -o Rtemp.mol"))
  system("obabel Rtemp.mol -imol -osvg -O Rtemp.svg")
  system("cairosvg Rtemp.svg -o Rtemp.cairo.svg")
  assign(id_n, readPicture("Rtemp.cairo.svg"))
  return(get(id_n))
}
## grid 10(n) structure candidate
top <- function(smiles_list, save_id_number=0){
  if(file.exists("structure_check")){
    cat("Dir exist\n")
  }else{
    dir.create("structure_check")
  }
  if(file.exists(paste0("structure_check/", save_id_number, ".svg"))){
    system(paste0("xdg-open ", "structure_check/", save_id_number, ".svg"))
    return(cat("Image exist >>>", paste0("structure_check/", save_id_number, ".svg") ,"\n"))
  }
  ## draw structure in grid
  structure_num=0
  for(i in smiles_list){
    structure_num=structure_num+1
    assign(paste0("project_structure_", structure_num), parse_structure(i, paste0("medium_p", structure_
  }
  svg(paste0("structure_check/", save_id_number, ".svg"), height=2, width=2*structure_num)
  for(i in 1:structure_num){
    grid.picture( get(paste0("project_structure_", i)) , x=i/(structure_num+1), y=0.5, height=1, width=
  }
  dev.off()
  system(paste0("xdg-open ", "structure_check/", save_id_number, ".svg"))
  cat("Save Image >>>", paste0("structure_check/", save_id_number, ".svg"), "\n")
}
## html table get
tab <- function(data_branch, list_col=c("id", "similarity", "name", "formula")){
  t <- gt(data_branch[, colnames(data_branch) %in% list_col]) %>%
  opt_table_font(
    font=list(google_font(name="Times"))
    ) %>%
  tab_header(
    title = md("**Features structure list**")
    #subtitle = md("`gtcars` is an R dataset")
  ) %>%
  opt_align_table_header(align = "left") %>%
  opt_table_lines(extent = c("none")) %>%
  cols_align(
    align = "left",
    columns = everything()
  ) %>%
  tab_style(
      style = cell_borders(
        sides = c("top", "bottom"),
        color = "black",
        weight = px(1.5),
        style = "solid"
      ),
      locations = cells_column_labels()
  ) %>%
  tab_style(
      style = cell_text(v_align="top"),
      locations = cells_column_labels(
        columns = everything()
        #rows = 1:nrow(data_branch)
    )
  ) %>%
  tab_style(
      style = cell_borders(
        sides = c("bottom"),
        color = "black",
        weight = px(1.5),
        style = "solid"
      ),
      locations = cells_body(
        columns=everything(),
        rows=nrow(data_branch)
      )
   ) %>%
  tab_style(
      style = cell_text(v_align="top"),
      locations = cells_body(
        columns = everything()
        #rows = 1:nrow(data_branch)
      )
   )
  return(t)
}
#####################
#####################
```

```
database <- read.csv(file="fingerid_candidate_top10.tsv", header=T, sep="\t")
## catch and draw structure in db
smi <- function(id, n=id){
  smiles <- database[which(database$"id"==id), colnames(database) %in% "smiles"]
  smiles <- top(smiles,n)
  return(smiles)
}
## smi_batch_save
smi_batch <- function(id, n=id){
  smiles <- database[which(database$"id"==id), colnames(database) %in% "smiles"]
  smiles <- top(smiles,n)
}
## catch and pretty table in db
sht <- function(id){
  table <- database[which(database$"id"==id),] %>%
    tab()
  return(table)
}
## replace the candidate structure annotation
rp <- function(data_pip, rp_id, rank_n,
               col=c("similarity",
                     "name",
                     "formula",
                     "xlogp",
                     "smiles",
                     "inchi",
                     "inchikey2D",
                     "links")){
  origin_order=colnames(data_pip)
  data_pip<-data_pip[,order(colnames(data_pip))]
  database<-database[,order(colnames(database))]
  data_pip[which(data_pip$"id"==rp_id),
       colnames(data_pip) %in% col] <- database[which(database$"id"==rp_id)[rank_n], colnames(database)
  data_pip<-data_pip[, order(factor(colnames(data_pip), levels=origin_order))]
  return(data_pip)
}
cut_line <- function(data_pip, cut_id){
  data_pip <- data_pip[which(data_pip$"id"!=cut_id), ]
  return(data_pip)
}
data_save <- function(data_pip, file_name="df"){
  time_temp <- date()
  write.table(data_pip, file=paste0(file_name, "_", time_temp, ".tsv"),
              sep="\t", col.names=T, row.names=F, quote=F)
}
workflow <- function(id_list, sleep_time=5, data_pip=data){
  cut_id_list=c()
  system("echo > cut.list")
  for(i in id_list){
    smi_batch(i)
    cut_else=scan("", nlines=1, what="c")
    if(class(try(cut_else))=="try_error"){
      cat("Try again\n")
      cut_else=scan("", nlines=1, what="c")
    }
    if(cut_else=="s"){
      cat("Skip\n")
      next
    }
    if(cut_else=="c"){
      cut_id_list=c(cut_id_list, i)
      data_pip=cut_line(data_pip, i)
      system(paste0("echo ", i, " >> cut.list"))
      cat("Cut the line of ", i, "\n")
    }else if(is.na(as.numeric(cut_else))==F){
      data_pip=rp(data_pip, i, as.numeric(cut_else))
      cat("Replace id", i, "candidate\n")
    }
  }
  return(data_pip)
}
####################
####################
datapath="com_1.tsv"
data=read.csv(file=datapath, header=T, sep="\t")
```

55 File: pretty_table.R

```
## pretty table
library(tidyverse)
library(reshape2)
library(gt)
library(Hmisc)
library(stringr)
data_save <- function(data_pip, file_name="df"){
  time_temp <- date()
  write.table(data_pip, file=paste0(file_name, "_", time_temp, ".tsv"),
              sep="\t", col.names=T, row.names=F, quote=F)
}

# datapath0="com_compound.tsv"
datapath0="merge.tsv"
#datapath1="indraw_lignans_and_iridoids.tsv"
datapath1="merge_lignans_and_iridoids.tsv"
#datapath11="iridoids_Mon Jan  3 19:56:31 2022.tsv"
#datapath1="com_carboxylic_acids.tsv"
datapath2="stat_classification.tsv"
# dataset <- read.csv(file=datapath0, header=T, sep="\t")
# dataset <- dataset[which(dataset$m.z > 193 & dataset$m.z <194),]
# write.table(dataset[,colnames(dataset) %in% c("id", "smiles")], file="test.tsv", sep="\t", col.names=
data0 <- read.csv(file=datapath0, header=T, sep="\t")
data0 <- data0[order(data0$id),]
data1 <- read.csv(file=datapath1, header=T, sep="\t")
data1 <- data1[order(data1$id),]
origin_order=colnames(data1)
data1 <- merge(data1, data0[,colnames(data0) %in% c("id", "IUPACName")], by="id", all.x=T, sort=T)
data1 <- mutate(data1, extra_name=ifelse(
                                        name!="null", name,
                                        ifelse(
                                               is.na(IUPACName)==T, "null", IUPACName
                                        )
)
)
data1 <- data1[, !(colnames(data1) %in% c("name", "IUPACName"))]
colnames(data1)[which(colnames(data1)=="extra_name")]="name"
data1 <- data1[, order(factor(colnames(data1), levels=origin_order))]
data_save(data1, "chem_supple")
################################################################
#data11 <- read.csv(file=datapath11, header=T, sep="\t")
#data1 <- rbind(data1, data11)
data2 <- read.csv(file=datapath2, header=T, sep="\t")
data <- merge(data1, data2, by="id", all.x=T, sort=T)
data_branch <- data[which(data$similarity>0.45), ]
data_branch <- data_branch[, colnames(data_branch) %in%
    c("id", "rt", "m.z", "variety", "pro.raw", "similarity", "name", "formula", "inchikey2D", "Index")]
data_branch <- data_branch[,c(which(colnames(data_branch)!="Index"), which(colnames(data_branch)=="Inde
data_branch <- data_branch[order(data_branch$inchikey2D, -data_branch$similarity), ]
## data_branch <- data_branch[!duplicated(data_branch$inchikey2D), ]
data_branch <- data_branch[order(data_branch$id, -data_branch$similarity), ]
data_branch <- data_branch[!duplicated(data_branch$id), ]
colnames(data_branch) <- capitalize(colnames(data_branch))
colnames(data_branch)[c(1:6, 10)] <- c("ID", "RT", "m/z", "Processing Variations", "Pro/Raw(peak area)"
data_branch$Number <- seq(nrow(data_branch))
data_branch <- data_branch[c(ncol(data_branch), 1:(ncol(data_branch)-1))]
#data_branch <- data_branch[order(data_branch[,colnames(data_branch) %in% c("m/z")]),]
data_branch$Name <- str_wrap(data_branch$Name, width=40)
###################################################################
t <- gt(data_branch) %>%
  opt_table_font(
    font=list(google_font(name="Times"))
    ) %>%
  tab_header(
    title = md("**Features of lignans and iridoids of _Eucommia ulmoides_ identified in LC-MS negative 
    #subtitle = md("`gtcars` is an R dataset")
  ) %>%
  opt_align_table_header(align = "left") %>%
  tab_footnote(
               footnote = "These features (compounds) were mainly obtained by phase I clustering of MCn
               locations = cells_title(
                                       groups = c("title")
               )
  ) %>%
  opt_table_lines(extent = c("none")) %>%
    cols_align(
               align = "left",
    columns = everything()
  ) %>%
  cols_width(
    Name ~ px(300)
    # ends_with("r") ~ px(100),
    # starts_with("date") ~ px(200),
    # everything() ~ px(60)
  ) %>%
  tab_style(
      style = cell_borders(
        sides = c("top", "bottom"),
        color = "black",
        weight = px(1.5),
        style = "solid"
      ),
      locations = cells_column_labels()
  ) %>%
  tab_style(
      style = cell_text(v_align="top"),
      locations = cells_column_labels(
        columns = everything()
        #rows = 1:nrow(data_branch)
    )
  ) %>%
  tab_style(
      style = cell_borders(
        sides = c("bottom"),
        color = "black",
        weight = px(1.5),
        style = "solid"
      ),
      locations = cells_body(
        columns=everything(),
        rows=nrow(data_branch)
      )
   ) %>%
  tab_style(
      style = cell_text(v_align="top"),
      locations = cells_body(
        columns = everything()
        #rows = 1:nrow(data_branch)
      )
   )
```

56 File: pubchem_hub_merge.R

```
library(tidyverse)
data_list <- list.files(path = ".", pattern = "*smiles.csv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)
```

```
df <- data.frame()
for(i in data_list){
  data <- read.csv(file=i, header=T, check.names=F, sep=",")
  df <- rbind(df, data)
}
file <- read.csv(file="cid_metadata.tsv", check.names=F, sep="\t", header=T)
df <- merge(df, file, by.x="CID", by.y="cid", all.x=T, sort=T)
#file2 <- read.csv(file="../fingerid_first_score.tsv", check.names=F, sep="\t", header=T)
file2 <- read.csv(file="fingerid_first_score.tsv", check.names=F, sep="\t", header=T)
df <- merge(df, file2, all.x=T, by="id", sort=T)
df <- df[!duplicated(df$id), ]
write.table(df, file="merge.tsv", col.names=T, row.names=F, quote=F, sep="\t")
```

57 File: heatmap.R

```
library(pheatmap)
map<- read.table(file= "hot_image-1.tsv",header= T,row.names= 1,sep= "\t")
test=pheatmap(map,
    fontsize=5,cellheight=10,cellwidth=12,border_color="#0a0404",
    treeheight_row=15,treeheight_col=15,
    clustering_method = "complete",
    color=colorRampPalette(c("#e60612","#e60612","white","#4a85c5","#4a85c5"))(100))
pdf("re_heatmap.pdf")
test
dev.off()

################echo /media/wizard/Seagate/MDMN/sep_neg0628/*_*_*/ | xargs -n 1 cp -v compound.config
################
################
#heatmap
## clustering_method 参数设定不同聚类方法，默认为"complete", 可以设定为'ward', 'ward.D', 'ward.D2', 'single
## clustering_distance_rows = "correlation" 参数设定行聚类距离方法为 Pearson corralation，默认为欧氏距离"euc
#clustering_distance_rows = "correlation")
#clustering_method = "ward")
library(pheatmap)
map<- read.table(file= "hot_image.tsv",header= T,row.names= 1,sep= "\t")
group= read.table(file= "group.tsv",header= T,sep="\t")
test=pheatmap(map,annotation_row=group[1],
    fontsize=5,cellheight=10,cellwidth=12,border_color="#0a0404",
    treeheight_row=15,treeheight_col=15,
    clustering_method = "complete",
    color=colorRampPalette(c("#e60612","#e60612","white","#4a85c5","#4a85c5"))(100))
pdf("heatmap.pdf")
test
dev.off()

#################
#################
#################
# pca ggbiplot



################################

library(ggbiplot)
library(ggsci)
library(scales)
library(ggrepel)
library(ggplot2)
library(tidyverse)
library(dplyr)
library(stringr)
library(ggforce)
library(ggpmisc)

datas=list.files(path = ".", pattern = "*pca.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)
savename=strsplit(datas[1],split=".tsv")
source<- read.table(file= datas[1],header= T,row.names= 1,sep= "\t")
class<- read.table(file= "metadata_level.tsv",header= T,row.names= 1,sep= "\t")
pca<- prcomp(source, scale. = TRUE)
pca_anno <- as.data.frame(pca[5])
pca<-cbind(pca_anno,class)

p <- ggplot(pca, aes(x=x.PC1, y=x.PC2, fill=level)) +
    geom_point() +
    #stat_ellipse(aes(color=class$subgroup), level = 0.95) +
    scale_color_npg() +
    scale_fill_npg() +
    guides(fill= guide_colourbar(order = 1)) +

    theme(legend.position = "right",text=element_text(family="serif"))

ggsave(p,file=paste0(savename,"_level.svg"))

#################
#################
##pca  ------------ropls
library(ropls)
library(ggbiplot)
library(ggsci)
library(scales)
source<- read.table(file= "fecal_pos_YH_oplsda.tsv",header= T,row.names= 1,sep= "\t")
class<- read.table(file= "metadata_YH_oplsda.tsv",header= T,row.names= 1,sep= "\t")
pca <- opls(x = source)
df <- pca@scoreMN
df <- as.data.frame(df)
p <- ggplot(df, aes(x=p1, y=p2, fill=class$subgroup)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1) +
    stat_ellipse(aes(color=class$subgroup), level = 0.95) +
    scale_color_npg() +
    scale_fill_npg() +
    theme(legend.position = "right",text=element_text(family="serif"))

svg("raw_and_pro_YH_pca_Nversion.svg", height=6, width=10)
p
dev.off()

#################
#################
##pls-da

library(ropls)
library(ggbiplot)
library(ggsci)
library(scales)
data <- read.table(file= "fecal_pos_pca.tsv",header= T,row.names= 1,sep= "\t",check.names=F)
class <- read.table(file= "metadata.tsv",header= T,row.names= 1,sep= "\t",check.names=F)
plsda <- opls(x = data, y = class[, "subgroup"], orthoI = 0)
df <- plsda@scoreMN
df <- as.data.frame(df)
p <- ggplot(df, aes(x=p1, y=p2, fill=class$subgroup)) +
    geom_point(alpha=0.8, size=3, shape=21, stroke=0.1) +
    stat_ellipse(aes(color=class$subgroup), level = 0.95) +
    scale_color_npg() +
    scale_fill_npg() +
    theme(legend.position = "right",text=element_text(family="serif"))

svg("pls_da_ggplot.svg", height=6, width=10)
p
dev.off()

#################
#################
## opls-da
##


#################
#################
#################




#################
#################
#################
#boxplot
library(ggplot2)
source<-read.table(file="chang",header=T,sep="\t")
boxplot<-ggplot(source,aes(x=days,y=length,fill=days))+
  stat_boxplot(geom="errorbar",width=0.3)+
  geom_boxplot(width=0.7)+
  geom_dotplot(binaxis ="y",
               stackdir="center",
               #position="jitter",
               dotsize = 0.4,)+
  stat_summary(fun.y="mean",geom="point",shape=23,size=3,fill="grey")
pdf("output_boxplot.pdf")
boxplot
```

```
dev.off()

#################
#################
#################
#bar
library(ggplot2)
source<-read.table(file="bar.tsv",header=T,sep="\t")
mzscale=1200
colwidth=mzscale/200
barplot<-ggplot(source, aes(x=mz, y=abun)) +
geom_bar(stat="identity",width=colwidth,fill=ifelse(source$abun>0,'black','red')) +
geom_point(size=0.95,color=ifelse(source$abun>0,'black','red')) +
xlim(0,mzscale) +
theme(panel.background=element_rect(fill='transparent',color='white'),
    panel.grid=element_line(color='grey85'), plot.margin = unit(c(3, 1, 3, 1), "cm"))
pdf("output_bar.pdf")
barplot
dev.off()

################
################
################
行列布局
grid.newpage()
pushViewport(viewport(layout = grid.layout(3, 3)))
print(p.scatter, vp=viewport(layout.pos.row=2:3, layout.pos.col=1:2))
print(p.hist.len, vp=viewport(layout.pos.row=1, layout.pos.col=1:2))
print(p.hist.wid, vp=viewport(layout.pos.row=2:3, layout.pos.col=3))

################
################
################

library(grid)
library(ggplot2)
# prepare ggplot charts
p.hist.len <- ggplot(iris) + geom_histogram(aes(x=Sepal.Length))
p.hist.wid <- ggplot(iris) + geom_histogram(aes(x=Sepal.Width)) + coord_flip()
p.scatter <- ggplot(iris) + geom_point(aes(x=Sepal.Length, y=Sepal.Width))
```

```
# create viewports
library(grid)
grid.newpage()
vp1 <- viewport(x=0, y=0, width=5, height=5)
vp2 <- viewport(x=0, y=0, width=2, height=2)

# direct the charts into the specified viewport
s1 <- print(p, vp=vp1)
s2 <- print(l, vp=vp2)

################
################

library("grImport2")
readPicture("file")
grid.picture(var)

################
################
################
library(visNetwork)
lists = read.table(file="net.csv",header=T,sep="\t")
nodes = read.table()
edges = read.table()

network = visNetwork(nodes, edges, height = "100%", width = "100%")
htmlwidgets::saveWidget(network, "network.html")

################
################
################
#Radial bar plot

library(tidyverse)
data <- read.table(file=args[1],header=T,sep="\t")
data <- data.frame(
  individual=paste( "Mister ", seq(1,60), sep=""),
  group=c( rep('A', 10), rep('B', 30), rep('C', 14), rep('D', 6)),
  value1=sample( seq(10,100), 60, replace=T),
  value2=sample( seq(10,100), 60, replace=T),
  value3=sample( seq(10,100), 60, replace=T)
)
data <- data %>% gather(key = "observation", value="value", -c(1,2))
nObsType <- nlevels(as.factor(data$observation))
data <- data %>% arrange(group, individual)
data$id <- rep( seq(1, nrow(data)/nObsType), each=nObsType)
################
label_data <- data %>% group_by(id, individual) %>% summarize(tot=sum(value))
number_of_bar <- nrow(label_data)
angle <- 90 - 360 * (label_data$id-0.5) /number_of_bar
label_data$hjust <- ifelse( angle < -90, 1, 0)
label_data$angle <- ifelse(angle < -90, angle+180, angle)
p <- ggplot(data) +
  geom_bar(aes(x=as.factor(id), y=value, fill=observation), stat="identity", alpha=0.5) +
  scale_fill_brewer(palette = "Paired") +
  ylim(-150,max(label_data$tot, na.rm=T)) +
  theme_minimal() +
  theme(
    axis.text = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    plot.margin = unit(rep(-1,4), "cm")
  ) +
  coord_polar() +
  geom_text(data=label_data, aes(x=id, y=tot+10, label=individual, hjust=hjust), color="black", fontfac

png('tr_tst2.png',width=300,height=300,units="px",bg="transparent")
p
dev.off()

################
nodes(imagepath,shape"image")
nodes <- data.frame(id = 1:4,
                    shape = c("image", "circularImage"),
                    label = "I'm an image")




visNetwork(nodes, edges) %>%
  visPhysics(stabilization = FALSE) %>%
  visNodes(shapeProperties = list(useBorderWithImage = FALSE)) %>%
  visEdges(smooth = FALSE) %>%
  #visIgraphLayout() %>%
  visOptions(highlightNearest = TRUE, nodesIdSelection = TRUE, selectedBy = "group", manipulation = TRUI
  visLayout(randomSeed = 123)

########################
#bubble plot
#######################

library(ggplot2)
library(tidyverse)
library(dplyr)
library(stringr)
library(ggforce)
library(ggpmisc)
data <- read.csv(file="fecal_pos_mzmine.tsv",header=T,sep="\t")
p <- ggplot(data, aes(x=rt, y=xlogp, size=similarity, fill=m.z, color=str_wrap(classification,30))) +
  geom_smooth(method = "lm", se=TRUE, color = "black", fill = "skyblue", alpha=0.3, size=0.4, formula =
  geom_point(alpha=0.4, shape=21, stroke=0.1) +
  geom_mark_ellipse(aes(color=str_wrap(classification,30)), alpha=0.1, size=0.2, expand=0.02, fill=NA)
  labs(size="Similarity") +
  labs(colour="Classification") +
  coord_cartesian(clip = "off") +
  scale_size_continuous( trans="exp", range=c(2, 5)) +
  theme_minimal() +
  theme(text=element_text(family="serif"))
pdf("bubble.pdf",width=7,height=5)
p
dev.off()

####################

library(ggplot2)
library(tidyverse)
library(dplyr)
library(stringr)
library(ggforce)
library(ggpmisc)
data <- read.csv(file="0.5_com_compound.tsv",header=T,sep="\t")
```

```r
p <- ggplot(data, aes(x=rt, y=xlogp, size=similarity, fill=m.z, color=classification)) +
  geom_smooth(method = "lm", se=TRUE, color = "black", fill = "skyblue", alpha=0.3, size=0.4, formula =
  geom_point(alpha=0.4, shape=21, stroke=0.1) +
  scale_size_continuous(trans="exp", range=c(2, 5)) +
   guides(
   colour = "none",
   fill = guide_colourbar(order = 1),
   size = guide_legend(order = 2)
 ) +
  theme_minimal() +
  theme(text=element_text(family="serif"))
pdf("sep_bubble.pdf",width=7,height=5)
p
dev.off()

###################

library(ggplot2)
library(tidyverse)
library(dplyr)
library(stringr)
library(ggforce)
library(ggpmisc)
data <- read.csv(file="0.5_com_compound.tsv",header=T,sep="\t")
data_sep <- data[which(data$m.z<=800 & data$m.z>=400),]
p <- ggplot(data_sep, aes(x=rt, y=xlogp, size=similarity, fill=m.z, color=classification)) +
  geom_smooth(method = "lm", se=TRUE, color = "black", fill = "skyblue", alpha=0.3, size=0.4, formula =

  stat_cor(method = "pearson",label.x = 3, label.y = 30) +

  stat_poly_eq(
    aes(label = ..eq.label..),
    formula = formula,parse = TRUE, geom = "text",label.x = 3,label.y = 28, hjust = 0) +

  geom_point(alpha=0.4, shape=21, stroke=0.1) +
  scale_size_continuous(trans="exp", range=c(2, 5)) +
   guides(
   colour = "none",
   fill = guide_colourbar(order = 1),
   size = guide_legend(order = 2)
 ) +
  theme_minimal() +
  theme(text=element_text(family="serif"))
pdf("sep_bubble_400_800.pdf",width=7,height=5)
p
dev.off()

###################
#rank
library(tidyverse)
data <- read.csv(file="rank.tsv",header=T,sep="\t")
p <- ggplot(data) +
  geom_point(aes(x=rank, y=log10_raw, size=similarity),fill="#99CCFF",alpha=0.4, shape=21, color="black
  geom_point(aes(x=rank, y=log10_pro, size=similarity),fill="#FF6666",alpha=0.4, shape=21, color="black
  scale_size_continuous(trans="exp", range=c(1, 3)) +
   guides(
   colour = "none",
   size = guide_legend(order = 2)
 ) +
  theme_minimal() +
  labs(y = "log10(area)")
  theme(text=element_text(family="serif"))
pdf("rank.pdf",width=7,height=5)
p
dev.off()

##################

library(ggplot2)

library(ggforce)

library(ggrepel)

n=list.files(path = ".", pattern = "*.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)

for (x in n) {

file <- strsplit(x, split=".tsv")
```

```
savename=paste0(file,".svg")

source<-read.csv(file=paste0(file,".tsv"),header=T,sep="\t")

source_anno=source[which(source$rel.intensity<0),]

barplot<-ggplot(source, aes(x=mz, y=rel.intensity)) +

  geom_bar(stat="identity",width=max(source$mz)/150,fill=ifelse(source$rel.intensity>0,"black","red")) +

  geom_point(size=1.3,color=ifelse(source$rel.intensity>0,"black","red"),alpha=ifelse(source$match==0,0

  xlim(0,max(source$mz, na.rm=T)+50) +

  annotate("text", x = 10, y = 90, label = paste0("Precursor m/z: ",source[1,colnames(source) %in% c("pr
            color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +

  annotate("text", x = 10, y = 78, label = paste0("RT (min): ", source[1,colnames(source) %in% c("RT..mi
            color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +

  annotate("text", x = 10, y = 66, label = paste0("Tanimoto similarity: ", source[1,colnames(source) %in
            color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +

  geom_label_repel(data=source_anno, aes(x=mz, y=rel.intensity, label=round(mz,2)),
                      color="black", alpha=0.5, fontface="bold", size=2, angle= 0,
                      direction="both", ylim=c(0, -100), segment.size = 0.2, segment.alpha = 0.3,
                      inherit.aes = FALSE, hjust = 0,
                      family="Times") +

  # annotate("curve", x = source_anno$mz+1, xend = source_anno$mz+4, y = source_anno$rel.intensity-3, y
  #           colour = "black", curvature =0, size=0.1, alpha=0.5, arrow = arrow(length = unit(0.5, "mm"

  theme(text=element_text(family="serif"),

  panel.background=element_rect(fill="transparent",color="white"),

  panel.grid=element_line(color="grey85"),

  plot.margin = unit(c(3, 1, 3, 1), "cm"))

ggsave(barplot,file=savename,width=8,height=6.5)
```

```
print(file)}

##################






 #########################
 ### line

 library(ggplot2)

 library(ggrepel)

 library(ggsci)

 list <- list.files(path = ".", pattern = "*.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)

 for(file in list){

 savename <- strsplit(file, split=".tsv")

 data <- read.csv(file=file,header=T,sep="\t")

 data <- data[which(data$group!=""),]

 data_anno_mz <- data[1,colnames(data) %in% c("mz")]

 data_anno_rt <- data[1,colnames(data) %in% c("center_rt")]

 tolerance = 0.005

 data_label <- data[which(data$label==1),]

 anno_x_range <- c(0, max(data$rt))

 anno_y_range <- c(0, max(data$intensity))
```

```
delta <- max(data$rt)-min(data$rt)

p <- ggplot(data,aes(x=rt, y=intensity, group=sample, colour=group)) +

 geom_line(alpha=0.8) +

 geom_label_repel(data=data_label, aes(x=rt, y=intensity, label=sample),
                    color="black", alpha=0.5, fontface="bold", size=2, angle= 0, xlim=anno_x_range, yl
                    direction="both", segment.size = 0.2, segment.alpha = 0.3, force = 1,
                    nudge_x = runif(1, min=delta*(1/18), max=delta*(1/10)), nudge_y = max(data$intensi
                    inherit.aes = FALSE, hjust = 0,
                    family="Times") +

 scale_color_npg() +

 scale_fill_npg() +

 labs(color="Peak attribution", x="RT (min)", y="Intensity") +

 annotate("text", x = min(data$rt), y = max(data$intensity)*(16/20),
       label = paste0("Precursor m/z: ",data_anno_mz-tolerance," ~ ",data_anno_mz+tolerance),
           color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +

 annotate("text", x = min(data$rt), y = max(data$intensity)*(15/20),
       label = paste0("RT (min): ",data_anno_rt),
           color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +

 #theme_minimal() +

 theme(text=element_text(family="serif"),

   legend.position = c(0.85,0.75),

   #axis.line = element_line(colour = "black", size=0.2),

   legend.background = element_rect(fill = "transparent", color = "transparent"),

   plot.margin = unit(c(3, 1, 3, 1), "cm"))

ggsave(p,file=paste0(savename,".svg"),width=8,height=6.5)
```

```
print(paste0("Finish >>> ",file))}

#######################################
# python Draw smiles

######################
######################
#change svg to pdf

library(rsvg)

files=list.files(path = ".", pattern = "*.svg$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)

for(file in files){

  name_set=strsplit(file, split=".svg")

  id=name_set[1]

  savename=paste0(id,".pdf")

  rsvg_pdf(file, savename)
  }

########################

library(staplr)

ms1=list.files(path = "EIC_rt_during", pattern = "*.pdf$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)

for(file in ms1){

id=strsplit(file, split=".pdf")

ms1=paste0("EIC_rt_during/",id,".pdf")

ms2=paste0("ms2_figures_label/",id,".pdf")
```

```
 structure=paste0("structure_2d/",id,".pdf")

 gather=c(ms1, ms2, structure)

 savename=paste0("report_pdf/",file)

 staple_pdf(input_files=gather, output_filepath= savename)
 }


########################
###t.test and volcano


########################
### Pearson correlation coefficient



#########################
### linear regression



#########################
### volcano plot



######## network

library(tidyverse)
library(igraph)
library(ggraph)
library(tidygraph)
```

58 File: annotate_child_nebula.R

```
annotate_child_nebulae <-
  function(
           nebula_name,
           compound_class_list = .MCn.nebula_class,
           write_output = T,
           output = paste0(.MCn.output, "/", .MCn.results),
           layout = "fr",
           height = "auto",
           width = "auto",
           plot_nodes_id = T,
           plot_structure = T,
           plot_ppcp = T,
           ratio_df = NULL,
           merge_image = T,
           return_plot = F,
           ...
           ){
    cat("[INFO] MCnebula run: annotate_child_nebulae\n")
    ## ---------------------------------------------------------------------
    ## all nodes in graph
    nodes <- dplyr::filter(.MCn.nebula_index, name == nebula_name)$".id"
    ## get top compound class (nodes_color data)
    ## as well as, collate metadata
    metadata <- lapply(compound_class_list, head, n = 1) %>%
      data.table::rbindlist(idcol = T) %>% # as data.frame
      dplyr::filter(.id %in% nodes) %>% # filter via nodes
      dplyr::select(.id, name) %>%
      dplyr::rename(vis_class = name)
    ## push environment name into parent.env, let some data could be catch in sub-environment via 'get'
    assign("envir_meta", environment(), envir = parent.env(environment()))
    ## gather data for annotation (nebula_name, hierarchy)
    hierarchy <- dplyr::filter(.MCn.nebula_index, name == nebula_name) %>%
      head(n = 1)
    anno = c(nebula_index = nebula_name, hierarchy = hierarchy$hierarchy)
    ## ---------------------------------------------------------------------
    ## set a environment to store layout data
    envir_layout <- new.env()
    ## set to remove nodes or not (set to 0, remove)
    if(plot_ppcp == T | plot_structure == T){
      remove_nodes = 0
    }else{
      remove_nodes = NULL
    }
```

```
## plot origin network (child network, with legend)
p <- grid_child_nebula(
                       .MCn.child_graph_list[[nebula_name]],
                       anno = anno,
                       print_into = F,
                       layout = layout,
                       ## save layout data in this environment
                       save_layout_df = envir_layout,
                       ## remove origin nodes
                       remove_nodes = remove_nodes,
                       ...)
## ----------------------------------------------------------------------
## whether plot pie diagram
if(is.null(ratio_df) == F){
  if(is.data.frame(ratio_df) == T){
    plot_ratio = T
  }else{
    cat("is.data.frame(ratio_df) == F\n")
    plot_ratio = F
  }
}else{
  plot_ratio = F
}
## ------------------------------------------------------------------------
## tmp dir
tmp_dir <- paste0(output, "/", "tmp")
if(file.exists(tmp_dir) == F){
  dir.create(tmp_dir)
}
## add annotation ---------------------------------------------------------
## nodes id
if(plot_nodes_id == T & (plot_ppcp == F)){
  p <- p + ggraph::geom_node_text(aes(label = name), size = 1)
}
## add annotation ---------------------------------------------------------
## require ChemmineOB and ChemmineR
with_structure <- 0
if(requireNamespace("ChemmineOB", quietly = T)){
  ## structure
  tmp_stru <- paste0(tmp_dir, "/", "structure")
  if(file.exists(tmp_stru) == F){
    dir.create(tmp_stru)
  }
  if(plot_structure == T){
    with_structure <- 1
    batch_mode_structure(metadata = metadata, tmp_stru = tmp_stru)
  }
}
## add annotation ----------------------------------------------------------
## re draw nodes with or without ppcp bar
tmp_ppcp <- paste0(tmp_dir, "/", "ppcp")
if(file.exists(tmp_ppcp) == F){
  dir.create(tmp_ppcp)
}
if(plot_ppcp == T | plot_structure == T | plot_ratio == T){
  batch_mode_nodes(
                   metadata = metadata,
                   tmp_ppcp = tmp_ppcp,
                   with_structure = with_structure,
                   plot_ppcp = plot_ppcp,
                   plot_ratio = plot_ratio,
                   ratio_df = ratio_df,
                   ...)
}
## -------------------------------------------------------------------------
## merge image
if(merge_image == T){
  if(requireNamespace("ggimage", quietly = T) &
     requireNamespace("gridExtra", quietly = T)){
    ## remove legend of size
    p <- p + ggplot2::guides(size = "none")
    merge_image(p, envir_layout$layout_n, tmp_ppcp)
  }
}
## -------------------------------------------------------------------------
## write_output ## estimate width
if(write_output == T){
  if(height == "auto" | width == "auto"){
    ## estimate width upon legend number of 'fill'
    n = length(unique(metadata$vis_class))
    height = 8
    width = ifelse(n <= 17, 10, ## 'class' less than 17
                      ifelse(n <= 34, 12.5,
                             ifelse(n <= 51, 15, 18)))
    }
    ## output
    ggplot2::ggsave(p, file = paste0(output, "/", nebula_name, "_graph.svg"),
           width = width, height = height)
  }
  cat("[INFO] MCnebula Job Done: annotate_child_nebulae\n")
  if(return_plot == T){
    return(p)
  }
}
## function gather all subview
gather_subview <-
function(
         subview,
         x,
         y,
         width,
         height,
         p = get("p", envir = get("envir_meta"))
         ){
  p <- p + ggimage::geom_subview(x = x, y = y, width = width, height = height,
                         subview = subview)
  assign("p", p, envir = get("envir_meta"))
  return("Done")
  ##
}
## funtion merge image, involves nodes (may include ppcp bar), structure, and network layout (with edge
merge_image <-
function(
         p, ## ggplot2 object
         layout_n,
         tmp_ppcp,
         ...
         ){
  ## ----------------------------------------------------------------------
  ## check svg image
  df <- dplyr::select(layout_n, x, y, name, tanimotoSimilarity) %>%
    dplyr::mutate(nodes_path = paste0(tmp_ppcp, "/", name, ".svg"),
                  check_nodes = file.exists(nodes_path)) %>%
    dplyr::filter(check_nodes == T)
  cat("## read_cairo_svg:", nrow(df), "(number)\n")
  ## read svg image
  subview_list <- pbapply::pblapply(df$name, base_read_cairo,
                                    path = tmp_ppcp,
                                    ...)
  ## ---------------------------------------------------------------------
  ## calculate width and height for subview, according to attributes of tanimotoSimilarity
  df <- dplyr::mutate(df,
                      width = ifelse(is.na(tanimotoSimilarity) == T, 1,
                                     1 + tanimotoSimilarity),
                      height = width)
  ## ---------------------------------------------------------------------
  ## as subview
  cat("## Advance visualization: gather_subview\n")
  pbapply::pbmapply(
                    gather_subview, ## function
                    subview_list,
                    df$x,
                    df$y,
                    df$width,
                    df$height
  )
}
```

59 File: batch_mode_nodes.R

```
batch_mode_nodes <-
  function(
           metadata,
           tmp_ppcp,
           with_structure = 0,
           plot_ppcp = plot_ppcp,
           plot_ratio = F,
           ratio_df = NULL,
           ...
           ){
    ## remove exist files
    lapply(list.files(tmp_ppcp, full.names = T), file.remove)
    ## ---------------------------------------------------------------------
    ## nodes color, which parallel to the nodes color in plot of visualize_child_nebula function
```

```r
  meta_color <- dplyr::select(metadata, vis_class) %>%
    dplyr::distinct() %>%
    dplyr::arrange(vis_class)
  meta_color$nodes_color <- .MCn.palette[1:nrow(meta_color)]
  ## gather color data
  meta_ppcp <- merge(metadata, meta_color, by = "vis_class", all.x = T)
  ## ----------------------------------------------------------------------
  ## pick ppcp_dataset
  ppcp_dataset = .MCn.ppcp_dataset[which(names(.MCn.ppcp_dataset) %in% meta_ppcp$".id")]
  ## sort data
  meta_ppcp$".id" <- factor(meta_ppcp$".id",
                            levels = names(ppcp_dataset))
  meta_ppcp <- meta_ppcp[order(meta_ppcp$".id"), ]
  ## ----------------------------------------------------------------------
  ## ratio_df, extra peak area data
  if(plot_ratio == T){
    ratio_df <- dplyr::mutate(ratio_df, .id = as.character(.id))
    ratio_df <- merge(dplyr::select(meta_ppcp, .id), ratio_df, all.x = T, by = ".id", sort = F)
    assign("envir_nebula", environment(), parent.env(environment()))
    ## get list data
    ratio_df_list <- lapply(ratio_df$".id", by_group_for_list,
                            df = get("ratio_df", envir = get("envir_nebula")),
                            col = ".id")
  }else{
    ratio_df_list <- rep(0, nrow(meta_ppcp))
  }
  ## ----------------------------------------------------------------------
  cat("## annotate_child_nebulae: batch_mode_nodes\n")
  pbapply::pbmapply(
                  base_vis_nodes, # function
                  ppcp_dataset, # main 1
                  meta_ppcp$nodes_color, # main 2
                  names(ppcp_dataset), # main 3, key_id
                  ratio_df_list, # main 4, draw pie diagram
                  MoreArgs = list(
                                path = normalizePath(tmp_ppcp),
                                with_structure = with_structure,
                                plot_ppcp = plot_ppcp,
                                plot_ratio = plot_ratio,
                                ...))
}
```

```r
## function mannually draw nodes
base_vis_nodes <-
  function(
           ppcp, ## main 1
           nodes_color, ## main 2
           key_id = NULL, ## main 3
           ratio_df = NULL, ## main 4, draw pie diagram
           plot_ratio = F,
           plot_nodes_id = T,
           plot_ppcp = T,
           label_color = "black",
           with_structure = 0,
           path = ".",
           class_index = unique(.MCn.nebula_index$relativeIndex),
           palette = colorRampPalette(c(.MCn.palette))(length(class_index)),
           palette_stat = .MCn.palette_stat,
           size_adjust = 0.7
           ){
    ## ---------------------------------------------------------------------
    ## filter via class_index
    ppcp <- ppcp[ppcp$relativeIndex %in% class_index, ]
    ## as factor, for painting color
    ppcp$relativeIndex <- factor(ppcp$relativeIndex, levels = sort(ppcp$relativeIndex))
    ppcp$num <- seq(1, nrow(ppcp))
    if(plot_ppcp == F){
      ppcp$V1 = 0
    }
    ## ---------------------------------------------------------------------
    ## plot nodes
    p <- ggplot(ppcp, aes(x = num, y = V1)) +
      ## nodes color
      ggplot2::geom_ribbon(fill = nodes_color,
                           aes(ymin = -5, ymax = 0,
                               x = ifelse(num == 1, 0,
                                          ifelse(num == nrow(ppcp), num + 1, num)))) +
      ## border color
      ggplot2::geom_ribbon(fill = "black",
                           aes(ymin = 0, ymax = 1.1,
                               x = ifelse(num == 1, 0,
                                          ifelse(num == nrow(ppcp), num + 1, num)))) +
        ## ppcp bar plot
        ggplot2::geom_col(alpha = 1, aes(fill = relativeIndex), color = "white", size = 0.25) +
        ## nodes border ratio
        ggplot2::ylim(-5, 1.3) +
        ## Polar coordinate transformation
        ggplot2::coord_polar() +
        ggplot2::theme_minimal() +
        ggplot2::theme(
              text = element_text(family="Times"),
              axis.ticks = element_blank(),
              axis.text = element_blank(),
              axis.title = element_blank(),
              panel.grid = element_blank(),
              ## remove legend
              legend.position = "none",
              panel.border = element_blank(),
              plot.margin =unit(c(0,0,0,0),"cm"),
              panel.spacing =unit(c(0,0,0,0),"cm")
        )
    ## ---------------------------------------------------------------------
    ## draw pie diagram
    if(plot_ratio == T){
      ratio_df <- reshape2::melt(ratio_df, id.vars = ".id", variable.name = "group", value.name = "valu
      ## value stack
      ratio_df <- dplyr::mutate(ratio_df,
                                xend = stack_sum(ratio_df$value),
                                x = stack_sum(c(0, ratio_df$value[1:(nrow(ratio_df)-1)])))
      ## normalize x axis range and x value
      n_factor = (max(ppcp$num) + 1)/max(ratio_df$xend)
      ratio_df <- dplyr::mutate(ratio_df,
                         midd = (x + value/2) * n_factor,
                         width = value * n_factor)
      ## add pie plot into ggplot2 project
      p <- p + ggplot2::geom_tile(data = ratio_df, size = 0.2, color = "white",
                                  aes(y = -2.5, x = midd, width = width, height = 2.5, fill = group)) +
        ## add 'fill' palette
        ggplot2::scale_fill_manual(values = c(palette, palette_stat[1:nrow(ratio_df)]))
    }else{
      ## add 'fill' palette
      p <- p + ggplot2::scale_fill_manual(values = palette)
    }
    ## ---------------------------------------------------------------------
```

```
    ## generate Graphics Device
    savepath = paste0(path, "/", key_id, ".svg")
    svglite::svglite(savepath, bg = "transparent")
    ## print nodes
    print(p)
    ## ----------------------------------------------------------------------
    ## print structure or not
    if(with_structure == 1){
      s_file = paste0(normalizePath(paste0(path, "/../structure")), "/", key_id, ".svg")
      if(file.exists(s_file)){
        ## via grImport2 import Cairo svg
        ps <- grImport2::readPicture(file = s_file)
        ## grid draw
        grImport2::grid.picture(ps, width = size_adjust, height = size_adjust)
      }
    }
    ## ----------------------------------------------------------------------
    ## grid nodes ID in nodes
    if(plot_nodes_id == T){
      ## a grid object
      ps <- grid::textGrob(paste0("ID:", key_id),
                           y = 0.25,
                           gp = grid::gpar(fontfamily = "Times", fontsize = 20, col = label_color))
      grid::grid.draw(ps)
    }
    ## ----------------------------------------------------------------------
    dev.off()
    # as cairo svg
    rsvg::rsvg_svg(savepath, savepath)
    ## ----------------------------------------------------------------------
  }
## function read cairo svg
base_read_cairo <-
  function(
           key_id,
           path,
           suffix = ".svg"
           ){
    prefix <- c()
    svg <- grImport2::grobify(grImport2::readPicture(paste0(path, "/", key_id, suffix)))
    svg <- gridExtra::arrangeGrob(svg)
    return(svg)
  }
stack_sum <-
  function(
           vector
           ){
    stack <- c()
    for(i in 1:length(vector)){
      stack[i] <- sum(vector[1:i])
    }
    return(stack)
  }
```

60 File: batch_mode_structure.R

```
batch_mode_structure <-
  function(
           metadata,
           tmp_stru
           ){
    ## ---------------------------------------------------------------------
    ## collate metadata
    meta_stru <- dplyr::mutate(metadata,
                               stru_file = paste0(tmp_stru, "/", .id, ".svg"),
                               stru_check = file.exists(stru_file))
    meta_stru <- merge(meta_stru, .MCn.structure_set[, c(".id", "smiles")], by = ".id", all.x = T)
    meta_stru <- dplyr::filter(meta_stru, is.na(smiles) == F)
    cat("## STAT of structure set:",
        paste0(nrow(meta_stru), "(compounds with structure)", "/", nrow(metadata), "(all compounds)"),
        "\n")
    meta_stru <- dplyr::filter(meta_stru, stru_check == F)
    ## ---------------------------------------------------------------------
    if(nrow(meta_stru) > 0){
      pbapply::pbmapply(
                        base_vis_structure, # function
                        meta_stru$".id", # key_id
                        meta_stru$smiles, # smiles
                        MoreArgs = list( path = normalizePath(tmp_stru) ))
    }
  }
## function draw structure
```

```
base_vis_structure <-
  function(
           key_id,
           smiles,
           path,
           to_file = paste0(path, "/", key_id, ".svg")
           ){
    ## openbabal. only support for linux
    ChemmineOB::convertToImage("SMI", "SVG", source = smiles, toFile = to_file)
    ## as transparent bg
    svg <- data.table::fread(file = to_file, sep = "", quote ="", header = F)
    svg$V1 = sub('fill="white"', 'fill="transparent"', svg$V1)
    write.table(x = svg, file = to_file, sep = "", col.names = F, row.names = F, quote = F)
    ## convert as cairo svg
    rsvg::rsvg_svg(to_file, to_file)
    return("Done")
  }
```

61 File: build_classes_tree_list.R

```
build_classes_tree_list <-
  function(
           class_index="canopus.tsv", path=.MCn.sirius
           ){
    data <- read_tsv(paste0(path, "/", class_index))
    ## ---------------------------------------------------------------------
    ## separate each levels of classes into sub-list
    root <- data[which(data$parentId==""), ]
    list <- list()
    n = 1
    list[[n]] <- root %>% dplyr::as_tibble()
    df <- data[data$parentId %in% root$id, ]
    ## ---------------------------------------------------------------------
    while(nrow(df) > 0){
      n = n + 1
      list[[n]] <- df %>% dplyr::as_tibble()
      df <- data[data$parentId %in% df$id, ]
    }
    .MCn.class_tree_list <<- list
    cat("INFO: Classification Index in.MCn.sirius project --->", class_index, "\nA total of 11 levels.
        Use following arguments to get some specific classes:
```

```
        .MCn.class_tree_list[[3]] >>> superclass
        .MCn.class_tree_list[[4]] >>> class
        .MCn.class_tree_list[[5]] >>> subclass
        .MCn.class_tree_list[[6]] >>> level 5 \n")
}
```

62 File: collate_ppcp.R

```
## main function
collate_ppcp <-
  function(
           formula_adduct = .MCn.formula_set,
           path = .MCn.sirius,
           dirs = "all",
           output = paste0(.MCn.output, "/", .MCn.results),
           write_output = T,
           nebula_class = T,
           nebula_index = T,
           ...
           ){
    cat("[INFO] MCnebula run: collate_ppcp\n")
    ## ----------------------------------------------------------------------
    ## check dirs ---- canopus
    cat("## collate_ppcp: check_dir\n")
    if(dirs == "all"){
      dirs <- list.files(path = path, pattern="^[0-9](.*)_(.*)_(.*)$", full.names = F)
      check <- pbapply::pbsapply(dirs, check_dir, file = "canopus") %>% unname
    }else{
      check <- pbapply::pbsapply(dirs, check_dir, file = "canopus") %>% unname
    }
    ## ----------------------------------------------------------------------
    ## lock on file location
    meta_dir <- dirs[which(check == T)] %>%
      data.frame() %>%
      dplyr::rename(dir = ".") %>%
      dplyr::mutate(.id = sapply(dir, grep_id)) %>%
      merge(formula_adduct, by = ".id", all.x = T, sort = F) %>%
      dplyr::mutate(adduct_trans = gsub(" ", "", adduct),
             target = paste0(precursorFormula, "_", adduct_trans, ".fpt"),
             full.name = paste0(path, "/", dir, "/", "canopus", "/", target),
             ## these files need to be check and filter (whether exist)
```

```
          ## note that some formula is no fingerprint computed
          ppcp = file.exists(full.name))
    meta_dir_filter <- dplyr::filter(meta_dir, ppcp == T)
    cat("## STAT of PPCP dataset:",
        paste0(nrow(meta_dir_filter), "(formula with PPCP)", "/", nrow(meta_dir), "(all formula)"),
        "\n")
  ## ---------------------------------------------------------------------
  ## load all ppcp dataset
  ppcp_dataset <- pbapply::pblapply(meta_dir_filter$full.name, read_fpt)
  names(ppcp_dataset) <- meta_dir_filter$".id"
  .MCn.ppcp_dataset <<- ppcp_dataset
  ## ---------------------------------------------------------------------
  ## summarize nebula_class
  if(nebula_class == T){
    cat("# Collate_ppcp |", date(), "| USE Method: method_summarize_nebula_class.\n")
    metadata <- data.table::rbindlist(.MCn.class_tree_list, idcol = T) %>%
      dplyr::rename(hierarchy = .id)
    .MCn.class_tree_data <<- dplyr::as_tibble(metadata)
    ## transmit environment
    assign("envir_meta", environment(), envir = parent.env(environment()))
    ## get nebula classes
    nebula_class <- pbapply::pblapply(ppcp_dataset, method_summarize_nebula_class,
                             class_data_type = "classes_tree_data",
                             ...)
    .MCn.nebula_class <<- nebula_class
  }
  ## ---------------------------------------------------------------------
  if(nebula_index == T){
    cat("# Collate_ppcp |", date(), "| USE Method: method_summarize_nebula_index.\n")
    ## gather all nebula classes
    nebula_index <- method_summarize_nebula_index(ppcp_dataset,
                                               ...)
    .MCn.nebula_index <<- nebula_index
  ## ---------------------------------------------------------------------
    if(write_output == T){
      write_tsv(nebula_index, file = paste0(output, "/", "nebula_index.tsv"))
    }
  }
  cat("[INFO] MCnebula Job Done: collate_ppcp.\n")
  return(nebula_index)
}
```

63 File: collate_structure.R

```
## main function
collate_structure <-
  function(
           dirs = "all",
           path = .MCn.sirius,
           output = paste0(.MCn.output, "/", .MCn.results),
           write_output = T,
           write_picked_formula_adduct = T,
           collate_method = "method_pick_formula_excellent", # "top_score", "top_similarity", "top_zodi
           ...
           ){
  cat( paste0("[INFO] MCnebula run: collate_structure\n") )
  ## ---------------------------------------------------------------------
  ## check dirs
  cat("## collate_structure: check_dir\n")
  if(dirs == "all"){
    dirs <- list.files(path = path, pattern="^[0-9](.*)_(.*)_(.*)$", full.names = F)
    check <- pbapply::pbsapply(dirs, check_dir) %>% unname
  }else{
    check <- pbapply::pbsapply(dirs, check_dir) %>% unname
  }
  dirs <- dirs[which(check == T)]
  ## build a new envir to place data
  formula_cache <- new.env()
  structure_cache <- new.env()
  ## ---------------------------------------------------------------------
  cat("## collate_structure:", paste0(collate_method), "\n")
  method_fun <- match.fun(collate_method)
  ## method
  pbapply::pblapply(dirs, method_fun,
                    return_formula = F,
                    ## the data are placed into cache envir
                    formula_cache = formula_cache,
                    structure_cache = structure_cache,
                    ...)
  ## ---------------------------------------------------------------------
  ## structure collate
  structure_dataset <- eapply(structure_cache, data.table) %>%
    data.table::rbindlist(idcol=T)
  .MCn.structure_set <<- dplyr::mutate(structure_dataset,
                                        tanimotoSimilarity = as.numeric(tanimotoSimilarity)) %>%
    dplyr::as_tibble()
  if(write_output == T){
    write_tsv( structure_dataset, paste0(output, "/", collate_method, ".structure.tsv"))
  }
  ## formula_adduct collate
  formula_adduct_set <- eapply(formula_cache, data.table) %>%
    data.table::rbindlist(idcol = T)
  .MCn.formula_set <<- dplyr::filter(formula_adduct_set, is.na(precursorFormula) == F)
  if(write_picked_formula_adduct == T){
    write_tsv(formula_adduct_set, paste0(output, "/picked_", collate_method, ".tsv"))
  }
  ## ----------------------------------------------------------------
  cat( paste0("[INFO] MCnebula Job Done: collate_structure.\n") )
}
## extra function in this module
## ----
grep_id <- function(x, sep = "_"){
  v <- unlist(strsplit(x, split="_"))
  id <- v[length(v)]
  return(id)
}
## ----
check_dir <- function(dir, path = .MCn.sirius, file = "compound.info"){
  if(file.exists(paste0(path, "/", dir, "/", file)) == T){
    check = T
  }else{
    check = F
  }
  return(check)
}
```

64 File: generate_child_nebulae.R

```
generate_child_nebulae <-
  function(
           nodes = .MCn.parent_nodes,
           edges = .MCn.parent_edges,
           max_edges = 5,
           nebula_index = .MCn.nebula_index,
           output = paste0(.MCn.output, "/", .MCn.results),
          output_format = "graphml",
          ...
          ){
    ##
    cat("[INFO] MCnebula run: generate_child_nebulae\n")
    assign("envir_nebula", environment(), envir = parent.env(environment()))
    ## get names of all classes
    names <- unique(nebula_index$name)
    ## for using lapply, first, trans the data.frame into list
    nebula_index <- lapply(unique(nebula_index$relativeIndex), by_group_for_list)
    ## push names
    names(nebula_index) <- names
    ## facet parent nebula
    ## create dir for placing graph file
    dir = paste0(output, "/", "child_nebula")
    if(file.exists(dir) == F){
      dir.create(dir)
    }
    .MCn.child_graph_list <<- pbapply::pblapply(nebula_index, separate_nebula,
                      output = dir,
                      max_edges = max_edges,
                      ...)
    cat("[INFO] MCnebula Job Done: generate_child_nebulae\n")
  }
by_group_for_list <-
  function(
          x,
          df = get("nebula_index", envir = get("envir_nebula")),
          col = "relativeIndex"
          ){
    df <- df[which(df[[col]] == x),]
    return(df)
  }
separate_nebula <-
  function(
          df,
          nodes = get("nodes", envir = get("envir_nebula")),
          edges = get("edges", envir = get("envir_nebula")),
          write_output = T,
          output = paste0(.MCn.output, "/", .MCn.results, "/", "child_nebula"),
          output_format = "graphml",
           max_edges = 5,
           write_extra = F
           ){
    id <- unique(df$".id")
    ## get the child nebula name
    name <- df[1, "name"]
    nodes <- nodes[nodes$".id" %in% id, ]
    edges <- edges[edges$".id_1" %in% id & edges$".id_2" %in% id, ]
    ## an edges number cut-off
    edges <- better_vis_nebula(edges, max_edges = max_edges)
    child_nebula <- igraph::graph_from_data_frame(edges, directed = T, vertices = nodes)
    if(write_output == T){
      write_graph(child_nebula,
                  file = paste0(output, "/", name, ".", output_format),
                  format = output_format)
      if(write_extra == T){
        write_tsv(edges, paste0(output, "/", name, "_edges.tsv"))
        write_tsv(nodes, paste0(output, "/", name, "_nodes.tsv"))
      }
    }
    return(child_nebula)
  }
better_vis_nebula <-
  function(
           edges,
           max_edges = 5
           ){
    ## order
    edges <- dplyr::arrange(edges, desc(edges[,3]))
    ta <- table(c(edges[[1]], edges[[2]]))
    ## at least loop number
    n <- length(which(ta > max_edges))
    if(n == 0){
      return(edges)
    }
    ## ----------------------------------------------------------------------
    ## copy data for override
    df <- dplyr::mutate(edges[, 1:2], SEQ = 1:nrow(edges))
    assign("envir_meta", environment(), envir = parent.env(environment()))
    ## use sapply instead of while loop
    continue = 1
```

```r
    sapply(1:n, edges_cut_off, max_edges = max_edges)
    ## ---------------------------------------------------------------------
    edges <- edges[df$SEQ, ]
    return(edges)
  }
edges_cut_off <-
  function(
           i,
           max_edges = 5
           ){
    continue = get("continue", envir = get("envir_meta"))
    if(continue == 1){
      edges = get("df", envir = get("envir_meta"))
      ## stat edges number of an id
      ta <- table(c(edges[[1]], edges[[2]]))
      ## ---------------------------------------------------------------------
      if(max(ta) > max_edges){ ## greater than threshold, hence perform exclude
        ## select an id to exclude its excess edges
        key_id <- names(ta[which(ta == max(ta))])[1]
        ## get SEQ of the edges which need to be excluded
        incude_id_edges <- edges[which(edges[[1]] == key_id | edges[[2]] == key_id),]
        exclude_edges_seq <- incude_id_edges[-(1:max_edges), ]$SEQ
        ## exclude edges
        edges <- edges[which(!edges$SEQ %in% exclude_edges_seq), ]
        assign("df", edges, envir = get("envir_meta"))
      }else{
      ## ---------------------------------------------------------------------
        ## signature of stop exclude
        assign("continue", 0, envir = get("envir_meta"))
      }
    }else{
      return()
    }
  }
```

65 File: generate_parent_nebula.R

```r
generate_parent_nebula <-
  function(
           write_output = T,
           output_format = "graphml",
           output = paste0(.MCn.output, "/", .MCn.results),
           edges_file = paste0(output, "/parent_nebula/parent_nebula_edges.tsv"), # exists edges file
           edges_method = "method_formula_based_spec_compare", # or NULL
           nodes_attributes = .MCn.formula_set,
           nodes_other_attributes = .MCn.structure_set,
           edge_filter = 0.5,
           cpu_cores = 8,
           ...
           ){
    cat("[INFO] MCnebula run: generate_parent_nebula\n")
    ## main body
    ## ---------------------------------------------------------------------
    ## generate edges data
    if(is.null(edges_method) == T){
      ## no edges_method
      cat("# generate_parent_nebula: no edges_uethod used\n")
      edges <- dplyr::as_tibble(cbind(".id_1" = nodes_other_attributes$".id",
                     ".id_2" = nodes_other_attributes$".id")) %>%
        dplyr::mutate(dotproduct = 1, mass_diff = 0)
    }else if(edges_method == "method_formula_based_spec_compare"){
      ## with edges_method
      if(is.null(edges_file) == F & file.exists(edges_file)){
        cat("# generate_parent_nebula: file.exists(edges_file) == T. Escape from time-consuming computat
        edges <- read_tsv(edges_file) %>%
          dplyr::mutate_at(c(".id_1", ".id_2"), as.character) %>%
          dplyr::mutate_at(c(colnames(edges)[3:4]), as.numeric)
      }else{
        edges = method_formula_based_spec_compare(edge_filter = edge_filter, cpu_cores = cpu_cores, ...
      }
    }
    ## ---------------------------------------------------------------------
    ## generate nodes data
    nodes <- nodes_attributes
    ## additional nodes attributes
    if(is.null(nodes_other_attributes) == F){
      nodes <- merge(nodes, nodes_other_attributes, by = ".id", all.x = T, sort = T) %>%
        ## rename the column name, otherwise the column will be choosed as key column in igraph
        dplyr::rename(compound_name = name)
    }
    ## ---------------------------------------------------------------------
    ## graph
```

```r
    parent_nebula <- igraph::graph_from_data_frame(edges, directed = T, vertices = nodes)
    if(write_output == T){
      dir = paste0(output, "/", "parent_nebula")
      if(file.exists(dir) == F){
        dir.create(dir)
      }
      write_graph(parent_nebula,
                  file = paste0(dir, "/", "parent_nebula.", output_format),
                  format = output_format)
      write_tsv(edges, paste0(dir, "/", "parent_nebula_edges.tsv"))
      write_tsv(nodes, paste0(dir, "/", "parent_nebula_nodes.tsv"))
    }
    ## ---------------------------------------------------------------------
    ## set as global var for next stage
    .MCn.parent_graph <<- parent_nebula
    .MCn.parent_nodes <<- nodes %>% as_tibble()
    .MCn.parent_edges <<- edges %>% as_tibble()
    cat("[INFO] MCnebula Job Done: generate_parent_nebula\n")
  }
```

66 File: get_formula.R

```r
get_formula <-
  function(
           key_id,
           exclude_element = NULL, ## e.g., c("S", "B", "P", "Si")
           formula_method = "top_zodiac",
           rank = 1:5, # or "all"
           ppm_error = 20,
           return_col = c("rank", "precursorFormula", "molecularFormula",
                          "adduct", "ZodiacScore", "massErrorPrecursor(ppm)"),
           ...
           ){
    path <- list.files(path = .MCn.sirius, pattern=paste0("*_", key_id, "$"), full.names=T)
    file <- read_tsv(paste0(path, "/", "formula_candidates.tsv"))
    file$rank <- as.numeric(file$rank)
    ## ---------------------------------------------------------------------
    if("ZodiacScore" %in% colnames(file) == F){
      file$ZodiacScore = 0
    }
    ## ---------------------------------------------------------------------
    if(is.null(exclude_element) == F){
      file <- file[!unname(sapply(file$precursorFormula, grep_element,
                                  exclude_element = exclude_element)), ]
    }
    ## ---------------------------------------------------------------------
    if(formula_method == "top_zodiac"){
      if(rank[1] == "all"){
        rank <- unique(file$rank)
      }
      df <- file[which(file$rank %in% rank & abs(file$"massErrorPrecursor(ppm)") <= ppm_error), c(retur
    }
    return(df)
}
## ---------------------------------------------------------------------
grep_element <-
  function(
           formula,
           exclude_element = c("S", "P", "B")
           ){
    if(length(grep(paste(exclude_element, collapse = "|"), formula)) == 1){
      return(T)
    }else{
      return(F)
    }
  }
```

67 File: get_ppcp.R

```
get_ppcp <-
  function(
           key_id = NULL,
           dir = NULL,
           precursor_formula = "method_pick_formula_excellent",
           adduct = NULL,
           reformat = T,
           filter = T,
           filter_threshold = 0.1,
           class_index = "canopus.tsv",
           ...
           ){
    ## get dir path
    if(is.null(dir) == T & is.null(key_id) == T){
      return()
    }else if(is.null(dir) == T){
      dir <- get_dir(key_id)
    }
    ## ---------------------------------------------------
    ## aquire formula via the method
    if( precursor_formula == "method_pick_formula_excellent" ){
      meta <- method_pick_formula_excellent(dir = dir)
      precursor_formula <- meta$precursorFormula
      adduct <- meta$adduct
    }
    ## ---------------------------------------------------
    ## read ppcp data
    file <- list.files(path = paste0(.MCn.sirius, "/", dir, "/", "canopus"),
                       pattern = paste0("^", precursor_formula, "(.*)", escape_ch(adduct), "(.*)", ".fp
                       full.names = T)
    ppcp <- read_fpt(file)
    ## ---------------------------------------------------
    ## reformat section
    if(reformat == F){
      return(ppcp)
    }
    ## check meta list
    if(exists(".MCn.class_tree_list") == F){
      build_classes_tree_list(class_index = class_index)
    }
    ## get the environment name for lapply function to invoke data
    assign(paste0("envir_", key_id), environment(), envir = parent.env(environment()))
    ## merge with meta table, and filter
    ppcp <- lapply(.MCn.class_tree_list, merge_class_ppcp,
                   ## parameter
                   key_id = key_id, filter = filter, filter_threshold = filter_threshold)
    return(ppcp)
  }
## a small function to get data of ppcp
read_fpt <- function(file){
  fpt = data.table::fread(input = file, header = F, quote = "")
  fpt$relativeIndex = seq(0, nrow(fpt) - 1)
  return(fpt)
}
```

```r
## specific character in adduct description need to be revise, for pattern matching
escape_ch <- function(x){
  x <- gsub("\\[", "\\\\\\[", x)
  x <- gsub("\\]", "\\\\\\]", x)
  x <- gsub("\\+", "\\\\\\+", x)
  x <- gsub("\\-", "\\\\\\-", x)
  x <- gsub(" ", "", x)
  return(x)
}
## the function to merge raw ppcp with meta list
merge_class_ppcp <-
  function(
           class,
           filter = T,
           filter_threshold = 0.1,
           key_id = NULL,
           values = get("ppcp", envir = get(paste0("envir_", key_id))),
           filter_col = "V1"
           ){
    df <- merge(class, values, all.x = T, by = "relativeIndex", sort = F)
    df <- df[which(df[[filter_col]] > ifelse(filter == T, filter_threshold, 0)),] %>%
      dplyr::as_tibble()
    return(df)
  }
```

68 File: get_structure.R

```r
get_structure <-
  function(
           key_id,
           precursor_formula = NULL,
           adduct = NULL,
           structure_method = "top_score", # or top_similarity
           order = T,
           return_row = 1:10, # or "all"
           path = .MCn.sirius,
           as_tibble = F,
           ...
           ){
    path <- list.files(path = path, pattern = paste0("*_", key_id, "$"), full.names = T)
    files <- list.files(paste0(path, "/fingerid"),
                        pattern = paste0(precursor_formula, "(.*)", escape_ch(adduct), "(.*).tsv$"),
                        full.names = F)
    ## ---------------------------------------------------
    ## read file
    list <- lapply(paste0(path, "/fingerid/", files), read_tsv)
    names(list) <- gsub(".tsv", "", files)
    ## ---------------------------------------------------
    ## reformat the data
    if(order == T){
      ## bind row as data frame
      df <- data.table::rbindlist(list, idcol = T)
      colnames(df)[which(colnames(df) == ".id")] <- "file_name"
      if(nrow(df) == 0){
        return(df)
      }
      ## ---------------------------------------------------
      ## order upon CSI:fingerID score
      if(structure_method == "top_score"){
        df <- df[order(-df$score),]
        df$structure_rank = as.numeric(1:nrow(df))
      ## order upon tanimoto similarity
      }else if(structure_method == "top_similarity"){
        df <- df[order(-df$similarity),]
        df$structure_rank = as.numeric(1:nrow(df))
      }
      ## ---------------------------------------------------
      if(as_tibble == T){
        df <- dplyr::as_tibble(df)
      }
      # return with top n
      if(return_row[1] != "all"){
        return(df[which(1:nrow(df) %in% return_row),])
      }else{
        return(df)
      }
    }
    return(list)
}
```

69 File: initialize_mcnebula.R

```r
initialize_mcnebula <-
  function(
           sirius_path,
           output_path = sirius_path,
           output_file = "mcnebula_results",
           palette = unique(c(ggsci::pal_simpsons()(16), ggsci::pal_igv("default")(51))),
           palette_stat = palette,
           palette_label = colorRampPalette(c("#C6DBEFFF", "#3182BDFF", "red"))(10),
           rm_var = F
           ){
    if(rm_var == T){
      ls(envir = .GlobalEnv, pattern = "^.MCn.(.*)", all.names = T) %>%
        rm(envir = .GlobalEnv)
    }
    if(file.exists(sirius_path)==F | file.exists(output_path)==F){
      cat("File path not find.\n")
      return()
    }
    if(file.exists(paste0(sirius_path, "/", ".format"))==F){
      cat("SIRIUS project not find.\n")
      return()
    }
    .MCn.sirius <<- sirius_path
    .MCn.output <<- output_path
    .MCn.results <<- output_file
    .MCn.palette <<- palette
    .MCn.palette_stat <<- palette_stat
    .MCn.palette_label <<- palette_label
    dir.create(paste0(.MCn.output, "/", .MCn.results))
    cat("MCnebula project has initialized at ->", .MCn.output, "\n")
  }
```

70 File: method_formula_based_spec_compare.R

```r
## main function
method_formula_based_spec_compare <-
  function(
           path = .MCn.sirius,
           dirs = "all",
           cpu_cores = 8,
           compare_fun = "dotproduct",
           precursor_mass_diff = T,
           edge_filter = 0.3,
           only_identical_class = T, ## only identical classes will be compared (according to results o
           min_hierarchy = 5, # only hierarchy >= 5 (class, subclass...) will be considered)
           filter_only_max = 2000, # only nodes number >= 2000, do zoidacScore and tanimotoSimilarity f
           min_zodiac = 0.9, # only ZodiacScore >= 0.5 ...
           min_tanimoto = 0.4, # only the top structure tanimotoSimilarity >= 0.4 ...
           ...
           ){
    ## ------------------------------------------------------------------------------------------
    ## check dirs ---- spectra
    if(dirs == "all"){
      dirs <- list.files(path = path, pattern="^[0-9](.*)_(.*)_(.*)$", full.names = F)
      cat("## method_formula_based_spec_compare: check_dir\n")
      check <- pbapply::pbsapply(dirs, check_dir, file = "spectra") %>% unname
    }else{
      check <- pbapply::pbsapply(dirs, check_dir, file = "spectra") %>% unname
    }
    ## ------------------------------------------------------------------------------------------
    ## lock on file location
    meta_dir <- dirs[which(check == T)] %>%
      data.frame() %>%
      dplyr::rename(dir = ".") %>%
      dplyr::mutate(.id = sapply(dir, grep_id)) %>%
      merge(.MCn.formula_set, by = ".id", all.x = T) %>%
      merge(.MCn.structure_set[, c(".id", "tanimotoSimilarity")], by = ".id", all.x = T)
    ## some .id were Avoid time-consuming calculation
    if(nrow(meta_dir) >= filter_only_max){
      meta_dir <- dplyr::filter(meta_dir, ZodiacScore >= min_zodiac, tanimotoSimilarity >= min_tanimoto
    }
    meta_dir <- dplyr::mutate(meta_dir, adduct_trans = gsub(" ", "", adduct),
                              target = paste0(precursorFormula, "_", adduct_trans, ".tsv"),
                              full.name = paste0(path, "/", dir, "/", "spectra", "/", target),
                              ## these files need to be check and filter (whether exist)
                              spectra = file.exists(full.name))
    meta_dir_filter <- dplyr::filter(meta_dir, spectra == T)
    cat("## STAT of spectra dataset:",
        paste0(nrow(meta_dir_filter), "(formula with match spectra)", "/", nrow(meta_dir), "(all formula
        "\n")
```

```
## ---------------------------------------------------------------------------------------------
## load all spectra dataset
spectra_cache <- new.env()
pbapply::pbmapply(read_as_spectrum2, # function
                  meta_dir_filter$full.name,
                  meta_dir_filter$".id",
                  MoreArgs = list(
                                  cache = spectra_cache
                                  ))
## ---------------------------------------------------------------------------------------------
## enumeration combination
if(only_identical_class == T){
  ## enumeration combination in each hierarchy
  combn <- dplyr::filter(.MCn.nebula_index,
                         hierarchy >= min_hierarchy,
                         .id %in% meta_dir_filter$".id") %>%
    dplyr::group_by(hierarchy) %>%
    ## dispose in each group
    dplyr::summarise_at(c(".id"), unique) %>%
    dplyr::summarise_at(c(".id"), sort) %>%
    ## enumerate possible
    dplyr::summarise_at(c(".id"), combn_edge) %>%
    dplyr::ungroup() %>%
    dplyr::select(.id) %>%
    dplyr::distinct()
  ## this column has two sub-colunm
  combn <- dplyr::as_tibble(combn$".id")
## ---------------------------------------------------------------------------------------------
## this cost too much time !!!
}else{
  combn <- combn_edge(meta_dir_filter$".id")
}
## ---------------------------------------------------------------------------------------------
## compareSpectra (ms2) (via MSnbase)
cat("## Method part: compare_spectra: sum:", nrow(combn), "\n")
combn[[compare_fun]] <- pbapply::pbapply(combn, 1, couple_ms2_compare,
                                         fun = compare_fun,
                                         cl = cpu_cores,
                                         cache = spectra_cache,
                                         ...)
## filter via spectra similarity (edge_filter)
```

```
    ## ------------------------------------------------------------------------------------------
    combn <- combn[which(combn[[compare_fun]] >= edge_filter), ]
    if(precursor_mass_diff == T){
      ## dir
      info_path = paste0(path, "/", meta_dir_filter$dir, "/", "compound.info")
      ## load file
      cat("## Method part: load compound info file\n")
      meta_dir_filter$compound_mass <- pbapply::pblapply(info_path, get_precursor_mass) %>%
        unlist()
      ## transmit the environment name to parent.env for lapply
      assign("envir_spectra", environment(), envir = parent.env(environment()))
      ## compute mass difference
      cat("## Method part: diff_precursor_mass: sum:", nrow(combn), "\n")
      combn[["mass_diff"]] <- pbapply::pbapply(combn[,1:2], 1, precursor_mass_diff)
    }
    combn <- dplyr::as_tibble(combn)
    return(combn)
    ## ------------------------------------------------------------------------------------------
  }
read_as_spectrum2 <-
  function(filename,
           key_id,
           cache = spectra_cache){
    file <- read_tsv(filename)
    file <- new("Spectrum2", mz = file$mz, intensity = file$rel.intensity)
    assign(paste0(key_id), file, envir = cache)
    return()
  }
combn_edge <-
  function(x){
    combn <- combn(x, 2)
      combn <- t(combn)
      combn <- data.frame(combn)
      colnames(combn) <- c(".id_1", ".id_2")
      return(combn)
  }
couple_ms2_compare <-
  function(x,
           fun = "dotproduct",
           cache = spectra_cache,
           ...){
```

```
    simi <- MSnbase::compareSpectra(get(x[1], envir = cache),
                                    get(x[2], envir = cache),
                                    fun = fun,
                                    ...)
    return(simi)
  }
get_precursor_mass <-
  function(path){
    df <- read_tsv(path)
    mass <- df[which(df$index == "ionMass"), 2]
    mass <- as.numeric(mass)
    return(mass)
  }
precursor_mass_diff <-
  function(x, df = get("meta_dir_filter", envir = get("envir_spectra"))){
    ##
    x1 = df[which(df$".id" == x[1]), "compound_mass"]
    x2 = df[which(df$".id" == x[2]), "compound_mass"]
    x = x2 - x1
    return(x)
  }
```

71 File: method_pick_formula_excellent.R

```
method_pick_formula_excellent <-
  function(
           dir = NULL,
           key_id = NULL,
           return_formula = T,
           return_structure = F,
           formula_cache = NULL,
           structure_cache = NULL,
           exclude_element = NULL,
           ppm_error = 20,
           fc = 1.5,
           formula_info = c("precursorFormula", "molecularFormula", "adduct", "ZodiacScore")
           ){
    ## check parameter
    if(is.null(key_id) == T & is.null(dir) == T){
      return(0)
    }
    if(is.null(key_id) == F){
      dir = get_dir(key_id)
    }else{
      key_id = grep_id(dir)
    }
    ## ------------------------------------------------------------------
    ## first, test formula
    ## check whether.MCn.sirius compute formula for this id
    if(file.exists(paste0(.MCn.sirius, "/", dir, "/", "formula_candidates.tsv")) == F){
      return(0)
    }
    ## ------------------------------------------------------------------
    formula_df <- get_formula(key_id,
                              rank = "all",
                              ppm_error = ppm_error,
                              exclude_element = exclude_element)
    ## sometimes the top formula is not rank 1, but must be top
    n = formula_df[1, "rank"]
    formula_zodiac_rank1 <- formula_df[ which(formula_df$rank == n), c(formula_info) ]
    ## check whether there are fingerid structure files exist.
    structure_df = ""
    if(file.exists(paste0(.MCn.sirius, "/", dir, "/", "fingerid")) == T){
      check = T
      structure_df <- try(get_structure(key_id, return_row= "all"), silent = T)
    }else{
      check = F
    }
    if(class(structure_df)[1] != "try-error" & check == T){
      ## In sirius workflow, if some enforce setting (e.g., adduct enfoce) have done, max mass error (p
      structure_df <- structure_df[structure_df$molecularFormula %in% formula_df$molecularFormula, ]
      ## get formula and adduct type (select from zodiac top score formula or formula of top score stru
      if(nrow(structure_df) > 0){
        ## get top score (of structure) formula
        top_struc_formula <- structure_df[1,]$molecularFormula
        ## ------------------- get score
        formula_structure_rank1 <-
          formula_df[ which(formula_df$molecularFormula == top_struc_formula), c(formula_info)]
        score_rank1_zodiac <- formula_zodiac_rank1[1,]$ZodiacScore %>% # get score
          as.numeric()
        score_rank1_structure <- formula_structure_rank1[1,]$ZodiacScore %>% # get score
          as.numeric()
        ## -------------------- comparation
        if( score_rank1_zodiac >= (score_rank1_structure * fc) ){
          use_zodiac = T
          ## sometimes rank 1 zodiac formulae are plural, due to complex adduct type
          ## hence based on structure score to filter them
          structure_df <- structure_df[structure_df$molecularFormula %in% formula_zodiac_rank1$molecula
        }else{
          use_zodiac = F
        }
        ## -------------------- aquisition
        if( nrow(structure_df) > 0 ){
          ## in rank 1 zodiac formula, top score (structure) formula were picked
          structure_pick <- structure_df[1,]
          formula_adduct <-
            formula_df[which(formula_df$molecularFormula == structure_pick$molecularFormula), c(formula_
        }else{
          structure_pick <- data.frame()
          formula_adduct <- formula_zodiac_rank1[1,]
        }
      }else{ # nrow(structure_df) == 0, no results
        use_zodiac = T
        structure_pick <- data.frame()
        formula_adduct <- formula_zodiac_rank1[1,]
      }
    }else{ # try-error or check == F, no such files
      use_zodiac = T
      structure_pick <- data.frame()
      formula_adduct <- formula_zodiac_rank1[1,]
    }
    ## ------------------------------------------------------------------
    ## add annotation col
    formula_adduct <- dplyr::mutate(formula_adduct, use_zodiac = use_zodiac)
    ## ------------------------------------------------------------------
    if(is.null(formula_cache) == F & is.null(structure_cache) == F){
      assign(paste0(key_id), formula_adduct, envir = formula_cache)
      assign(paste0(key_id), structure_pick, envir = structure_cache)
    }
    ## return data
    if(return_formula == T){
      return(formula_adduct)
    }else if(return_structure == T){
      return(structure_pick)
    }
  }
## get dir from key_id
get_dir <- function(key_id, path = .MCn.sirius){
  dir <- list.files(path = path, pattern=paste0("^[0-9](.*)_(.*)_", key_id, "$"), full.names = F)
  check <- check_dir(dir)
  if(check == T){
    return(dir)
  }
}
```

72 File: method__rerank__binning__cluster.R

```
#' @title FUNCTION_TITLE
#' @description FUNCTION_DESCRIPTION
#' @param apset PARAM_DESCRIPTION
#' @param cluster_cutoff PARAM_DESCRIPTION, Default: seq(0.9, 0.1, by = -0.1)
#' @param least_size PARAM_DESCRIPTION, Default: 3
#' @return OUTPUT_DESCRIPTION
#' @details DETAILS
#' @examples
#' \dontrun{
#' if(interactive()){
#'  #EXAMPLE1
#'  }
#' }
#' @seealso
#'  \code{\link[ChemmineR]{cmp.cluster}}
#'  \code{\link[dplyr]{select}}, \code{\link[dplyr]{mutate}}, \code{\link[dplyr]{arrange}}, \code{\link
#'  \code{\link[reshape2]{melt}}
#'  \code{\link[tidyr]{separate}}
#' @rdname method_rerank_binning_cluster
#' @export
#' @importFrom ChemmineR cmp.cluster
#' @importFrom dplyr select mutate arrange filter distinct
#' @importFrom reshape2 melt
#' @importFrom tidyr separate
method_rerank_binning_cluster <-
  function(
           apset,
```

```r
            reference_compound,
            cluster_cutoff = seq(0.9, 0.1, by = -0.01),
            least_size = 3
            ){
    ## cluster via function of ChemmineR
    cat("## method_rerank_binning_cluster: ChemmineR::cmp.cluster\n")
    meta_rank <- ChemmineR::cmp.cluster(db = apset, cutoff = cluster_cutoff)
    ## ----------------------------------------------------------------------
    ## preparation for calculating
    ## -------------------------------------
    ## get cluster ID
    meta_rank <- dplyr::select(meta_rank, ids, starts_with("CLID"))
    ## reshape the data as long data frame
    meta_rank <- reshape2::melt(meta_rank, id.var = "ids", variable.name = "name", value.name = "number
    ## get the cutoff of cluster results
    meta_rank <- dplyr::mutate(meta_rank, cutoff = stringr::str_extract(name, "(?<=_).{1,}$"),
                               ## get origin .id
                               .id = stringr::str_extract(ids, ".*(?=_)"),
                               ## get origin structure_rank
                               structure_rank = stringr::str_extract(ids, "(?<=_)[0-9]{1,}$"),
                               ## whether reference compound
                               reference = ifelse(.id %in% reference_compound$.id & structure_rank == 1
    ## merge to get tanimotoSimilarity of reference compound
    meta_rank <- merge(meta_rank,
                       reference_compound[, c(".id", "tanimotoSimilarity")],
                       by = ".id", all.x = T)
    ## if the compound is reference compound, assign tanimotoSimilarity as reference_score
    meta_rank <- dplyr::mutate(meta_rank,
                               reference_score = ifelse(reference, tanimotoSimilarity, 0),
                               ## set group, according to cutoff and cluster ID
                               group = paste0(name, "_", number))
    ## ----------------------------------------------------------------------
    ## to calculate score of each cluster, set group
    cluster_score <- dplyr::group_by(meta_rank, group)
    ## calculate
    cluster_score <- dplyr::summarise_at(cluster_score, "reference_score", sum)
    ## rename
    cluster_score <- dplyr::rename(cluster_score, cluster_score = reference_score)
    ## ----------------------------------------------------------------------
    ## merge to get cluster_score
    meta_rank <- merge(meta_rank, cluster_score, by = "group", all.x = T)
    ## according to cutoff, re-size the score
    meta_rank <- dplyr::mutate(meta_rank, cluster_score = cluster_score * as.numeric(cutoff)^3)
    ## group via id and structure_rank
    meta_rank <- dplyr::group_by(meta_rank, ids)
    ## for all scores of one id, get sum score
    meta_rank <- dplyr::summarise_at(meta_rank, "cluster_score", sum)
    ## ---------------------------------------------------------------------
    ## reformat and output
    ## ------------------------------------
    meta_rank <- dplyr::mutate(meta_rank,
                               ## get origin .id
                               .id = stringr::str_extract(ids, ".*(?=_)"),
                               ## get origin structure_rank
                               structure_rank = stringr::str_extract(ids, "(?<=_)[0-9]{1,}$"))
    ## by id as list
    meta_rank <- by_group_as_list(meta_rank, ".id")
    ## normalize the score, via dividing by top score
    meta_rank <- lapply(meta_rank, function(df){
                          dplyr::mutate(df, norm_score = cluster_score / max(cluster_score))
                               })
    meta_rank <- data.table::rbindlist(meta_rank)
    ## as tabble
    meta_rank <- dplyr::as_tibble(meta_rank)
    return(meta_rank)
    ## ---------------------------------------------------------------------
    meta_rank <- ChemmineR::cmp.cluster(db = apset, cutoff = cluster_cutoff) %>%
      dplyr::select(ids, starts_with("CLSZ_")) %>%
      ## convert into long table
      reshape2::melt(id.var = "ids", variable.name = "cutoff", value.name = "size") %>%
      ## get 'cutoff' and as.numeric
      dplyr::mutate(cutoff = as.numeric(gsub("CLSZ_", "", cutoff))) %>%
      ## get '.id' and 'structure_rank'
      tidyr::separate(col = "ids", into = c(".id", "structure_rank"), sep = "_", remove = T) %>%
      dplyr::mutate(structure_rank = as.numeric(structure_rank)) %>%
      dplyr::arrange(desc(cutoff), desc(size), structure_rank) %>%
      ## at least, the size of cluster reach 'least_size', contribute to re-rank
      dplyr::filter(!(cutoff >= min(cluster_cutoff) & size <= least_size)) %>%
      ## for each .id, only the top 1 (according to 'cutoff', 'size', 'structure_rank', sequentialy) re
      dplyr::distinct(.id, .keep_all = T)
    return(meta_rank)
  }
```

73 File: method_rerank_jarvis_patrick_cluster.R

```r
#' @title FUNCTION_TITLE
#' @description FUNCTION_DESCRIPTION
#' @param apset PARAM_DESCRIPTION
#' @param cluster_cutoff PARAM_DESCRIPTION, Default: 0.7
#' @param generate_neighbors PARAM_DESCRIPTION, Default: 20
#' @param share_numbers PARAM_DESCRIPTION, Default: 10
#' @param share_limite PARAM_DESCRIPTION, Default: 'average'
#' @param ... PARAM_DESCRIPTION
#' @return OUTPUT_DESCRIPTION
#' @details DETAILS
#' @examples
#' \dontrun{
#' if(interactive()){
#'  #EXAMPLE1
#'  }
#' }
#' @seealso
#'  \code{\link[ChemmineR]{jarvisPatrick}}
#'  \code{\link[dplyr]{rename}}, \code{\link[dplyr]{mutate}}, \code{\link[dplyr]{arrange}}, \code{\link
#'  \code{\link[tidyr]{separate}}
#' @rdname method_rerank_jarvis_patrick_cluster
#' @export
#' @importFrom ChemmineR jarvisPatrick
#' @importFrom dplyr rename mutate arrange distinct
#' @importFrom tidyr separate
method_rerank_jarvis_patrick_cluster <-
  function(
           apset,
           cluster_cutoff = 0.7,
           generate_neighbors = 20,
           share_numbers = 10,
           share_limite = "average",
           ...
           ){
    ## cluster via function of ChemmineR
    cat("## method_rerank_jarvis_patrick_cluster: ChemmineR::jarvisPatrick\n")
    meta_rank <- ChemmineR::jarvisPatrick(nearestNeighbors(apset,
                                                           numNbrs = generate_neighbors,
                                                           cutoff = cluster_cutoff),
                                          k = share_numbers,
                                linkage = share_limite,
                                mode = "a1b") %>%
      vector_as_df() %>%
      merge(., vector_as_df(table(.$expr)), by.x = "expr", by.y = "ids", all.x = T) %>%
      dplyr::rename(cluster_id = expr, size = expr.y) %>%
      tidyr::separate(col = "ids", into = c(".id", "structure_rank"), sep = "_", remove = T) %>%
      dplyr::mutate(structure_rank = as.numeric(structure_rank)) %>%
      dplyr::arrange(desc(size), structure_rank) %>%
      ## for each .id, only the top 1 (according to 'size', 'structure_rank', sequentialy) retain
      dplyr::distinct(.id, .keep_all = T)
    return(meta_rank)
  }
vector_as_df <-
  function(
           vector
           ){
    df <- data.table(cbind(ids = names(vector), expr = unname(vector)))
    return(df)
  }
```

74 File: method_summarize_nebula_class.R

```
method_summarize_nebula_class <-
  function(
           data,
           ppcp_threshold = 0.5,
           max_number = 5,
           hierarchy_priority = c(6, 5, 4, 3), ## level 5, subclass, class, superclass
           class_data_type = "classes_tree_list", ## or "classes_tree_data"
           ...
           ){
    ## input data
    if(class_data_type == "classes_tree_list"){
      class_data = .MCn.class_tree_list
      metadata <- data.table::rbindlist(class_data, idcol = T) %>%
        dplyr::rename(hierarchy = .id)
    }else if(class_data_type == "classes_tree_data"){
      metadata <- class_data <- get("metadata", envir = get("envir_meta"))
    }
    ## main body
    df <- dplyr::filter(data, V1 >= ppcp_threshold) %>%
      merge(metadata[, 1:5], all.x = T, by = "relativeIndex", sort = F) %>%
      dplyr::filter(hierarchy %in% hierarchy_priority)
    df <- df[order(factor(df$hierarchy, levels = hierarchy_priority), -df$V1), ] %>%
      head(n = max_number)
    return(df)
  }
```

75 File: method_summarize_nebula_index.R

```
method_summarize_nebula_index <-
  function(
           ppcp_dataset,
           nebula_class = .MCn.nebula_class,
           ppcp_threshold = 0.5,
           # min number of compounds allowed exist in a child-nebula. if less than, filter the nebula
           min_possess = 10,
           ## max percentage of compounds allowed exist in a child-nebula
           max_possess_pct = 0.2,
           identical_merge = T,
           identical_factor = 0.8,
           ## if set to 4 (class), the level of or below this hierarchy (e.g., subclass) will perform m
           merge_allowed_hierarchy = c("top_level" = 4),
           ...
           ){
    classes <- data.table::rbindlist(nebula_class) %>%
      dplyr::distinct(relativeIndex)
    ## get classes
    classes <- classes$relativeIndex
    ## environment for lapply function
    assign("envir_classes", environment(), envir = parent.env(environment()))
    cat("## Method part: class_retrieve\n")
    index_list <- pbapply::pblapply(ppcp_dataset, class_retrieve,
                                    ...)
    index_df <- data.table::rbindlist(index_list, idcol = T)
    ## ---------------------------------------------------------------------
    ## filter via max_possess and min_possess
    stat <- table(index_df$relativeIndex)
    stat <- stat[which(stat >= min_possess & stat <= max_possess_pct * length(unique(index_df$".id")))]
    index_df <- index_df %>%
      dplyr::filter(relativeIndex %in% names(stat))
    ## gather with classes annotation
    index_df <- data.table::rbindlist(.MCn.class_tree_list, idcol = T) %>%
      dplyr::rename(hierarchy = .id) %>%
      dplyr::select(relativeIndex, name, hierarchy) %>%
      merge(index_df, by = "relativeIndex", all.y = T, sort = F)
    ## ---------------------------------------------------------------------
    if(identical_merge == T){
      ## filter identical or similar classes
      ## enumerate combination
      class_for_merge <- index_df %>%
        dplyr::filter(hierarchy >= merge_allowed_hierarchy[["top_level"]]) %>%
        dplyr::distinct(relativeIndex) %>%
        unlist() %>%
        combn(m = 2) %>%
        t() %>%
        data.frame()
      cat("## Method part: identical_filter\n")
      discard = pbapply(class_for_merge, 1, identical_filter,
                        identical_factor = identical_factor,
                        ...) %>%
        unlist() %>%
        unique()
    }
    index_df <- index_df[!index_df$relativeIndex %in% discard, ]
    ## cluster id in each classes
    nebula_index <- dplyr::group_by(index_df, relativeIndex)
    return(nebula_index)
  }
class_retrieve <-
  function(
           data,
           the_relativeIndex = get("classes", envir = get("envir_classes")),
           ppcp_threshold = 0.5
           ){
    ##
    classes <- the_relativeIndex
    data <- dplyr::filter(data, relativeIndex %in% classes, V1 >= ppcp_threshold)
    return(data)
  }
identical_filter <-
  function(
           couple,
           index_df = get("index_df", envir = get("envir_classes")),
           identical_factor = 0.7
           ){
    ##
    x = unique(index_df[which(index_df$relativeIndex %in% couple[1]), ]$".id")
    y = unique(index_df[which(index_df$relativeIndex %in% couple[2]), ]$".id")
    p_x = table(x %in% y)
    p_y = table(y %in% x)
    if("TRUE" %in% names(p_x) == F | "TRUE" %in% names(p_y) == F){
      return()
    }
    p_x = prop.table(p_x)[["TRUE"]]
    p_y = prop.table(p_y)[["TRUE"]]
    if(p_x >= identical_factor & p_y >= identical_factor){
      idn = ifelse(length(x) >= length(y), couple[2], couple[1])
      return(idn)
    }else{
      return()
    }
  }
```

76 File: n_method_pick_formula_excellent.R

```
#' @title FUNCTION_TITLE
#' @description FUNCTION_DESCRIPTION
#' @param dir PARAM_DESCRIPTION, Default: NULL
#' @param key_id PARAM_DESCRIPTION, Default: NULL
#' @param return_formula PARAM_DESCRIPTION, Default: T
#' @param return_structure PARAM_DESCRIPTION, Default: F
#' @param formula_cache PARAM_DESCRIPTION, Default: NULL
#' @param structure_cache PARAM_DESCRIPTION, Default: NULL
#' @param exclude_element PARAM_DESCRIPTION, Default: NULL
#' @param ppm_error PARAM_DESCRIPTION, Default: 20
#' @param fc PARAM_DESCRIPTION, Default: 1.5
#' @param formula_info PARAM_DESCRIPTION, Default: c("precursorFormula", "molecularFormula", "adduct",
#' @return OUTPUT_DESCRIPTION
#' @details DETAILS
#' @examples
#' \dontrun{
#' if(interactive()){
#'  #EXAMPLE1
#'  }
#' }
#' @seealso
#'  \code{\link[dplyr]{mutate}}
#' @rdname method_pick_formula_excellent
#' @export
#' @importFrom dplyr mutate
method_pick_formula_excellent <-
  function(
           key_id = NULL,
           dir = NULL,
           return_formula = T,
           return_structure = F,
           formula_cache = NULL,
           structure_cache = NULL,
           exclude_element = NULL,
           ppm_error = 20,
           fc = 1.5,
           formula_info = c("precursorFormula", "molecularFormula", "adduct", "ZodiacScore")
           ){
    ## ----------------------------------------------------------------------
    if(is.null(dir) == F){
      if(length(dir) >= 1){
        meta <- data.table::data.table(dir = dir)
        meta <- dplyr::mutate(meta, key_id = grep_id(dir))
      }
    }else{
      meta <- data.table::data.table(key_id = key_id, dir = get_dir(key_id))
    }
    ## ----------------------------------------------------------------------
    meta <- dplyr::mutate(meta, formula_dir = paste0(.MCn.sirius, "/", dir, "/", "formula_candidates.ts
                          fingerid = paste0(.MCn.sirius, "/", dir, "/", "fingerid"))
    ## ------------------------------------
    cat("## method part: check_dir: formula set\n")
    check <- pbapply::pblapply(meta$formula_dir, file.exists)
    meta$check_fm <- check
    ## ------------------------------------
    cat("## method part: check_dir: fingerid\n")
    check <- pbapply::pblapply(meta$fingerid, file.exists)
    meta$check_fd <- check
    ## ------------------------------------
  }
```

77 File: nebula_re_rank.R

```
nebula_re_rank <-
  function(
           nebula_name,
           top_n = 10,
           match_pattern = c("precursorFormula"), ## or c("precursorFormula", "adduct")
           collate_factor = 0.85,
           cluster_method = "method_rerank_binning_cluster",
           revise_MCn_formula_set = T,
           revise_MCn_structure_set = T,
           ...
           ){
    cat("[INFO] MCnebula run: nebula_re_rank\n")
    ## get formula
    id_set <- dplyr::filter(.MCn.nebula_index, name == nebula_name)
    formula_adduct <- dplyr::filter(.MCn.formula_set, .id %in% id_set$".id")
    ## ---------------------------------------------------------------------
    ## match patern
    if("precursorFormula" %in% match_pattern == F){
      formula_adduct$precursorFormula = NULL
    }
    if("adduct" %in% match_pattern == F){
      formula_adduct$adduct = NULL
    }
    ## ---------------------------------------------------------------------
    ## catch file
    ## due to the return data type of mapply, lapply is used instead
    ## first, as list
    assign("envir_nebula", environment(), envir = parent.env(environment()))
    formula_adduct <- lapply(formula_adduct$".id", by_group_for_list,
                             col = ".id",
                             df = get("formula_adduct", envir = get("envir_nebula")))
    ## then, use lapply match file
    cat("## netbula_re_rank: get_structure\n")
    structure_set <- pbapply::pblapply(formula_adduct, df_get_structure,
                                       top_n = top_n,
                                       collate_factor = collate_factor,
                                       ...)
    structure_set <- data.table::rbindlist(structure_set, fill = T)
    cat("## STAT of structure_set:",
        paste0(nrow(structure_set), " (structure sum)/", length(unique(structure_set$".id")), "(.id sum
    ## ---------------------------------------------------------------------
    ## get sdfset, and further get apset via ChemmineR
    cat("## convert data: SMILES_set -> SDF_set -> AP_set\n")
    sdfset <- smiles_to_sdfset(structure_set)
    apset <- sdf2ap(sdfset)
    ## ---------------------------------------------------------------------
    ## cluster method
    method_fun <- match.fun(cluster_method)
    meta_rank <- method_fun(apset, ...)
    print(meta_rank)
    ## ---------------------------------------------------------------------
    structure_set <- merge(structure_set,
                           meta_rank[, c(".id", "structure_rank", "size")],
                           by = c(".id", "structure_rank"), all.x = T) %>%
      dplyr::filter(is.na(size) == F) %>%
      dplyr::as_tibble()
    ## ---------------------------------------------------------------------
    ## revise .GlobalVar .MCn.formula_set
    if(revise_MCn_formula_set == T){
      ## prepare replace data
      rp <- dplyr::arrange(structure_set, .id) %>%
        tidyr::separate(col = "file_name", sep = "_", into = c("precursorFormula", "adduct")) %>%
        dplyr::mutate(adduct = gsub("\\+(?!$)", " \\+ ", adduct, perl = T),
                      adduct = gsub("\\-(?!$)", " \\- ", adduct, perl = T)) %>%
        dplyr::select(.id, precursorFormula, adduct, molecularFormula)
      ## replace
      fset <- dplyr::arrange(.MCn.formula_set, .id)
      fset[fset$".id" %in% rp$".id", c(".id", "precursorFormula", "adduct", "molecularFormula")] <- rp
      .MCn.formula_set <<- fset
    }
    ## ---------------------------------------------------------------------
    ## revise .GlobalVar .MCn.structure_set -------
    if(revise_MCn_structure_set == T){
      sset <- dplyr::arrange(.MCn.structure_set, .id)
      ## prepare replace data
      rp <- dplyr::arrange(structure_set, .id) %>%
        dplyr::select(colnames(sset))
      ## replace
```

```
      sset <- dplyr::distinct(rbind(rp, sset), .id, .keep_all = T)
      .MCn.structure_set <<- sset
      ## rename exist structure picture -------
      tmp_stru <- paste0(.MCn.output, "/", .MCn.results, "/tmp/structure")
      if(file.exists(tmp_stru) == T){
        lapply(paste0(tmp_stru, "/", rp$".id", ".svg"), rename_file)
      }
    }
    ## ----------------------------------------------------------------------
    cat("[INFO] MCnebula Job Done: nebula_re_rank\n")
    return(structure_set)
  }
smiles_to_sdfset <-
  function(
           structure_set
           ){
    ##
    smiles_set <- structure_set$smiles
    names(smiles_set) <- paste0(structure_set$".id", "_", structure_set$structure_rank)
    ## this function automaticly set the vector name as name of each subset
    sdf_set <- ChemmineR::smiles2sdf(smiles_set)
    return(sdf_set)
  }
df_get_structure <-
  function(
           x,
           top_n = 10,
           collate_factor = 0.85,
           ...
           ){
    df <- get_structure(
                        x[[".id"]],
                        x[["precursorFormula"]],
                        x[["adduct"]],
                        return_row = 1:top_n,
                        ...)
    if(nrow(df) == 0){
      return(df)
    }
    df <- dplyr::mutate(df, .id = x[[".id"]]) ## add key_id
    top_simi <- df[1, "tanimotoSimilarity"]
    df <- dplyr::filter(df, tanimotoSimilarity >= top_simi * collate_factor)
    return(df)
  }
rename_file <-
  function(
           file,
           suffix = "prefix"
           ){
    if(file.exists(file) == T){
      file.rename(file, paste0(file, ".", suffix))
    }
  }
```

78 File: read_tsv.R

```
read_tsv <- function(path){
  file <- data.table::fread(input=path, sep="\t", header=T, quote="", check.names=F)
  return(file)
}
write_tsv <-
  function(x, filename){
    write.table(x, file = filename, sep = "\t", col.names = T, row.names = F, quote = F)
  }
```

79 File: test.R

```
compare_score <-
  function(
           structure_df,
           formula_df,
           formula_zodiac_rank1,
           formula_info,
           fc
           ){
    if(is.na(fc) == F){
      ## get top score (of structure) formula
      top_struc_formula <- structure_df[1,]$molecularFormula
      ## ---------------------------------------------------------------------
      ## ------------------- get score
      formula_structure_rank1 <-
```

```
        formula_df[ molecularFormula == top_struc_formula, formula_info, with = F]
      ## get score
      ## the data.frame is a data.table project
      score_rank1_zodiac <- as.numeric(formula_zodiac_rank1[1, ZodiacScore])
      score_rank1_structure <- as.numeric(formula_structure_rank1[1, ZodiacScore])
      ## ------------------------------------------------------------------
      ## ------------------------------------------------------------------
      ## ------------------- comparation
      if( score_rank1_zodiac >= (score_rank1_structure * fc) ){
        use_zodiac = T
        ## sometimes rank 1 zodiac formulae are plural, due to complex adduct type
        ## hence based on structure score to filter them
        structure_df <- structure_df[molecularFormula %in% formula_zodiac_rank1$molecularFormula, ]
      }else{
        use_zodiac = F
      }
    }else{
      use_zodiac = F
    }
    list <- list(structure_df, use_zodiac)
    return(list)
  }
```

80 File: vis_via_molconvert.R

```
vis_via_molconvert_nebulae <-
  function(
           nebula_name
           ){
    df <- dplyr::filter(.MCn.nebula_index, name == nebula_name)
    stru <- dplyr::filter(.MCn.structure_set, .id %in% df$".id")
    vis_via_molconvert(stru$smiles, stru$".id")
    return("Done")
  }
vis_via_molconvert <-
  function(
           smiles_set,
           id_set,
           output = paste0(.MCn.output, "/", .MCn.results, "/tmp/structure")
           ){
    if(file.exists(output) == F){
      dir.create(paste0(.MCn.output, "/", .MCn.results))
      dir.create(output)
    }
    pbapply::pbmapply(molconvert_structure,
           smiles_set,
           id_set,
           MoreArgs = list(output = output)
    )
    return("Done")
  }
## ----------------------------------------------------------------------
molconvert_structure <-
  function(
           smiles,
           id,
           output = paste0(.MCn.output, "/", .MCn.results, "/tmp/structure")
           ){
    file = tempfile()
    system(paste0("molconvert mol \"", smiles, "\" -o ", file))
    system(paste0("obabel ", file, " -imol -osvg -O ", file, " > /dev/null 2>&1"))
    system(paste0("sed -i \'s/white/transparent/g\' ", file))
    system(paste0("cairosvg -f svg ", file, " -o ", output, "/", id, ".svg"))
    return()
  }
```

81 File: visualize_child_nebulae.R

```
visualize_child_nebulae <-
  function(
           graph_list = .MCn.child_graph_list,
           compound_class_list = .MCn.nebula_class,
           write_output = T,
           output = paste0(.MCn.output, "/", .MCn.results),
           layout = "fr",
           width = 23,
           height = 30,
           ...
           ){
    cat("[INFO] MCnebula run: visualize_child_nebulae\n")
    ## get top compound class (nodes_color data)
    metadata <- lapply(compound_class_list, head, n = 1) %>%
      data.table::rbindlist(idcol = T) %>%
      dplyr::select(.id, name) %>%
      dplyr::rename(vis_class = name)
    assign("envir_meta", environment(), envir = parent.env(environment()))
    ## draw network via ggplot, and print into grid palette
    ## number of child_nebulae
    n = length(graph_list)
    ## specification of grid (cols * rows)
    cols = n^(1/2)
    if(round(cols) != cols){
      cols = round(cols)
      rows = cols + 1
    }else{
      rows = cols
    }
    ## grid position of all child_nebulae
    graph_anno <- names(graph_list) %>% # names
      dplyr::as_tibble() %>%
      dplyr::rename(nebula_index = value) %>%
      merge(.MCn.class_tree_data[,c("name", "hierarchy")], by.x = "nebula_index", by.y = "name", all.x
      dplyr::arrange(desc(hierarchy)) %>%
      ## calculate position
      dplyr::mutate(seq = 1:n,
                    col = ifelse(seq %% cols != 0, seq %% cols, cols),
                    row = (seq - col)/cols + 1)
    ## as list
    nebula_index <- graph_anno$nebula_index
    graph_anno <- lapply(nebula_index, by_group_for_list,
                         df = get("graph_anno", envir = get("envir_meta")),
                         col = "nebula_index")
    ## re-order the graph list according to annotation
    graph_list <- lapply(nebula_index,
                         function(x){
                           graph_list[[x]]
                         })
    ## prepare grid palette
    svglite::svglite(paste0(output, "/", "child_nebulae.svg"), width = width, height = height)
    grid::grid.newpage()
    grid::pushViewport(viewport(layout = grid.layout(rows, cols)))
    ## draw child_nebulae in grid
    pbapply::pbmapply(grid_child_nebula, ## function
                      graph_list, ## graph list
                      graph_anno, ## graph annotation
                      MoreArgs = list( ## args
                                       layout = layout,
                                       ...
                                       ))
    dev.off()
    cat("[INFO] MCnebula Job Done: visualize_child_nebulae\n")
  }
grid_child_nebula <-
  function(
           graph,
           anno,
           layout = "fr",
           class = get("metadata", envir = get("envir_meta")),
           title_palette = .MCn.palette_label,
           print_into = T,
           save_layout_df = NULL,
           remove_nodes = NULL,
           ...
           ){
    ## reformat graph, add with class
    graph <- tidygraph::as_tbl_graph(graph)
    ## nodes ------------------------------------------------------
    nodes <- merge(graph, class, by.x = "name" , by.y = ".id", all.x = TRUE, sort = F)
    nodes <- dplyr::as_tibble(nodes)
    ## edges ------------------------------------------------------
    edges <- dplyr::as_tibble(tidygraph::activate(graph, edges))
    if(nrow(edges) >= 1){
      ## "dotproduct" or other attributes of compare spectra method.
      edges <- dplyr::rename(edges, similarity = 3)
    }else{
      edges <- dplyr::mutate(edges, similarity = NA)
    }
    ## ------------------------------------------------------------
    ## gather nodes and edges
    graph <- tidygraph::tbl_graph(nodes = nodes, edges = edges)
    ## create network layout
    layout_n <- ggraph::create_layout(graph, layout = layout, ...)
    if(is.null(save_layout_df) == F){
      assign("layout_n", layout_n, envir = save_layout_df)
    }
    ## color palette
    palette <- .MCn.palette
    ## plot
    p <- base_vis_c_nebula(layout_n, palette,
                           title = anno[["nebula_index"]],
                           title_fill = title_palette[as.numeric(anno[["hierarchy"]])],
                           remove_nodes = remove_nodes,
                           ...)
    if(print_into == F){
      return(p)
    }
    print(p + ggplot2::guides(size="none", fill="none"),
          vp = grid::viewport(layout.pos.row = anno[["row"]], layout.pos.col = anno[["col"]]))
  }
## base visualization method
base_vis_c_nebula <-
  function(
           nebula,
           title = NULL,
           palette = .MCn.palette,
           title_fill = "grey",
           nodes_size_range = c(3, 7),
           edges_width_range = c(0.1, 0.7),
           title_size = 20,
           remove_nodes = NULL
           ){
    if(is.null(remove_nodes) == F){
      nodes_size_range = 0
    }
    p <- ggraph::ggraph(nebula) +
      ggraph::geom_edge_fan(aes(edge_width = similarity), color = "black", show.legend = F) +
      ggraph::geom_node_point(
                      aes(
                          size = ifelse(is.na(tanimotoSimilarity) == F, tanimotoSimilarity, 0.2),
                          fill = vis_class
                          ),
                      shape = 21
                      ) +
      ggplot2::scale_fill_manual(values = palette) +
      ggraph::scale_edge_width(range = edges_width_range) +
      ggplot2::scale_size(range = nodes_size_range) +
      ggplot2::guides(fill = guide_legend(override.aes = list(size = 5))) +
      ggplot2::ggtitle(stringr::str_wrap(title, width = 30)) +
      ggplot2::labs(size = "Tanimoto\nsimilarity", fill = "Compound class") +
      ggplot2::theme_grey() +
      ggplot2::theme(
            text = element_text(family = "Times"),
            axis.ticks = element_blank(),
            axis.text = element_blank(),
            axis.title = element_blank(),
            panel.background = element_rect(fill = "white"),
            panel.grid = element_blank(),
            plot.title = ggtext::element_textbox(
                                             size = title_size,
                                             color = "white", fill = title_fill, box.color = "white",
                                             halign = 0.5, linetype = 1, r = unit(5, "pt"), width = unit(1,
                                             padding = margin(2, 0, 1, 0), margin = margin(3, 3, 3, 3)
            )
      )
    return(p)
  }
```

82 File: visualize_nebula.R

```
visualize_nebula <-
  function(
           nebula_name,
           ...
           ){
    p <- annotate_child_nebulae(
                                nebula_name = nebula_name,
                                write_output = F,
                                plot_structure = F,
                                plot_ppcp = F,
                                merge_image = F,
                                return_plot = T,
                                ...)
    return(p)
  }
```

83 File: visualize_parent_nebula.R

```r
visualize_parent_nebula <-
  function(
           graph = .MCn.parent_graph,
           write_output = T,
           output = paste0(.MCn.output, "/", .MCn.results),
           layout = "mds",
           nodes_color = c("hierarchy" = 4), ## default, use superclass as color.
           width = 15,
           height = 12,
           return_plot = F,
           ...
           ){
    cat("[INFO] MCnebula run: visualize_parent_nebula\n")
    ## get nodes_color data
    metadata = .MCn.class_tree_data
    assign("envir_meta", environment(), envir = parent.env(environment()))
    cat("# Visualize_parent_nebula |", date(), "| USE Method: method_summarize_nebula_index.\n")
    class <- pbapply::pblapply(.MCn.ppcp_dataset, method_summarize_nebula_class,
                               class_data_type = "classes_tree_data",
                               max_number = 1,
                               hierarchy_priority = nodes_color[["hierarchy"]] )
    class <- data.table::rbindlist(class, idcol = T) %>%
      dplyr::select(.id, name) %>%
      dplyr::rename(vis_class = name)
    ## reformat graph, add with class
    graph <- tidygraph::as_tbl_graph(graph)
    nodes <- graph %>%
      tidygraph::activate(nodes) %>%
      merge(class, by.x = "name" , by.y = ".id", all.x=TRUE, sort=F) %>%
      dplyr::mutate(vis_class = ifelse(is.na(vis_class) == T, "Unknown", vis_class)) %>%
      dplyr::as_tibble()
    edges <- graph %>%
      tidygraph::activate(edges) %>%
      ## rename the col of value of compare spectra
      dplyr::rename(similarity = 3) %>%
      dplyr::as_tibble()
    graph <- tidygraph::tbl_graph(nodes = nodes, edges = edges)
    ## create network layout
    layout_n <- ggraph::create_layout(graph, layout = layout, ...)
    ## palette
    palette <- .MCn.palette
    ## draw network via ggraph
    p <- base_vis_p_nebula(layout_n, palette)
    ## write_output
    if(write_output == T){
      ggplot2::ggsave(p, file = paste0(output, "/", "parent_nebula", "/", "parent_nebula.svg"),
             width = width, height = height)
    }
    cat("[INFO] MCnebula Job Done: visualize_parent_nebula\n")
    if(return_plot == T){
      return(p)
    }
  }
## base visualization method
base_vis_p_nebula <-
  function(
           nebula,
           palette = .MCn.palette,
           ...
           ){
    p <- ggraph::ggraph(nebula) +
      ggraph::geom_edge_fan(aes(edge_width = similarity), color = "lightblue", show.legend = F) +
      ggraph::geom_node_point(
                        aes(
                            size = ifelse(is.na(tanimotoSimilarity) == F, tanimotoSimilarity, 0.2),
                            fill = stringr::str_wrap(vis_class, width = 25)
                            ),
                        shape = 21
                        ) +
      ggplot2::scale_fill_manual(values = palette) +
      ggraph::scale_edge_width(range = c(0.1, 0.7)) +
      ggplot2::guides(fill = guide_legend(override.aes = list(size = 5))) +
      ggplot2::labs(fill="Class", size="Tanimoto similarity") +
      ggplot2::theme_grey() +
      ggplot2::theme(
            text = element_text(family = "Times"),
            axis.ticks = element_blank(),
            axis.text = element_blank(),
            axis.title = element_blank(),
            panel.background = element_rect(fill = "white"),
            legend.key.width = unit(1, "cm"),
            legend.key.height = unit(1.8, "cm"),
            legend.title = element_text(size = 20, face = "bold"),
            legend.text = element_text(size = 20),
            legend.background = element_rect(fill = "transparent"),
            panel.grid = element_blank(),
            strip.text = element_text(size = 20, face = "bold")
      )
    return(p)
  }
```

84 File: rcdk.R

```
###########  R rcdk structure

####  wd ~/operation/back/0703_all/results

library(tidyverse)

library(rcdk)

data <- read.csv(file="fingerid_first_score.tsv", header=T,sep="\t")

data <- data[which(is.na(data$smiles)==F), colnames(data) %in% c("id", "smiles")]

for(i in 1:nrow(data)){

id <- data[i, 1]

stru <- data[i, 2]

mols <- parse.smiles(stru)

#mols <- generate.2d.coordinates(mols[[1]])

#mols <- get.smiles(mols,smiles.flavors(c('CxSmiles')))

write.molecules(mols, paste0("structure_2d/smiles_draw/", id, "_"), together = F, write.props = F)

}
```

85 File: read_xml.R

```
library(XML)

xml <- xmlToDataFrame("sites.xml")

write.table(xml, file = paste0("data",".tsv"), quote = FALSE, append = FALSE, sep = "\t", col.names = T
```

86 File: sirius_db.R

```
library(tidyverse)
path1="cnumber_cid.tsv"
path2="merge_smiles.tsv"
df1 <- read.csv(file=path1, sep="\t", header=T)
df2 <- read.csv(file=path2, sep="\t", header=T)
df <- merge(df1, df2, all.y=T, by.x="pubchem.id", by.y="CID", sort=T)
df <- df[, c(3,2,1)]
write.table(df, file="sirius_db.tsv", sep="\t", col.names=F, row.names=F, quote=F)
```

87 File: superstart.R

```
library(usethis)
library(devtools)
library(progress)
## data reformat
library(pbapply)
library(data.table)
library(dplyr)
library(stringr)
load_all("~/MCnebula/R")
load_all("~/extra/")

library(ggplot2)
library(ggsci)
library(grid)
```

88 File: ttest_volcano.R

```
library(outliers)
grubbs<-function(x){
  x<-round(x,4)
  grubbs_outliers<-c()
  grubbs_p.value<-c()
  grubbs_g.value<-c()
  grubbs_g<-c()
  grubbs_minormax<-c()
  grubbs_pvalue<-c()
  grubbs_p<-0
  while(grubbs_p<0.05){
    grubbs_outliers<-c(grubbs_outliers,grubbs_minormax)
    grubbs_p.value<-c(grubbs_p.value,grubbs_pvalue)
    grubbs_g<-c(grubbs_g,grubbs_g.value)
    if(sum(x==grubbs_minormax)!=0)x<-x[-which(x==grubbs_minormax)]
    if(sd(x)==0) break
    grubbs_test<-grubbs.test(x,type=10,opposite=F,two.sided=F)
    grubbs_p<-grubbs_test$p.value
    grubbs_pvalue<-grubbs_test$p.value
    grubbs_g.value<-grubbs_test$statistic[1]
    grubbs_a<-strsplit(grubbs_test$alternative," ",fixed=T)
    grubbs_minormax<-as.numeric(unlist(grubbs_a)[3])
  }
  outliner_res<-data.frame(outliers=grubbs_outliers,gvalue=grubbs_g,pvalue=grubbs_p.value)
  return(outliner_res)
}
dixon<-function(x){
    dixon_p.value<-c()
    dixon_q.value<-c()
    dixon_q<-c()
    dixon_pvalue<-c()
    dixon_outliers<-c()
    dixon_minormax<-c()
    dixon_p<-0
    while(dixon_p<0.05){
      dixon_outliers<-c(dixon_outliers,dixon_minormax)
      dixon_p.value<-c(dixon_p.value,dixon_pvalue)
      dixon_q<-c(dixon_q,dixon_q.value)
      if(sum(x==dixon_minormax)!=0)x<-x[-which(x==dixon_minormax)]
      if(sd(x)==0) break
      dixon_test<-dixon.test(x,type=0,opposite=F,two.sided=F)
      dixon_p<-dixon_test$p.value
      dixon_pvalue<-dixon_test$p.value
      dixon_q.value<-dixon_test$statistic[1]
      dixon_a<-strsplit(dixon_test$alternative," ",fixed=T)
      dixon_minormax<-as.numeric(unlist(dixon_a)[3])
    }
    outliner_res<-data.frame(outliers=dixon_outliers,qvalue=dixon_q,pvalue=dixon_p.value)
    return(outliner_res)
  }
savepath="volcano_ttest"
if(file.exists(savepath)==FALSE){dir.create(savepath)}
data <- read.csv(file="fecal_pos_volcano.tsv",header=T,sep="\t",check.names=F)
group <- unique(data$subgroup)
id_set=colnames(data)
list=rbind(c("model","raw"),c("model","pro"),c("raw","pro"),c("model","control"))
the_filter=0
for(row in 1:nrow(list)){
double=list[row,]
escape=0
    for(k in c("low","medium","high")){
    if(double[1]!="model"){compare2=paste0(double[1],"_",k)}else{compare2=double[1]}
    if(double[2]!="model"){compare1=paste0(double[2],"_",k)}else{compare1=double[2]}
    if(double[1]=="model" & double[2]=="control"){compare2="model"; compare1="control"; escape=1}
    sink(paste0(savepath,"/",compare2,"@",compare1),append = FALSE, split = FALSE)
    print(paste0("|id|p-value|number|fold:",compare2,"/",compare1,"|"))
    for(i in 2:ncol(data)){
    X=data[which(data$subgroup==compare1),i]
    n=try(out<-dixon(round(X,3)))
    if(class(n)=="try-error"){n=try(out<-grubbs(round(X,3)))}
    if(class(n)=="try-error"){out=NULL}
    if(is.null(out$outliers)){x_filter=X}else{x_filter=X[which(X!=c(out$outliers))];the_filter=the_filt
      num_x=length(x_filter)
    Y=data[which(data$subgroup==compare2),i]
    n=try(out<-dixon(round(Y,3)))
    if(class(n)=="try-error"){n=try(out<-grubbs(round(Y,3)))}
    if(class(n)=="try-error"){out=NULL}
    if(is.null(out$outliers)){y_filter=Y}else{y_filter=Y[which(Y!=c(out$outliers))];the_filter=the_filt
      num_y=length(y_filter)
    stat=t.test(X, Y, var.equal = T, paired = F)
    fold=mean(y_filter)/mean(x_filter)
    print(paste0("|",id_set[i],"|",stat$p.value,"|",num_x,"_",num_y,"|",fold,"|"))}
    sink()
    print(paste0("fold change: ",compare2," divided by ",compare1))
    if(escape==1){break}
    }
}
print(the_filter)
```

89 File: violin.R

```
library(ggplot2)
library(ggrepel)
library(ggsci)
library(tidyr)
library(ggforce)
library(reshape2)
library(ggthemes)
library(stringr)
file="for_violin_0.5.tsv"
dfpath="fingerid_first_score.tsv"
df <- read.csv(file=dfpath, header=T, sep="\t")
df <- df[,colnames(df) %in% c("id", "similarity")]
data <- read.csv(file=file,header=T,sep="\t")
colnames(data)[3:4]=c("Raw-Eucommia", "Pro-Eucommia")
data <- melt(data, id.vars=c("id","classification"), variable.name="condition", value.name="expr")
data$expr[which(data$expr==-Inf)] <- 0
data <- merge(data, df, all.x=T, by="id", sort=F)
data <- data[which(data$similarity>0.5), ]
p <- ggplot(data, aes(x=classification, y=expr, fill=ifelse(condition=="Raw-Eucommia", "1", "2"))) +
   geom_violin(trim=F,color="transparent") +
   geom_boxplot(width=0.2,position=position_dodge(0.9)) +
   guides(fill="none") +
   coord_flip() +
   scale_fill_manual(values = c("#4DBBD5FF", "#E64B35FF")) +
   scale_x_discrete(limits = rev(levels(as.factor(data$classification)))) +
   labs(x="Classification", y="Log10(peak area)(Tanimoto similarity > 0.5; Posterior probability > 0.5
   theme(text=element_text(family="Times"),
       axis.title = element_text(face="bold"),
       axis.title.x = element_text(hjust=1.7),
       axis.text.x = element_text(angle=0),
       axis.text = element_text(size=12)) +
   facet_wrap(~condition, ncol=2)
 ggsave(p, file="violin.svg", width=8, height=18)
```

```
  #ggsave(p, file="violin.svg", width=10, height=15)
  ####################  geom_density
  list=grep("lignan|Iridoid", data$classification, ignore.case=T)
  df <- data[list,]
  df$classification <- str_wrap(df$classification, width=25)
  pp <- ggplot() +
    geom_density(data=df, aes(x=expr, fill=condition), alpha=0.3) +
    facet_grid(classification~.,
           scales="free"
    ) +
    scale_fill_manual(values = c("#4DBBD5FF", "#DC0000FF")) +
    theme_classic() +
    labs(x="Log10(peak area)", y="Density") +
    guides(ncol=2) +
    theme(text=element_text(family="Times"),
        axis.title = element_text(face="bold", size=15),
        axis.text.x = element_text(angle=0),
        axis.text = element_text(size=15),
        legend.text = element_text(size=15),
        legend.title = element_blank(),
        legend.position = "bottom",
        strip.text = element_text(size = 12, face = "bold"),
        strip.background=element_rect(color="white")
        )
  ggsave(pp, file="density_free.svg", width=8, height=14)
```

90 File: vip_p_facet.R

```
library(ggplot2)
library(ggsci)

args<-commandArgs(TRUE)

setwd(args[1])

df <- read.csv(file="vip_p_facet.tsv",header= T,sep= "\t",check.names=FALSE)

colnames(df)[3]="p"

anno=unique(df[,colnames(df) %in% c("group","g1_from","g2_deep")])
```

```
p <- ggplot(df, aes(x=p, y=vip, color=FC)) +

    geom_point(alpha=0.8, size=1.5, stroke=0) +

    xlim(0,0.5) +

    scale_color_gradientn(limits = c(-5, +5),
                          breaks = c(-3, 0, +3),
                          colours = c("#E6550DFF", "#79AF97FF", "#3182BDFF")) +
    #scale_color_npg() +
    #scale_fill_npg() +

    labs(y="VIP", x="p-value", color="FC") +

    geom_hline(yintercept = 1,linetype=4,size=0.8) +

    geom_vline(xintercept = 0.05,linetype=4,size=0.8) +

    geom_text(data=anno, aes(x=max(df$p)/4, y=max(df$vip), label=group),
            hjust=0.5, color="black", fontface="bold",alpha=0.6, size=2, inherit.aes = FALSE, family="T

    ggtitle("VIP-P") +

    facet_grid(g2_deep ~ g1_from) +

    theme(legend.position = "right",text=element_text(family="serif"), plot.title = element_text(hjust =

ggsave(p,file="vip_p_facet.svg")
```

91 File: vip_p.R

```
library(ggplot2)
library(ggsci)

args<-commandArgs(TRUE)

setwd(args[1])

datas=list.files(path = ".", pattern = "*.vip_p$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
```

```r
            ignore.case = FALSE, include.dirs = FALSE)

for(i in 1:length(datas)){

savename=strsplit(datas[i],split=".vip_p")

df <- read.csv(file=datas[i],header= T,sep= "\t",check.names=FALSE)

truenames=colnames(df)

set=unlist(strsplit(truenames[4],split=": "))

legend=paste0("VIP-P-FC: ",set[2])

colnames(df)=c("id","vip","p","fold")

p <- ggplot(df, aes(x=p, y=vip, color=fold)) +

    geom_point(alpha=0.8, size=3, stroke=0) +

    scale_color_gradientn(limits = c(-5, +5),
                          breaks = c(-3, 0, +3),
                          colours = c("#E6550DFF", "#79AF97FF", "#3182BDFF")) +
    #scale_color_npg() +
    #scale_fill_npg() +

    labs(y="VIP", x="p-value", color="FC") +

    geom_hline(yintercept = 1,linetype=4,size=0.8) +

    geom_vline(xintercept = 0.05,linetype=4,size=0.8) +

    ggtitle(legend) +

    #xlim(0,0.5) +

    #scale_x_continuous(limits=c(-60 ,60)) +

    #scale_y_continuous(limits=c(-60 ,60)) +

    #facet_grid(g2_deep ~ g1_from) +
    theme(legend.position = "right",text=element_text(family="serif"), plot.title = element_text(hjust =

ggsave(p,file=paste0(savename,"_vip_p.svg"))
}
```

92 File: volcano_facet.R

```
library(ggplot2)
library(ggrepel)

setwd("volcano")

data <- read.csv(file="volcano_facet",header=T,sep="\t")

#data <- na.omit(data)

savename="volcano_facet"

title="Volcano grid"

data$change <- factor(ifelse(data$p.value < 0.05 & abs(data$fold) >= 1,
                      ifelse(data$fold >= 1,"up","down"),"stable"),
                      levels = c("up","down","stable"))

p <- ggplot(data,aes(x=fold, y=-log10(p.value),color = change)) +

  geom_point(alpha=0.8, stroke=0, size=1.5) +

  scale_color_manual(values = c("down"="#4DBBD5FF","stable"="#8491B4FF","up"="#DC0000FF")) +

  #xlim(-10,10) +

  ylim(1,max(-log10(data$p.value))) +

  geom_hline(yintercept = -log10(0.05),linetype=4,size=0.8) +

  geom_vline(xintercept = c(-1,1),linetype=4,size=0.8) +

  labs(x = "log2(foldchange)",y="-log10(p-value)",title=title) +
  geom_text_repel(data = data[data$p.value<0.01 & abs(data$fold) >= 2,],
                  aes(label = id),size = 3,family="Times") +

  #annotate("text", x = -10, y = max(-log10(data$p.value))*0.8, label = paste0("Based on ",nrow(data),"
  #           color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +

  theme(text=element_text(family="serif"),

    #axis.line = element_line(colour = "black", size=0.2),

    #plot.margin = unit(c(3, 1, 3, 1), "cm")

    plot.title = element_text(hjust = 0.5)) +

  facet_grid(g2_deep ~ g1_from)

 #svg(paste0(savename,".svg"),width=8,height=6.5)
 #p
 #dev.off()

 ggsave(p,file=paste0(savename,".svg"),width=8,height=6.5)
```

93 File: volcano.R

```
library(ggplot2)
library(ggrepel)

setwd("volcano")

files=list.files(path = ".", pattern = "*.tsv$", all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE)

for(file in files){

data <- read.csv(file=file,header=T,sep="\t",check.names=F)

#data <- na.omit(data)

savename=strsplit(file, split=".tsv")
```

```r
set=colnames(data)

title=set[3]

colnames(data)=c("id","p.value","fold")

data$change <- factor(ifelse(data$p.value < 0.05 & abs(data$fold) >= 1,
                      ifelse(data$fold >= 1,"up","down"),"stable"),
                      levels = c("up","down","stable"))

p <- ggplot(data,aes(x=fold, y=-log10(p.value),color = change)) +

  geom_point(alpha=0.8, stroke=0, size=3) +

  scale_color_manual(values = c("down"="#4DBBD5FF","stable"="#8491B4FF","up"="#DC0000FF")) +

  #xlim(-10,10) +

  ylim(1,max(-log10(data$p.value))) +

  geom_hline(yintercept = -log10(0.05),linetype=4,size=0.8) +

  geom_vline(xintercept = c(-1,1),linetype=4,size=0.8) +

  labs(x = "log2(foldchange)",y="-log10(p-value)",title=title) +

  geom_text_repel(data = data[data$p.value<0.01 & abs(data$fold) >= 2,],
                  aes(label = id),size = 3,family="Times") +

  #annotate("text", x = -10, y = max(-log10(data$p.value))*0.8, label = paste0("Based on ",nrow(data),"
  #           color="black",size = 3, fontface="bold", family="Times", hjust = 0 ) +

  theme(text=element_text(family="serif"),

    #axis.line = element_line(colour = "black", size=0.2),

    #plot.margin = unit(c(3, 1, 3, 1), "cm")

    plot.title = element_text(hjust = 0.5))

 #svg(paste0(savename,".svg"),width=8,height=6.5)
 #p
 #dev.off()

 ggsave(p,file=paste0(savename,".svg"),width=8,height=6.5)}
```

94 File: xcms.R

```
library(xcms)
setwd("/media/wizard/back/nanjing_sample/fecal/pos_mzml")
metadata <- read.csv(file="EIC_metadata.tsv",header=T,sep="\t")
dda_file=list.files(path = ".", pattern = "*.mzML$", all.files = FALSE,
          full.names = FALSE, recursive = FALSE,
          ignore.case = FALSE, include.dirs = FALSE)
dir.create("EIC")
tolerance=0.005
for(filename in dda_file){
dda_data <- readMSData(filename, mode = "onDisk")
dir.create(paste0("EIC/EIC_",filename))
  for(number in 1:nrow(metadata)){
  id <- metadata[number,colnames(metadata) %in% c("id")]
  mz <- metadata[number,colnames(metadata) %in% c("m.z")]
  mzrange <- c(as.numeric(mz)-tolerance,as.numeric(mz)+tolerance)
  ex_data <- chromatogram(dda_data, msLevel = 1L, mz = mzrange, aggregationFun = "max")
  ex_data_1 <- ex_data[1,1]
  if(number==1){
   write.table(rtime(ex_data_1),paste0("EIC/EIC_",filename,"/rt",".tsv"),col.names = FALSE,sep="\t")}
  write.table(intensity(ex_data_1),paste0("EIC/EIC_",filename,"/",id,"_intensity",".tsv"),col.names =
  print(paste0(filename," >>> ",number,"/",nrow(metadata))))}}
```

95 File: zhu_heatmap.R

```
### ggplot2_heatmap.R

library(ggplot2)
library(ggtree)
library(aplot)
library(tidyr)
library(rayshader)
library(ggsci)
library(stringr)
```

```r
file="heatmap_1.tsv"

savename="zhu_heatmap.svg"

data <- read.csv(file=file,header=T,sep="\t",row.names=1, check.name=F)

colnames(data) <- str_wrap(colnames(data), width=30)

phr <- hclust(dist(data)) %>% ggtree(layout="rectangular", branch.length="none")

phc <- hclust(dist(t(data))) %>% ggtree(layout="rectangular", branch.length="none") + layout_dendrogram

data$names <- rownames(data)

p1 <- gather(data, 1:(ncol(data)-1), key="condition", value='expr')  ## keep the last col

pp <- ggplot(p1,aes(x=condition,y=names,fill=expr)) +
  geom_tile() +
  #theme_minimal()+
  scale_fill_gradient2() +
  #scale_fill_npg() +
  scale_y_discrete(position="right") +
  labs(x="Compounds", y="Sample", fill="Log10(Peak area)") +
  theme(
    axis.text.x = element_text(angle=90, hjust=1),
    axis.text = element_text(size=20),
    axis.title = element_text(size=20, face="bold"),
    #axis.text = element_blank(),
    legend.title = element_text(size=20, face="bold"),
    legend.text = element_text(size=20),
    legend.key.width = unit(2, "cm"),
        legend.key.height = unit(4, "cm"),
        text=element_text(family="serif")
    #axis.title.y = element_text(size = 14),
    #plot.title = element_text(hjust = 1,vjust=-40,size=14)
    )

pp_com <- pp %>%
  insert_left(phr, width=.1) %>%
  insert_top(phc, height=.1)
```

```
ggsave(pp_com,file=savename, width=20, height=20)

#plot_gg(pp_com, multicore = TRUE, width = 20 ,height=20, scale=250) # 加载图形
#render_depth(focallength=100,focus=0.72)

library(ggupset)

file="instance_norm_random.tsv"

data <- read.csv(file=file,header=T,sep="\t", check.name=F)

data$upset_x=paste0(data$names, "_", data$condition)

p <- ggplot(data, aes(x=factor(upset_x), y=expr)) +
  geom_col() +
  #scale_x_mergelist(sep = "_") +
  axis_combmatrix(sep = "_") +
  #coord_flip() +
  labs(x="Feature link", y="Normalized ftalign similarity") +
  theme(
    #axis.text.x = element_text(angle=90),
    axis.text.y = element_text(size=20, angle=90),
    axis.title.x = element_text(size=30, face="bold", angle=180),
    axis.title.y = element_text(size=30, face="bold"),
    #axis.text = element_blank(),
    legend.title = element_text(size=20, face="bold"),
    legend.text = element_text(size=20),
    #legend.key.width = unit(2, "cm"),
        #legend.key.height = unit(4, "cm"),
        text=element_text(family="serif")
    #axis.title.y = element_text(size = 14),
    #plot.title = element_text(hjust = 1,vjust=-40,size=14)
    ) +
  theme_combmatrix(combmatrix.panel.point.size = 5,
           #combmatrix.panel.margin = unit(c(0.5,0.5),"pt"),
           combmatrix.panel.line.size = 2,
           combmatrix.label.height = unit(500, "pt"),
           combmatrix.label.text = element_text(family="serif", angle=145, size=12, face="bold", hjust=
           combmatrix.label.make_space = F)

ggsave(p, file="test1.pdf", width=20, height=20)
```