Bash codes of ‘practice’

Contents

1 File: back_eucommia_analysis.sh

```
### Similar-fragmentation-network based multidimensional data analysis
mzmine_path="../0924_neg_RT.tsv.csv"
savepath="results/0924_neg_RT.tsv"
version=$(echo | awk '{l=asort(a,b);print l}')
    if [ $version == 0 ]
    then
    mzmine_data=$(awk -F , '
        {
        if(NR==1)
            {
            for(i=4; i<=NF; i++)
                {
                col_sample[$i]=i
                }
            l=asorti(col_sample,b)
            \
            printf $1 "\t" $2 "\t" $3
            \
            for(i=1; i<l; i++)
                {
                printf b[i] "\t"
                }
            printf b[l] "\n"
            }
        if(NR>=2)
            {
            printf $1"\t" $2"\t" $3
            \
            for(i=1; i<l; i++)
                {
```

```
                printf $col_sample[b[i]] "\t"
                }
            printf $col_sample[b[l]] "\n"
            }
        }' $mzmine_path)
    echo "$mzmine_data" > $savepath
    fi;
```

```
#############
climit="0.95"
numlimit="0.2"
data1="results/canopus_pp_filter.tsv"
data2="r_network/mzmine_table.tsv"
compar1="Raw"
compar2="Pro"
log=10
log_to=2
stat=$(awk -F $'\t' -v OFS=$'\t' '
    {
    if(NR==1)
        {
        for(i=2; i<=NF; i++)
            {
            class[i]=$i
            num[i]=0
            }
        }
    if(NR>=2)
        {
        for(i=2; i<=NF; i++)
            {
            if($i>='$climit')
                {
                num[i]+=1;
                id[i,NR]=$1;
                }
            }
        }
    }
    END{
    for(i=2; i<=NF; i++)
        {
```

```
        p=num[i]/(NR-1)
        \
        if(p<='$numlimit') #
            {
            printf class[i]"\t"  num[i]"\t"
            \
            for(j=2; j<=NR; j++)
                {
                if(id[i,j]!="")
                    {
                    printf id[i,j]"@";
                    }
                }
            printf "\n"
            }
        }
    }' $data1 | sed 's/@$//g')
awk -F ["\t"@] -v OFS=$'\t' '
    BEGIN{
        maxNF=0
        \
        rows=0
        \
        compar1n=0
        \
        compar2n=0
        }
    {
    if(NR==FNR)
        {
        rows+=1
        if(maxNF<NF)
            {
            maxNF=NF
            }
        class[FNR]=$1
        \
        num[FNR]=$2
        \
        for(j=3; j<=NF; j++)
            {
```

```
            id[FNR,j]=$j
            }
        }
    else
        {
        if(FNR==1)
            {
            for(i=1; i<=NF; i++)
                {
                if(($i~/'$compar1'/))
                    {
                    colum1[i]=i
                    compar1n+=1
                    }
                if(($i~/'$compar2'/))
                    {
                    colum2[i]=i
                    compar2n+=1
                    }
                if(($i~/retention/))
                    {
                    rtcolum=i
                    }
                if(($i~/m\/z/))
                    {
                    mzcolum=i
                    }
                }
            }
        if(FNR>=2)
            {
            sum[1,$1]=0
            \
            sum[2,$1]=0
            \
            rt[$1]=$rtcolum
            \
            mz[$1]=$mzcolum
            \
            for(i=1; i<=NF; i++)
                {
```

```
            if(i==colum1[i] && colum1[i]!="")
                {
                sum[1,$1]+=$i
                }
            if(i==colum2[i] && colum2[i]!="")
                {
                sum[2,$1]+=$i
                }
            }
        }
    }
}
END{
    printf "class\t"  "id\t"  "log'$log'_raw\t"  "log'$log'_pro\t" \
    \
    "log'$log'_delta_area\t"  "pro_to_raw\t"  "log'$log_to'_pro_to_raw\t"  "variety\t"  "number\t"
    \
    printf "rt\t"  "m/z\n" >> "boxplot.tsv"
    \
    for(i=1; i<=rows; i++)
        {
        for(j=3; j<=maxNF; j++)
            {
            if(id[i,j]!="")
                {
                raw=(sum[1,id[i,j]]/compar1n)
                \
                pro=(sum[2,id[i,j]]/compar2n)
                \
                log_raw=log(raw)/log('$log')
                \
                log_pro=log(pro)/log('$log')
                \
                delta_area=pro-raw
                \
                if(raw!=0)
                    {
                    to_raw=pro/raw
                    \
                    norm_to_raw=log(to_raw)/log('$log_to')
                    }
                }
            }
        }
    }' $data1 $data2 $data3
###################################
```

```
#### stat num
class_pp_limit=0.9
data="for_sun_$class_pp_limit.tsv"
savepath="for_violin_${class_pp_limit}.tsv"
awk -F $'\t' '
    {
    if(FNR==1)
        {
        printf $0"\n" > "'$savepath'"
        }
    num[$2]+=1
    data[FNR]=$0
    class[FNR]=$2
    }
    END{
        for(i in num)
            {
            print i,num[i]
            if(num[i]>=50)
                {
                for(j in class)
                    {
                    if(class[j]==i)
                    {
                    printf data[j]"\n" >> "'$savepath'"
                    }
                    }
                }
            }
        }' $data
#################
```

```
                    else
                        {
                        to_raw="infinity"
                        \
                        norm_to_raw="infinity"
                        }
                    if(delta_area>0)
                        {
                        norm_delta=log(delta_area)/log('$log')
                        \
                        variety="increase"
                        }
                    else if(delta_area<0)
                        {
                        norm_delta=-1*(log(-1*delta_area)/log('$log'))
                        \
                        variety="decrease"
                        }
                    else
                        {
                        norm_delta=0
                        }
                    printf class[i]"\t"  id[i,j]"\t"  log_raw"\t"  log_pro"\t" \
                    \
                    norm_delta"\t"  to_raw"\t"  norm_to_raw"\t"  variety"\t"  num[i]"\t" \
                    \
                    rt[id[i,j]]"\t"  mz[id[i,j]]"\n" >> "boxplot.tsv"
                    }
                }
            }
        }' <(echo "$stat") $data2
#################
```

2 File: blender.sh

```
source activate parl
tmux new -s parl
tmux attach -t parl
cd ~/ParlAI; python setup.py develop
python projects/personachat/scripts/kvmemnn_interactive.py
python parlai/scripts/safe_interactive.py -mf zoo:blender/blender_9B/model -t blended_skill_talk
```

3 File: cas_reformate.sh

```
###############
data="cas"
savepath="cas_arrange.tsv"
awk -F "[:]||[']ᅵ[,][ ][']||['][,][ ][\"]||[\"][,][ ][\"]||[\"][,][ ][']" '
    {
    if($1~/^\[/)
        {
        name=$1
        id_num+=1
        id[id_num]=name
        print id_num, id[id_num]
        }
    if($1~/BEGIN_compound/)
        {
        num[id_num]+=1
        getline;
        for(i=1; i<=NF; i++)
            {
            if($i ~ /[0-9](.*)-[0-9][0-9]-[0-9](.*)$/ || $i ~ /^CAS/)
                {
                data[id_num,num[id_num]]=$i
                break;
                }
            }
        }
    }
    END{
        printf "number\t"  "name\t"  "cas\n" > "'$savepath'"
        for(i=1; i<=id_num; i++)
            {
            printf i"\t"id[i] >> "'$savepath'"
            print i"\t"id[i]
            for(j=1; j<=num[i]; j++)
                {
                printf "\t"data[i,j] >> "'$savepath'"
                }
            printf "\n" >> "'$savepath'"
            }
        }' $data
```

```
##############
data="cas_arrange.tsv"
sed -i -e 's/\[//g; s/\]//g; s/{//g; s/}//g' $data
```

4 File: clamscan.sh

```
clamscan -r --bell -i /home
```

5 File: colorful_line_eucommia_bash.sh

```
datapath="/media/wizard/back/thermo_mzML_0518/EIC"
mkdir $datapath/EIC_merge
echo "" > $datapath/file.tsv
data1="$datapath/../metadata.tsv"
data2="$datapath/EIC*.mzML/*.tsv"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if(NR>=2)
        {
          total_id[FNR]=$1
        }
    }
  if(FILENAME~/intensity/)
    {
      if(FNR==1)
        {
          n=split(FILENAME,f,"[/]")
          split(f[n-1], g,"[_]")
          samplename=g[2]
          split(f[n], a, "[_]")
          id=a[1]
          if(samplename!=p_samplename)
            {
              num_sample+=1
              sample[num_sample]=samplename
            }
          p_samplename=samplename
        }
      if(FNR>=1)
        {
          data_scan[samplename,id,FNR]=$1
          if($2!="NA")
            {
              data_intensity[samplename,id,FNR]=$2
            }
        else
          {
            data_intensity[samplename,id,FNR]="0"
          }
      }
  }
if(FILENAME~/rt.tsv/)
  {
    if(FNR==1)
      {
        n=split(FILENAME,f,"[/]")
        split(f[n-1], g,"[_]")
        samplename=g[2]
        printf samplename"\n"
      }
    if(FNR>=1)
      {
        data_scan[samplename,FNR]=$1
        data_rt[samplename,FNR]=$2
        max_rows[samplename]=FNR
      }
  }
}
END{
for(i in total_id)
  {
    printf "rt\t"  "intensity\t"  "sample\n" > "'$datapath'/EIC_merge/" total_id[i] ".tsv"
    for(j in sample)
      {
        for(k=1; k<=max_rows[sample[j]]; k++)
          {
            printf data_rt[sample[j],k]"\t"  data_intensity[sample[j],total_id[i],k]"\t"  sample[j]"\n"
              >> "'$datapath'/EIC_merge/" total_id[i] ".tsv"
            }
        }
    }
}' $data1 $data2
```

```
######################################
mkdir results/EIC_rt_during
data1="/media/wizard/back/thermo_mzML_0518/EIC/metadata.tsv"
data2="results/re_neg_RT.tsv"
data3="/media/wizard/back/thermo_mzML_0518/EIC/EIC_merge/*.tsv"
savepath="results/EIC_rt_during/"
excess_time="0.1"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      group_name[$1]=$2
    }
  if(FILENAME~/results/)
    {
      if(FNR==1)
        {
          p_file=FILENAME
          for(i=1; i<=NF; i++)
            {
              if($i~/ID/)
                {
                  col_id=i
                }
              if($i~/m\/z/)
                {
                  col_mz=i
                }
              if($i~/retention/)
                {
                  col_rt=i
                }
              if($i~/start$/)
                {
                  split($i,a,"[ ]")
                  samplename=a[1]
                  # print samplename
                  col_start[samplename]=i
        }
      if($i~/end$/)
        {
          split($i,a,"[ ]")
          samplename=a[1]
          col_end[samplename]=i
        }
    }
  }
if(FNR>=2)
  {
    mz[$col_id]=$col_mz
    # print $col_mz
    center_rt[$col_id]=$col_rt
    for(i in col_start)
      {
        if($col_start[i]!="0")
          {
            rt_start[$col_id,i]=$col_start[i]
            # print $col_start[i]
          }
      else if(reference_sample[$col_id]=="")
        {
          for(j in col_start)
            {
              if($col_start[j]!="0")
                {
                  rt_start[$col_id,i]=$col_start[j]
                  reference_sample[$col_id]=j
                  break;
                }
            }
        }
    else
      {
        rt_start[$col_id,i]=$col_start[reference_sample[$col_id]]
      }
  }
for(i in col_end)
  {
    if($col_end[i]!="0")
```

```
            {
              rt_end[$col_id,i]=$col_end[i]
            }
        else
          {
            rt_end[$col_id,i]=$col_end[reference_sample[$col_id]]
          }
      }
  }
}
if(FILENAME~/EIC_merge/)
  {
    if(FNR==1)
      {
        close(p_file)
        p_file=FILENAME
        close("'$savepath'" p_id ".tsv")
        num_id+=1
        n=split(FILENAME,a,"[/]||[.]")
        id=a[n-1]
        # print id
        p_id=id
        if(num_id==1)
          {
            for(i=1; i<=NF; i++)
              {
                if($i~/^rt/)
                  {
                    col_rt=i
                  }
                if($i~/intensity/)
                  {
                    col_intensity=i
                  }
                if($i~/sample/)
                  {
                    col_sample=i
                  }
              }
          }
        printf $0"\t" "group\t" "label\t" "color\t" "mz\t" "center_rt\n" > "'$savepath'" id ".tsv"
      }
    if(FNR>=2)
      {
        rt_min=($col_rt)/60
        if(threshold[id,$col_sample]=="")
          {
            threshold[id,$col_sample]=rt_start[id,$col_sample]+(rt_end[id,$col_sample]-rt_start[id,$col_
          }
        if(rt_min>=rt_start[id,$col_sample]-"'$excess_time'" && rt_min<=rt_end[id,$col_sample]+"'$excess
          {
            if(rt_min+0>=threshold[id,$col_sample] && end_sig[id,$col_sample]!="1")
              {
                label_sig[id,$col_sample]=1
                end_sig[id,$col_sample]=1
              }
          else
            {
              label_sig[id,$col_sample]=0
            }
          if(rt_min+0>=rt_start[id,$col_sample]+0 && rt_min+0<=rt_end[id,$col_sample]+0)
            {
              color[id,$col_sample]=group_name[$col_sample]
            }
        else
          {
            color[id,$col_sample]="Non feature"
          }
        printf sprintf("%.2f",rt_min)"\t"  $col_intensity"\t"  $col_sample"\t"  group_name[$col_sample]
        start_FNR[id]+=1
        if(start_FNR[id]=="1")
          {
            printf "\t"sprintf("%.4f",mz[id])  "\t"sprintf("%.2f",center_rt[id]) >> "'$savepath'" id ".
          }
        printf "\n" >> "'$savepath'" id ".tsv"
      }
  }
}
}' $data1 $data2 $data3
```

6 File: convert_margin.sh

```
convert cluster_accuracy_bar.png \
-define trim:edges=north,east,south,west \
-background white \
-fuzz 90% \
-trim \
test.png
```

7 File: display.sh

```
read -p "The Display Mode >>> " parameter
if [ $parameter == 0 ]
then
  i=0
elif [ $parameter == 1 ]
then
  i=9
fi
xrandr -s $i
```

8 File: eucommia_analysis.sh

```
## new algorithm (transform principle)
```

```
######## here the low pp will be filtered
data="results/stat_classification.tsv"
stat=$(awk -F $'\t' '
{
  if(NR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/^id$/)
            {
              col_id=i
            }
          if($i~/^definition$/)
            {
              col_definition=i
            }
        }
    }
  if(NR>=2)
    {
      if($col_definition!="null")
        {
          id_class[$col_id]=$col_definition
          class_num[$col_definition]+=1
        }
    }
}
END{
for(i in class_num)
  {
    printf i"\t"  class_num[i]"\t"
    for(j in id_class)
      {
        if(id_class[j]==i)
          {
            printf j"@"
          }
      }
    printf "\n"
  }
}' $data | sed 's/@$//g')
data2="results/re_neg_RT.tsv"
echo "" > results/neg_RT.tsv
data3="results/neg_RT.tsv"
compar1="Raw"
compar2="Pro"
log=10
log_to=2
awk -F ["\t"@] -v OFS=$'\t' '
BEGIN{
maxNF=0
rows=0
compar1n=0
compar2n=0
}
{
  if(NR==FNR)
```

```
    {
      rows+=1
      if(maxNF<NF)
        {
          maxNF=NF
        }
      class[FNR]=$1
      num[FNR]=$2
      for(j=3; j<=NF; j++)
        {
          id[FNR,j]=$j
        }
    }
else
  {
    if(FNR==1 && FILENAME~/re_/)
      {
        for(i=1; i<=NF; i++)
          {
            if(($i~/'$compar1'/) && $i~/area/)
              {
                colum1[i]=i
                compar1n+=1
              }
            if(($i~/'$compar2'/) && $i~/area/)
              {
                colum2[i]=i
                compar2n+=1
              }
            if(($i~/retention/))
              {
                rtcolum=i
              }
            if(($i~/m\/z/))
              {
                mzcolum=i
              }
          }
      }
    if(FNR>=2 && $0!="")
      {
        if(sum[1,$1]=="" && sum[2,$1]=="") ######## revise data
          {
            sum[1,$1]=0
            sum[2,$1]=0
            rt[$1]=$rtcolum
            mz[$1]=$mzcolum
            for(i=1; i<=NF; i++)
              {
                if(i==colum1[i] && colum1[i]!="")
                  {
                    sum[1,$1]+=$i
                  }
                if(i==colum2[i] && colum2[i]!="")
                  {
                    sum[2,$1]+=$i
                  }
              }
          }
      }
  }
}
END{
printf "class\t"  "id\t"  "log'$log'_raw\t"  "log'$log'_pro\t" \
  "log'$log'_delta_area\t"  "pro_to_raw\t"  "log'$log_to'_pro_to_raw\t"  "variety\t"  "number\t" > "algo
  printf "rt\t"  "m/z\n" >> "algorithm.tsv"
  for(i=1; i<=rows; i++)
    {
      for(j=3; j<=maxNF; j++)
        {
          if(id[i,j]!="")
            {
              raw=(sum[1,id[i,j]]/compar1n)
              pro=(sum[2,id[i,j]]/compar2n)
              log_raw=log(raw)/log('$log')
              log_pro=log(pro)/log('$log')
              delta_area=pro-raw
              if(raw!=0)
                {
                  to_raw=pro/raw
                  norm_to_raw=log(to_raw)/log('$log_to')
                }
      else
       {
        to_raw="infinity"
        norm_to_raw="infinity"
       }
      if(delta_area+0>0)
       {
        norm_delta=log(delta_area)/log('$log')
        variety="increase"
       }
     else if(delta_area<0)
      {
       norm_delta=-1*(log(-1*delta_area)/log('$log'))
       variety="decrease"
      }
    else
     {
      norm_delta=0
     }
    printf class[i]"\t"  id[i,j]"\t"  log_raw"\t"  log_pro"\t" \
     norm_delta"\t"  to_raw"\t"  norm_to_raw"\t"  variety"\t"  num[i]"\t" \
     rt[id[i,j]]"\t"  mz[id[i,j]]"\n" >> "algorithm.tsv"
    }
  }
}
}' <(echo "$stat") $data2 $data3
```

```
#######################
data="algorithm.tsv"
awk -F $'\t' -v OFS=$'\t' '
{
 if(NR==1)
  {
   print "id","rt","m/z","classification","variety","pro/raw","log10_raw","log10_pro","norm_delta" >
   for(i=1; i<=NF; i++)
    {
     if($i~/^id/)
      {
       col_id=i
      }
     if($i~/^rt/)
      {
              col_rt=i
            }
          if($i~/m\/z/)
            {
              col_mz=i
            }
          if($i~/class/)
            {
              col_class=i
            }
          if($i~/variety/)
            {
              col_variety=i
            }
          if($i~/^pro_to_raw/)
            {
              col_ratio=i
            }
          if($i~/log10_raw/)
            {
              col_log10_raw=i
            }
          if($i~/log10_pro/)
            {
              col_log10_pro=i
            }
          if($i~/delta_area/)
            {
              col_delta=i
            }
        }
    }
  if(NR>=2)
    {
      print $col_id, $col_rt, sprintf("%.5f",$col_mz), $col_class, $col_variety, $col_ratio, $col_log10
        $col_delta \
        >> "compound.tsv"
      }
  }' $data
```

```
########################
data="results/fingerid_first_score.tsv"
list="compound.tsv"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/^id/)
                {
                  col_id=i
                }
              if($i~/similarity/)
                {
                  col_simi=i
                }
              if($i~/name/)
                {
                  col_name=i
                }
              if($i~/formula/)
                {
                  col_formula=i
                }
              if($i~/^inchi$/)
                {
                  col_inchi=i
                }
              if($i~/smiles/)
                {
                  col_smiles=i
                }
              if($i~/score/)
                {
                  col_score=i
                }
              if($i~/xlogp/)
                {
```

```
                col_xlogp=i
              }
            if($i~/inchikey2D/)
              {
                col_inchikey2D=i
              }
            if($i~/links/)
              {
                col_links=i
              }
          }
      }
    if(FNR>=2)
      {
        simi[$col_id]=$col_simi
        name[$col_id]=$col_name
        formula[$col_id]=$col_formula
        smiles[$col_id]=$col_smiles
        inchi[$col_id]=$col_inchi
        inchikey2D[$col_id]=$col_inchikey2D
        score[$col_id]=$col_score
        xlogp[$col_id]=$col_xlogp
        links[$col_id]=$col_links
      }
  }
if(NR!=FNR)
  {
    if(FNR==1)
      {
        for(i=1; i<=NF; i++)
          {
            if($i~/^id/)
              {
                col_list_id=i
              }
            printf $i"\t" > "com_compound.tsv"
          }
        printf "similarity\t"  "name\t"  "formula\t"  "xlogp\t"  "smiles\t" \
          "inchi\t"  "inchikey2D\t"  "links\n" >> "com_compound.tsv"
        }
      if(FNR>=2)
```

```
          {
            for(i=1; i<=NF; i++)
              {
                printf $i"\t" >> "com_compound.tsv"
              }
            printf simi[$col_list_id]"\t"  name[$col_list_id]"\t"  formula[$col_list_id]"\t"  xlogp[$col
              smiles[$col_list_id]"\t" inchi[$col_list_id]"\t"  inchikey2D[$col_list_id]"\t" \
              links[$col_list_id]"\n" >> "com_compound.tsv"
            }
        }
    }' $data $list
  ##############################
  #search
  data="com_compound.tsv"
  savename="com_lignans_and_iridoids.tsv"
  awk -F $'\t' '
  {
    if(FNR==1)
      {
        for(i=1; i<=NF; i++)
          {
            if($i~/class/)
              {
                col_classification=i
              }
          }
        printf $0"\n" > "'$savename'"
      }
    if(FNR>=2)
      {
        if(tolower($col_classification)~/iridoid/)
          {
            printf $0"\n" >> "'$savename'"
          }
        if(tolower($col_classification)~/lignan/)
          {
            printf $0"\n" >> "'$savename'"
          }
      }
  }' $data
### filter_class
```

```
###########################
```

```
###########################
```

```
###################

 ########################
 ###sun.tsv
 ###matrix
 #   id classification  log10_raw   log10_pro
 #  147 Lignans, neolignans and related compounds   6.11895 6.51289
 #  147 Furofuran lignans   6.11895 6.51289
 #  147 O-methylated flavonoids 6.11895 6.51289
 #  147 Coumaric acids and derivatives  6.11895 6.51289
 #  147 Amino acids and derivatives 6.11895 6.51289
 #  147 Terpene glycosides  6.11895 6.51289
 data="for_sun.tsv"
 awk -F $'\t' '
 {
   if(NR==1)
     {
       for(i=1; i<=NF; i++)
         {
           if($i~/^id/)
             {
               col_id=i
             }
           if($i~/class/)
             {
               col_class=i
             }
           if($i~/log10_raw/)
             {
               col_log_raw=i
             }
           if($i~/log10_pro/)
             {
               col_log_pro=i
             }
         }
     }
   if(NR>=2)
```

```
     {
       id[FNR]=$col_id
       class[FNR]=$col_class
       uniq_class[$col_class]=$col_class
       log_raw[FNR]=$col_log_raw
       log_pro[FNR]=$col_log_pro
     }
 }
END{
printf "name\t"  "group\t"
for(i=2; i<=NR; i++)
  {
    if(i<NR)
      {
        printf "raw_"id[i]"\t"  "pro_"id[i]"\t"
      }
    if(i==NR)
      {
        printf "raw_"id[i]"\t"  "pro_"id[i]"\n"
      }
  }
for(i in uniq_class)
  {
    printf uniq_class[i]"\t"  "NA\t"
    {
      for(j=2; j<=NR; j++)
        {
          if(j<NR)
            {
              if(class[j]==uniq_class[i])
                {
                  if(log_raw[j]!="-inf")
                    {
                      printf log_raw[j]*(-1)"\t"  log_pro[j]"\t"
                    }
                else
                  {
                    printf 0"\t"  log_pro[j]"\t"
                  }
              }
          else
```

```
            {
              printf "0\t"  "0\t"
            }
        }
    else
      {
        if(class[j]==uniq_class[i])
          {
            if(log_raw[j]!="-inf")
              {
                printf log_raw[j]*(-1)"\t"  log_pro[j]"\n"
              }
          else
            {
              printf 0"\t"  log_pro[j]"\n"
            }
        }
    else
      {
        printf "0\t"  "0\n"
      }
  }
}
}
}
}
' $data > sun.tsv
```

```
##########
sort -t $'\t' -k 7 -n com_lignans_and_iridoids.tsv > test.tsv
awk -F $'\t' '
{
  if(NR==1)
    {
      printf "rank\t"  "log10_pro\t"  $0"\n"
      rank=255
      for(i=1; i<=NF; i++)
        {
          if($i~/pro\/raw/)
            {
              col_ratio=i
            }
          if($i~/log10_raw/)
            {
              col_log_raw=i
            }
        }
    }
  if(NR>=2)
    {
      printf rank"\t"  $col_log_raw+log($col_log_raw)/log(10)"\t"  $0"\n"
      rank-=1
    }
}' test.tsv > rank.tsv
```

```
######################################
##xcms
data="com_lignans_and_iridoids.tsv"
savepath="../thermo_mzML_0518/EIC_metadata.tsv"
awk -F $'\t' '
{
  if(NR==1)
    {
      for(i=i; i<=NF; i++)
        {
          if($i~/^id$/)
            {
              col_id=i
            }
          if($i~/m\/z/)
            {
              col_mz=i
            }
        }
      printf "id\t"  "m/z\n" > "'$savepath'"
    }
  if(NR>=2)
    {
      printf $col_id"\t"  $col_mz"\n" >> "'$savepath'"
    }
}' $data
```

```
######################################
```

```
######################################
#peak during time Correction
data1="results/neg_RT.tsv"
data2="results/0924_neg_RT.tsv"
savepath="results/re_neg_RT.tsv"
mz_tolerance=0.005
rt_tolerance=0.1
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/ID/)
                {
                  col_id=i
                }
              if($i~/retention/)
                {
                  col_rt=i
                }
              if($i~/m\/z/)
                {
                  col_mz=i
                }
            }
        }
      if(FNR>=2)
        {
          data1_mz[$col_id]=$col_mz
          data1_rt[$col_id]=$col_rt
          set[$col_id]=$col_id
          dataset[$col_id]=$0
        }
    }
  if(NR>FNR)
    {
      if(FNR==1)
        {
```

```
        printf $0"\n" > "'$savepath'"
        for(i=1; i<=NF; i++)
          {
            if($i~/ID/)
              {
                col_id=i
              }
            if($i~/retention/)
              {
                col_rt=i
              }
            if($i~/m\/z/)
              {
                col_mz=i
              }
          }
      }
    if(FNR>=2)
      {
        data2_mz[$col_id]=$col_mz
        data2_rt[$col_id]=$col_rt
        for(i in data1_mz)
          {
            if(data1_mz[i]<=$col_mz+"'$mz_tolerance'" && data1_mz[i]>=$col_mz-"'$mz_tolerance'")
              {
                if(data1_rt[i]<=$col_rt+"'$rt_tolerance'" && data1_rt[i]>=$col_rt-"'$rt_tolerance'")
                  {
                    data1_num[i]+=1
                    delete set[i];
                    print "data1",i,">>>","data2",$col_id,">>>",data1_num[i]
                    printf i"\t" data1_mz[i]"\t" data1_rt[i] >> "'$savepath'"
                    for(j=4; j<=NF; j++)
                      {
                        printf "\t"$j >> "'$savepath'"
                      }
                    printf "\n" >> "'$savepath'"
                  }
              }
          }
      }
  }
```

```
}
END{
for(i in set)
  {
    printf dataset[i]"\n" >> "'$savepath'"
    printf set[i]"\n"
  }
}' $data1 $data2
```

```
######################################
datapath="/media/wizard/back/thermo_mzML_0518/EIC"
mkdir $datapath/EIC_merge
echo "" > $datapath/file.tsv
data1="$datapath/../metadata.tsv"
data2="$datapath/EIC*.mzML/*.tsv"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if(NR>=2)
        {
          total_id[FNR]=$1
        }
    }
  if(FILENAME~/intensity/)
    {
      if(FNR==1)
        {
          n=split(FILENAME,f,"[/]")
          split(f[n-1], g,"[_]")
          samplename=g[2]
          split(f[n], a, "[_]")
          id=a[1]
          if(samplename!=p_samplename)
            {
              num_sample+=1
              sample[num_sample]=samplename
            }
          p_samplename=samplename
        }
      if(FNR>=1)
        {
          data_scan[samplename,id,FNR]=$1
          if($2!="NA")
            {
              data_intensity[samplename,id,FNR]=$2
            }
        else
          {
            data_intensity[samplename,id,FNR]="0"
          }
      }
  }
if(FILENAME~/rt.tsv/)
  {
    if(FNR==1)
      {
        n=split(FILENAME,f,"[/]")
        split(f[n-1], g,"[_]")
        samplename=g[2]
        printf samplename"\n"
      }
    if(FNR>=1)
      {
        data_scan[samplename,FNR]=$1
        data_rt[samplename,FNR]=$2
        max_rows[samplename]=FNR
      }
  }
}
END{
for(i in total_id)
  {
    printf "rt\t"  "intensity\t"  "sample\n" > "'$datapath'/EIC_merge/" total_id[i] ".tsv"
    for(j in sample)
      {
        for(k=1; k<=max_rows[sample[j]]; k++)
          {
            printf data_rt[sample[j],k]"\t"  data_intensity[sample[j],total_id[i],k]"\t"  sample[j]"\n"
              >> "'$datapath'/EIC_merge/" total_id[i] ".tsv"
            }
        }
    }
```

```
}' $data1 $data2
```

```
########################################
mkdir results/EIC_rt_during
data1="/media/wizard/back/thermo_mzML_0518/EIC/metadata.tsv"
data2="results/re_neg_RT.tsv"
data3="/media/wizard/back/thermo_mzML_0518/EIC/EIC_merge/*.tsv"
savepath="results/EIC_rt_during/"
excess_time="0.1"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      group_name[$1]=$2
    }
  if(FILENAME~/results/)
    {
      if(FNR==1)
        {
          p_file=FILENAME
          for(i=1; i<=NF; i++)
            {
              if($i~/ID/)
                {
                  col_id=i
                }
              if($i~/m\/z/)
                {
                  col_mz=i
                }
              if($i~/retention/)
                {
                  col_rt=i
                }
              if($i~/start$/)
                {
                  split($i,a,"[ ]")
                  samplename=a[1]
                  # print samplename
                  col_start[samplename]=i
                }
              if($i~/end$/)
                {
                  split($i,a,"[ ]")
                  samplename=a[1]
                  col_end[samplename]=i
                }
            }
        }
      if(FNR>=2)
        {
          mz[$col_id]=$col_mz
          # print $col_mz
          center_rt[$col_id]=$col_rt
          for(i in col_start)
            {
              if($col_start[i]!="0")
                {
                  rt_start[$col_id,i]=$col_start[i]
                  # print $col_start[i]
                }
            else if(reference_sample[$col_id]=="")
              {
                for(j in col_start)
                  {
                    if($col_start[j]!="0")
                      {
                        rt_start[$col_id,i]=$col_start[j]
                        reference_sample[$col_id]=j
                        break;
                      }
                  }
              }
          else
            {
              rt_start[$col_id,i]=$col_start[reference_sample[$col_id]]
            }
        }
      for(i in col_end)
        {
          if($col_end[i]!="0")
            {
              rt_end[$col_id,i]=$col_end[i]
```

```
            }
        else
          {
            rt_end[$col_id,i]=$col_end[reference_sample[$col_id]]
          }
      }
  }
}
if(FILENAME~/EIC_merge/)
  {
    if(FNR==1)
      {
        close(p_file)
        p_file=FILENAME
        close("'$savepath'" p_id ".tsv")
        num_id+=1
        n=split(FILENAME,a,"[/]||[.]")
        id=a[n-1]
        # print id
        p_id=id
        if(num_id==1)
          {
            for(i=1; i<=NF; i++)
              {
                if($i~/^rt/)
                  {
                    col_rt=i
                  }
                if($i~/intensity/)
                  {
                    col_intensity=i
                  }
                if($i~/sample/)
                  {
                    col_sample=i
                  }
              }
          }
        printf $0"\t" "group\t" "label\t" "color\t" "mz\t" "center_rt\n" > "'$savepath'" id ".tsv"
      }
    if(FNR>=2)
```

```
      {
        rt_min=($col_rt)/60
        if(threshold[id,$col_sample]=="")
          {
            threshold[id,$col_sample]=rt_start[id,$col_sample]+(rt_end[id,$col_sample]-rt_start[id,$col_
          }
        if(rt_min>=rt_start[id,$col_sample]-"'$excess_time'" && rt_min<=rt_end[id,$col_sample]+"'$exces
          {
            if(rt_min+0>=threshold[id,$col_sample] && end_sig[id,$col_sample]!="1")
              {
                label_sig[id,$col_sample]=1
                end_sig[id,$col_sample]=1
              }
          else
            {
              label_sig[id,$col_sample]=0
            }
          if(rt_min+0>=rt_start[id,$col_sample]+0 && rt_min+0<=rt_end[id,$col_sample]+0)
            {
              color[id,$col_sample]=group_name[$col_sample]
            }
        else
          {
            color[id,$col_sample]="Non feature"
          }
        printf sprintf("%.2f",rt_min)"\t"  $col_intensity"\t"  $col_sample"\t"  group_name[$col_sample]
        start_FNR[id]+=1
        if(start_FNR[id]=="1")
          {
            printf "\t"sprintf("%.4f",mz[id])  "\t"sprintf("%.2f",center_rt[id]) >> "'$savepath'" id ".t
          }
        printf "\n" >> "'$savepath'" id ".tsv"
      }
  }
}
}' $data1 $data2 $data3
```

```
##############################
```

```
#### violin plot
mkdir multi_pp_class
```

```
###################################
data1="filter_0_class.tsv" #lignans and iridoids
data2="results/canopus_pp.tsv"
data3="com_compound.tsv"
ex_export="com_1011.tsv"
similarity_limit="0.4"
```

```
###################################
for i in 0.4 0.5 0.6 0.7 0.8 0.9 0.95 0.99
do
  class_pp_limit=$i
  #class_pp_limit="0.9"
  savename="multi_pp_class/for_sun_$class_pp_limit.tsv"
  awk -F $'\t' '
  {
    if(NR==FNR)
      {
        filter_class[$1]=$2
      }
    if(FILENAME~/canopus/)
      {
        if(FNR==1)
          {
            col_id=1
            for(i=2; i<=NF; i++)
              {
                for(j in filter_class)
                  {
                    if(j==$i)
                      {
                        col_class[j]=i
                        print i
                      }
                  }
              }
          }
        if(FNR>=2)
          {
            for(i in col_class)
              {
                if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
                  {
```

```
                class_set[i,$col_id]=i
                print i,$col_class[i]
              }
            }
          }
        }
      if(FILENAME~/'$data3'/)
        {
          if(FNR==1)
            {
              for(i=1; i<=NF; i++)
                {
                  if($i~/^id/)
                    {
                      col_id=i
                    }
                  if($i~/log10_raw/)
                    {
                      col_log_raw=i
                    }
                  if($i~/log10_pro/)
                    {
                      col_log_pro=i
                    }
                  if($i~/similarity/)
                    {
                      col_similarity=i
                    }
                }
              printf "id\t"  "classification\t"  "log10_raw\t"  "log10_pro\n" > "'$savename'"
              printf $0"\n" > "'$ex_export'"
            }
          if(FNR>=2)
            {
              if($col_similarity+0 >= "'$similarity_limit'"+0)
                {
                  for(i in class_set)
                    {
                      if(i~"\034"$col_id"$")
                        {
                          printf $col_id"\t"  class_set[i]"\t"  $col_log_raw"\t"  $col_log_pro"\n" >> "'$
                          printf $0"\n" >> "'$ex_export'"
                        }
                    }
                }
            }
        }
    }' $data1 $data2 $data3
  done
```

```
###################################
```

```
#### stat num
for i in 0.4 0.5 0.6 0.7 0.8 0.9 0.95 0.99
do
  class_pp_limit=$i
  data="multi_pp_class/for_sun_$class_pp_limit.tsv"
  savepath="multi_pp_class/for_violin_${class_pp_limit}_pattern.tsv"
  awk -F $'\t' '
  {
    if(FNR==1)
      {
        printf $0"\n" > "'$savepath'"
      }
    num[$2]+=1
    data[FNR]=$0
    class[FNR]=$2
  }
END{
for(i in num)
  {
    print i,num[i]
    if(num[i]+0>=50)
      {
        for(j in class)
          {
            if(class[j]==i)
              {
                printf data[j]"\n" >> "'$savepath'"
              }
          }
      }
  }
}' $data
done;
```

```
#################
```

```
######################################  the former is network_facet
for i in 0.4 0.5 0.6 0.7 0.8 0.9 0.95 0.99
do
  class_pp_limit=$i
  mkdir results/network_facet_ladder2_$class_pp_limit
  data1="multi_pp_class/for_violin_${class_pp_limit}_pattern.tsv"
  data2="results/source_target_tree_0.4.tsv" # "results/source_target_tree_0.4.tsv"
  save_class="results/filter_child_class.tsv"
  savepath="results/network_facet_ladder2_$class_pp_limit/"
  awk -F $'\t' '
  {
    if(NR==FNR)
      {
        if(FNR==1)
          {
            for(i=1; i<=NF; i++)
              {
                if($i~/classification/)
                  {
                    col_class=i
                  }
                if($i~/^id$/)
                  {
                    col_id=i
                  }
              }
          }
        if(FNR>=2)
          {
            class[$col_class]=$col_class
            class_id[$col_class,$col_id]=$col_id
            stat_id[$col_class,$col_id]=$col_id
            belong[$col_class,$col_id]=$col_class
          }
      }
    if(NR>FNR)
      {
        if(FNR==1)
          {
            printf "" > "'$save_class'"
            for(i in class)
              {
                printf i"\n" >> "'$save_class'"
                printf $0"\t"  "facet\n" > "'$savepath'" i ".tsv"
              }
          }
        if(FNR>=2)
          {
            for(i in class)
              {
                if( class_id[i,$1]==$1 && class_id[i,$2]==$2 )
                  {
                    printf $0"\t"  i"\n" >> "'$savepath'" i ".tsv"
                    delete stat_id[i,$1]
                    delete stat_id[i,$2]
                  }
              }
          }
      }
  }
END{
for(i in stat_id)
  {
    ## source target similarity delta_mz fp fp class
    printf stat_id[i]"\t" stat_id[i]"\t"  "1\t"  "0\t"  "null\t"  "null\t"  belong[i]"\n" >> "'$savepat
  }
}' $data1 $data2
done;
```

```
#################################
### for ring plot
data1="canopus.tsv"
data2="results/filter_child_class.tsv"
data3="results/canopus_pp.tsv"
savepath="results/canopus_pp_filter.tsv"
awk -F $'\t' '
{
  if(FILENAME~/canopus.tsv/)
    {
```

```
    if(FNR==1)
      {
        for(i=1; i<=NF; i++)
          {
            if($i~/absolute/)
              {
                col_index=i
              }
            if($i~/^id/)
              {
                col_chemid=i
              }
            if($i~/name/)
              {
                col_name=i
              }
            if($i~/description/)
              {
                col_des=i
              }
          }
      }
    if(FNR>2)
      {
        ab_index[$col_name]=$col_index
        chemid[$col_name]=$col_chemid
        des[$col_name]=$col_des
      }
  }
if(FILENAME~/filter_child_class/)
  {
    class[$1]=$1
    if(FNR==1)
      {
        printf "index\t"  "chem_id\t"  "name\t"  "description\n" > "results/child_class.tsv"
      }
    printf ab_index[$1]"\t"  chemid[$1]"\t"  $1"\t"  des[$1]"\n" >> "results/child_class.tsv"
  }
if(FILENAME~/canopus_pp/)
  {
    if(FNR==1)
```

```
        {
          printf $1 > "'$savepath'"
          for(i=2; i<=NF; i++)
            {
              if(class[$i]!="")
                {
                  n+=1
                  printf "\tC"ab_index[$i] >> "'$savepath'"
                  col_set[n]=i
                }
            }
          printf "\n" >> "'$savepath'"
        }
      if(FNR>=2)
        {
          printf $1 >> "'$savepath'"
          for(i=1; i<=n; i++)
            {
              printf "\t"$col_set[i] >> "'$savepath'"
            }
          printf "\n" >> "'$savepath'"
        }
    }
}' $data1 $data2 $data3
```

```
#################################
## Eucommia peak erea normalized
# accoding: com_lignans_and_iridoids.tsv
data=com_lignans_and_iridoids.tsv
awk -F $'\t' '
{
}' $data
```

```
#### ############  instance for 3d plot
### step1 RT ~ intensity
Rscript ~/Downloads/codes/instance_3d_xcms.R
```

```
#################  json tree
mkdir json_tree
data1="com_lignans_and_iridoids.tsv"
id=2268
formula=C17H24O10
data2="/media/wizard/back/0703_all/*_$id/trees/$formula*.json" ####### in media
```

```
cp $data2 json_tree/tree_$id.json
# id    rt  m/z classification  variety pro/raw
# 3918  13.2883753333333    701.22926   Terpene glycosides  increase    52.5192
# 2529  11.5328771666667    699.24926   Lignan glycosides   increase    4.73482
# 3674  7.13257936666667    551.16124   Iridoid O-glycosides    increase    3.91387
# 3380  12.6588768333333    613.21304   Terpene glycosides  increase    3.9104
data="json_tree/tree_$id.json"
savepath="json_tree/"
awk -F "[ ][:][ ]|||[,]" '
{
  if($0~/"root"/)
    {
      root=$2
    }
  if($0~/"id"/)
    {
      id[$2]=$2;
      the_id=$2
      getline;
      formula[the_id]=$2
    }
  if($0~/"source"/)
    {
      source[$2]=$2
      link[$2]+=1
      the_source=$2
      getline;
      target[the_source, link[the_source]]=$2
      getline;
      formula_edge[the_source, link[the_source]]=$2
    }
}
END{
printf "id\t" "label\n" > "'$savepath'"  "nodes_'$id'.tsv"
printf "from\t" "to\t" "label\n" > "'$savepath'"  "edges_'$id'.tsv"
```

```
#####
printf "root\t" root"\n" >> "'$savepath'"  "nodes_'$id'.tsv"
for(i in id)
  {
    printf i"\t" formula[i]"\n" >> "'$savepath'"  "nodes_'$id'.tsv"
  }
```

```
for(i in source)
  {
    for(j=1; j<=link[i]; j++)
      {
        printf i"\t"  target[i,j]"\t" formula_edge[i,j]"\n" >> "'$savepath'"  "edges_'$id'.tsv"
      }
  }
}' $data
```

```
######################
Rscript ~/Downloads/codes/json_tree.R
```

```
######################
## Image reshape
```

```
######################
metadata="canopus_neg.tsv"
#data="/media/wizard/back/0703_all/490_initial_8_neg_495/canopus/C17H24O10_[M-H]-.fpt"
savepath="canopus_parent_index.tsv"
awk -F $'\t' '
{
  if(FNR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/^id/)
            {
              col_id=i
            }
          if($i~/^parent/)
            {
              col_parent=i
            }
        }
    }
  if(FNR>=2)
    {
      parent[$col_id]=$col_parent
      rows[FNR]=$col_id
      #print rows[FNR]
    }
}
END{
```

```
print "END"
printf "id\t"  "parentid\t"  "num\n" > "'$savepath'"
for(i=2; i<=FNR; i++)
  {
    root=rows[i]
    num[root]+=1
    index_id[rows[i]]=root
    while(root!="")
      {
        if(parent[root]!="")
          {
            index_id[rows[i]]=parent[root]"-"index_id[rows[i]]
            num[parent[root]]+=1
          }
        root=parent[root]
      }
  }
for(i=2; i<=FNR; i++)
  {
    printf rows[i]"\t"  index_id[rows[i]]"\t"  num[rows[i]]"\n" > "'$savepath'"
  }
}' $metadata
```

```
######################
```

```
#### 1028 violin plot
```

```
###################################
data1="results/filter_child_class.tsv"
data2="results/canopus_pp.tsv"
data3="com_compound.tsv"
ex_export="com_1011.tsv"
similarity_limit="0.4"
```

```
###################################
class_pp_limit="0.5"
savename="for_sun_$class_pp_limit.tsv"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      filter_class[$1]=$2
    }
```

```
if(FILENAME~/canopus/)
  {
    if(FNR==1)
      {
        col_id=1
        for(i=2; i<=NF; i++)
          {
            for(j in filter_class)
              {
                if(j==$i)
                  {
                    col_class[j]=i
                    print i
                  }
              }
          }
      }
    if(FNR>=2)
      {
        for(i in col_class)
          {
            if(sprintf("%.3f",$col_class[i])+0 > "'$class_pp_limit'"+0)
              {
                class_set[i,$col_id]=i
                print i,$col_class[i]
              }
          }
      }
  }
if(FILENAME~/'$data3'/)
  {
    if(FNR==1)
      {
        for(i=1; i<=NF; i++)
          {
            if($i~/^id/)
              {
                col_id=i
              }
            if($i~/log10_raw/)
              {
                  col_log_raw=i
                }
              if($i~/log10_pro/)
                {
                  col_log_pro=i
                }
              if($i~/similarity/)
                {
                  col_similarity=i
                }
            }
          printf "id\t"  "classification\t"  "log10_raw\t"  "log10_pro\n" > "'$savename'"
          printf $0"\n" > "'$ex_export'"
        }
      if(FNR>=2)
        {
          if($col_similarity+0 > "'$similarity_limit'"+0)
            {
              for(i in class_set)
                {
                  if(i~"\034"$col_id"$")
                    {
                      printf $col_id"\t"  class_set[i]"\t"  $col_log_raw"\t"  $col_log_pro"\n" >> "'$sa
                      printf $0"\n" >> "'$ex_export'"
                    }
                }
            }
        }
    }
}' $data1 $data2 $data3
```

```
###################################
```

```
#### stat num
class_pp_limit=0.5
data="for_sun_$class_pp_limit.tsv"
savepath="for_violin_${class_pp_limit}.tsv"
awk -F $'\t' '
{
  if(FNR==1)
    {
      printf $0"\n" > "'$savepath'"
    }
  num[$2]+=1
  data[FNR]=$0
  class[FNR]=$2
}
END{
for(i in num)
  {
    print i,num[i]
    if(num[i]+0>=10)
      {
        for(j in class)
          {
            if(class[j]==i)
              {
                printf data[j]"\n" >> "'$savepath'"
              }
          }
      }
  }
}' $data
```

```
#################
```

```
#################
data1="filter_class.csv" #lignans and iridoids
data2="results/canopus_pp.tsv"
data3="com_compound.tsv"
ex_export="results/com_lignans_and_iridoids.tsv"
similarity_limit="0.4"
class_pp_limit="0.5"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      filter_class[$1]=$1
    }
  if(FILENAME~/canopus/)
    {
      if(FNR==1)
        {
          col_id=1
          for(i=2; i<=NF; i++)
            {
              for(j in filter_class)
                {
                  if(j==$i)
                    {
                      col_class[j]=i
                      print i
                    }
                }
            }
        }
      if(FNR>=2)
        {
          for(i in col_class)
            {
              if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
                {
                  class_set[i,$col_id]=i
                  print i,$col_class[i]
                }
            }
        }
    }
  if(FILENAME~/'$data3'/)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/^id/)
                {
                  col_id=i
                }
              if($i~/log10_raw/)
                {
                  col_log_raw=i
                }
              if($i~/log10_pro/)
                {
                  col_log_pro=i
                }
              if($i~/similarity/)
                {
                  col_similarity=i
                }
            }
          printf $0"\n" > "'$ex_export'"
        }
      if(FNR>=2)
        {
          if($col_similarity+0 >= "'$similarity_limit'")
            {
              for(i in class_set)
                {
                  if(i~"\034"$col_id"$")
                    {
                      printf $0"\n" >> "'$ex_export'"
                    }
                }
            }
        }
    }
}' $data1 $data2 $data3
```

9 File: for_com_lignans_and_iridoids.sh

```
list=$(awk '{print NR}' filter_class.csv)
for i in $list
do
  data1=$(awk '{if(NR=='$i'){printf $0}}' filter_class.csv) #lignans and iridoids
  data2="results/canopus_pp.tsv"
  data3="com_compound.tsv"
  ex_export="results/com_${i}.tsv"
  similarity_limit="0.4"
  class_pp_limit="0.5"
  awk -F $'\t' '
  {
    if(NR==FNR)
      {
        filter_class[$1]=$1
        index_class=$1
      }
    if(FILENAME~/canopus/)
      {
        if(FNR==1)
          {
            col_id=1
            for(i=2; i<=NF; i++)
              {
                for(j in filter_class)
                  {
                    if(j==$i)
                      {
                        col_class[j]=i
                        print i
                      }
                  }
              }
          }
        if(FNR>=2)
          {
            for(i in col_class)
              {
                if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
                  {
                    class_set[i,$col_id]=i
                    print i,$col_class[i]
                  }
              }
          }
      }
    if(FILENAME~/'$data3'/)
      {
        if(FNR==1)
          {
            for(i=1; i<=NF; i++)
              {
                if($i~/^id/)
                  {
                    col_id=i
                  }
                if($i~/log10_raw/)
```

```
                  {
                    col_log_raw=i
                  }
                if($i~/log10_pro/)
                  {
                    col_log_pro=i
                  }
                if($i~/similarity/)
                  {
                    col_similarity=i
                  }
              }
            printf "Index\t"$0"\n" > "'$ex_export'"
          }
        if(FNR>=2)
          {
            if($col_similarity+0 >= "'$similarity_limit'")
              {
                for(i in class_set)
                  {
                    if(i~"\034"$col_id"$")
                      {
                        printf index_class"\t"$0"\n" >> "'$ex_export'"
                      }
                  }
              }
          }
      }
  }' <(echo "$data1") $data2 $data3
done;
```

10 File: for_gnps_facet.sh

```
plimit=0
until [[ "$plimit" > "0.5" ]] && [[ "$plimit" < "0.99" ]]
do
  read -p "If the minimum alignment similarity is greater than 0.3, this step will automatic calculate 
done;
plimit2=0
until [[ "$plimit2" > "0.1" ]] && [[ "$plimit2" < "$plimit" ]]
do
```

```
  read -p "The minimum posterior probability of the molecular fingerprint(plimit_2) to be controled. 0.
done;
tlimit=$(ls temp/ftaligntemp/filter_net_* | awk -F _ '{print $NF}')
check_rep=$(awk 'END{print NR}' <(echo "$tlimit"))
if [[ "$check_rep" > "1" ]]
then
  tlimit=0
  until [ -f temp/ftaligntemp/filter_net_$tlimit ]
  do
    read -p "Plural and different fragment_tree_network files were found to exist locally. Please selec
  done;
fi;
if [ -f temp/ftaligntemp/refilter_net_$tlimit ]
then rm temp/ftaligntemp/refilter_net_$tlimit
fi;
if [ -f temp/ftaligntemp/fpsample ]
then rm temp/ftaligntemp/fpsample
fi;
echo "Running delta_fingerprint computation..."
echo "Aquiring data from sirius index..."
data1="*_*/compound.info"
data2="temp/ftaligntemp/filter_net_$tlimit"
echo "Run fragment_tree_network_delta."
savepath="temp/ms_data_$tlimit"
echo "step 1: ms data"
if ! [ -f $savepath ]
then
  awk -F "[\t]||[:]||[_]" -v OFS=$'\t' '
  BEGIN{
  printf "..."
  printf "id\t"  "m/z\t"  "rt\n" > "results/mz_and_rt.tsv"
}
{
  if(FILENAME~/compound.info/)
    {
      if(FNR==1)
        {
          printf "Info: catch >>> "FILENAME"\n"
        }
      if($1=="name")
        {
          i+=1;
          id[i]=$NF;
          n=split($NF,a,"[_]")
          printf a[n]"\t" >> "results/mz_and_rt.tsv"
        }
      if($1=="ionMass")
        {
          mz[id[i]]=$NF
          printf sprintf("%.4f",$NF)"\t" >> "results/mz_and_rt.tsv"
        }
      if($1=="ionType")
        {
          type[id[i]]=$NF
        }
      if($1=="rt")
        {
          rt[id[i]]=$2;
          printf sprintf("%.2f",$2/60)"\n" >> "results/mz_and_rt.tsv"
        }
    }
  if(FILENAME~/ftaligntemp/)
    {
      #source  target  ftalign  delta_mz  delta_rt  source_iontype  target_iontype;
      if(FNR==1)
        {
          printf "" > "'$savepath'"
        }
      print $1,$2,$3,sprintf("%.3f",mz[$2]-mz[$1]),sprintf("%.2f",rt[$2]-rt[$1]),type[$1],type[$2]  >>
    }
}' $data1 $data2
fi;
```

```
########################
echo "step 2: data path"
source_file="temp/Mo_filename"
data="temp/ms_data_$tlimit"
savepath="temp/datapath_$tlimit"
if ! [ -f $savepath ]
then
  awk -F $'\t' -v OFS=$'\t' '
  {
    if(NR==FNR)
```

```
      {
        n=split($2, a, "[_]")
        file[a[n]]=$2
        formu_type[a[n]]=$3
      }
    if(NR!=FNR)
      {
        #<path>sourceFormula  <path>targetFormula
        path1=file[$1]"/fingerprints/"formu_type[$1]".fpt"
        path2=file[$2]"/fingerprints/"formu_type[$2]".fpt"
        data[path1]=path1
        data[path2]=path2
      }
  }
END{
for(i in data)
  {
    n+=1
    print "Check file:",n
    if(getline<i==-1)
      {
        printf "Escape filename: " i "\n"
      }
  else
    {
      printf i" " > "'$savepath'"
    }
  close(i)
}
print "Sum:",n
}' $source_file $data
fi;
datapath=$(cat temp/datapath_$tlimit)
echo "step 3: fingerprint"
data_allfp="temp/data_allfp_$tlimit"
if ! [ -f $data_allfp ]
then
  awk -F $'\t' '
  BEGIN{
  n=0
}
```

```
{
  if(FNR==1)
    {
      if(n>1)
        {
          close(file)
        }
      file=FILENAME;
      print "Get fingerprints: ",FILENAME
      n+=1;
      printf FILENAME"\n"$0"\n" > "'$data_allfp'"
    }
else
  {
    print $0 > "'$data_allfp'"
  }
}' $datapath
fi;
```

```
############################
awk -F $'\t' -v OFS=$'\t' '
{
  if(NR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/absolute/)
            {
              col_index=i
            }
          if($i~/description/)
            {
              col_description=i
            }
        }
    }
  if(NR>=2)
    {
      print $col_index,$col_description
    }
}' csi_fingerid.tsv > temp/ftaligntemp/pos_fingerprint_index;
awk -F $'\t' -v OFS=$'\t' '
```

```
{
  if(NR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/absolute/)
            {
              col_index=i
            }
          if($i~/description/)
            {
              col_description=i
            }
        }
    }
  if(NR>=2)
    {
      print $col_index,$col_description
    }
}' csi_fingerid_neg.tsv > temp/ftaligntemp/neg_fingerprint_index;
posindex="temp/ftaligntemp/pos_fingerprint_index"
negindex="temp/ftaligntemp/neg_fingerprint_index"
data="temp/ms_data_$tlimit"
echo "step 4: merge data"
awk -F $'\t' '
BEGIN{
file=0
x=0
count=0
f=0
posnum=0
negnum=0
}
{
  if(FNR==1)
    {
      file+=1
    }
  if(NR==FNR)
    {
      if(($1~/fingerprint/))
```

```
        {
          if(x+0>f+0)
            {
              f=x  # calculate the max index.
            }
          count+=1;  # calculate the all fingerprints file number.
          split($1,a,"[/]"); e=split(a[1],b,"[_]"); id=b[e]; #catch the id.
          x=0;
        }
    else
      {
        x+=1;
        fp[id,x]=$1;
      }
  }
if(file==2)
  {
    pos[FNR]=$1
    posnum+=1
  }
if(file==3)
  {
    neg[FNR]=$1
    negnum+=1
  }
if(file==4)
  {
    if("'$tlimit'"+0 >= 0.3)
      {
        if(fp[$1,1]!="" && fp[$2,1]!="")
          {
            n1=split($6, g, "[]]");
            n2=split($7, h, "[]]");
            if(g[n1]=="+" && h[n2]=="+")
              {
                for(x=1; x<=posnum; x++)
                  {
                    if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
                      {
                        data_s[FNR,x]=pos[x]
                      }
```

```
         else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
          {
           data_t[FNR,x]=pos[x]
          }
        }
      }
     if(g[n1]=="-" && h[n2]=="-")
      {
       for(x=1; x<=negnum; x++)
        {
         if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
          {
           data_s[FNR,x]=neg[x]
          }
        else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
         {
          data_t[FNR,x]=neg[x]
         }
       }
     }
    if(g[n1]=="-" && h[n2]=="+" || h[n2]=="-" && g[n1]=="+")
     {
      for(i=1; i<=posnum; i++)
       {
        for(j=1; j<=negnum; j++)
         {
          if(pos[i]==neg[j])
           {
            mirror[i]=j  #if pos=i, the identical index of neg is mirror[i]
           }
         }
       }
     }
    if(g[n1]=="-" && h[n2]=="+")
     {
      for(x=1; x<=f; x++)
       {
        if(fp[$1,mirror[x]]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
         {
          data_s[FNR,x]=neg[mirror[x]]
         }
       else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,mirror[x]]+0<='$plimit2'+0)
        {
         data_t[FNR,x]=neg[mirror[x]]
        }
      }
    }
   if(h[n2]=="-" && g[n1]=="+")
    {
     for(x=1; x<=f; x++)
      {
       if(fp[$1,x]+0>= '$plimit'+0 && fp[$2,mirror[x]]+0<='$plimit2'+0)
        {
         data_s[FNR,x]=neg[mirror[x]]
        }
      else if(fp[$2,mirror[x]]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
       {
        data_t[FNR,x]=neg[mirror[x]]
       }
     }
   }
 }
}
if(FNR==1)
 {
  printf "source\t"  "target\t"  "ftalign_similarity\t"  "delta_m/z\t"  "source_fp_uniq\t"  "target_f
 }
else
 {
  printf $1"\t"  $2"\t"  $3"\t"  $4"\t";
  if('$tlimit'+0 >= 0.3)
   {
    if(fp[$1,1]=="" || fp[$2,1]=="")
     {
      printf "NA@NA\n"
     }
   else
    {
     printf "source:"  #the source fingerprint uniq.
     for(x=1; x<=f; x++)
      {
       if(data_s[FNR,x]!="")
```

```
                {
                  printf data_s[FNR,x]","
                }
            };
          printf "@target:"  #the target fingerprint uniq.
          for(x=1; x<=f; x++)
            {
              if(data_t[FNR,x]!="")
                {
                  printf data_t[FNR,x]","
                }
            }
          printf "\n"
        }
    }
else
  {
    printf "NA@NA\n"
  }
}
}
}' $data_allfp $posindex $negindex $data | sed -e 's/,@/\t/g; s/,$//g; s/@/\t/g'  > results/source_targe
echo "All instances have written into <results/source_target_tree_$tlimit.tsv>."
```

```
#####################################
echo "step 5: separate child-nebula from parent-nebula."
data="results/stat_classification.tsv"
savepath="temp/filter_0_class.tsv"
awk -F $'\t' '
{
  if(FNR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/^definition$/)
            {
              col_class=i
            }
        }
    }
  if(FNR>=2)
    {
      class[$col_class]=$col_class
    }
}
END{
for(i in class)
  {
    printf class[i]"\n" > "'$savepath'"
  }
}' $data
```

```
######################################
data1="temp/filter_0_class.tsv"
data2="results/canopus_pp.tsv"
data3="results/fingerid_first_score.tsv"
until [[ "$similarity_limit" > 0 ]] && [[ "$similarity_limit" < 1 ]]
do
  read -p "Please set the Tanimoto similarity threshold contribute to child-nebula. >>> " similarity_li
done;
until [[ "$definition_limit" > 0.01 ]] && [[ "$definition_limit" < 1 ]]
do
  read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>> " def
done;
class_pp_limit=$definition_limit
savepath="temp/idenfication_filter_$class_pp_limit.tsv"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      filter_class[$1]=$1
    }
  if(FILENAME~/canopus/)
    {
      if(FNR==1)
        {
          col_id=1
          for(i=2; i<=NF; i++)
            {
              for(j in filter_class)
                {
                  if(j==$i)
                    {
                      col_class[j]=i
                      print i
                    }
                }
            }
        }
      if(FNR>=2)
        {
          for(i in col_class)
            {
              if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
                {
                  class_set[i,$col_id]=i
                  print "ID: ",$1,i,$col_class[i]
                }
            }
        }
    }
  if(FILENAME~/fingerid_first_score/)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/^id/)
                {
                  col_id=i
                }
              if($i~/similarity/)
                {
                  col_similarity=i
                }
            }
          printf "class_nebula_facet\t"  $0"\n" > "'$savepath'"
        }
      if(FNR>=2)
        {
          if($col_similarity+0 >= "'$similarity_limit'"+0)
            {
              for(i in class_set)
                {
                  if(i~"\034"$col_id"$")
```

```
                    {
                      printf class_set[i]"\t"  $0"\n" > "'$savepath'"
                    }
                }
            }
        }
    }
}' $data1 $data2 $data3
```

```
###################################
data="temp/idenfication_filter_$class_pp_limit.tsv"
savepath="temp/idenfication_filter2_${class_pp_limit}.tsv"
until [[ "$num_limit_1" -gt 0 ]]
do
  read -p "Please enter the features number threshold contribute to child-nebula (min number). >>> " nu
done;
until [[ "$num_limit_2" -gt "$num_limit_1" ]]
do
  read -p "Please enter the features number threshold contribute to child-nebula (max number). >>> " nu
done;
awk -F $'\t' '
{
  if(FNR==1)
    {
      printf $0"\n" > "'$savepath'"
      for(i=1; i<=NF; i++)
        {
          if($i~/class_nebula_facet/)
            {
              col_class=i
            }
        }
    }
  if(FNR>=2)
    {
      num[$col_class]+=1
      data[FNR]=$0
      class[FNR]=$col_class
    }
}
END{
for(i in num)
```

```
  {
    if(num[i]+0 >= '$num_limit_1'+0 && num[i]+0 <= '$num_limit_2')
      {
        printf "The nodes number of the child-nebula is " num[i] ".\n"
        for(j in class)
          {
            if(class[j]==i)
              {
                printf data[j]"\n" >> "'$savepath'"
              }
          }
      }
  }
}' $data
```

```
##################################
mkdir results/network_facet_$class_pp_limit
data1="temp/idenfication_filter2_${class_pp_limit}.tsv"
data2="results/source_target_tree_$tlimit.tsv" # "results/source_target_tree_0.4.tsv"
save_class="results/filter_child_class.tsv"
savepath="results/network_facet_$class_pp_limit/"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/class_nebula_facet/)
                {
                  col_class=i
                }
              if($i~/^id$/)
                {
                  col_id=i
                }
            }
        }
      if(FNR>=2)
        {
          class[$col_class]=$col_class
          class_id[$col_class,$col_id]=$col_id
          stat_id[$col_class,$col_id]=$col_id
          belong[$col_class,$col_id]=$col_class
        }
    }
  if(NR>FNR)
    {
      if(FNR==1)
        {
          for(i in class)
            {
              printf i"\n" > "'$save_class'"
              printf $0"\t"  "facet\n" > "'$savepath'" i ".tsv"
            }
        }
      if(FNR>=2)
        {
          for(i in class)
            {
              if( class_id[i,$1]==$1 && class_id[i,$2]==$2 )
                {
                  printf $0"\t"  i"\n" >> "'$savepath'" i ".tsv"
                  delete stat_id[i,$1]
                  delete stat_id[i,$2]
                }
            }
        }
    }
}
END{
for(i in stat_id)
  {
    ## source target similarity delta_mz fp fp class
    printf stat_id[i]"\t" stat_id[i]"\t"  "1\t"  "0\t"  "null\t"  "null\t"  belong[i]"\n"  >> "'$savepa
  }
}' $data1 $data2
```

```
#####################
data1="canopus.tsv"
data2="results/filter_child_class.tsv"
data3="results/canopus_pp.tsv"
savepath="results/canopus_pp_filter.tsv"
```

```
awk -F $'\t' '
{
  if(FILENAME~/canopus.tsv/)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/absolute/)
                {
                  col_index=i
                }
              if($i~/^id/)
                {
                  col_chemid=i
                }
              if($i~/name/)
                {
                  col_name=i
                }
              if($i~/description/)
                {
                  col_des=i
                }
            }
        }
      if(FNR>2)
        {
          ab_index[$col_name]=$col_index
          chemid[$col_name]=$col_chemid
          des[$col_name]=$col_des
        }
    }
  if(FILENAME~/filter_child_class/)
    {
      class[$1]=$1
      if(FNR==1)
        {
          printf "index\t"  "chem_id\t"  "name\t"  "description\n" > "results/child_class.tsv"
        }
      printf ab_index[$1]"\t"  chemid[$1]"\t"  $1"\t"  des[$1]"\n" >> "results/child_class.tsv"
    }
  if(FILENAME~/canopus_pp/)
    {
      if(FNR==1)
        {
          printf $1 > "'$savepath'"
          for(i=2; i<=NF; i++)
            {
              if(class[$i]!="")
                {
                  n+=1
                  printf "\tC"ab_index[$i] >> "'$savepath'"
                  col_set[n]=i
                }
            }
          printf "\n" >> "'$savepath'"
        }
      if(FNR>=2)
        {
          printf $1 >> "'$savepath'"
          for(i=1; i<=n; i++)
            {
              printf "\t"$col_set[i] >> "'$savepath'"
            }
          printf "\n" >> "'$savepath'"
        }
    }
}' $data1 $data2 $data3
mv results/network_facet_$class_pp_limit results/gnps_network_facet_$class_pp_limit
```

11 File: for_violin.sh

```
data1="results/filter_child_class.tsv"
data2="results/canopus_pp.tsv"
data3="com_compound.tsv"
ex_export="com_1011.tsv"
similarity_limit="0.4"
```

```
###################################
class_pp_limit="0.5"
savename="for_sun_$class_pp_limit.tsv"
```

```
awk -F $'\t' '
{
  if(NR==FNR)
    {
      filter_class[$1]=$2
    }
  if(FILENAME~/canopus/)
    {
      if(FNR==1)
        {
          col_id=1
          for(i=2; i<=NF; i++)
            {
              for(j in filter_class)
                {
                  if(j==$i)
                    {
                      col_class[j]=i
                      print i
                    }
                }
            }
        }
      if(FNR>=2)
        {
          for(i in col_class)
            {
              if(sprintf("%.3f",$col_class[i])+0 > "'$class_pp_limit'"+0)
                {
                  class_set[i,$col_id]=i
                  print i,$col_class[i]
                }
            }
        }
    }
  if(FILENAME~/'$data3'/)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/^id/)
                {
                  col_id=i
                }
              if($i~/log10_raw/)
                {
                  col_log_raw=i
                }
              if($i~/log10_pro/)
                {
                  col_log_pro=i
                }
              if($i~/similarity/)
                {
                  col_similarity=i
                }
            }
          printf "id\t"  "classification\t"  "log10_raw\t"  "log10_pro\n" > "'$savename'"
          printf $0"\n" > "'$ex_export'"
        }
      if(FNR>=2)
        {
          if($col_similarity+0 > "'$similarity_limit'"+0)
            {
              for(i in class_set)
                {
                  if(i~"\034"$col_id"$")
                    {
                      printf $col_id"\t"  class_set[i]"\t"  $col_log_raw"\t"  $col_log_pro"\n" >> "'$sa
                      printf $0"\n" >> "'$ex_export'"
                    }
                }
            }
        }
    }
}' $data1 $data2 $data3
```

```
####################################
```

```
#### stat num
class_pp_limit=0.5
data="for_sun_$class_pp_limit.tsv"
```

```
savepath="for_violin_${class_pp_limit}.tsv"
awk -F $'\t' '
{
  if(FNR==1)
    {
      printf $0"\n" > "'$savepath'"
    }
  num[$2]+=1
  data[FNR]=$0
  class[FNR]=$2
}
END{
for(i in num)
  {
    print i,num[i]
    if(num[i]+0>=10)
      {
        for(j in class)
          {
            if(class[j]==i)
              {
                printf data[j]"\n" >> "'$savepath'"
              }
          }
      }
  }
}' $data
```

12 File: ftp_delete.sh

```
#!/bin/bash
path=$1
PUTFILE=$2
ftp -ivn <<EOF
passive
open ccms-ftp01.ucsd.edu 21
user Yellow xxk123456
binary
cd $path
prompt
delete $PUTFILE
```

```
by
EOF
echo "up file end . . ."
```

13 File: ftp_upload_target_dir.sh

```
#!/bin/bash
path=$1
PUTFILE=$2
target=$3
ftp -ivn <<EOF
passive
open ccms-ftp01.ucsd.edu 21
user Yellow xxk123456
binary
cd $target
lcd $path
prompt
put $PUTFILE
by
EOF
echo "up file end . . ."
```

14 File: ftp_upload.sh

```
#!/bin/bash
path=$1
PUTFILE=$2
ftp -ivn <<EOF
passive
open ccms-ftp01.ucsd.edu 21
user Yellow xxk123456
binary
cd .
lcd $path
prompt
put $PUTFILE
by
EOF
echo "up file end . . ."
```

15 File: git.sh

```
## Wed Mar  2 10:57:48 AM CST 2022
# git config --global user.name "chi-med-pro"
# git config --global user.email "202011114011074@zcmu.edu.cn"
## ssh-keygen -t rsa -C "202011114011074@zcmu.edu.cn"
## ssh-add ~/.ssh/id_rsa_r
git remote add origin git@github.com:chi-med-pro/MCnebula.git
git add .
git commit -m "initial submit"
git pull origin master
# git add/rm .gitignore
# git rebase --continue
git push -u origin master
```

16 File: install_name_iupuc.sh

```
# download the smi_to_iupac model
URL="https://bwsyncandshare.kit.edu/s/bRWQ4Y8bZmWnBEt/download"
GZ="./chem_iupac/download"

if [ -d "./chem_iupac/download" ]; then
    echo "Dataset has already been downloaded."
else
    wget "$URL" -P "./chem_iupac/"

    if [ -f $GZ ]; then
        echo "Dataset successfully downloaded."
    else
        echo "Dataset not successfully downloaded."
        exit
    fi
    tar zxvf $GZ -C ./chem_iupac/
fi



# download opsin
URL="https://github.com/dan2097/opsin/releases/download/2.4.0/opsin-2.4.0-jar-with-dependencies.jar"
GZ="./chem_iupac/opsin/opsin-2.4.0-jar-with-dependencies.jar"

if [ -d "./chem_iupac/opsin/opsin-2.4.0-jar-with-dependencies.jar" ]; then
```

```
    echo "Opsin has already been downloaded."
else
    wget "$URL" -P "./chem_iupac/opsin/"

    if [ -f $GZ ]; then
        echo "Opsin successfully downloaded."
    else
        echo "Opsin not successfully downloaded."
        exit
    fi
fi
```

17 File: kegg_catch_smiles.sh

```
# location="~/operation/re_fecal_neg/kegg""
cd ~/operation/re_fecal_neg/kegg
savepath="cnumber_cid.tsv"
awk -F "[:][ ]" '
{
  if(FNR==1)
    {
      printf "kegg number\t" "pubchem id" > "'$savepath'"
    }
  if($0~/BEGIN_COMPOUND/)
    {
      getline
      if($0~/C[0-9]*/)
        {
          cnum=$0
          next_sub=1
          print "Catch C number: ", $0
          printf "\n"cnum > "'$savepath'"
        }
    else
      {
        next_sub=0
        print "Invalid sublist."
      }
    }
  if($1~/PubChem/ && next_sub==1)
    {
      printf "\t"$2 > "'$savepath'"
    }
}' dblink.list
## filter the blank and output the list
data="cnumber_cid.tsv"
nlimit=100
list=$(awk -F $'\t' '
{
  if($1~/C[0-9]*/)
    {
      if($2=="")
        {
          print "Escape CID of",$1
        }
      if($2!="")
        {
          gsub(/ /,"",$2)
          if(n>'$nlimit')
            {
              n=0
              print cid_set
            }
          n+=1
          if(n==1)
            {
              cid_set=$2
            }
        else
          {
            cid_set=cid_set","$2
          }
      }
  }
}
END{
print cid_set
}' $data)
## nrow
nrow=$(awk 'END{print NR}' <(echo "$list"))
##
for i in $(seq $nrow)
```

```
do
  cids=$(awk '{if(NR=='$i'){print $0}}' <(echo "$list"))
  check=$(echo "$cids" | awk '{if($0~/^[0-9]/){printf 0}else{printf 1}}')
  while [ $check == 0 ]
  do
    echo "Try catch pubchem API (${i}/${nrow})..."
    if [ -f ${i}_smiles.csv ]
    then
      check=$(awk '
      {
        if(FNR==1)
          {
            if($0~/CID/)
              {
                printf "1"
              }
          else
            {
              printf "0"
            }
        }
    }
  END{
  if(FNR==0)
    {
      printf "0"
    }
}' ${i}_smiles.csv)
    fi;
if [ $check == 0 ]
then
  curl --connect-timeout 20 --retry 100 --retry-delay 30 https://pubchem.ncbi.nlm.nih.gov/rest/pug/compo
fi;
done;
done;
## merge and reformat
awk -F "[,]" '
{
  if(NR==1)
    {
      printf $1
      for(i=2; i<=NF; i++)
        {
          printf "\t"$i
        }
      printf "\n"
    }
  if(FNR>=2)
    {
      printf $1
      for(i=2; i<=NF; i++)
        {
          printf "\t"$i
        }
      printf "\n"
    }
}' *_smiles.csv | sed 's/\"//g' | awk '{if($2!=""){print $0}}' > merge_smiles.tsv
## as sirius db
Rscript ~/Downloads/codes/sirius_db.R
```

18 File: ladder.sh

```
similarity_limit=0.4
num_limit_1=30
num_limit_2=500
tlimit=0.4
for i in 0.4 0.5 0.6 0.7 0.8 0.9 0.95 0.99
do
definition_limit=$i
data1="temp/filter_0_class.tsv"
data2="results/canopus_pp.tsv"
data3="results/fingerid_first_score.tsv"
class_pp_limit=$definition_limit
savepath="temp/idenfication_filter_$class_pp_limit.tsv"
awk -F $'\t' '
  {
  if(NR==FNR)
    {
    filter_class[$1]=$1
    }
  if(FILENAME~/canopus/)
    {
```

```
    if(FNR==1)
      {
      col_id=1
      for(i=2; i<=NF; i++)
        {
        for(j in filter_class)
          {
          if(j==$i)
            {
            col_class[j]=i
            print i
            }
          }
        }
      }
    if(FNR>=2)
      {
      for(i in col_class)
        {
        if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
          {
          class_set[i,$col_id]=i
          print "ID: ",$1,i,$col_class[i]
          }
        }
      }
    }
  if(FILENAME~/fingerid_first_score/)
    {
    if(FNR==1)
      {
      for(i=1; i<=NF; i++)
        {
        if($i~/^id/)
        {
        col_id=i
        }
      if($i~/similarity/)
        {
        col_similarity=i
        }
        }
      printf "class_nebula_facet\t"  $0"\n" > "'$savepath'"
      }
    if(FNR>=2)
      {
      if($col_similarity+0 >= "'$similarity_limit'"+0)
        {
       for(i in class_set)
         {
         if(i~"\034"$col_id"$")
           {
           printf class_set[i]"\t"  $0"\n" > "'$savepath'"
           }
         }
       }
      }
    }
  }' $data1 $data2 $data3
```

```
###################################
 data="temp/idenfication_filter_$class_pp_limit.tsv"
 savepath="temp/idenfication_filter2_${class_pp_limit}.tsv"
 awk -F $'\t' '
   {
   if(FNR==1)
     {
     printf $0"\n" > "'$savepath'"
     for(i=1; i<=NF; i++)
       {
       if($i~/class_nebula_facet/)
         {
         col_class=i
         }
       }
     }
 if(FNR>=2)
   {
   num[$col_class]+=1
   data[FNR]=$0
     class[FNR]=$col_class
   }
  }
   END{
     for(i in num)
       {
       if(num[i]+0 >= '$num_limit_1'+0 && num[i]+0 <= '$num_limit_2')
         {
         printf "The nodes number of the child-nebula is " num[i] ".\n"
         for(j in class)
           {
           if(class[j]==i)
             {
             printf data[j]"\n" >> "'$savepath'"
             }
           }
         }
       }
     }' $data
 ##################################
 mkdir results/network_facet_$class_pp_limit
 data1="temp/idenfication_filter2_${class_pp_limit}.tsv"
 data2="results/source_target_tree_$tlimit.tsv" # "results/source_target_tree_0.4.tsv"
 save_class="results/filter_child_class.tsv"
 savepath="results/network_facet_$class_pp_limit/"
 awk -F $'\t' '
   {
   if(NR==FNR)
     {
     if(FNR==1)
       {
       for(i=1; i<=NF; i++)
         {
         if($i~/class_nebula_facet/)
           {
           col_class=i
           }
         if($i~/^id$/)
           {
           col_id=i
           }
         }
       }
     if(FNR>=2)
```

```
      {
      class[$col_class]=$col_class
      class_id[$col_class,$col_id]=$col_id
      stat_id[$col_class,$col_id]=$col_id
      belong[$col_class,$col_id]=$col_class
      }
    }
  if(NR>FNR)
    {
    if(FNR==1)
      {
      for(i in class)
        {
        printf i"\n" > "'$save_class'"
        printf $0"\t"  "facet\n" > "'$savepath'" i ".tsv"
        }
      }
    if(FNR>=2)
      {
      for(i in class)
        {
        if( class_id[i,$1]==$1 && class_id[i,$2]==$2 )
          {
          printf $0"\t"  i"\n" >> "'$savepath'" i ".tsv"
          delete stat_id[i,$1]
          delete stat_id[i,$2]
          }
        }
      }
    }
  }
  END{
    for(i in stat_id)
      {
      ## source target similarity delta_mz fp fp class
      printf stat_id[i]"\t" stat_id[i]"\t"  "1\t"  "0\t"  "null\t"  "null\t"  belong[i]"\n" \
      \
      >> "'$savepath'" belong[i] ".tsv"
      }
    }' $data1 $data2
done;
```

19 File: marvin.sh

```
### marvin
## location ~/operation/back/0703_all
data="results/fingerid_first_score.tsv"
savepath="results/structure_2d/smiles_draw"
awk -F $'\t' '
{
  if(FNR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/^id/)
            {
              col_id=i
            }
          if($i~/smiles/)
            {
              col_smiles=i
            }
        }
    }
  if(FNR>=2)
    {
      system("molconvert mol \""  $col_smiles  "\" -o '$savepath'/"  $col_id  ".mol")
      close("molconvert mol \""  $col_smiles  "\" -o '$savepath'/"  $col_id  ".mol")
      printf "Info: the ID is "$col_id". Row number:"FNR".\n"
    }
}' $data
```

```
####################################
### use obabel to convert mol format
ls results/structure_2d/smiles_draw/[0-9]*.mol | awk -F $'\t' '
{
  split($0, a, "[.][m][o][l]")
  system("obabel "  $0  " -imol -osvg -O "  a[1]  ".mol.svg")
  close("obabel "  $0  " -imol -osvg -O "  a[1]  ".mol.svg")
  print a[1]".svg"
}'
```

```
####################################
sed -i -e 's/white/transparent/g; s/stroke-width="2.0"/stroke-width="4.0"/g;' results/structure_2d/smil
```

```
###################################
ls results/structure_2d/smiles_draw/[0-9]*.mol.svg | awk '
{
  if($0~/cairo/)
    {
      next
    }
else
  {
    system("cairosvg "  $0  " -o "  $0 ".cairo.svg")
    close("cairosvg "  $0  " -o "  $0 ".cairo.svg")
    printf "Info: the filename is "$0"\n"
  }
}'
```

```
##################################
```

```
##################################              candidate structure
#mkdir results/structure_2d/candidate
data="results/structure_2d/candidate/*_class.tsv"
id_set=$(awk -F $'\t' '
{
  if(FNR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/"group_sub"/ || $i~/^group_sub$/)
            {
              col_group_sub=i
              #print i,NF
            }
          if($i~/"group"/ || $i~/^group$/)
            {
              col_group=i
            }
        }
    }
  if(FNR>=2)
    {
      id[$col_group_sub]=$col_group_sub
      id[$col_group]=$col_group
    }
}
END{
for(i in id)
  {
    n+=1
    if(n==1)
      {
        printf i
      }
  else
    {
      printf "_"i
    }
}
}' $data)
```

```
##################################
path="/media/wizard/back/0703_all"
awk '
BEGIN{
path="'$path'"
file_set="'$id_set'"
n=split(file_set, a, "[_]")
for(i=1; i<=n; i++)
  {
    "ls -d "  path  "/*_" a[i] | getline
    close("ls -d "  path  "/*_" a[i] | getline)
    name=name""$0"/structure_candidates.tsv "
  }
printf name > "tmp"
}'
```

```
#################################
savepath="results/structure_2d/candidate"
can_num=30
awk -F $'\t' '
{
  if(FNR==1)
    {
      n=split(FILENAME, a, "[/]")
      for(i=1; i<=n; i++)
        {
          if(a[i]~/[0-9](.*)_(.*)_(.*)[0-9]/)
            {
              file=a[i]
            }
        }
      n=split(file, b, "[_]")
      id=b[n]
      for(i=1; i<=NF; i++)
        {
          if($i~/smiles/)
            {
              col_smiles=i
            }
        }
    }
  if(FNR>=2 && FNR <='$can_num')
    {
      num[id]+=1
      system("molconvert mol \"" $col_smiles "\" -o '$savepath'/" id "_can_" num[id] ".mol")
      close("molconvert mol \"" $col_smiles "\" -o '$savepath'/" id "_can_" num[id] ".mol")
      printf id " >>> " num[id] "\n"
    }
}' $(cat tmp)
```

```
#############################
ls results/structure_2d/candidate/3918_*.mol | awk -F $'\t' '
{
  split($0, a, "[.][m][o][l]")
  system("obabel "  $0  " -imol -osvg -O "  a[1]  ".mol.svg")
  close("obabel "  $0  " -imol -osvg -O "  a[1]  ".mol.svg")
  print a[1]".svg"
}'
```

```
############################
sed -i -e 's/white/transparent/g' results/structure_2d/candidate/3918_*.mol.svg
```

```
###################################
ls results/structure_2d/candidate/3918_*.mol.svg | awk '
{
  if($0~/cairo/)
    {
      next
    }
else
  {
    system("cairosvg "  $0  " -o "  $0 ".cairo.svg")
    close("cairosvg "  $0  " -o "  $0 ".cairo.svg")
    printf "Info: the filename is "$0"\n"
  }
}'
```

```
####################################
```

20 File: MCnebula_1.0.sh

```
## bin/bash

echo "We are all in the gutter,
but some of us are looking at the stars.";
PS3='Please select the workflow to be executed. >>> '
select command in \
  "MCnebula_workflow" \
  "structure_extract" \
  "classification_extract_sum" \
  "classification_extract_filter" \
  "fragment_tree_network" \
  "fragment_tree_network_delta" \
  "structure_candidate_top10" \
  "(beta)" \
  "exit"
do
  if [[ $command == "MCnebula_workflow" ]]
  then
    confirm=0
    until [[ $confirm == "yes" ]] || [[ $confirm == "no" ]]
    do
      read -p "Running all proccesses sequentially? [yes/no] >>> " confirm
    done;
    if [[ $confirm == "no" ]]
    then exit
    fi;
    default="structure_extract classification_extract_sum classification_extract_filter fragment_tree_n
    list=$(echo $default);
  else list=$( echo $command)
```

```
fi;
for option in $(echo $list)
do
  case $option in
    ###################################
    ###################################
    ###################################
    ###################################
    structure_extract)
    echo "Run structure_extract."
    projectpath=0
    until [ -d $projectpath ] && [ -f $projectpath/canopus_summary.tsv ] && [ -f $projectpath/.format
    do
      read -p "Please input the path of the sirius project >>> " projectpath;
    done;
    cd $projectpath;
    mkdir results;
    mkdir temp;
    mkdir temp/fintemp;
    if [ -f temp/Mo_filename ]
    then rm temp/Mo_filename
    fi;
    data="*_*/fingerid/*.tsv"
    awk -F $'\t' '
    BEGIN{
    all_id=0
    max_id=-1
    p_id="null"
    printf "Info: loading the data..."
  }
{
  if(FNR==1)
    {
      if(NR>FNR)
        {
          close(pfile)
        }
      f=split(FILENAME,a,"[/]");
      n=split(a[1],b,"[_]");
      id=b[n];
      if(id!=p_id)
        {
          all_id+=1
          if(id>max_id)
            {
              max_id=id
            }
        }
      file[id]=a[1]
      split(a[f], g, "[.]")
      formu_type[id]=g[1]
      if(all_id==1)
        {
          for(i=1; i<=NF; i++)
            {
              if(($i~/inchikey2D/))
                {
                  col_2D=i
                }
              if($i=="inchi")
                {
                  col_inchi=i
                }
              if(($i~/Formula/))
                {
                  col_formu=i
                }
              if($i=="score")
                {
                  col_score=i
                }
              if($i=="name")
                {
                  col_name=i
                }
              if($i=="smiles")
                {
                  col_smiles=i
                }
              if($i=="xlogp")
                {
                  col_x=i
```

```
                  }
                if(($i~/imilarity/))
                  {
                    col_simi=i
                  }
                if($i~/links/)
                  {
                    col_links=i
                  }
              }
          }
        pfile=FILENAME
        printf g[1]"\t"formu_type[id]"\n"
        printf "Info: the data of " id " (ID) has input. The filename is " FILENAME ".\n"
      }
    if(FNR>=2)
      {
        row_score=$col_score
        row_simi=$col_simi
        if(max["score",id]=="" || max["score",id]+0<row_score+0)
          {
            max["score",id]=row_score
            name["score",id]=$col_name
            formula["score",id]=$col_formu
            formu_type["score",id]=formu_type[id]
            simi["score",id]=$col_simi
            smiles["score",id]=$col_smiles
            inchi["score",id]=$col_inchi
            in2D["score",id]=$col_2D
            score["score",id]=$col_score
            xlogp["score",id]=$col_x
            links["score",id]=$col_links
          }
        if(max["simi",id]=="" || max["simi",id]+0<row_simi+0)
          {
            max["simi",id]=row_simi
            name["simi",id]=$col_name
            formula["simi",id]=$col_formu
            formu_type["simi",id]=formu_type[id]
            simi["simi",id]=$col_simi
            smiles["simi",id]=$col_smiles
            inchi["simi",id]=$col_inchi
            in2D["simi",id]=$col_2D
            score["simi",id]=$col_score
            xlogp["simi",id]=$col_x
            links["simi",id]=$col_links
          }
      }
  }
END{
printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t"  "score\t"
for(l in file)
  {
    if(max["score",l]!="")
      {
        printf l"\t"  name["score",l]"\t"  formula["score",l]"\t"  simi["score",l]"\t"  smiles["score",l
        printf formula["score",l]"\t"  file[l]"\t"  formu_type["score",l]"\n" > "temp/Mo_filename"
      }
    if(max["simi",l]!="" && max["simi",l]!=max["score",l])
      {
        printf l"\t"  name["simi",l]"\t"  formula["simi",l]"\t"  simi["simi",l]"\t"  smiles["simi",l]"\t
      }
  }
}' $data
data_sum="results/fingerid_sum.tsv"
awk -F $'\t' '
{
  if(NR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/id/)
            {
              col_id=i
            }
          if($i~/score/)
            {
              col_score=i
            }
        }
    }
  if(NR>=2)
```

```
    {
      id=$col_id
      if(max_id+0<id || max_id=="")
        {
          max_id=id
        }
      if(score[id]+0<$col_score+0 || score[id]=="")
        {
          score[id]=$col_score
          data[id]=$0
        }
    }
}
END{
printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t"  "score\t"
for(i=1; i<=max_id; i++)
  {
    if(data[i]!="")
      {
        printf data[i] "\n" >> "results/fingerid_first_score.tsv"
      }
  }
}' $data_sum
echo "structure_extract results have been successfully assembled into <results/fingerid_sum.tsv> and <r
;;
```

```
####################################
```

```
####################################
```

```
####################################
```

```
####################################
classification_extract_sum)
echo "Run classification_extract_sum.";
if [ -f temp/Mo_filename ] && [ -f canopus.tsv ] && [ -f canopus_neg.tsv ] && [ -f .format ]
then
  echo "Project path acknowledged."
else
  until [ -d $projectpath ] && [ -f $projectpath/canopus.tsv ] && [ -f $projectpath/.format ]
  do
    read -p "Please input the path of the sirius project >>> " projectpath;
  done;
  cd $projectpath;
fi;
data1="temp/Mo_filename"
data2="canopus.tsv"
data3="canopus_neg.tsv"
datas=$(awk -F $'\t' '
{
  x=$2"/canopus/"$3".fpt"
  if(getline < x == 1)
    {
      printf x" "
    }
  close(x)
}' $data1)
```

```
##################
awk -F $'\t' '
{
  if(NR==FNR)
    {
      i=split($2,s,"[_]")
      the_id[FNR]=s[i]
      n=FNR
    }
  if(FILENAME ~ /canopus.tsv/)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i ~ /^name/)
                {
                  col_class=i
                }
              if($i ~ /absolute/)
                {
                  col_abindex=i
                }
            }
        }
      if(FNR>=2)
        {
```

```
        abindex[1,FNR]=$col_abindex
        indexset[$col_abindex]=$col_class
      }
  }
 if(FILENAME ~ /canopus_neg.tsv/)
  {
    p_filename=FILENAME
    if(FNR==1)
      {
        for(i=1; i<=NF; i++)
          {
            if($i ~ /name/)
              {
                col_class=i
              }
            if($i ~ /absolute/)
              {
                col_abindex=i
              }
          }
      }
    if(FNR>=2)
      {
        abindex[2,FNR]=$col_abindex
        indexset[$col_abindex]=$col_class
      }
  }
 if(FILENAME ~ /.fpt/)
  {
    if(FNR==1)
      {
        close(p_filename)
        printf "Info: data_file name of " p_filename " has been input.\n"
        p_filename=FILENAME
        split(FILENAME,a,"[/]");
        m=split(a[1],b,"[_]");
        id=b[m];
        if(FILENAME ~ /\+.fpt/)
          {
            ion=1
          }
          if(FILENAME ~ /\-.fpt/)
            {
              ion=2
            }
        }
      pp[id,abindex[ion,FNR+1]]=sprintf("%.3f",$1)
    }
}
END{
printf "id" > "results/canopus_pp.tsv"
for(i in indexset)
  {
    ord+=1
    orderlist[ord]=i
    printf "\t"indexset[i] >> "results/canopus_pp.tsv"
  }
printf "\n" >> "results/canopus_pp.tsv"
for(i=1; i<=n; i++)
  {
    printf the_id[i] >> "results/canopus_pp.tsv"
    for(j=1; j<=ord; j++)
      {
        if(pp[the_id[i],orderlist[j]]=="")
          {
            pp[the_id[i],orderlist[j]]=0
          }
        printf "\t"pp[the_id[i],orderlist[j]] >> "results/canopus_pp.tsv"
      }
    printf "\n" >> "results/canopus_pp.tsv"
  }
}' $data1 $data2 $data3 $datas
echo "classiication_extract_sum have been successfully written into <results/canopus_pp.tsv>"
;;
```

```
####################################
```

```
####################################
```

```
####################################
classification_extract_filter)
echo "Run classification_extract_filter.";
if [ -f results/canopus_pp.tsv ]
then
```

```
  echo "Project path acknowledged."
else
  until [ -d $projectpath ] && [ -f $projectpath/results/canopus_pp.tsv ] && [ -f $projectpath/results/c
  do
    read -p "Please input the path of the sirius project >>> " projectpath;
  done;
  cd $projectpath;
fi;
check=0
until [[ "$check" == "yes" ]] || [[ "$check" == "no" ]]
do
  read -p "Collate the posterior probabilities of the classification of each feature? [yes/no] >>> " che
done;
if [[ $check == "yes" ]]
then
  definition_limit=0
  until [[ "$definition_limit" > "0.01" ]] && [[ "$definition_limit" < "1" ]]
  do
    read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>> " de
  done;
  data1="canopus_summary.tsv"
  data2="canopus.tsv"
  data3="results/canopus_pp.tsv"
  awk -F $'\t' '
  {
    if(NR==FNR)
      {
        if(FNR==1)
          {
            p_file=FILENAME
            for(i=1; i<=NF; i++)
              {
                if($i~/name/)
                  {
                    col_id=i
                  }
                if($i~/specific/)
                  {
                    col_specific=i
                  }
                if($i~/level/)
```

```
       {
        col_level=i
       }
      if($i~/subclass/)
       {
        col_subclass=i
       }
      if($i~/^class/)
       {
        col_class=i
       }
      if($i~/superclass/)
       {
        col_superclass=i
       }
     }
   }
  if(FNR>=2)
   {
    n=split($col_id,a,"[_]")
    id=a[n]
    specific[id]=$col_specific
    level[id]=$col_level
    subclass[id]=$col_subclass
    class[id]=$col_class
    superclass[id]=$col_superclass
   }
 }
if(FILENAME~/canopus.tsv/)
 {
  if(FNR==1)
   {
    close(p_file)
    p_file=FILENAME
    for(i=1; i<=NF; i++)
     {
      if($i~/name/)
       {
        col_name=i
       }
      if($i~/description/)
```

```
              {
                col_description=i
              }
            }
        }
      if(FNR>=2)
        {
          description[$col_name]=$col_description
        }
    }
  if(FILENAME~"'$data3'")
    {
      if(FNR==1)
        {
          close(p_file)
          for(i=1; i<=NF; i++)
            {
              if($i~/^id$/)
                {
                  col_id=i
                }
              if(i>=2)
                {
                  col_class_name[i]=$i
                }
            }
          printf "id\t"  "definition_source\t"  "definition\t"  "definition_pp\t"  "definition_descri
        }
      if(FNR>=2)
        {
          for(i=2; i<=NF; i++)
            {
              c_pp[col_class_name[i]]=sprintf("%.4f",$i)
            }
          if(level[$col_id] != "" && c_pp[level[$col_id]] >= "'$definition_limit'")
            {
              definition_source="level_5"
              definition=level[$col_id]
            }
        else if(subclass[$col_id] != "" && c_pp[subclass[$col_id]] >= "'$definition_limit'")
          {
              definition_source="subclass"
              definition=subclass[$col_id]
            }
        else if(class[$col_id] != "" && c_pp[class[$col_id]] >= "'$definition_limit'")
          {
            definition_source="class"
            definition=class[$col_id]
          }
      else if(superclass[$col_id] != "" && c_pp[superclass[$col_id]] >= "'$definition_limit'")
        {
          definition_source="superclass"
          definition=superclass[$col_id]
        }
    else
      {
        definition_source="null"
        definition="null"
        c_pp[definition]="null"
        description[definition]="null"
      }
    printf $col_id"\t"  definition_source"\t"  definition"\t"  c_pp[definition]"\t"  description[defini
  }
}
}' $data1 $data2 $data3
echo "classification_extract_filter have been successfully written into <results/stat_classification.ts
fi;
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
fragment_tree_network)
echo "Run fragment_tree_network.";
if [ -d temp ] && [ -f ftalign.tsv ] && [ -f .format ]
then
  echo "Project path acknowledged."
else
  until [ -d $projectpath ] && [ -f $projectpath/ftalign.tsv ] && [ -f $projectpath/.format ]
  do
```

```
    read -p "Please input the path of the sirius project. Make sure you have moved the fragment tree ali
  done;
  cd $projectpath;
fi;
echo "Please enter the minimum alignment similarity(tlimit) that you want to filter."
tlimit=0
until [[ "$tlimit" > "0.01" ]] && [[ "$tlimit" < "0.99" ]]
do
  read -p "0.4-0.7 is recommended >>> " tlimit;
done;
if ! [ -d temp/ftaligntemp ]
then
  mkdir temp/ftaligntemp
fi;
data="ftalign.tsv"
savepath="temp/ftaligntemp/tmp"
awk -F $'\t' -v OFS=$'\t' '
{
  printf "Info: NR = " NR ". FNR = " FNR ".\n"
  if(NR==FNR)
    {
      for(i=1; i<=NF; i++)
        {
          if(NR==1 && i!=1 || NR!=1 && i==1)
            {
              n=split($i,x,"[_]")
              raw[NR,i]=x[n]
            }
        else
          {
            if(NR==i)
              {
                raw_norm[NR,i]=$i
              }
          }
      }
  }
if(NR>FNR && FNR>=2)
  {
    for(i=2; i<=NF; i++)
      {
        norm1=$i/raw_norm[i,i]
        norm2=$i/raw_norm[FNR,FNR]
        norms=sprintf("%.2f", ((norm1+norm2)/2))
        if((norms+0 > '$tlimit'+0))
          {
            if(raw[FNR,1]+0>=raw[1,i]+0)
              {
                print raw[FNR,1], raw[1,i], norms > "'$savepath'"
              }
            if(raw[FNR,1]+0<raw[1,i]+0)
              {
                print raw[1,i], raw[FNR,1], norms > "'$savepath'"
              }
          }
      }
  }
}' $data $data
sort -u $savepath > temp/ftaligntemp/tmp2
savepath="temp/ftaligntemp/filter_net_$tlimit"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if($1!=$2)
        {
          replink[$1]=$1
          replink[$2]=$2
        }
    }
  if(NR!=FNR)
    {
      if($1!=$2)
        {
          printf $0"\n" > "'$savepath'"
        }
      if($1==$2)
        {
          if(replink[$1]=="")
            {
              printf $0"\n" > "'$savepath'"
            }
        }
    }
}' temp/ftaligntemp/tmp2 temp/ftaligntemp/tmp2
echo "fragment_tree_network have been successfully written into <temp/ftaligntemp/filter_net_$tlimit>"
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
fragment_tree_network_delta)
if [ -d temp ] && [ -f ftalign.tsv ] && [ -f .format ]
then
  echo "Project path acknowledged."
else
  until [ -d $projectpath ] && [ -f $projectpath/ftalign.tsv ] && [ -f $projectpath/.format ]
  do
    read -p "Please input the path of the sirius project >>> " projectpath;
  done;
  cd $projectpath;
fi;
echo "The following module attempts to compute the differential fingerprints of connected clusters base
```

```
###################
```

```
###################
```

```
###################
plimit=0
until [[ "$plimit" > "0.5" ]] && [[ "$plimit" < "0.99" ]]
do
  read -p "If the minimum alignment similarity is greater than 0.3, this step will automatic calculate 
done;
plimit2=0
until [[ "$plimit2" > "0.1" ]] && [[ "$plimit2" < "$plimit" ]]
do
  read -p "The minimum posterior probability of the molecular fingerprint(plimit_2) to be controled. 0.
done;
tlimit=$(ls temp/ftaligntemp/filter_net_* | awk -F _ '{print $NF}')
check_rep=$(awk 'END{print NR}' <(echo "$tlimit"))
if [[ "$check_rep" > "1" ]]
then
```

```
  tlimit=0
  until [ -f temp/ftaligntemp/filter_net_$tlimit ]
  do
    read -p "Plural and different fragment_tree_network files were found to exist locally. Please selec
  done;
fi;
if [ -f temp/ftaligntemp/refilter_net_$tlimit ]
then rm temp/ftaligntemp/refilter_net_$tlimit
fi;
if [ -f temp/ftaligntemp/fpsample ]
then rm temp/ftaligntemp/fpsample
fi;
echo "Running delta_fingerprint computation..."
echo "Aquiring data from sirius index..."
data1="*_*/compound.info"
data2="temp/ftaligntemp/filter_net_$tlimit"
echo "Run fragment_tree_network_delta."
savepath="temp/ms_data_$tlimit"
echo "step 1: ms data"
if ! [ -f $savepath ]
then
  awk -F "[\t]||[:]||[_]" -v OFS=$'\t' '
  BEGIN{
  printf "..."
  printf "id\t"  "m/z\t"  "rt\n" > "results/mz_and_rt.tsv"
}
{
  if(FILENAME~/compound.info/)
    {
      if(FNR==1)
        {
          printf "Info: catch >>> "FILENAME"\n"
        }
      if($1=="name")
        {
          i+=1;
          id[i]=$NF;
          n=split($NF,a,"[_]")
          printf a[n]"\t" >> "results/mz_and_rt.tsv"
        }
      if($1=="ionMass")
```

```
        {
          mz[id[i]]=$NF
          printf sprintf("%.4f",$NF)"\t" >> "results/mz_and_rt.tsv"
        }
      if($1=="ionType")
        {
          type[id[i]]=$NF
        }
      if($1=="rt")
        {
          rt[id[i]]=$2;
          printf sprintf("%.2f",$2/60)"\n" >> "results/mz_and_rt.tsv"
        }
    }
  if(FILENAME~/ftaligntemp/)
    {
      #source  target  ftalign  delta_mz  delta_rt  source_iontype  target_iontype;
      if(FNR==1)
        {
          printf "" > "'$savepath'"
        }
      print $1,$2,$3,sprintf("%.3f",mz[$2]-mz[$1]),sprintf("%.2f",rt[$2]-rt[$1]),type[$1],type[$2]  >>
    }
}' $data1 $data2
fi;
```

```
########################
echo "step 2: data path"
source_file="temp/Mo_filename"
data="temp/ms_data_$tlimit"
savepath="temp/datapath_$tlimit"
if ! [ -f $savepath ]
then
  awk -F $'\t' -v OFS=$'\t' '
  {
    if(NR==FNR)
      {
        n=split($2, a, "[_]")
        file[a[n]]=$2
        formu_type[a[n]]=$3
      }
    if(NR!=FNR)
```

```
      {
        #<path>sourceFormula  <path>targetFormula
        path1=file[$1]"/fingerprints/"formu_type[$1]".fpt"
        path2=file[$2]"/fingerprints/"formu_type[$2]".fpt"
        data[path1]=path1
        data[path2]=path2
      }
  }
END{
for(i in data)
  {
    n+=1
    print "Check file:",n
    if(getline<i==-1)
      {
        printf "Escape filename: " i "\n"
      }
  else
    {
      printf i" " > "'$savepath'"
    }
  close(i)
}
print "Sum:",n
}' $source_file $data
fi;
datapath=$(cat temp/datapath_$tlimit)
echo "step 3: fingerprint"
data_allfp="temp/data_allfp_$tlimit"
if ! [ -f $data_allfp ]
then
  awk -F $'\t' '
  BEGIN{
  n=0
}
{
  if(FNR==1)
    {
      if(n>1)
        {
          close(file)
        }
      file=FILENAME;
      print "Get fingerprints: ",FILENAME
      n+=1;
      printf FILENAME"\n"$0"\n" > "'$data_allfp'"
    }
else
  {
    print $0 > "'$data_allfp'"
  }
}' $datapath
fi;
```

```
###########################
awk -F $'\t' -v OFS=$'\t' '
{
  if(NR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/absolute/)
            {
              col_index=i
            }
          if($i~/description/)
            {
              col_description=i
            }
        }
    }
  if(NR>=2)
    {
      print $col_index,$col_description
    }
}' csi_fingerid.tsv > temp/ftaligntemp/pos_fingerprint_index;
awk -F $'\t' -v OFS=$'\t' '
{
  if(NR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/absolute/)
```

```
            {
              col_index=i
            }
          if($i~/description/)
            {
              col_description=i
            }
        }
    }
  if(NR>=2)
    {
      print $col_index,$col_description
    }
}' csi_fingerid_neg.tsv > temp/ftaligntemp/neg_fingerprint_index;
posindex="temp/ftaligntemp/pos_fingerprint_index"
negindex="temp/ftaligntemp/neg_fingerprint_index"
data="temp/ms_data_$tlimit"
echo "step 4: merge data"
awk -F $'\t' '
BEGIN{
file=0
x=0
count=0
f=0
posnum=0
negnum=0
}
{
  if(FNR==1)
    {
      file+=1
    }
  if(NR==FNR)
    {
      if(($1~/fingerprint/))
        {
          if(x+0>f+0)
            {
              f=x  # calculate the max index.
            }
          count+=1;  # calculate the all fingerprints file number.
          split($1,a,"[/]"); e=split(a[1],b,"[_]"); id=b[e]; #catch the id.
          x=0;
        }
    else
      {
        x+=1;
        fp[id,x]=$1;
      }
  }
if(file==2)
  {
    pos[FNR]=$1
    posnum+=1
  }
if(file==3)
  {
    neg[FNR]=$1
    negnum+=1
  }
if(file==4)
  {
    if("'$tlimit'"+0 >= 0.3)
      {
        if(fp[$1,1]!="" && fp[$2,1]!="")
          {
            n1=split($6, g, "[]]");
            n2=split($7, h, "[]]");
            if(g[n1]=="+" && h[n2]=="+")
              {
                for(x=1; x<=posnum; x++)
                  {
                    if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
                      {
                        data_s[FNR,x]=pos[x]
                      }
                  else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
                    {
                      data_t[FNR,x]=pos[x]
                    }
                }
            }
```

```
     if(g[n1]=="-" && h[n2]=="-")
       {
         for(x=1; x<=negnum; x++)
           {
             if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
               {
                 data_s[FNR,x]=neg[x]
               }
           else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
             {
               data_t[FNR,x]=neg[x]
             }
         }
     }
   if(g[n1]=="-" && h[n2]=="+" || h[n2]=="-" && g[n1]=="+")
     {
       for(i=1; i<=posnum; i++)
         {
           for(j=1; j<=negnum; j++)
             {
               if(pos[i]==neg[j])
                 {
                   mirror[i]=j  #if pos=i, the identical index of neg is mirror[i]
                 }
             }
         }
     }
   if(g[n1]=="-" && h[n2]=="+")
     {
       for(x=1; x<=f; x++)
         {
           if(fp[$1,mirror[x]]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
             {
               data_s[FNR,x]=neg[mirror[x]]
             }
         else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,mirror[x]]+0<='$plimit2'+0)
           {
             data_t[FNR,x]=neg[mirror[x]]
           }
       }
   }
```

```
      if(h[n2]=="-" && g[n1]=="+")
        {
          for(x=1; x<=f; x++)
            {
              if(fp[$1,x]+0>= '$plimit'+0 && fp[$2,mirror[x]]+0<='$plimit2'+0)
                {
                  data_s[FNR,x]=neg[mirror[x]]
                }
            else if(fp[$2,mirror[x]]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
              {
                data_t[FNR,x]=neg[mirror[x]]
              }
          }
      }
  }
}
if(FNR==1)
  {
    printf "source\t"  "target\t"  "ftalign_similarity\t"  "delta_m/z\t"  "source_fp_uniq\t"  "target_f
  }
else
  {
    printf $1"\t"  $2"\t"  $3"\t"  $4"\t";
    if('$tlimit'+0 >= 0.3)
      {
        if(fp[$1,1]=="" || fp[$2,1]=="")
          {
            printf "NA@NA\n"
          }
      else
        {
          printf "source:"  #the source fingerprint uniq.
          for(x=1; x<=f; x++)
            {
              if(data_s[FNR,x]!="")
                {
                  printf data_s[FNR,x]","
                }
            };
          printf "@target:"  #the target fingerprint uniq.
          for(x=1; x<=f; x++)
```

```
            {
              if(data_t[FNR,x]!="")
                {
                  printf data_t[FNR,x]","
                }
            }
          printf "\n"
        }
    }
else
  {
    printf "NA@NA\n"
  }
}
}
}' $data_allfp $posindex $negindex $data | sed -e 's/,@/\t/g; s/,$//g; s/@/\t/g'  > results/source_targe
echo "All instances have written into <results/source_target_tree_$tlimit.tsv>."
```

```
#####################################
echo "step 5: separate child-nebula from parent-nebula."
data="results/stat_classification.tsv"
savepath="temp/filter_0_class.tsv"
awk -F $'\t' '
{
  if(FNR==1)
    {
      for(i=1; i<=NF; i++)
        {
          if($i~/^definition$/)
            {
              col_class=i
            }
        }
    }
  if(FNR>=2)
    {
      class[$col_class]=$col_class
    }
}
END{
for(i in class)
  {
    printf class[i]"\n" > "'$savepath'"
  }
}' $data
```

```
######################################
data1="temp/filter_0_class.tsv"
data2="results/canopus_pp.tsv"
data3="results/fingerid_first_score.tsv"
until [[ "$similarity_limit" > 0 ]] && [[ "$similarity_limit" < 1 ]]
do
  read -p "Please set the Tanimoto similarity threshold contribute to child-nebula. >>> " similarity_li
done;
until [[ "$definition_limit" > 0.01 ]] && [[ "$definition_limit" < 1 ]]
do
  read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>> " defi
done;
class_pp_limit=$definition_limit
savepath="temp/idenfication_filter_$class_pp_limit.tsv"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      filter_class[$1]=$1
    }
  if(FILENAME~/canopus/)
    {
      if(FNR==1)
        {
          col_id=1
          for(i=2; i<=NF; i++)
            {
              for(j in filter_class)
                {
                  if(j==$i)
                    {
                      col_class[j]=i
                      print i
                    }
                }
            }
        }
      if(FNR>=2)
        {
          for(i in col_class)
            {
              if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
                {
                  class_set[i,$col_id]=i
                  print "ID: ",$1,i,$col_class[i]
                }
            }
        }
    }
  if(FILENAME~/fingerid_first_score/)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/^id/)
                {
                  col_id=i
                }
              if($i~/similarity/)
                {
                  col_similarity=i
                }
            }
          printf "class_nebula_facet\t"  $0"\n" > "'$savepath'"
        }
      if(FNR>=2)
        {
          if($col_similarity+0 >= "'$similarity_limit'"+0)
            {
              for(i in class_set)
                {
                  if(i~"\034"$col_id"$")
                    {
                      printf class_set[i]"\t"  $0"\n" > "'$savepath'"
                    }
                }
            }
        }
    }
}' $data1 $data2 $data3
```

```
####################################
data="temp/idenfication_filter_$class_pp_limit.tsv"
savepath="temp/idenfication_filter2_${class_pp_limit}.tsv"
until [[ "$num_limit_1" -gt 0 ]]
do
  read -p "Please enter the features number threshold contribute to child-nebula (min number). >>> " nu
done;
until [[ "$num_limit_2" -gt "$num_limit_1" ]]
do
  read -p "Please enter the features number threshold contribute to child-nebula (max number). >>> " nu
done;
awk -F $'\t' '
{
  if(FNR==1)
    {
      printf $0"\n" > "'$savepath'"
      for(i=1; i<=NF; i++)
        {
          if($i~/class_nebula_facet/)
            {
              col_class=i
            }
        }
    }
  if(FNR>=2)
    {
      num[$col_class]+=1
      data[FNR]=$0
      class[FNR]=$col_class
    }
}
END{
for(i in num)
  {
    if(num[i]+0 >= '$num_limit_1'+0 && num[i]+0 <= '$num_limit_2')
      {
        printf "The nodes number of the child-nebula is " num[i] ".\n"
        for(j in class)
          {
            if(class[j]==i)
              {
                printf data[j]"\n" >> "'$savepath'"
              }
          }
      }
  }
}' $data
```

```
##################################
mkdir results/network_facet_$class_pp_limit
data1="temp/idenfication_filter2_${class_pp_limit}.tsv"
data2="results/source_target_tree_$tlimit.tsv" # "results/source_target_tree_0.4.tsv"
save_class="results/filter_child_class.tsv"
savepath="results/network_facet_$class_pp_limit/"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/class_nebula_facet/)
                {
                  col_class=i
                }
              if($i~/^id$/)
                {
                  col_id=i
                }
            }
        }
      if(FNR>=2)
        {
          class[$col_class]=$col_class
          class_id[$col_class,$col_id]=$col_id
          stat_id[$col_class,$col_id]=$col_id
          belong[$col_class,$col_id]=$col_class
        }
    }
  if(NR>FNR)
```

```
    {
      if(FNR==1)
        {
          for(i in class)
            {
              printf i"\n" > "'$save_class'"
              printf $0"\t"  "facet\n" > "'$savepath'" i ".tsv"
            }
        }
      if(FNR>=2)
        {
          for(i in class)
            {
              if( class_id[i,$1]==$1 && class_id[i,$2]==$2 )
                {
                  printf $0"\t"  i"\n" >> "'$savepath'" i ".tsv"
                  delete stat_id[i,$1]
                  delete stat_id[i,$2]
                }
            }
        }
    }
}
END{
for(i in stat_id)
  {
    ## source target similarity delta_mz fp fp class
    printf stat_id[i]"\t" stat_id[i]"\t"  "1\t"  "0\t"  "null\t"  "null\t"  belong[i]"\n"  >> "'$savepa
  }
}' $data1 $data2
```

```
#####################
data1="canopus.tsv"
data2="results/filter_child_class.tsv"
data3="results/canopus_pp.tsv"
savepath="results/canopus_pp_filter.tsv"
awk -F $'\t' '
{
  if(FILENAME~/canopus.tsv/)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i~/absolute/)
                {
                  col_index=i
                }
              if($i~/^id/)
                {
                  col_chemid=i
                }
              if($i~/name/)
                {
                  col_name=i
                }
              if($i~/description/)
                {
                  col_des=i
                }
            }
        }
      if(FNR>2)
        {
          ab_index[$col_name]=$col_index
          chemid[$col_name]=$col_chemid
          des[$col_name]=$col_des
        }
    }
  if(FILENAME~/filter_child_class/)
    {
      class[$1]=$1
      if(FNR==1)
        {
          printf "index\t"  "chem_id\t"  "name\t"  "description\n" > "results/child_class.tsv"
        }
      printf ab_index[$1]"\t"  chemid[$1]"\t"  $1"\t"  des[$1]"\n" >> "results/child_class.tsv"
    }
  if(FILENAME~/canopus_pp/)
    {
      if(FNR==1)
        {
          printf $1 > "'$savepath'"
          for(i=2; i<=NF; i++)
            {
              if(class[$i]!="")
                {
                  n+=1
                  printf "\tC"ab_index[$i] >> "'$savepath'"
                  col_set[n]=i
                }
            }
          printf "\n" >> "'$savepath'"
        }
      if(FNR>=2)
        {
          printf $1 >> "'$savepath'"
          for(i=1; i<=n; i++)
            {
              printf "\t"$col_set[i] >> "'$savepath'"
            }
          printf "\n" >> "'$savepath'"
        }
    }
}' $data1 $data2 $data3
;;
```

```
###################################
```

```
###################################
structure_candidate_top10)
echo "structure_candidate_top10 (awk version  3.1)."
projectpath=0
until [ -d $projectpath ] && [ -f $projectpath/.format ]
do
  read -p "Please input the path of the sirius project >>> " projectpath;
done;
cd $projectpath;
if [ -f results ]
then echo "Expand into target dir <results>"
else mkdir results
  mkdir temp
  mkdir temp/fintemp;
fi;
## gather all structure candidate
```

```
data="*_*/fingerid/*.tsv"
awk -F $'\t' '
BEGIN{
all_id=0
max_id=-1
p_id="null"
printf "Info: loading the data..."
}
{
  if(FNR==1)
    {
      if(NR>FNR)
        {
          close(pfile)
        }
      f=split(FILENAME,a,"[/]");
      n=split(a[1],b,"[_]");
      id=b[n];
      if(id!=p_id)
        {
          all_id+=1
          if(id>max_id)
            {
              max_id=id
            }
        }
      file[id]=a[1]
      split(a[f], g, "[.]")
      formu_type[id]=g[1]
      if(all_id==1)
        {
          printf "id\t" "name\t" "formula\t" "similarity\t" "smiles\t" "inchi\t" "inchikey2D\t"
          targetfile="results/fingerid_candidate_all.tsv"
          for(i=1; i<=NF; i++)
            {
              if(($i~/inchikey2D/))
                {
                  col_2D=i
                }
              if($i=="inchi")
                {
```

```
                col_inchi=i
              }
            if(($i~/Formula/))
              {
                col_formu=i
              }
            if($i=="score")
              {
                col_score=i
              }
            if($i=="name")
              {
                col_name=i
              }
            if($i=="smiles")
              {
                col_smiles=i
              }
            if($i=="xlogp")
              {
                col_x=i
              }
            if(($i~/imilarity/))
              {
                col_simi=i
              }
            if($i~/links/)
              {
                col_links=i
              }
          }
      }
    pfile=FILENAME
    printf g[1]"\t"formu_type[id]"\n"
    printf "Info: the data of " id " (ID) has input. The filename is " FILENAME ".\n"
  }
 if(FNR>=2)
  {
  ##  "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t"  "score\t"
  printf id"\t" $col_name"\t" $col_formu"\t" $col_simi"\t" $col_smiles"\t" $col_inchi"\t" $col_2D"\t"
  }
```

```
} ' $data
## sort the candidates to get top 10
data="results/fingerid_candidate_all.tsv"
version=0
version=$(awk 'BEGIN{a[1]=1; asort(a, b); print b[1]}')
if [ $version != 1 ]
then
  echo "Awk version   3.1. Function <asort> not available."
fi
awk -F $'\t' '
{
  if(FNR==1)
    {
      printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t"  "sco
      targetfile="results/fingerid_candidate_top10.tsv"
      for(i=1; i<=NF; i++)
        {
          if($i~/^id$/)
            {
              col_id=i
            }
          if($i~/^score/)
            {
              col_score=i
            }
        }
    }
  if(FNR>=2)
    {
      if(id!=$col_id)
        {
          ## output data top10
          if(id!="")
            {
              j=asort(score, a)
              for(i=j; i>=(j-9); i--)
                {
                  if(i==0)
                    {
                      break
                    }
```

```
         if(data[a[i]]=="")
          {
           print "yes"
          }
         printf data[a[i]]"\n" > targetfile
        }
       delete data
       delete score
      }
     ## gather first row of the id
     n=1
     score[n]=$col_score
     if(data[$col_score]!="")
      {
       $col_score+=0.00000001
      }
     data[$col_score]=$0
    }
  else
    {
     n+=1
     score[n]=$col_score
     if(data[$col_score]!="")
      {
       $col_score+=0.00000001
      }
     data[$col_score]=$0
    }
   id=$col_id
  }
}
END{
{
 j=asort(score, a)
 for(i=j; i>=j-9; i--)
  {
   if(i==0)
    {
     break
    }
   printf data[a[i]]"\n" > targetfile
```

```
    }
}
}' $data
exit
;;
```

```
###################################
```

```
###################################
"(beta)")
exit;
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
exit)
echo "The mystery of creation is like the darkness of night--it is great.
Delusions of knowledge are like the fog of the morning."
exit;
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
*)
echo "error"
exit;
;;
esac;
done;#(for)
done;#(select)
```

21 File: mcnebula1023.sh

```
##########################
```

```
##########################
echo "We are all in the gutter,
but some of us are looking at the stars.";

PS3='Please select the workflow to be executed. >>> '

select command in \
    \
    "default" \
    \
    "structure_extract" \
    \
    "classification_extract_sum" \
    \
    "classification_extract_filter" \
    \
    "fragment_tree_network" \
    \
    "fragment_tree_network_delta" \
    \
    "compound_idenfication" \
    \
    "double_ion_network" \
    \
    "exit"
do

    if [[ $command == "default" ]]
    then

    confirm=0

        until [[ $confirm == "yes" ]] || [[ $confirm == "no" ]]
        do
        read -p "Running all proccesses sequentially? [yes/no] >>> " confirm
        done;
        if [[ $confirm == "no" ]]
        then exit
        fi;

    default="structure_extract classification_extract_sum classification_extract_filter fragment_tree_n

    list=$(echo $default);
    else list=$( echo $command)
    fi;

    for option in $(echo $list)
    do
        case $option in
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
structure_extract)
echo "Run structure_extract."
projectpath=0
    until [ -d $projectpath ] && [ -f $projectpath/canopus_summary.tsv ] && [ -f $projectpath/.format ]
    do
    read -p "Please input the path of the sirius project >>> " projectpath;
    done;

cd $projectpath;
mkdir results;
mkdir temp;
mkdir temp/fintemp;

    if [ -f temp/Mo_filename ]
    then rm temp/Mo_filename
    fi;

data="*_*/fingerid/*.tsv"

awk -F $'\t' '
    BEGIN{
        all_id=0
        \
        max_id=-1
        \
        p_id="null"
        \
        printf "Info: loading the data..."
        }
```

```
{
if(FNR==1)
    {
    if(NR>FNR)
        {
        close(pfile)
        }
    f=split(FILENAME,a,"[/]");

    n=split(a[1],b,"[_]");

    id=b[n];

    if(id!=p_id)
        {
        all_id+=1
        \
        if(id>max_id)
            {
            max_id=id
            }
        }
    file[id]=a[1]

    split(a[f], g, "[.]")

    formu_type[id]=g[1]

    if(all_id==1)
        {
        for(i=1; i<=NF; i++)
            {
            if(($i~/inchikey2D/))
                {
                col_2D=i
                }
            if($i=="inchi")
                {
                col_inchi=i
                }
            if(($i~/Formula/))
```

```
                    {
                    col_formu=i
                    }
                if($i=="score")
                    {
                    col_score=i
                    }
                if($i=="name")
                    {
                    col_name=i
                    }
                if($i=="smiles")
                    {
                    col_smiles=i
                    }
                if($i=="xlogp")
                    {
                    col_x=i
                    }
                if(($i~/imilarity/))
                    {
                    col_simi=i
                    }
                if($i~/links/)
                    {
                    col_links=i
                    }
                }
            }
        pfile=FILENAME

        printf g[1]"\t"formu_type[id]"\n"

        printf "Info: the data of " id " (ID) has input. The filename is " FILENAME ".\n"
        }
    if(FNR>=2)
        {
        row_score=$col_score

        row_simi=$col_simi
        if(max["score",id]=="" || max["score",id]+0<row_score+0)
            {
            max["score",id]=row_score

            name["score",id]=$col_name

            formula["score",id]=$col_formu

            simi["score",id]=$col_simi

            smiles["score",id]=$col_smiles

            inchi["score",id]=$col_inchi

            in2D["score",id]=$col_2D

            score["score",id]=$col_score

            xlogp["score",id]=$col_x

            links["score",id]=$col_links
            }
        if(max["simi",id]=="" || max["simi",id]+0<row_simi+0)
            {
            max["simi",id]=row_simi

            name["simi",id]=$col_name

            formula["simi",id]=$col_formu

            simi["simi",id]=$col_simi

            smiles["simi",id]=$col_smiles

            inchi["simi",id]=$col_inchi

            in2D["simi",id]=$col_2D

            score["simi",id]=$col_score

            xlogp["simi",id]=$col_x
            links["simi",id]=$col_links
            }
        }
    }
    END{
        printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t" \
        \
        "score\t"  "xlogp\t"  "links\n" > "results/fingerid_sum.tsv"
        \
        printf "" > "temp/Mo_filename"

        for(l in file)
            {
            if(max["score",l]!="")
                {
                printf l"\t" \
                \
                name["score",l]"\t"  formula["score",l]"\t"  simi["score",l]"\t" \
                \
                smiles["score",l]"\t"  inchi["score",l]"\t"  in2D["score",l]"\t" \
                \
                score["score",l]"\t"  xlogp["score",l]"\t"  links["score",l]"\n" >> "results/fingerid_su

                printf formula["score",l]"\t"  file[l]"\t"  formu_type[l]"\n" >> "temp/Mo_filename"
                }
            if(max["simi",l]!="" && max["simi",l]!=max["score",l])
                {
                printf l"\t" \
                \
                name["simi",l]"\t"  formula["simi",l]"\t"  simi["simi",l]"\t" \
                \
                smiles["simi",l]"\t"  inchi["simi",l]"\t"  in2D["simi",l]"\t" \
                \
                score["simi",l]"\t"  xlogp["simi",l]"\t"  links["simi",l]"\n" >> "results/fingerid_sum.t
                }
            }
        }' $data

data_sum="results/fingerid_sum.tsv"

awk -F $'\t' '
```

```
{
if(NR==1)
    {
    for(i=1; i<=NF; i++)
        {
        if($i~/id/)
            {
            col_id=i
            }
        if($i~/score/)
            {
            col_score=i
            }
        }
    }
if(NR>=2)
    {
    id=$col_id
    \
    if(max_id+0<id || max_id=="")
        {
        max_id=id
        }
    if(score[id]+0<$col_score+0 || score[id]=="")
        {
        score[id]=$col_score
        \
        data[id]=$0
        }
    }
}
END{
    printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t" \
    \
    "score\t"  "xlogp\t"  "links\n" > "results/fingerid_first_score.tsv"
    \
    for(i=1; i<=max_id; i++)
        {
        if(data[i]!="")
            {
            printf data[i] "\n" >> "results/fingerid_first_score.tsv"
```

```
                }
            }
        }' $data_sum

echo "structure_extract results have been successfully assembled into <results/fingerid_sum.tsv> and <r
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
classification_extract_sum)

echo "Run classification_extract_sum.";
    if [ -f temp/Mo_filename ] && [ -f canopus.tsv ] && [ -f canopus_neg.tsv ] && [ -f .format ]
    then
    echo "Project path acknowledged."
    else
        until [ -d $projectpath ] && [ -f $projectpath/canopus.tsv ] && [ -f $projectpath/.format ]
        do
        read -p "Please input the path of the sirius project >>> " projectpath;
        done;
        cd $projectpath;
    fi;

data1="temp/Mo_filename"

data2="canopus.tsv"

data3="canopus_neg.tsv"

datas=$(awk -F $'\t' '
    {
    if(getline < $2"/canopus/"$3".fpt"==-1)
        {
        printf ""
        }
    else
        {
        printf $2"/canopus/"$3".fpt "
        }
    close($2"/canopus/"$3".fpt")
    }' $data1)

awk -F $'\t' '
    {
    if(NR==FNR)
        {
        file[FNR]=$2

        mo[$2]=$1

        i=split($2,s,"[_]")

        the_id[FNR]=s[i]

        n=FNR
        }
    if((FILENAME ~ /'$data2'/))
        {
        if(FNR==1)
            {
            close("'$data1'")

            printf "Info: data_file name of '$data1' has been input.\n"

            for(i=1; i<=NF; i++)
                {
                if(($i ~ /name/))
                    {
                    col_class=i
                    }
                if(($i ~ /absolute/))
                    {
                    col_abindex=i
                    }
                }
            }
        if(FNR>=2)
            {
            abindex[1,FNR]=$col_abindex
            class[1,$col_abindex]=$col_class

            rows_data2=FNR
            }
        }
    if((FILENAME ~ /'$data3'/))
        {
        if(FNR==1)
            {
            close("'$data2'")

            printf "Info: data_file name of '$data2' has been input.\n"

            for(i=1; i<=NF; i++)
                {
                if(($i ~ /name/))
                    {
                    col_class=i
                    }
                if(($i ~ /absolute/))
                    {
                    col_abindex=i
                    }
                }
            }
        if(FNR>=2)
            {
            abindex[2,FNR]=$col_abindex

            class[2,$col_abindex]=$col_class

            rows_data3=FNR
            }
        }
    if((FILENAME ~ /fpt/))
        {
        if(FNR==1)
            {
            if(p_filename=="")
                {
                close("'$data3'")
                }
            else if(FILENAME!=p_filename)
                {
                close(p_filename)

                printf "Info: data_file name of " p_filename " has been input.\n"
                }
            p_filename=FILENAME
            \
            for(i=1; i<=n; i++)
                {
                if((FILENAME ~ file[i]) && (FILENAME ~ mo[file[i]]))
                    {
                    break
                    }
                else
                    {
                    x=i+1
                    }
                }
            if(x==n+1)
                {
                nextfile
                }
            split(FILENAME,a,"[/]");

            m=split(a[1],b,"[_]");

            id=b[m];

            if((FILENAME ~ /\+.fpt/))
                {
                ion=1
                }
            else
                {
                ion=2
                }
            pp[id,abindex[ion,2]]=$1
            }
        if(FNR>=2)
```

```
            {
            pp[id,abindex[ion,FNR+1]]=$1
            }
        }
    }
    END{
        if(abindex[1,rows_data2] > abindex[2,rows_data3])
            {
            maxindex=abindex[1,rows_data2]
            }
        else
            {
            maxindex=abindex[2,rows_data3]
            }
        printf "id\t" > "results/canopus_pp.tsv"
        \
        for(i=0; i<=maxindex; i++)
            {
            if(class[1,i]!="" && i!=maxindex)
                {
                printf class[1,i]"\t" >> "results/canopus_pp.tsv"
                }
            else if(class[2,i]!="" && i!=maxindex)
                {
                printf class[2,i]"\t" >> "results/canopus_pp.tsv"
                }
            else if(i==maxindex && class[1,i]!="")
                {
                printf class[1,i]"\n" >> "results/canopus_pp.tsv"
                }
            else if(i==maxindex && class[2,i]!="")
                {
                printf class[2,i]"\n" >> "results/canopus_pp.tsv"
                }
            }
        for(i=1; i<=n; i++)
            {
            printf the_id[i]"\t" >> "results/canopus_pp.tsv"

            for(j=0; j<=maxindex; j++)
                {
```

```
                if(class[1,j]!="" && j!=maxindex || class[2,j]!="" && j!=maxindex)
                    {
                    if(pp[the_id[i],j]=="")
                        {
                        printf 0"\t" >> "results/canopus_pp.tsv"
                        }
                    else
                        {
                        printf pp[the_id[i],j]"\t" >> "results/canopus_pp.tsv"
                        }
                    }
                else if(class[1,j]!="" && j==maxindex || class[2,j]!="" && j==maxindex)
                    {
                    if(pp[the_id[i],j]=="")
                        {
                        printf 0"\n" >> "results/canopus_pp.tsv"
                        }
                    else
                        {
                        printf pp[the_id[i],j]"\n" >> "results/canopus_pp.tsv"
                        }
                    }
                }
            }
        }' $data1 $data2 $data3 $datas

echo "classification_extract_sum have been successfully written into <results/canopus_pp.tsv>"
;;
```

```
###################################
```

```
###################################
```

```
###################################


classification_extract_filter)
echo "Run classification_extract_filter.";
    if [ -f results/canopus_pp.tsv ]
    then
    echo "Project path acknowledged."
    else
        until [ -d $projectpath ] && [ -f $projectpath/results/canopus_pp.tsv ] && [ -f $projectpath/re
        do
        read -p "Please input the path of the sirius project >>> " projectpath;
        done;
        cd $projectpath;
    fi;

check=0
    until [[ "$check" == "yes" ]] || [[ "$check" == "no" ]]
    do
    read -p "Collate the posterior probabilities of the classification of each feature? [yes/no] >>> " c
    done;

    if [[ $check == "yes" ]]
    then
    definition_limit=0

        until [[ "$definition_limit" > "0.01" ]] && [[ "$definition_limit" < "1" ]]
        do
        read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>>
        done;

    data1="canopus_summary.tsv"

    data2="canopus.tsv"

    data3="results/canopus_pp.tsv"

    awk -F $'\t' '
        {
        if(NR==FNR)
            {
            if(FNR==1)
                {
                p_file=FILENAME
                \
                for(i=1; i<=NF; i++)
                    {
                    if($i~/name/)
                        {
                        col_id=i
                        }
                    if($i~/specific/)
                        {
                        col_specific=i
                        }
                    if($i~/level/)
                        {
                        col_level=i
                        }
                    if($i~/subclass/)
                        {
                        col_subclass=i
                        }
                    if($i~/^class/)
                        {
                        col_class=i
                        }
                    if($i~/superclass/)
                        {
                        col_superclass=i
                        }
                    }
                }
            if(FNR>=2)
                {
                n=split($col_id,a,"[_]")
                \
                id=a[n]
                \
                specific[id]=$col_specific
                \
                level[id]=$col_level
                \
                subclass[id]=$col_subclass
                \
                class[id]=$col_class
                \
                superclass[id]=$col_superclass
                }
            }
        if(FILENAME~/canopus.tsv/)
            {
            if(FNR==1)
```

```
                {
                close(p_file)
                \
                p_file=FILENAME
                \
                for(i=1; i<=NF; i++)
                    {
                    if($i~/name/)
                        {
                        col_name=i
                        }
                    if($i~/description/)
                        {
                        col_description=i
                        }
                    }
                }
            if(FNR>=2)
                {
                description[$col_name]=$col_description
                }
            }
        if(FILENAME~"'$data3'")
            {
            if(FNR==1)
                {
                close(p_file)
                \
                for(i=1; i<=NF; i++)
                    {
                    if($i~/^id$/)
                        {
                        col_id=i
                        }
                    if(i>=2)
                        {
                        col_class_name[i]=$i
                        }
                    }
                printf "id\t"  "definition_source\t"  "definition\t"  "definition_pp\t"  "definition_de
                \
                "specific\t"  "specific_pp\t"  "level_5\t"  "level_5_pp\t" \
                \
                "subclass\t"  "subclass_pp\t"  "class\t"  "class_pp\t"  "superclass\t"  "superclass_pp\
                \
                > "results/stat_classification.tsv"
                }
            if(FNR>=2)
                {
                for(i=2; i<=NF; i++)
                    {
                    c_pp[col_class_name[i]]=sprintf("%.4f",$i)
                    }
                if(level[$col_id] != "" && c_pp[level[$col_id]] >= "'$definition_limit'")
                    {
                    definition_source="level_5"
                    \
                    definition=level[$col_id]
                    }
                else if(subclass[$col_id] != "" && c_pp[subclass[$col_id]] >= "'$definition_limit'")
                    {
                    definition_source="subclass"
                    \
                    definition=subclass[$col_id]
                    }
                else if(class[$col_id] != "" && c_pp[class[$col_id]] >= "'$definition_limit'")
                    {
                    definition_source="class"
                    \
                    definition=class[$col_id]
                    }
                else if(superclass[$col_id] != "" && c_pp[superclass[$col_id]] >= "'$definition_limit'"
                    {
                    definition_source="superclass"
                    \
                    definition=superclass[$col_id]
                    }
                else
                    {
                    definition_source="null"
                    \
                    definition="null"
```

```
                    \
                    c_pp[definition]="null"
                    \
                    description[definition]="null"
                    }
                printf $col_id"\t"  definition_source"\t"  definition"\t"  c_pp[definition]"\t"  descri
                \
                specific[$col_id]"\t"  c_pp[specific[$col_id]]"\t"  level[$col_id]"\t"  c_pp[level[$col
                \
                subclass[$col_id]"\t"  c_pp[subclass[$col_id]]"\t"  class[$col_id]"\t"  c_pp[class[$col
                \
                superclass[$col_id]"\t"  c_pp[superclass[$col_id]]"\n" >> "results/stat_classification.
                }
            }
        }' $data1 $data2 $data3

    echo "classification_extract_filter have been successfully written into <results/stat_classificatio

    fi;
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
fragment_tree_network)

echo "Run fragment_tree_network.";

    if [ -d temp ] && [ -f ftalign.tsv ] && [ -f .format ]
    then
    echo "Project path acknowledged."

    else
        until [ -d $projectpath ] && [ -f $projectpath/ftalign.tsv ] && [ -f $projectpath/.format ]
        do
        read -p "Please input the path of the sirius project. Make sure you have moved the fragment tre
        done;
        cd $projectpath;
    fi;
```

```
echo "Please enter the minimum alignment similarity(tlimit) that you want to filter."

tlimit=0
    until [[ "$tlimit" > "0.01" ]] && [[ "$tlimit" < "0.99" ]]
    do
    read -p "0.4-0.7 is recommended >>> " tlimit;
    done;

    if ! [ -d temp/ftaligntemp ]
    then
    mkdir temp/ftaligntemp
    fi;

data="ftalign.tsv"

savepath="temp/ftaligntemp/tmp"

awk -F $'\t' -v OFS=$'\t' '
    {
    print NR,FNR

    if(NR==FNR)
        {
        for(i=1; i<=NF; i++)
            {
            if(NR==1 && i!=1 || NR!=1 && i==1)
                {
                n=split($i,x,"[_]")
                \
                raw[NR,i]=x[n]
                }
            else
                {
                if(NR==i)
                    {
                    raw_norm[NR,i]=$i
                    }
                }
            }
        }
    if(NR>FNR && FNR>=2)
```

```
        {
        for(i=2; i<=NF; i++)
            {
            norm1=${i}/raw_norm[i,i]

            norm2=${i}/raw_norm[FNR,FNR]

            norms=sprintf("%.2f", ((norm1+norm2)/2))

            if((norms+0 > '$tlimit'+0) && (norms+0 < 1+0))
                {
                if(raw[FNR,1]+0>=raw[1,i]+0)
                    {
                    print raw[FNR,1], raw[1,i], norms > "'$savepath'"
                    }
                if(raw[FNR,1]+0<raw[1,i]+0)
                    {
                    print raw[1,i], raw[FNR,1], norms > "'$savepath'"
                    }
                }
            }
        }
    }' $data $data

sort -u $savepath > temp/ftaligntemp/filter_net_$tlimit

echo "fragment_tree_network have been successfully written into <temp/ftaligntemp/filter_net_$tlimit>"
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################

fragment_tree_network_delta)

    if [ -d temp ] && [ -f ftalign.tsv ] && [ -f .format ]
    then
    echo "Project path acknowledged."
    else
```

```
        until [ -d $projectpath ] && [ -f $projectpath/ftalign.tsv ] && [ -f $projectpath/.format ]
        do
        read -p "Please input the path of the sirius project >>> " projectpath;
        done;
    cd $projectpath;
    fi;
echo "The following module attempts to compute the differential fingerprints of connected clusters base
```

```
###################
```

```
###################
```

```
###################

plimit=0
    until [[ "$plimit" > "0.5" ]] && [[ "$plimit" < "0.99" ]]
    do
    read -p "If the minimum alignment similarity is greater than 0.3, this step will automatic calculate
    done;

plimit2=0
    until [[ "$plimit2" > "0.1" ]] && [[ "$plimit2" < "$plimit" ]]
    do
    read -p "The minimum posterior probability of the molecular fingerprint(plimit_2) to be controled. 
    done;

tlimit=$(ls temp/ftaligntemp/filter_net_* | awk -F _ '{print $NF}')

check_rep=$(awk 'END{print NR}' <(echo "$tlimit"))

    if [[ "$check_rep" > "1" ]]
    then
    tlimit=0
        until [ -f temp/ftaligntemp/filter_net_$tlimit ]
        do
        read -p "Plural and different fragment_tree_network files were found to exist locally. Please s
        done;
    fi;


    if [ -f temp/ftaligntemp/refilter_net_$tlimit ]
    then rm temp/ftaligntemp/refilter_net_$tlimit
    fi;
    if [ -f temp/ftaligntemp/fpsample ]
    then rm temp/ftaligntemp/fpsample
    fi;

echo "Running delta_fingerprint computation..."

echo "Aquiring data from sirius index..."

data1="*_*/compound.info"

data2="temp/ftaligntemp/filter_net_$tlimit"

echo "Run fragment_tree_network_delta."

savepath="temp/ms_data_$tlimit"

echo "step 1: ms data"

    if ! [ -f $savepath ]
    then

    awk -F "[\t]||[:]||[_]" -v OFS=$'\t' '

        BEGIN{
            printf "..."

            printf "id\t"  "m/z\t"  "rt\n" > "results/mz_and_rt.tsv"
            }
        {
        if(FILENAME~/compound.info/)
            {
            if(FNR==1)
                {
                printf "Info: catch >>> "FILENAME"\n"
                }
            if($1=="name")
                {
                i+=1;

                id[i]=$NF;
                n=split($NF,a,"[_]")

                printf a[n]"\t" >> "results/mz_and_rt.tsv"
                }
            if($1=="ionMass")
                {
                mz[id[i]]=$NF

                printf sprintf("%.4f",$NF)"\t" >> "results/mz_and_rt.tsv"
                }
            if($1=="ionType")
                {
                type[id[i]]=$NF
                }
            if($1=="rt")
                {
                rt[id[i]]=$2;

                printf sprintf("%.2f",$2/60)"\n" >> "results/mz_and_rt.tsv"
                }
            }
        if(FILENAME~/ftaligntemp/)
            {
            #source  target  ftalign  delta_mz  delta_rt  source_iontype  target_iontype;

            if(FNR==1)
                {
                printf "" > "'$savepath'"
                }
            print $1,$2,$3,sprintf("%.3f",mz[$2]-mz[$1]),sprintf("%.2f",rt[$2]-rt[$1]),type[$1],type[$2
            \
            >> "'$savepath'"
            }
        }' $data1 $data2
     fi;
```

```
########################

echo "step 2: data path"

source_file="temp/Mo_filename"
```

```
data="temp/ms_data_$tlimit"

savepath="temp/datapath_$tlimit"

    if ! [ -f $savepath ]
    then

    awk -F $'\t' -v OFS=$'\t' '
        {
        if(NR==FNR)
            {
            n=split($2, a, "[_]")

            file[a[n]]=$2

            formu_type[a[n]]=$3
            }
        if(NR!=FNR)
            {
            #<path>sourceFormula  <path>targetFormula

            path1=file[$1]"/fingerprints/"formu_type[$1]".fpt"

            path2=file[$2]"/fingerprints/"formu_type[$2]".fpt"

            data[path1]=path1

            data[path2]=path2
            }
        }
        END{
            for(i in data)
                {
                n+=1

                print "Check file:",n

                if(getline<i==-1)
                    {
                    printf "Escape filename: " i "\n"
                    }
                else
                    {
                    printf i" " > "'$savepath'"
                    }
                close(i)
                }
            print "Sum:",n

            }' $source_file $data
    fi;

datapath=$(cat temp/datapath_$tlimit)

echo "step 3: fingerprint"

data_allfp="temp/data_allfp_$tlimit"

    if ! [ -f $data_allfp ]
    then

    awk -F $'\t' '
        BEGIN{
            n=0
            }
        {
        if(FNR==1)
            {
            if(n>1)
                {
                close(file)
                }
            file=FILENAME;

            print "Get fingerprints: ",FILENAME

            n+=1;

            printf FILENAME"\n"$0"\n" > "'$data_allfp'"
            }
        else
            {
```

```
            print $0 > "'$data_allfp'"
            }
        }' $datapath

    fi;
```

```
###########################



awk -F $'\t' -v OFS=$'\t' '
    {
    if(NR==1)
        {
        for(i=1; i<=NF; i++)
            {
            if($i~/absolute/)
                {
                col_index=i
                }
            if($i~/description/)
                {
                col_description=i
                }
            }
        }
    if(NR>=2)
        {
        print $col_index,$col_description
        }
    }' csi_fingerid.tsv > temp/ftaligntemp/pos_fingerprint_index;

awk -F $'\t' -v OFS=$'\t' '
    {
    if(NR==1)
        {
        for(i=1; i<=NF; i++)
            {
            if($i~/absolute/)
                {
                col_index=i
                }
            if($i~/description/)
```

```
                {
                col_description=i
                }
            }
        }
    if(NR>=2)
        {
        print $col_index,$col_description
        }
    }' csi_fingerid_neg.tsv > temp/ftaligntemp/neg_fingerprint_index;

posindex="temp/ftaligntemp/pos_fingerprint_index"

negindex="temp/ftaligntemp/neg_fingerprint_index"

data="temp/ms_data_$tlimit"

echo "step 4: merge data"

awk -F $'\t' '
    BEGIN{
        file=0
        x=0
        count=0
        f=0
        posnum=0
        negnum=0
        }
    {
    if(FNR==1)
        {
        file+=1
        }
    if(NR==FNR)
        {
        if(($1~/fingerprint/))
            {
            if(x+0>f+0)
                {
                f=x  # calculate the max index.
                }
            count+=1;  # calculate the all fingerprints file number.
            \
            split($1,a,"[/]"); e=split(a[1],b,"[_]"); id=b[e]; #catch the id.
            \
            x=0;
            }
        else
            {
            x+=1;
            \
            fp[id,x]=$1;
            }
        }
    if(file==2)
        {
        pos[FNR]=$1

        posnum+=1
        }
    if(file==3)
        {
        neg[FNR]=$1

        negnum+=1
        }
    if(file==4)
        {
        if("'$tlimit'"+0 >= 0.3)
            {
            if(fp[$1,1]!="" && fp[$2,1]!="")
                {
                n1=split($6, g, "[]]");

                n2=split($7, h, "[]]");

                if(g[n1]=="+" && h[n2]=="+")
                    {
                    for(x=1; x<=posnum; x++)
                        {
                        if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
                            {
                            data_s[FNR,x]=pos[x]
                            }
                        else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
                            {
                            data_t[FNR,x]=pos[x]
                            }
                        }
                    }
                if(g[n1]=="-" && h[n2]=="-")
                    {
                    for(x=1; x<=negnum; x++)
                        {
                        if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
                            {
                            data_s[FNR,x]=neg[x]
                            }
                        else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
                            {
                            data_t[FNR,x]=neg[x]
                            }
                        }
                    }
                if(g[n1]=="-" && h[n2]=="+" || h[n2]=="-" && g[n1]=="+")
                    {
                    for(i=1; i<=posnum; i++)
                        {
                        for(j=1; j<=negnum; j++)
                            {
                            if(pos[i]==neg[j])
                                {
                                mirror[i]=j     #if pos=i, the identical index of neg is mirror[i]
                                }
                            }
                        }
                    }
                if(g[n1]=="-" && h[n2]=="+")
                    {
                    for(x=1; x<=f; x++)
                        {
                        if(fp[$1,mirror[x]]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
                            {
```

```
                    data_s[FNR,x]=neg[mirror[x]]
                    }
                else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,mirror[x]]+0<='$plimit2'+0)
                    {
                    data_t[FNR,x]=neg[mirror[x]]
                    }
                }
            }
        if(h[n2]=="-" && g[n1]=="+")
            {
            for(x=1; x<=f; x++)
                {
                if(fp[$1,x]+0>= '$plimit'+0 && fp[$2,mirror[x]]+0<='$plimit2'+0)
                    {
                    data_s[FNR,x]=neg[mirror[x]]
                    }
                else if(fp[$2,mirror[x]]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
                    {
                    data_t[FNR,x]=neg[mirror[x]]
                    }
                }
            }
        }
    }
if(FNR==1)
    {
    printf "source\t"  "target\t"  "ftalign_similarity\t"  "delta_m/z\t"  "source_fp_uniq\t"  "
    }
else
    {
    printf $1"\t"  $2"\t"  $3"\t"  $4"\t";
    \
    if('$tlimit'+0 >= 0.3)
        {
        if(fp[$1,1]=="" || fp[$2,1]=="")
            {
            printf "NA@NA\n"
            }
        else
            {
            printf "source:"  #the source fingerprint uniq.
```

```
                    \
                    for(x=1; x<=f; x++)
                        {
                        if(data_s[FNR,x]!="")
                            {
                            printf data_s[FNR,x]","
                            }
                        };
                    printf "@target:"  #the target fingerprint uniq.
                    \
                    for(x=1; x<=f; x++)
                        {
                        if(data_t[FNR,x]!="")
                            {
                            printf data_t[FNR,x]","
                            }
                        }
                    printf "\n"
                    }
                }
            else
                {
                printf "NA@NA\n"
                }
            }
        }
    }' $data_allfp $posindex $negindex $data | sed -e 's/,@/\t/g; s/,$//g; s/@/\t/g' \
    \
    > results/source_target_tree_$tlimit.tsv;

echo "All instances have written into <results/source_target_tree_$tlimit.tsv>."
```

```
#####################################

echo "step 5: separate child-nebula from parent-nebula."

data="results/stat_classification.tsv"

savepath="temp/filter_0_class.tsv"

awk -F $'\t' '
    {
```

```
   if(FNR==1)
       {
       for(i=1; i<=NF; i++)
           {
           if($i~/^definition$/)
               {
               col_class=i
               }
           }
       }
   if(FNR>=2)
       {
       class[$col_class]=$col_class
       }
   }
   END{
       for(i in class)
           {
           printf class[i]"\n" > "'$savepath'"
           }
       }' $data

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

data1="temp/filter_0_class.tsv" #lignans and iridoids

data2="results/canopus_pp.tsv"

data3="results/fingerid_first_score.tsv"

   until [[ "$similarity_limit" > 0 ]] && [[ "$similarity_limit" < 1 ]]
   do
   read -p "Please set the Tanimoto similarity threshold contribute to child-nebula. >>> " similarity_l
   done;

   until [[ "$definition_limit" > 0.01 ]] && [[ "$definition_limit" < 1 ]]
   do
   read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>> " de
   done;

class_pp_limit=$definition_limit
```

```
savepath="temp/idenfication_filter_$class_pp_limit.tsv"

awk -F $'\t' '
   {
   if(NR==FNR)
       {
       filter_class[$1]=$1
       }
   if(FILENAME~/canopus/)
       {
       if(FNR==1)
           {
           col_id=1

           for(i=2; i<=NF; i++)
               {
               for(j in filter_class)
                   {
                   if(j==$i)
                       {
                       col_class[j]=i

                       print i
                       }
                   }
               }
           }
       if(FNR>=2)
           {
           for(i in col_class)
               {
               if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
                   {
                   class_set[i,$col_id]=i

                   print "ID: ",$1,i,$col_class[i]
                   }
               }
           }
       }
   if(FILENAME~/fingerid_first_score/)
```

```
        {
        if(FNR==1)
            {
            for(i=1; i<=NF; i++)
                {
                if($i~/^id/)
                    {
                    col_id=i
                    }
                if($i~/similarity/)
                    {
                    col_similarity=i
                    }
                }
            printf "class_nebula_facet\t"  $0"\n" > "'$savepath'"
            }
        if(FNR>=2)
            {
            if($col_similarity+0 >= "'$similarity_limit'"+0)
                {
                for(i in class_set)
                    {
                    if(i~"\034"$col_id"$")
                        {
                        printf class_set[i]"\t"  $0"\n" > "'$savepath'"
                        }
                    }
                }
            }
        }
    }' $data1 $data2 $data3
```

```
##################################

 data="temp/idenfication_filter_$class_pp_limit.tsv"

 savepath="temp/idenfication_filter2_${class_pp_limit}.tsv"

    until [[ "$num_limit" -gt 0 ]]
    do
    read -p "Please enter the features number threshold contribute to child-nebula. >>> " num_limit;
    done;
```

```
awk -F $'\t' '
   {
   if(FNR==1)
       {
       printf $0"\n" > "'$savepath'"

       for(i=1; i<=NF; i++)
           {
           if($i~/class_nebula_facet/)
               {
               col_class=i
               }
           }
       }
   if(FNR>=2)
       {
       num[$col_class]+=1

       data[FNR]=$0

       class[FNR]=$col_class
       }
   }
   END{
       for(i in num)
           {
           print num[i]
           if(num[i] >= '$num_limit')
               {
               for(j in class)
                   {
                   if(class[j]==i)
                       {
                       printf data[j]"\n" >> "'$savepath'"
                       }
                   }
               }
           }
       }' $data

##################################
```

```
mkdir results/network_facet_$class_pp_limit

data1="temp/idenfication_filter2_${class_pp_limit}.tsv"

data2="results/source_target_tree_$tlimit.tsv" # "results/source_target_tree_0.4.tsv"

save_class="results/filter_child_class.tsv"

savepath="results/network_facet_$class_pp_limit/"

awk -F $'\t' '
   {
   if(NR==FNR)
       {
       if(FNR==1)
           {
           for(i=1; i<=NF; i++)
               {
               if($i~/class_nebula_facet/)
                   {
                   col_class=i
                   }
               if($i~/^id$/)
                   {
                   col_id=i
                   }
               }
           }
       if(FNR>=2)
           {
           class[$col_class]=$col_class

           class_id[$col_class,$col_id]=$col_id

           stat_id[$col_class,$col_id]=$col_id

           belong[$col_class,$col_id]=$col_class
           }
       }
   if(NR>FNR)
       {
        if(FNR==1)
            {
            for(i in class)
                {
                printf i"\n" > "'$save_class'"

                printf $0"\t"  "facet\n" > "'$savepath'" i ".tsv"
                }
            }
        if(FNR>=2)
            {
            for(i in class)
                {
                if( class_id[i,$1]==$1 && class_id[i,$2]==$2 )
                    {
                    printf $0"\t"  i"\n" >> "'$savepath'" i ".tsv"

                    delete stat_id[i,$1]

                    delete stat_id[i,$2]
                    }
                }
            }
        }
    }
    END{
        for(i in stat_id)
            {
            ## source target similarity delta_mz fp fp class

            printf stat_id[i]"\t" stat_id[i]"\t"  "1\t"  "0\t"  "null\t"  "null\t"  belong[i]"\n" \
            \
            >> "'$savepath'" belong[i] ".tsv"
            }
        }' $data1 $data2

;;
```

```
###################################
```

```
###################################
compound_idenfication)
```

```
echo "Run compound_idenfication."








echo "compound_idenfication "

exit

;;
```

```
##################################
```

```
##################################
double_ion_network)

exit;

;;
```

```
##################################
```

```
##################################
```

```
##################################
```

```
##################################
exit)
echo "The mystery of creation is like the darkness of night--it is great.
Delusions of knowledge are like the fog of the morning."
exit;
;;
```

```
##################################
```

```
##################################
```

```
##################################
```

```
##################################
*)
echo "error"
exit;
```

```
;;
esac;
done;#(for)
done;#(select)
```

22 File: mcnebula1110.sh

```
##########################
```

```
##########################
echo "We are all in the gutter,
but some of us are looking at the stars.";
PS3='Please select the workflow to be executed. >>> '
select command in \
   \
  "default" \
  \
  "structure_extract" \
  \
  "classification_extract_sum" \
  \
  "classification_extract_filter" \
  \
  "fragment_tree_network" \
  \
  "fragment_tree_network_delta" \
  \
  "compound_idenfication" \
  \
  "double_ion_network" \
  \
  "exit"
do
 if [[ $command == "default" ]]
 then
 confirm=0
   until [[ $confirm == "yes" ]] || [[ $confirm == "no" ]]
   do
   read -p "Running all proccesses sequentially? [yes/no] >>> " confirm
   done;
   if [[ $confirm == "no" ]]
```

```
    then exit
    fi;
  default="structure_extract classification_extract_sum classification_extract_filter fragment_tree_netw
  list=$(echo $default);
  else list=$( echo $command)
  fi;
  for option in $(echo $list)
  do
    case $option in
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
structure_extract)
echo "Run structure_extract."
projectpath=0
 until [ -d $projectpath ] && [ -f $projectpath/canopus_summary.tsv ] && [ -f $projectpath/.format ]
 do
 read -p "Please input the path of the sirius project >>> " projectpath;
 done;
cd $projectpath;
mkdir results;
mkdir temp;
mkdir temp/fintemp;
 if [ -f temp/Mo_filename ]
 then rm temp/Mo_filename
 fi;
data="*_*/fingerid/*.tsv"
awk -F $'\t' '
  BEGIN{
    all_id=0
    \
    max_id=-1
    \
    p_id="null"
    \
    printf "Info: loading the data..."
    }
  {
```

```
if(FNR==1)
  {
  if(NR>FNR)
    {
    close(pfile)
    }
  f=split(FILENAME,a,"[/]");
  n=split(a[1],b,"[_]");
  id=b[n];
  if(id!=p_id)
    {
    all_id+=1
    \
    if(id>max_id)
      {
      max_id=id
      }
    }
  file[id]=a[1]
  split(a[f], g, "[.]")
  formu_type[id]=g[1]
  if(all_id==1)
    {
    for(i=1; i<=NF; i++)
      {
      if(($i~/inchikey2D/))
        {
        col_2D=i
        }
      if($i=="inchi")
        {
        col_inchi=i
        }
      if(($i~/Formula/))
        {
        col_formu=i
        }
      if($i=="score")
        {
        col_score=i
        }
      if($i=="name")
        {
        col_name=i
        }
      if($i=="smiles")
        {
        col_smiles=i
        }
      if($i=="xlogp")
        {
        col_x=i
        }
      if(($i~/imilarity/))
        {
        col_simi=i
        }
      if($i~/links/)
        {
        col_links=i
        }
      }
    }
  pfile=FILENAME
  printf g[1]"\t"formu_type[id]"\n"
  printf "Info: the data of " id " (ID) has input. The filename is " FILENAME ".\n"
  }
if(FNR>=2)
  {
  row_score=$col_score
  row_simi=$col_simi
  if(max["score",id]=="" || max["score",id]+0<row_score+0)
   {
    max["score",id]=row_score
    name["score",id]=$col_name
    formula["score",id]=$col_formu
    simi["score",id]=$col_simi
    smiles["score",id]=$col_smiles
    inchi["score",id]=$col_inchi
    in2D["score",id]=$col_2D
    score["score",id]=$col_score
    xlogp["score",id]=$col_x
    links["score",id]=$col_links
    }
  if(max["simi",id]=="" || max["simi",id]+0<row_simi+0)
   {
    max["simi",id]=row_simi
    name["simi",id]=$col_name
    formula["simi",id]=$col_formu
    simi["simi",id]=$col_simi
    smiles["simi",id]=$col_smiles
    inchi["simi",id]=$col_inchi
    in2D["simi",id]=$col_2D
    score["simi",id]=$col_score
    xlogp["simi",id]=$col_x
    links["simi",id]=$col_links
    }
  }
}
END{
  printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t" \
  \
  "score\t"  "xlogp\t"  "links\n" > "results/fingerid_sum.tsv"
  \
  printf "" > "temp/Mo_filename"
  for(l in file)
    {
    if(max["score",l]!="")
      {
      printf l"\t" \
      \
      name["score",l]"\t"  formula["score",l]"\t"  simi["score",l]"\t" \
      \
      smiles["score",l]"\t"  inchi["score",l]"\t"  in2D["score",l]"\t" \
      \
      score["score",l]"\t"  xlogp["score",l]"\t"  links["score",l]"\n" >> "results/fingerid_sum.tsv"
      printf formula["score",l]"\t"  file[l]"\t"  formu_type[l]"\n" >> "temp/Mo_filename"
      }
    if(max["simi",l]!="" && max["simi",l]!=max["score",l])
      {
      printf l"\t" \
      \
      name["simi",l]"\t"  formula["simi",l]"\t"  simi["simi",l]"\t" \
      \
      smiles["simi",l]"\t"  inchi["simi",l]"\t"  in2D["simi",l]"\t" \
      \
      score["simi",l]"\t"  xlogp["simi",l]"\t"  links["simi",l]"\n" >> "results/fingerid_sum.tsv"
      }
    }
  }' $data
data_sum="results/fingerid_sum.tsv"
awk -F $'\t' '
{
if(NR==1)
  {
  for(i=1; i<=NF; i++)
    {
    if($i~/id/)
      {
      col_id=i
      }
    if($i~/score/)
      {
      col_score=i
      }
    }
  }
if(NR>=2)
  {
  id=$col_id
  \
  if(max_id+0<id || max_id=="")
    {
    max_id=id
    }
  if(score[id]+0<$col_score+0 || score[id]=="")
    {
    score[id]=$col_score
    \
    data[id]=$0
    }
  }
}
END{
  printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t" \
  \
  "score\t"  "xlogp\t"  "links\n" > "results/fingerid_first_score.tsv"
  \
  for(i=1; i<=max_id; i++)
    {
    if(data[i]!="")
      {
      printf data[i] "\n" >> "results/fingerid_first_score.tsv"
      }
    }
  }' $data_sum
echo "structure_extract results have been successfully assembled into <results/fingerid_sum.tsv> and <r
;;
```

```
####################################
```

```
####################################
```

```
####################################
```

```
####################################
classification_extract_sum)
echo "Run classification_extract_sum.";
 if [ -f temp/Mo_filename ] && [ -f canopus.tsv ] && [ -f canopus_neg.tsv ] && [ -f .format ]
 then
 echo "Project path acknowledged."
 else
   until [ -d $projectpath ] && [ -f $projectpath/canopus.tsv ] && [ -f $projectpath/.format ]
   do
   read -p "Please input the path of the sirius project >>> " projectpath;
   done;
   cd $projectpath;
 fi;
data1="temp/Mo_filename"
data2="canopus.tsv"
data3="canopus_neg.tsv"
datas=$(awk -F $'\t' '
  {
  x=$2"/canopus/"$3".fpt"
  if(getline < x == 1)
    {
    printf x" "
    }
  close(x)
  }' $data1)
awk -F $'\t' '
  {
  if(NR==FNR)
    {
    file[FNR]=$2
    mo[$2]=$1
    i=split($2,s,"[_]")
    the_id[FNR]=s[i]
    n=FNR
    }
  if((FILENAME ~ /'$data2'/))
    {
    if(FNR==1)
      {
      close("'$data1'")
      printf "Info: data_file name of '$data1' has been input.\n"
      for(i=1; i<=NF; i++)
        {
        if(($i ~ /name/))
          {
          col_class=i
          }
        if(($i ~ /absolute/))
          {
          col_abindex=i
          }
        }
      }
    if(FNR>=2)
      {
      abindex[1,FNR]=$col_abindex
      class[1,$col_abindex]=$col_class
      rows_data2=FNR
      }
    }
  if((FILENAME ~ /'$data3'/))
    {
    if(FNR==1)
      {
      close("'$data2'")
      printf "Info: data_file name of '$data2' has been input.\n"
      for(i=1; i<=NF; i++)
        {
        if(($i ~ /name/))
          {
          col_class=i
          }
        if(($i ~ /absolute/))
          {
          col_abindex=i
          }
        }
      }
    if(FNR>=2)
      {
      abindex[2,FNR]=$col_abindex
      class[2,$col_abindex]=$col_class
      rows_data3=FNR
      }
    }
  if((FILENAME ~ /fpt/))
    {
    if(FNR==1)
      {
      if(p_filename=="")
        {
        close("'$data3'")
        }
      else if(FILENAME!=p_filename)
        {
        close(p_filename)
        printf "Info: data_file name of " p_filename " has been input.\n"
        }
      p_filename=FILENAME
      for(i=1; i<=n; i++)
        {
        if((FILENAME ~ file[i]) && (FILENAME ~ mo[file[i]]))
          {
          break
          }
        else
          {
          x=i+1
          }
        }
      if(x==n+1)
        {
        nextfile
        }
      split(FILENAME,a,"[/]");
      m=split(a[1],b,"[_]");
      id=b[m];
      if(FILENAME ~ /\+.fpt/)
        {
        ion=1
        }
      if(FILENAME ~ /\-.fpt/)
        {
        ion=2
        }
      pp[id,abindex[ion,2]]=$1
      }
    if(FNR>=2)
      {
      pp[id,abindex[ion,FNR+1]]=$1
      }
    }
  }
  END{
    if(abindex[1,rows_data2]+0 > abindex[2,rows_data3]+0)
      {
      maxindex=abindex[1,rows_data2]
      }
    else
      {
      maxindex=abindex[2,rows_data3]
      }
    printf "id\t" > "results/canopus_pp.tsv"
    \
    for(i=0; i<=maxindex; i++)
```

```
    {
    if(class[1,i]!="" && i!=maxindex)
      {
      printf class[1,i]"\t" >> "results/canopus_pp.tsv"
      }
    else if(class[2,i]!="" && i!=maxindex)
      {
      printf class[2,i]"\t" >> "results/canopus_pp.tsv"
      }
    else if(i==maxindex && class[1,i]!="")
      {
      printf class[1,i]"\n" >> "results/canopus_pp.tsv"
      }
    else if(i==maxindex && class[2,i]!="")
      {
      printf class[2,i]"\n" >> "results/canopus_pp.tsv"
      }
    }
  for(i=1; i<=n; i++)
    {
    printf the_id[i]"\t" >> "results/canopus_pp.tsv"
    for(j=0; j<=maxindex; j++)
      {
      if(class[1,j]!="" && j!=maxindex || class[2,j]!="" && j!=maxindex)
        {
        if(pp[the_id[i],j]=="")
          {
          printf 0"\t" >> "results/canopus_pp.tsv"
          }
        else
          {
          printf pp[the_id[i],j]"\t" >> "results/canopus_pp.tsv"
          }
        }
      else if(class[1,j]!="" && j==maxindex || class[2,j]!="" && j==maxindex)
        {
        if(pp[the_id[i],j]=="")
          {
          printf 0"\n" >> "results/canopus_pp.tsv"
          }
        else
```

```
            {
            printf pp[the_id[i],j]"\n" >> "results/canopus_pp.tsv"
            }
          }
        }
      }
    }' $data1 $data2 $data3 $datas
echo "classification_extract_sum have been successfully written into <results/canopus_pp.tsv>"
;;
```

```
###################################
```

```
###################################
```

```
###################################
classification_extract_filter)
echo "Run classification_extract_filter.";
 if [ -f results/canopus_pp.tsv ]
 then
 echo "Project path acknowledged."
 else
   until [ -d $projectpath ] && [ -f $projectpath/results/canopus_pp.tsv ] && [ -f $projectpath/results/
   do
   read -p "Please input the path of the sirius project >>> " projectpath;
   done;
   cd $projectpath;
 fi;
check=0
 until [[ "$check" == "yes" ]] || [[ "$check" == "no" ]]
 do
 read -p "Collate the posterior probabilities of the classification of each feature? [yes/no] >>> " che
 done;
 if [[ $check == "yes" ]]
 then
 definition_limit=0
   until [[ "$definition_limit" > "0.01" ]] && [[ "$definition_limit" < "1" ]]
   do
   read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>> " de
   done;
 data1="canopus_summary.tsv"
  data2="canopus.tsv"
  data3="results/canopus_pp.tsv"
 awk -F $'\t' '
```

```
{
if(NR==FNR)
  {
  if(FNR==1)
    {
    p_file=FILENAME
    \
    for(i=1; i<=NF; i++)
      {
      if($i~/name/)
        {
        col_id=i
        }
      if($i~/specific/)
        {
        col_specific=i
        }
      if($i~/level/)
        {
        col_level=i
        }
      if($i~/subclass/)
        {
        col_subclass=i
        }
      if($i~/^class/)
        {
        col_class=i
        }
      if($i~/superclass/)
        {
        col_superclass=i
        }
      }
    }
  if(FNR>=2)
    {
    n=split($col_id,a,"[_]")
    \
    id=a[n]
    \
    specific[id]=$col_specific
    \
    level[id]=$col_level
    \
    subclass[id]=$col_subclass
    \
    class[id]=$col_class
    \
    superclass[id]=$col_superclass
    }
  }
if(FILENAME~/canopus.tsv/)
  {
  if(FNR==1)
    {
    close(p_file)
    \
    p_file=FILENAME
    \
    for(i=1; i<=NF; i++)
      {
      if($i~/name/)
        {
        col_name=i
        }
      if($i~/description/)
        {
        col_description=i
        }
      }
    }
  if(FNR>=2)
    {
    description[$col_name]=$col_description
    }
  }
if(FILENAME~"'$data3'")
  {
  if(FNR==1)
    {
    close(p_file)
    \
    for(i=1; i<=NF; i++)
      {
      if($i~/^id$/)
        {
        col_id=i
        }
      if(i>=2)
        {
        col_class_name[i]=$i
        }
      }
    printf "id\t"  "definition_source\t"  "definition\t"  "definition_pp\t"  "definition_description
    \
    "specific\t"  "specific_pp\t"  "level_5\t"  "level_5_pp\t" \
    \
    "subclass\t"  "subclass_pp\t"  "class\t"  "class_pp\t"  "superclass\t"  "superclass_pp\n" \
    \
    > "results/stat_classification.tsv"
    }
  if(FNR>=2)
    {
    for(i=2; i<=NF; i++)
      {
      c_pp[col_class_name[i]]=sprintf("%.4f",$i)
      }
    if(level[$col_id] != "" && c_pp[level[$col_id]] >= "'$definition_limit'")
      {
      definition_source="level_5"
      \
      definition=level[$col_id]
      }
    else if(subclass[$col_id] != "" && c_pp[subclass[$col_id]] >= "'$definition_limit'")
      {
      definition_source="subclass"
      \
      definition=subclass[$col_id]
      }
    else if(class[$col_id] != "" && c_pp[class[$col_id]] >= "'$definition_limit'")
      {
      definition_source="class"
```

```
         \
         definition=class[$col_id]
         }
       else if(superclass[$col_id] != "" && c_pp[superclass[$col_id]] >= "'$definition_limit'")
         {
         definition_source="superclass"
         \
         definition=superclass[$col_id]
         }
       else
         {
         definition_source="null"
         \
         definition="null"
         \
         c_pp[definition]="null"
         \
         description[definition]="null"
         }
       printf $col_id"\t"  definition_source"\t"  definition"\t"  c_pp[definition]"\t"  description[defi
       \
       specific[$col_id]"\t"  c_pp[specific[$col_id]]"\t"  level[$col_id]"\t"  c_pp[level[$col_id]]"\t"
       \
       subclass[$col_id]"\t"  c_pp[subclass[$col_id]]"\t"  class[$col_id]"\t"  c_pp[class[$col_id]]"\t"
       \
       superclass[$col_id]"\t"  c_pp[superclass[$col_id]]"\n" >> "results/stat_classification.tsv"
       }
     }
   }' $data1 $data2 $data3
  echo "classification_extract_filter have been successfully written into <results/stat_classification.
  fi;
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
fragment_tree_network)
echo "Run fragment_tree_network.";
 if [ -d temp ] && [ -f ftalign.tsv ] && [ -f .format ]
 then
 echo "Project path acknowledged."
 else
   until [ -d $projectpath ] && [ -f $projectpath/ftalign.tsv ] && [ -f $projectpath/.format ]
   do
   read -p "Please input the path of the sirius project. Make sure you have moved the fragment tree ali
   done;
   cd $projectpath;
 fi;
echo "Please enter the minimum alignment similarity(tlimit) that you want to filter."
tlimit=0
 until [[ "$tlimit" > "0.01" ]] && [[ "$tlimit" < "0.99" ]]
 do
 read -p "0.4-0.7 is recommended >>> " tlimit;
 done;
  if ! [ -d temp/ftaligntemp ]
 then
 mkdir temp/ftaligntemp
  fi;
data="ftalign.tsv"
savepath="temp/ftaligntemp/tmp"
awk -F $'\t' -v OFS=$'\t' '
  {
  printf "Info: NR = " NR ". FNR = " FNR ".\n"
  if(NR==FNR)
    {
   for(i=1; i<=NF; i++)
     {
     if(NR==1 && i!=1 || NR!=1 && i==1)
       {
       n=split($i,x,"[_]")
       \
       raw[NR,i]=x[n]
       }
     else
       {
       if(NR==i)
         {
         raw_norm[NR,i]=$i
         }
       }
```

```
      }
    }
  if(NR>FNR && FNR>=2)
    {
    for(i=2; i<=NF; i++)
      {
      norm1=$i/raw_norm[i,i]
      norm2=$i/raw_norm[FNR,FNR]
      norms=sprintf("%.2f", ((norm1+norm2)/2))
      if((norms+0 > '$tlimit'+0))
        {
        if(raw[FNR,1]+0>=raw[1,i]+0)
          {
          print raw[FNR,1], raw[1,i], norms > "'$savepath'"
          }
        if(raw[FNR,1]+0<raw[1,i]+0)
          {
          print raw[1,i], raw[FNR,1], norms > "'$savepath'"
          }
        }
      }
    }
  }' $data $data
sort -u $savepath > temp/ftaligntemp/tmp2
savepath="temp/ftaligntemp/filter_net_$tlimit"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if($1!=$2)
        {
          replink[$1]=$1
          replink[$2]=$2
        }
    }
  if(NR!=FNR)
    {
      if($1!=$2)
        {
          printf $0"\n" > "'$savepath'"
        }
      if($1==$2)
        {
          if(replink[$1]=="")
            {
              printf $0"\n" > "'$savepath'"
            }
        }
    }
}' temp/ftaligntemp/tmp2 temp/ftaligntemp/tmp2
echo "fragment_tree_network have been successfully written into <temp/ftaligntemp/filter_net_$tlimit>"
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
fragment_tree_network_delta)
 if [ -d temp ] && [ -f ftalign.tsv ] && [ -f .format ]
 then
 echo "Project path acknowledged."
 else
   until [ -d $projectpath ] && [ -f $projectpath/ftalign.tsv ] && [ -f $projectpath/.format ]
   do
   read -p "Please input the path of the sirius project >>> " projectpath;
   done;
 cd $projectpath;
 fi;
echo "The following module attempts to compute the differential fingerprints of connected clusters base
```

```
###################
```

```
###################
```

```
###################
plimit=0
 until [[ "$plimit" > "0.5" ]] && [[ "$plimit" < "0.99" ]]
 do
 read -p "If the minimum alignment similarity is greater than 0.3, this step will automatic calculate t
 done;
plimit2=0
 until [[ "$plimit2" > "0.1" ]] && [[ "$plimit2" < "$plimit" ]]
 do
 read -p "The minimum posterior probability of the molecular fingerprint(plimit_2) to be controled. 0.1
 done;
tlimit=$(ls temp/ftaligntemp/filter_net_* | awk -F _ '{print $NF}')
check_rep=$(awk 'END{print NR}' <(echo "$tlimit"))
 if [[ "$check_rep" > "1" ]]
 then
 tlimit=0
   until [ -f temp/ftaligntemp/filter_net_$tlimit ]
   do
   read -p "Plural and different fragment_tree_network files were found to exist locally. Please select
   done;
 fi;
 if [ -f temp/ftaligntemp/refilter_net_$tlimit ]
 then rm temp/ftaligntemp/refilter_net_$tlimit
 fi;
 if [ -f temp/ftaligntemp/fpsample ]
 then rm temp/ftaligntemp/fpsample
 fi;
echo "Running delta_fingerprint computation..."
echo "Aquiring data from sirius index..."
data1="*_*/compound.info"
data2="temp/ftaligntemp/filter_net_$tlimit"
echo "Run fragment_tree_network_delta."
savepath="temp/ms_data_$tlimit"
echo "step 1: ms data"
  if ! [ -f $savepath ]
  then
 awk -F "[\t]||[:]||[_]" -v OFS=$'\t' '
   BEGIN{
     printf "..."
     printf "id\t"  "m/z\t"  "rt\n" > "results/mz_and_rt.tsv"
     }
   {
   if(FILENAME~/compound.info/)
     {
     if(FNR==1)
       {
       printf "Info: catch >>> "FILENAME"\n"
       }
     if($1=="name")
       {
```

```
      i+=1;
      id[i]=$NF;
      n=split($NF,a,"[_]")
      printf a[n]"\t" >> "results/mz_and_rt.tsv"
      }
    if($1=="ionMass")
      {
      mz[id[i]]=$NF
      printf sprintf("%.4f",$NF)"\t" >> "results/mz_and_rt.tsv"
      }
    if($1=="ionType")
      {
      type[id[i]]=$NF
      }
    if($1=="rt")
      {
      rt[id[i]]=$2;
      printf sprintf("%.2f",$2/60)"\n" >> "results/mz_and_rt.tsv"
      }
    }
  if(FILENAME~/ftaligntemp/)
    {
    #source  target  ftalign  delta_mz  delta_rt  source_iontype  target_iontype;
    if(FNR==1)
      {
      printf "" > "'$savepath'"
      }
    print $1,$2,$3,sprintf("%.3f",mz[$2]-mz[$1]),sprintf("%.2f",rt[$2]-rt[$1]),type[$1],type[$2] \
    \
    >> "'$savepath'"
    }
  }' $data1 $data2
 fi;
```

```
########################
echo "step 2: data path"
source_file="temp/Mo_filename"
data="temp/ms_data_$tlimit"
savepath="temp/datapath_$tlimit"
  if ! [ -f $savepath ]
  then
 awk -F $'\t' -v OFS=$'\t' '
```

```
   {
   if(NR==FNR)
     {
     n=split($2, a, "[_]")
     file[a[n]]=$2
     formu_type[a[n]]=$3
     }
   if(NR!=FNR)
     {
     #<path>sourceFormula  <path>targetFormula
     path1=file[$1]"/fingerprints/"formu_type[$1]".fpt"
     path2=file[$2]"/fingerprints/"formu_type[$2]".fpt"
     data[path1]=path1
     data[path2]=path2
     }
   }
   END{
     for(i in data)
       {
       n+=1
       print "Check file:",n
       if(getline<i==-1)
         {
         printf "Escape filename: " i "\n"
         }
       else
         {
         printf i" " > "'$savepath'"
         }
       close(i)
       }
     print "Sum:",n
     }' $source_file $data
   fi;
datapath=$(cat temp/datapath_$tlimit)
echo "step 3: fingerprint"
data_allfp="temp/data_allfp_$tlimit"
  if ! [ -f $data_allfp ]
  then
 awk -F $'\t' '
   BEGIN{
     n=0
     }
   {
   if(FNR==1)
     {
     if(n>1)
       {
       close(file)
       }
     file=FILENAME;
     print "Get fingerprints: ",FILENAME
     n+=1;
     printf FILENAME"\n"$0"\n" > "'$data_allfp'"
     }
   else
     {
     print $0 > "'$data_allfp'"
     }
   }' $datapath
  fi;
```

```
###########################
awk -F $'\t' -v OFS=$'\t' '
  {
  if(NR==1)
    {
    for(i=1; i<=NF; i++)
      {
      if($i~/absolute/)
        {
        col_index=i
        }
      if($i~/description/)
        {
        col_description=i
        }
      }
    }
  if(NR>=2)
    {
    print $col_index,$col_description
    }
  }' csi_fingerid.tsv > temp/ftaligntemp/pos_fingerprint_index;
awk -F $'\t' -v OFS=$'\t' '
  {
  if(NR==1)
    {
    for(i=1; i<=NF; i++)
      {
      if($i~/absolute/)
        {
        col_index=i
        }
      if($i~/description/)
        {
        col_description=i
        }
      }
    }
  if(NR>=2)
    {
    print $col_index,$col_description
    }
  }' csi_fingerid_neg.tsv > temp/ftaligntemp/neg_fingerprint_index;
posindex="temp/ftaligntemp/pos_fingerprint_index"
negindex="temp/ftaligntemp/neg_fingerprint_index"
data="temp/ms_data_$tlimit"
echo "step 4: merge data"
awk -F $'\t' '
  BEGIN{
    file=0
    x=0
    count=0
    f=0
    posnum=0
    negnum=0
    }
  {
  if(FNR==1)
    {
    file+=1
    }
  if(NR==FNR)
    {
    if(($1~/fingerprint/))
      {
      if(x+0>f+0)
        {
        f=x  # calculate the max index.
        }
      count+=1;  # calculate the all fingerprints file number.
      \
      split($1,a,"[/]"); e=split(a[1],b,"[_]"); id=b[e]; #catch the id.
      \
      x=0;
      }
    else
      {
      x+=1;
      \
      fp[id,x]=$1;
      }
    }
  if(file==2)
    {
    pos[FNR]=$1
    posnum+=1
    }
  if(file==3)
    {
    neg[FNR]=$1
    negnum+=1
    }
  if(file==4)
    {
    if("'$tlimit'"+0 >= 0.3)
      {
     if(fp[$1,1]!="" && fp[$2,1]!="")
       {
       n1=split($6, g, "[]]");
       n2=split($7, h, "[]]");
       if(g[n1]=="+" && h[n2]=="+")
         {
         for(x=1; x<=posnum; x++)
           {
           if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
             {
             data_s[FNR,x]=pos[x]
             }
           else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
             {
             data_t[FNR,x]=pos[x]
             }
           }
         }
       if(g[n1]=="-" && h[n2]=="-")
         {
         for(x=1; x<=negnum; x++)
           {
           if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
             {
             data_s[FNR,x]=neg[x]
             }
           else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
             {
             data_t[FNR,x]=neg[x]
             }
           }
         }
       if(g[n1]=="-" && h[n2]=="+" || h[n2]=="-" && g[n1]=="+")
         {
         for(i=1; i<=posnum; i++)
           {
           for(j=1; j<=negnum; j++)
             {
             if(pos[i]==neg[j])
               {
               mirror[i]=j  #if pos=i, the identical index of neg is mirror[i]
               }
             }
           }
         }
       if(g[n1]=="-" && h[n2]=="+")
         {
         for(x=1; x<=f; x++)
```

```
          {
          if(fp[$1,mirror[x]]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
            {
            data_s[FNR,x]=neg[mirror[x]]
            }
          else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,mirror[x]]+0<='$plimit2'+0)
            {
            data_t[FNR,x]=neg[mirror[x]]
            }
          }
        }
      if(h[n2]=="-" && g[n1]=="+")
        {
        for(x=1; x<=f; x++)
          {
          if(fp[$1,x]+0>= '$plimit'+0 && fp[$2,mirror[x]]+0<='$plimit2'+0)
            {
            data_s[FNR,x]=neg[mirror[x]]
            }
          else if(fp[$2,mirror[x]]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
            {
            data_t[FNR,x]=neg[mirror[x]]
            }
          }
        }
      }
    }
   if(FNR==1)
     {
     printf "source\t"  "target\t"  "ftalign_similarity\t"  "delta_m/z\t"  "source_fp_uniq\t"  "target_
     }
   else
     {
     printf $1"\t"  $2"\t"  $3"\t"  $4"\t";
     \
     if('$tlimit'+0 >= 0.3)
       {
      if(fp[$1,1]=="" || fp[$2,1]=="")
        {
        printf "NA@NA\n"
        }
      else
        {
        printf "source:"  #the source fingerprint uniq.
        \
        for(x=1; x<=f; x++)
          {
          if(data_s[FNR,x]!="")
            {
            printf data_s[FNR,x]","
            }
          };
        printf "@target:"  #the target fingerprint uniq.
        \
        for(x=1; x<=f; x++)
          {
          if(data_t[FNR,x]!="")
            {
            printf data_t[FNR,x]","
            }
          }
        printf "\n"
        }
      }
    else
      {
      printf "NA@NA\n"
      }
     }
   }
  }' $data_allfp $posindex $negindex $data | sed -e 's/,@/\t/g; s/,$//g; s/@/\t/g' \
  \
  > results/source_target_tree_$tlimit.tsv;
echo "All instances have written into <results/source_target_tree_$tlimit.tsv>."
```

```
#####################################
echo "step 5: separate child-nebula from parent-nebula."
data="results/stat_classification.tsv"
savepath="temp/filter_0_class.tsv"
awk -F $'\t' '
  {
  if(FNR==1)
    {
```

```
    for(i=1; i<=NF; i++)
      {
      if($i~/^definition$/)
        {
        col_class=i
        }
      }
    }
  if(FNR>=2)
    {
    class[$col_class]=$col_class
    }
  }
  END{
   for(i in class)
      {
      printf class[i]"\n" > "'$savepath'"
      }
    }' $data
######################################
data1="temp/filter_0_class.tsv"
data2="results/canopus_pp.tsv"
data3="results/fingerid_first_score.tsv"
until [[ "$similarity_limit" > 0 ]] && [[ "$similarity_limit" < 1 ]]
do
read -p "Please set the Tanimoto similarity threshold contribute to child-nebula. >>> " similarity_lim
done;
 until [[ "$definition_limit" > 0.01 ]] && [[ "$definition_limit" < 1 ]]
 do
 read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>> " def
 done;
class_pp_limit=$definition_limit
savepath="temp/idenfication_filter_$class_pp_limit.tsv"
awk -F $'\t' '
  {
  if(NR==FNR)
    {
    filter_class[$1]=$1
    }
  if(FILENAME~/canopus/)
    {
```

```
    if(FNR==1)
      {
      col_id=1
      for(i=2; i<=NF; i++)
        {
        for(j in filter_class)
          {
          if(j==$i)
            {
            col_class[j]=i
            print i
            }
          }
        }
      }
    if(FNR>=2)
      {
      for(i in col_class)
        {
        if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
          {
          class_set[i,$col_id]=i
          print "ID: ",$1,i,$col_class[i]
          }
        }
      }
    }
  if(FILENAME~/fingerid_first_score/)
    {
    if(FNR==1)
      {
      for(i=1; i<=NF; i++)
        {
        if($i~/^id/)
        {
        col_id=i
        }
      if($i~/similarity/)
        {
        col_similarity=i
        }
        }
      printf "class_nebula_facet\t"  $0"\n" > "'$savepath'"
      }
    if(FNR>=2)
      {
      if($col_similarity+0 >= "'$similarity_limit'"+0)
        {
       for(i in class_set)
         {
         if(i~"\034"$col_id"$")
           {
           printf class_set[i]"\t"  $0"\n" > "'$savepath'"
           }
         }
       }
      }
    }
  }' $data1 $data2 $data3
```

```
###################################
 data="temp/idenfication_filter_$class_pp_limit.tsv"
 savepath="temp/idenfication_filter2_${class_pp_limit}.tsv"
  until [[ "$num_limit_1" -gt 0 ]]
  do
    read -p "Please enter the features number threshold contribute to child-nebula (min number). >>> " r
  done;
  until [[ "$num_limit_2" -gt "$num_limit_1" ]]
  do
    read -p "Please enter the features number threshold contribute to child-nebula (max number). >>> " r
  done;
 awk -F $'\t' '
   {
   if(FNR==1)
     {
     printf $0"\n" > "'$savepath'"
     for(i=1; i<=NF; i++)
       {
       if($i~/class_nebula_facet/)
         {
         col_class=i
         }
       }
     }
 if(FNR>=2)
   {
   num[$col_class]+=1
   data[FNR]=$0
     class[FNR]=$col_class
   }
  }
   END{
     for(i in num)
       {
       if(num[i]+0 >= '$num_limit_1'+0 && num[i]+0 <= '$num_limit_2')
         {
         printf "The nodes number of the child-nebula is " num[i] ".\n"
         for(j in class)
           {
           if(class[j]==i)
             {
             printf data[j]"\n" >> "'$savepath'"
             }
           }
         }
       }
     }' $data
 ##################################
 mkdir results/network_facet_$class_pp_limit
 data1="temp/idenfication_filter2_${class_pp_limit}.tsv"
 data2="results/source_target_tree_$tlimit.tsv" # "results/source_target_tree_0.4.tsv"
 save_class="results/filter_child_class.tsv"
 savepath="results/network_facet_$class_pp_limit/"
 awk -F $'\t' '
   {
   if(NR==FNR)
     {
     if(FNR==1)
       {
       for(i=1; i<=NF; i++)
         {
         if($i~/class_nebula_facet/)
           {
           col_class=i
           }
         if($i~/^id$/)
           {
           col_id=i
           }
         }
       }
     if(FNR>=2)
       {
       class[$col_class]=$col_class
       class_id[$col_class,$col_id]=$col_id
       stat_id[$col_class,$col_id]=$col_id
       belong[$col_class,$col_id]=$col_class
       }
     }
   if(NR>FNR)
     {
     if(FNR==1)
       {
       for(i in class)
         {
         printf i"\n" > "'$save_class'"
         printf $0"\t"  "facet\n" > "'$savepath'" i ".tsv"
         }
       }
     if(FNR>=2)
       {
       for(i in class)
         {
         if( class_id[i,$1]==$1 && class_id[i,$2]==$2 )
           {
           printf $0"\t"  i"\n" >> "'$savepath'" i ".tsv"
           delete stat_id[i,$1]
           delete stat_id[i,$2]
           }
         }
       }
     }
   }
   END{
     for(i in stat_id)
```

```
      {
      ## source target similarity delta_mz fp fp class
      printf stat_id[i]"\t" stat_id[i]"\t"  "1\t"  "0\t"  "null\t"  "null\t"  belong[i]"\n" \
      \
      >> "'$savepath'" belong[i] ".tsv"
      }
    }' $data1 $data2
#####################
data1="canopus.tsv"
data2="results/filter_child_class.tsv"
data3="results/canopus_pp.tsv"
savepath="results/canopus_pp_filter.tsv"
awk -F $'\t' '
   {
   if(FILENAME~/canopus.tsv/)
       {
       if(FNR==1)
           {
           for(i=1; i<=NF; i++)
               {
               if($i~/absolute/)
                   {
                   col_index=i
                   }
               if($i~/^id/)
                   {
                   col_chemid=i
                   }
               if($i~/name/)
                   {
                   col_name=i
                   }
               if($i~/description/)
                   {
                   col_des=i
                   }
               }
           }
       if(FNR>2)
           {
           ab_index[$col_name]=$col_index
```

```
            chemid[$col_name]=$col_chemid
            des[$col_name]=$col_des
            }
        }
    if(FILENAME~/filter_child_class/)
        {
        class[$1]=$1
        if(FNR==1)
            {
            printf "index\t"  "chem_id\t"  "name\t"  "description\n" > "results/child_class.tsv"
            }
        printf ab_index[$1]"\t"  chemid[$1]"\t"  $1"\t"  des[$1]"\n" >> "results/child_class.tsv"
        }
    if(FILENAME~/canopus_pp/)
        {
        if(FNR==1)
            {
            printf $1 > "'$savepath'"
            for(i=2; i<=NF; i++)
                {
                if(class[$i]!="")
                    {
                    n+=1
                    printf "\tC"ab_index[$i] >> "'$savepath'"
                    col_set[n]=i
                    }
                }
            printf "\n" >> "'$savepath'"
            }
        if(FNR>=2)
            {
            printf $1 >> "'$savepath'"
            for(i=1; i<=n; i++)
                {
                printf "\t"$col_set[i] >> "'$savepath'"
                }
            printf "\n" >> "'$savepath'"
            }
        }
    }' $data1 $data2 $data3
;;
```

```
###################################
```

```
###################################
compound_idenfication)
echo "Run compound_idenfication."
echo "compound_idenfication "
exit
;;
```

```
###################################
```

```
###################################
double_ion_network)
exit;
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
exit)
echo "The mystery of creation is like the darkness of night--it is great.
Delusions of knowledge are like the fog of the morning."
exit;
;;
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
*)
echo "error"
exit;
;;
esac;
done;#(for)
done;#(select)
```

23 File: mcnebula1126.sh

```
##########################
```

```
##########################
echo "We are all in the gutter,
but some of us are looking at the stars.";
PS3='Please select the workflow to be executed. >>> '
select command in \
   \
  "default" \
  \
  "structure_extract" \
  \
  "classification_extract_sum" \
  \
  "classification_extract_filter" \
  \
  "fragment_tree_network" \
  \
  "fragment_tree_network_delta" \
  \
  "compound_idenfication" \
  \
  "double_ion_network" \
  \
  "exit"
do
 if [[ $command == "default" ]]
 then
 confirm=0
   until [[ $confirm == "yes" ]] || [[ $confirm == "no" ]]
   do
   read -p "Running all proccesses sequentially? [yes/no] >>> " confirm
   done;
   if [[ $confirm == "no" ]]
   then exit
   fi;
 default="structure_extract classification_extract_sum classification_extract_filter fragment_tree_netw
 list=$(echo $default);
 else list=$( echo $command)
 fi;
 for option in $(echo $list)
 do
   case $option in
```

```
###################################
```

```
###################################
```

```
###################################
```

```
###################################
structure_extract)
echo "Run structure_extract."
projectpath=0
 until [ -d $projectpath ] && [ -f $projectpath/canopus_summary.tsv ] && [ -f $projectpath/.format ]
 do
 read -p "Please input the path of the sirius project >>> " projectpath;
 done;
cd $projectpath;
mkdir results;
mkdir temp;
mkdir temp/fintemp;
 if [ -f temp/Mo_filename ]
 then rm temp/Mo_filename
 fi;
data="*_*/fingerid/*.tsv"
awk -F $'\t' '
  BEGIN{
    all_id=0
    \
    max_id=-1
    \
    p_id="null"
    \
    printf "Info: loading the data..."
    }
  {
  if(FNR==1)
    {
    if(NR>FNR)
      {
      close(pfile)
      }
    f=split(FILENAME,a,"[/]");
    n=split(a[1],b,"[_]");
    id=b[n];
    if(id!=p_id)
      {
      all_id+=1
      \
      if(id>max_id)
        {
        max_id=id
        }
      }
    file[id]=a[1]
    split(a[f], g, "[.]")
    formu_type[id]=g[1]
    if(all_id==1)
      {
      for(i=1; i<=NF; i++)
        {
        if(($i~/inchikey2D/))
          {
          col_2D=i
          }
        if($i=="inchi")
          {
          col_inchi=i
          }
        if(($i~/Formula/))
          {
          col_formu=i
          }
        if($i=="score")
          {
          col_score=i
          }
        if($i=="name")
          {
          col_name=i
          }
        if($i=="smiles")
          {
```

```
          col_smiles=i
          }
        if($i=="xlogp")
          {
          col_x=i
          }
        if(($i~/imilarity/))
          {
          col_simi=i
          }
        if($i~/links/)
          {
          col_links=i
          }
        }
      }
    pfile=FILENAME
    printf g[1]"\t"formu_type[id]"\n"
    printf "Info: the data of " id " (ID) has input. The filename is " FILENAME ".\n"
    }
  if(FNR>=2)
    {
    row_score=$col_score
    row_simi=$col_simi
    if(max["score",id]=="" || max["score",id]+0<row_score+0)
     {
      max["score",id]=row_score
      name["score",id]=$col_name
      formula["score",id]=$col_formu
      formu_type["score",id]=formu_type[id]
      simi["score",id]=$col_simi
      smiles["score",id]=$col_smiles
      inchi["score",id]=$col_inchi
      in2D["score",id]=$col_2D
      score["score",id]=$col_score
      xlogp["score",id]=$col_x
      links["score",id]=$col_links
      }
    if(max["simi",id]=="" || max["simi",id]+0<row_simi+0)
     {
      max["simi",id]=row_simi
      name["simi",id]=$col_name
      formula["simi",id]=$col_formu
      formu_type["simi",id]=formu_type[id]
      simi["simi",id]=$col_simi
      smiles["simi",id]=$col_smiles
      inchi["simi",id]=$col_inchi
      in2D["simi",id]=$col_2D
      score["simi",id]=$col_score
      xlogp["simi",id]=$col_x
      links["simi",id]=$col_links
      }
    }
  }
  END{
    printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t" \
    "score\t"  "xlogp\t"  "links\n" > "results/fingerid_sum.tsv"
    for(l in file)
      {
      if(max["score",l]!="")
        {
        printf l"\t" \
        \
        name["score",l]"\t"  formula["score",l]"\t"  simi["score",l]"\t" \
        \
        smiles["score",l]"\t"  inchi["score",l]"\t"  in2D["score",l]"\t" \
        \
        score["score",l]"\t"  xlogp["score",l]"\t"  links["score",l]"\n" >> "results/fingerid_sum.tsv"
        printf formula["score",l]"\t"  file[l]"\t"  formu_type["score",l]"\n" > "temp/Mo_filename"
        }
      if(max["simi",l]!="" && max["simi",l]!=max["score",l])
        {
        printf l"\t" \
        \
        name["simi",l]"\t"  formula["simi",l]"\t"  simi["simi",l]"\t" \
        \
        smiles["simi",l]"\t"  inchi["simi",l]"\t"  in2D["simi",l]"\t" \
        \
        score["simi",l]"\t"  xlogp["simi",l]"\t"  links["simi",l]"\n" >> "results/fingerid_sum.tsv"
        }
      }
    }' $data
```

```
data_sum="results/fingerid_sum.tsv"
awk -F $'\t' '
  {
  if(NR==1)
    {
    for(i=1; i<=NF; i++)
      {
      if($i~/id/)
        {
        col_id=i
        }
      if($i~/score/)
        {
        col_score=i
        }
      }
    }
  if(NR>=2)
    {
    id=$col_id
    \
    if(max_id+0<id || max_id=="")
      {
      max_id=id
      }
    if(score[id]+0<$col_score+0 || score[id]=="")
      {
      score[id]=$col_score
      \
      data[id]=$0
      }
    }
  }
  END{
    printf "id\t"  "name\t"  "formula\t"  "similarity\t"  "smiles\t"  "inchi\t"  "inchikey2D\t" \
    \
    "score\t"  "xlogp\t"  "links\n" > "results/fingerid_first_score.tsv"
    \
    for(i=1; i<=max_id; i++)
      {
      if(data[i]!="")
```

```
        {
        printf data[i] "\n" >> "results/fingerid_first_score.tsv"
        }
      }
    }' $data_sum
echo "structure_extract results have been successfully assembled into <results/fingerid_sum.tsv> and <re
;;
```

```
####################################
```

```
####################################
```

```
####################################
```

```
####################################
classification_extract_sum)
echo "Run classification_extract_sum.";
 if [ -f temp/Mo_filename ] && [ -f canopus.tsv ] && [ -f canopus_neg.tsv ] && [ -f .format ]
 then
 echo "Project path acknowledged."
 else
   until [ -d $projectpath ] && [ -f $projectpath/canopus.tsv ] && [ -f $projectpath/.format ]
   do
   read -p "Please input the path of the sirius project >>> " projectpath;
   done;
   cd $projectpath;
 fi;
data1="temp/Mo_filename"
data2="canopus.tsv"
data3="canopus_neg.tsv"
datas=$(awk -F $'\t' '
  {
  x=$2"/canopus/"$3".fpt"
  if(getline < x == 1)
    {
    printf x" "
    }
  close(x)
  }' $data1)
```

```
##################
awk -F $'\t' '
{
  if(NR==FNR)
    {
      i=split($2,s,"[_]")
      the_id[FNR]=s[i]
      n=FNR
    }
  if(FILENAME ~ /canopus.tsv/)
    {
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i ~ /^name/)
                {
                  col_class=i
                }
              if($i ~ /absolute/)
                {
                  col_abindex=i
                }
            }
        }
      if(FNR>=2)
        {
          abindex[1,FNR]=$col_abindex
          indexset[$col_abindex]=$col_class
        }
    }
  if(FILENAME ~ /canopus_neg.tsv/)
    {
      p_filename=FILENAME
      if(FNR==1)
        {
          for(i=1; i<=NF; i++)
            {
              if($i ~ /name/)
                {
                  col_class=i
                }
              if($i ~ /absolute/)
                {
                  col_abindex=i
                }
            }
        }
      if(FNR>=2)
        {
          abindex[2,FNR]=$col_abindex
          indexset[$col_abindex]=$col_class
        }
    }
  if(FILENAME ~ /.fpt/)
    {
      if(FNR==1)
        {
          close(p_filename)
          printf "Info: data_file name of " p_filename " has been input.\n"
          p_filename=FILENAME
          split(FILENAME,a,"[/]");
          m=split(a[1],b,"[_]");
          id=b[m];
          if(FILENAME ~ /\+.fpt/)
            {
              ion=1
            }
          if(FILENAME ~ /\-.fpt/)
            {
              ion=2
            }
        }
      pp[id,abindex[ion,FNR+1]]=sprintf("%.3f",$1)
    }
}
END{
printf "id" > "results/canopus_pp.tsv"
for(i in indexset)
  {
    ord+=1
    orderlist[ord]=i
    printf "\t"indexset[i] >> "results/canopus_pp.tsv"
  }
printf "\n" >> "results/canopus_pp.tsv"
for(i=1; i<=n; i++)
```

```
  {
    printf the_id[i] >> "results/canopus_pp.tsv"
    for(j=1; j<=ord; j++)
      {
        if(pp[the_id[i],orderlist[j]]=="")
          {
            pp[the_id[i],orderlist[j]]=0
          }
        printf "\t"pp[the_id[i],orderlist[j]] >> "results/canopus_pp.tsv"
      }
    printf "\n" >> "results/canopus_pp.tsv"
  }
}' $data1 $data2 $data3 $datas
echo "classiication_extract_sum have been successfully written into <results/canopus_pp.tsv>"
;;
```

```
####################################
```

```
####################################
```

```
####################################
classification_extract_filter)
echo "Run classification_extract_filter.";
 if [ -f results/canopus_pp.tsv ]
 then
 echo "Project path acknowledged."
 else
   until [ -d $projectpath ] && [ -f $projectpath/results/canopus_pp.tsv ] && [ -f $projectpath/results
   do
   read -p "Please input the path of the sirius project >>> " projectpath;
   done;
   cd $projectpath;
 fi;
check=0
 until [[ "$check" == "yes" ]] || [[ "$check" == "no" ]]
 do
 read -p "Collate the posterior probabilities of the classification of each feature? [yes/no] >>> " che
 done;
 if [[ $check == "yes" ]]
 then
 definition_limit=0
   until [[ "$definition_limit" > "0.01" ]] && [[ "$definition_limit" < "1" ]]
   do
```

```
  read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>> " de
  done;
data1="canopus_summary.tsv"
 data2="canopus.tsv"
 data3="results/canopus_pp.tsv"
awk -F $'\t' '
  {
  if(NR==FNR)
    {
    if(FNR==1)
      {
      p_file=FILENAME
      \
      for(i=1; i<=NF; i++)
        {
        if($i~/name/)
          {
          col_id=i
          }
        if($i~/specific/)
          {
          col_specific=i
          }
        if($i~/level/)
          {
          col_level=i
          }
        if($i~/subclass/)
          {
          col_subclass=i
          }
        if($i~/^class/)
          {
          col_class=i
          }
        if($i~/superclass/)
          {
          col_superclass=i
          }
        }
      }
    if(FNR>=2)
      {
      n=split($col_id,a,"[_]")
      \
      id=a[n]
      \
      specific[id]=$col_specific
      \
      level[id]=$col_level
      \
      subclass[id]=$col_subclass
      \
      class[id]=$col_class
      \
      superclass[id]=$col_superclass
      }
    }
  if(FILENAME~/canopus.tsv/)
    {
    if(FNR==1)
      {
      close(p_file)
      \
      p_file=FILENAME
      \
      for(i=1; i<=NF; i++)
        {
        if($i~/name/)
          {
          col_name=i
          }
        if($i~/description/)
          {
          col_description=i
          }
        }
      }
    if(FNR>=2)
      {
      description[$col_name]=$col_description
      }
    }
  if(FILENAME~"'$data3'")
    {
    if(FNR==1)
      {
      close(p_file)
      \
      for(i=1; i<=NF; i++)
        {
        if($i~/^id$/)
          {
          col_id=i
          }
        if(i>=2)
          {
          col_class_name[i]=$i
          }
        }
      printf "id\t" "definition_source\t" "definition\t" "definition_pp\t" "definition_description
      \
      "specific\t" "specific_pp\t" "level_5\t" "level_5_pp\t" \
      \
      "subclass\t" "subclass_pp\t" "class\t" "class_pp\t" "superclass\t" "superclass_pp\n" \
      \
      > "results/stat_classification.tsv"
      }
    if(FNR>=2)
      {
      for(i=2; i<=NF; i++)
        {
        c_pp[col_class_name[i]]=sprintf("%.4f",$i)
        }
      if(level[$col_id] != "" && c_pp[level[$col_id]] >= "'$definition_limit'")
        {
        definition_source="level_5"
        \
        definition=level[$col_id]
        }
      else if(subclass[$col_id] != "" && c_pp[subclass[$col_id]] >= "'$definition_limit'")
        {
        definition_source="subclass"
```

```
         \
         definition=subclass[$col_id]
         }
       else if(class[$col_id] != "" && c_pp[class[$col_id]] >= "'$definition_limit'")
         {
         definition_source="class"
         \
         definition=class[$col_id]
         }
       else if(superclass[$col_id] != "" && c_pp[superclass[$col_id]] >= "'$definition_limit'")
         {
         definition_source="superclass"
         \
         definition=superclass[$col_id]
         }
       else
         {
         definition_source="null"
         \
         definition="null"
         \
         c_pp[definition]="null"
         \
         description[definition]="null"
         }
       printf $col_id"\t"  definition_source"\t"  definition"\t"  c_pp[definition]"\t"  description[defi
       \
       specific[$col_id]"\t"  c_pp[specific[$col_id]]"\t"  level[$col_id]"\t"  c_pp[level[$col_id]]"\t"
       \
       subclass[$col_id]"\t"  c_pp[subclass[$col_id]]"\t"  class[$col_id]"\t"  c_pp[class[$col_id]]"\t"
       \
       superclass[$col_id]"\t"  c_pp[superclass[$col_id]]"\n" >> "results/stat_classification.tsv"
       }
     }
   }' $data1 $data2 $data3
  echo "classification_extract_filter have been successfully written into <results/stat_classification.t
  fi;
;;
```

```
###################################
```

```
####################################
```

```
####################################
```

```
####################################
fragment_tree_network)
echo "Run fragment_tree_network.";
 if [ -d temp ] && [ -f ftalign.tsv ] && [ -f .format ]
 then
 echo "Project path acknowledged."
 else
   until [ -d $projectpath ] && [ -f $projectpath/ftalign.tsv ] && [ -f $projectpath/.format ]
   do
   read -p "Please input the path of the sirius project. Make sure you have moved the fragment tree ali
   done;
   cd $projectpath;
 fi;
echo "Please enter the minimum alignment similarity(tlimit) that you want to filter."
tlimit=0
 until [[ "$tlimit" > "0.01" ]] && [[ "$tlimit" < "0.99" ]]
 do
 read -p "0.4-0.7 is recommended >>> " tlimit;
 done;
  if ! [ -d temp/ftaligntemp ]
 then
 mkdir temp/ftaligntemp
  fi;
data="ftalign.tsv"
savepath="temp/ftaligntemp/tmp"
awk -F $'\t' -v OFS=$'\t' '
  {
  printf "Info: NR = " NR ". FNR = " FNR ".\n"
  if(NR==FNR)
    {
   for(i=1; i<=NF; i++)
     {
     if(NR==1 && i!=1 || NR!=1 && i==1)
       {
       n=split($i,x,"[_]")
       \
       raw[NR,i]=x[n]
       }
     else
       {
       if(NR==i)
         {
         raw_norm[NR,i]=$i
         }
       }
     }
   }
 if(NR>FNR && FNR>=2)
   {
   for(i=2; i<=NF; i++)
     {
     norm1=$i/raw_norm[i,i]
     norm2=$i/raw_norm[FNR,FNR]
     norms=sprintf("%.2f", ((norm1+norm2)/2))
     if((norms+0 > '$tlimit'+0))
       {
       if(raw[FNR,1]+0>=raw[1,i]+0)
         {
         print raw[FNR,1], raw[1,i], norms > "'$savepath'"
         }
       if(raw[FNR,1]+0<raw[1,i]+0)
         {
         print raw[1,i], raw[FNR,1], norms > "'$savepath'"
         }
       }
     }
   }
  }' $data $data
sort -u $savepath > temp/ftaligntemp/tmp2
savepath="temp/ftaligntemp/filter_net_$tlimit"
awk -F $'\t' '
{
  if(NR==FNR)
    {
      if($1!=$2)
        {
          replink[$1]=$1
          replink[$2]=$2
        }
    }
  if(NR!=FNR)
    {
      if($1!=$2)
        {
          printf $0"\n" > "'$savepath'"
        }
      if($1==$2)
        {
          if(replink[$1]=="")
            {
              printf $0"\n" > "'$savepath'"
            }
        }
    }
}' temp/ftaligntemp/tmp2 temp/ftaligntemp/tmp2
echo "fragment_tree_network have been successfully written into <temp/ftaligntemp/filter_net_$tlimit>"
;;
```

```
####################################
```

```
####################################
```

```
####################################
```

```
####################################
fragment_tree_network_delta)
 if [ -d temp ] && [ -f ftalign.tsv ] && [ -f .format ]
 then
 echo "Project path acknowledged."
 else
   until [ -d $projectpath ] && [ -f $projectpath/ftalign.tsv ] && [ -f $projectpath/.format ]
   do
   read -p "Please input the path of the sirius project >>> " projectpath;
   done;
 cd $projectpath;
 fi;
echo "The following module attempts to compute the differential fingerprints of connected clusters based
```

```
###################
```

```
###################
```

```
###################
plimit=0
```

```
 until [[ "$plimit" > "0.5" ]] && [[ "$plimit" < "0.99" ]]
 do
 read -p "If the minimum alignment similarity is greater than 0.3, this step will automatic calculate tl
 done;
plimit2=0
 until [[ "$plimit2" > "0.1" ]] && [[ "$plimit2" < "$plimit" ]]
 do
 read -p "The minimum posterior probability of the molecular fingerprint(plimit_2) to be controled. 0.1
 done;
tlimit=$(ls temp/ftaligntemp/filter_net_* | awk -F _ '{print $NF}')
check_rep=$(awk 'END{print NR}' <(echo "$tlimit"))
 if [[ "$check_rep" > "1" ]]
 then
 tlimit=0
   until [ -f temp/ftaligntemp/filter_net_$tlimit ]
   do
   read -p "Plural and different fragment_tree_network files were found to exist locally. Please select
   done;
 fi;
 if [ -f temp/ftaligntemp/refilter_net_$tlimit ]
 then rm temp/ftaligntemp/refilter_net_$tlimit
 fi;
 if [ -f temp/ftaligntemp/fpsample ]
 then rm temp/ftaligntemp/fpsample
 fi;
echo "Running delta_fingerprint computation..."
echo "Aquiring data from sirius index..."
data1="*_*/compound.info"
data2="temp/ftaligntemp/filter_net_$tlimit"
echo "Run fragment_tree_network_delta."
savepath="temp/ms_data_$tlimit"
echo "step 1: ms data"
  if ! [ -f $savepath ]
  then
 awk -F "[\t]||[:]||[_]" -v OFS=$'\t' '
   BEGIN{
     printf "..."
     printf "id\t"  "m/z\t"  "rt\n" > "results/mz_and_rt.tsv"
     }
   {
   if(FILENAME~/compound.info/)
```

```
    {
    if(FNR==1)
      {
      printf "Info: catch >>> "FILENAME"\n"
      }
    if($1=="name")
      {
      i+=1;
      id[i]=$NF;
      n=split($NF,a,"[_]")
      printf a[n]"\t" >> "results/mz_and_rt.tsv"
      }
    if($1=="ionMass")
      {
      mz[id[i]]=$NF
      printf sprintf("%.4f",$NF)"\t" >> "results/mz_and_rt.tsv"
      }
    if($1=="ionType")
      {
      type[id[i]]=$NF
      }
    if($1=="rt")
      {
      rt[id[i]]=$2;
      printf sprintf("%.2f",$2/60)"\n" >> "results/mz_and_rt.tsv"
      }
    }
  if(FILENAME~/ftaligntemp/)
    {
    #source  target  ftalign  delta_mz  delta_rt  source_iontype  target_iontype;
    if(FNR==1)
      {
      printf "" > "'$savepath'"
      }
    print $1,$2,$3,sprintf("%.3f",mz[$2]-mz[$1]),sprintf("%.2f",rt[$2]-rt[$1]),type[$1],type[$2] \
    \
    >> "'$savepath'"
    }
  }' $data1 $data2
fi;
```

```
########################
echo "step 2: data path"
source_file="temp/Mo_filename"
data="temp/ms_data_$tlimit"
savepath="temp/datapath_$tlimit"
  if ! [ -f $savepath ]
  then
 awk -F $'\t' -v OFS=$'\t' '
   {
   if(NR==FNR)
     {
     n=split($2, a, "[_]")
     file[a[n]]=$2
     formu_type[a[n]]=$3
     }
   if(NR!=FNR)
     {
     #<path>sourceFormula  <path>targetFormula
     path1=file[$1]"/fingerprints/"formu_type[$1]".fpt"
     path2=file[$2]"/fingerprints/"formu_type[$2]".fpt"
     data[path1]=path1
     data[path2]=path2
     }
   }
   END{
     for(i in data)
       {
       n+=1
       print "Check file:",n
       if(getline<i==-1)
         {
         printf "Escape filename: " i "\n"
         }
       else
         {
         printf i" " > "'$savepath'"
         }
       close(i)
       }
     print "Sum:",n
     }' $source_file $data
   fi;
datapath=$(cat temp/datapath_$tlimit)
echo "step 3: fingerprint"
data_allfp="temp/data_allfp_$tlimit"
  if ! [ -f $data_allfp ]
  then
 awk -F $'\t' '
   BEGIN{
     n=0
     }
   {
   if(FNR==1)
     {
     if(n>1)
       {
       close(file)
       }
     file=FILENAME;
     print "Get fingerprints: ",FILENAME
     n+=1;
     printf FILENAME"\n"$0"\n" > "'$data_allfp'"
     }
   else
     {
     print $0 > "'$data_allfp'"
     }
   }' $datapath
  fi;
```

```
###########################
awk -F $'\t' -v OFS=$'\t' '
  {
  if(NR==1)
    {
    for(i=1; i<=NF; i++)
      {
      if($i~/absolute/)
        {
        col_index=i
        }
      if($i~/description/)
        {
```

```
        col_description=i
        }
      }
    }
  if(NR>=2)
    {
    print $col_index,$col_description
    }
  }' csi_fingerid.tsv > temp/ftaligntemp/pos_fingerprint_index;
awk -F $'\t' -v OFS=$'\t' '
  {
  if(NR==1)
    {
    for(i=1; i<=NF; i++)
      {
      if($i~/absolute/)
        {
        col_index=i
        }
      if($i~/description/)
        {
        col_description=i
        }
      }
    }
  if(NR>=2)
    {
    print $col_index,$col_description
    }
  }' csi_fingerid_neg.tsv > temp/ftaligntemp/neg_fingerprint_index;
posindex="temp/ftaligntemp/pos_fingerprint_index"
negindex="temp/ftaligntemp/neg_fingerprint_index"
data="temp/ms_data_$tlimit"
echo "step 4: merge data"
awk -F $'\t' '
  BEGIN{
    file=0
    x=0
    count=0
    f=0
    posnum=0
  negnum=0
  }
{
if(FNR==1)
  {
  file+=1
  }
if(NR==FNR)
  {
  if(($1~/fingerprint/))
    {
    if(x+0>f+0)
      {
      f=x  # calculate the max index.
      }
    count+=1;  # calculate the all fingerprints file number.
    \
    split($1,a,"[/]"); e=split(a[1],b,"[_]"); id=b[e]; #catch the id.
    \
    x=0;
    }
  else
    {
    x+=1;
    \
    fp[id,x]=$1;
    }
  }
if(file==2)
  {
  pos[FNR]=$1
  posnum+=1
  }
if(file==3)
  {
  neg[FNR]=$1
  negnum+=1
  }
if(file==4)
  {
  if("'$tlimit'"+0 >= 0.3)
```

```
     {
    if(fp[$1,1]!="" && fp[$2,1]!="")
      {
      n1=split($6, g, "[]]");
      n2=split($7, h, "[]]");
      if(g[n1]=="+" && h[n2]=="+")
        {
        for(x=1; x<=posnum; x++)
          {
          if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
            {
            data_s[FNR,x]=pos[x]
            }
          else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
            {
            data_t[FNR,x]=pos[x]
            }
          }
        }
      if(g[n1]=="-" && h[n2]=="-")
        {
        for(x=1; x<=negnum; x++)
          {
          if(fp[$1,x]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
            {
            data_s[FNR,x]=neg[x]
            }
          else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
            {
            data_t[FNR,x]=neg[x]
            }
          }
        }
      if(g[n1]=="-" && h[n2]=="+" || h[n2]=="-" && g[n1]=="+")
        {
        for(i=1; i<=posnum; i++)
          {
          for(j=1; j<=negnum; j++)
            {
            if(pos[i]==neg[j])
              {
```

```
            mirror[i]=j  #if pos=i, the identical index of neg is mirror[i]
            }
          }
        }
      }
    if(g[n1]=="-" && h[n2]=="+")
      {
      for(x=1; x<=f; x++)
        {
        if(fp[$1,mirror[x]]+0>='$plimit'+0 && fp[$2,x]+0<='$plimit2'+0)
          {
          data_s[FNR,x]=neg[mirror[x]]
          }
        else if(fp[$2,x]+0>='$plimit'+0 && fp[$1,mirror[x]]+0<='$plimit2'+0)
          {
          data_t[FNR,x]=neg[mirror[x]]
          }
        }
      }
    if(h[n2]=="-" && g[n1]=="+")
      {
      for(x=1; x<=f; x++)
        {
        if(fp[$1,x]+0>= '$plimit'+0 && fp[$2,mirror[x]]+0<='$plimit2'+0)
          {
          data_s[FNR,x]=neg[mirror[x]]
          }
        else if(fp[$2,mirror[x]]+0>='$plimit'+0 && fp[$1,x]+0<='$plimit2'+0)
          {
          data_t[FNR,x]=neg[mirror[x]]
          }
        }
      }
    }
  }
 if(FNR==1)
   {
   printf "source\t"  "target\t"  "ftalign_similarity\t"  "delta_m/z\t"  "source_fp_uniq\t"  "target_
   }
 else
   {
      printf $1"\t"  $2"\t"  $3"\t"  $4"\t";
      \
      if('$tlimit'+0 >= 0.3)
        {
       if(fp[$1,1]=="" || fp[$2,1]=="")
         {
         printf "NA@NA\n"
         }
       else
         {
         printf "source:"  #the source fingerprint uniq.
         \
         for(x=1; x<=f; x++)
           {
           if(data_s[FNR,x]!="")
             {
             printf data_s[FNR,x]","
             }
           };
         printf "@target:"  #the target fingerprint uniq.
         \
         for(x=1; x<=f; x++)
           {
           if(data_t[FNR,x]!="")
             {
             printf data_t[FNR,x]","
             }
           }
         printf "\n"
         }
       }
     else
       {
       printf "NA@NA\n"
       }
      }
    }
  }' $data_allfp $posindex $negindex $data | sed -e 's/,@/\t/g; s/,$//g; s/@/\t/g' \
  \
  > results/source_target_tree_$tlimit.tsv;
echo "All instances have written into <results/source_target_tree_$tlimit.tsv>."
```

```
#####################################
echo "step 5: separate child-nebula from parent-nebula."
data="results/stat_classification.tsv"
savepath="temp/filter_0_class.tsv"
awk -F $'\t' '
  {
  if(FNR==1)
    {
    for(i=1; i<=NF; i++)
      {
      if($i~/^definition$/)
        {
        col_class=i
        }
      }
    }
  if(FNR>=2)
    {
    class[$col_class]=$col_class
    }
  }
  END{
   for(i in class)
      {
      printf class[i]"\n" > "'$savepath'"
      }
    }' $data
 #####################################
 data1="temp/filter_0_class.tsv"
 data2="results/canopus_pp.tsv"
 data3="results/fingerid_first_score.tsv"
 until [[ "$similarity_limit" > 0 ]] && [[ "$similarity_limit" < 1 ]]
 do
 read -p "Please set the Tanimoto similarity threshold contribute to child-nebula. >>> " similarity_lim
 done;
  until [[ "$definition_limit" > 0.01 ]] && [[ "$definition_limit" < 1 ]]
  do
  read -p "Please enter the classes posterior probabilities limition. 0.5~0.99 may work well. >>> " def
  done;
 class_pp_limit=$definition_limit
 savepath="temp/idenfication_filter_$class_pp_limit.tsv"
```

```
awk -F $'\t' '
  {
  if(NR==FNR)
    {
    filter_class[$1]=$1
    }
  if(FILENAME~/canopus/)
    {
    if(FNR==1)
      {
      col_id=1
      for(i=2; i<=NF; i++)
        {
        for(j in filter_class)
          {
          if(j==$i)
            {
            col_class[j]=i
            print i
            }
          }
        }
      }
    if(FNR>=2)
      {
      for(i in col_class)
        {
        if(sprintf("%.3f",$col_class[i])+0>="'$class_pp_limit'"+0)
          {
          class_set[i,$col_id]=i
          print "ID: ",$1,i,$col_class[i]
          }
        }
      }
    }
  if(FILENAME~/fingerid_first_score/)
    {
    if(FNR==1)
      {
      for(i=1; i<=NF; i++)
        {
        if($i~/^id/)
        {
        col_id=i
        }
      if($i~/similarity/)
        {
        col_similarity=i
        }
        }
      printf "class_nebula_facet\t"  $0"\n" > "'$savepath'"
      }
    if(FNR>=2)
      {
      if($col_similarity+0 >= "'$similarity_limit'"+0)
        {
       for(i in class_set)
         {
         if(i~"\034"$col_id"$")
           {
           printf class_set[i]"\t"  $0"\n" > "'$savepath'"
           }
         }
       }
      }
    }
  }' $data1 $data2 $data3
```

```
###################################
 data="temp/idenfication_filter_$class_pp_limit.tsv"
 savepath="temp/idenfication_filter2_${class_pp_limit}.tsv"
  until [[ "$num_limit_1" -gt 0 ]]
  do
    read -p "Please enter the features number threshold contribute to child-nebula (min number). >>> " r
  done;
  until [[ "$num_limit_2" -gt "$num_limit_1" ]]
  do
    read -p "Please enter the features number threshold contribute to child-nebula (max number). >>> " r
  done;
 awk -F $'\t' '
   {
   if(FNR==1)
     {
```

```
    printf $0"\n" > "'$savepath'"
    for(i=1; i<=NF; i++)
      {
      if($i~/class_nebula_facet/)
        {
        col_class=i
        }
      }
    }
if(FNR>=2)
  {
  num[$col_class]+=1
  data[FNR]=$0
    class[FNR]=$col_class
  }
 }
  END{
    for(i in num)
      {
      if(num[i]+0 >= '$num_limit_1'+0 && num[i]+0 <= '$num_limit_2')
        {
        printf "The nodes number of the child-nebula is " num[i] ".\n"
        for(j in class)
          {
          if(class[j]==i)
            {
            printf data[j]"\n" >> "'$savepath'"
            }
          }
        }
      }
    }' $data
##################################
mkdir results/network_facet_$class_pp_limit
data1="temp/idenfication_filter2_${class_pp_limit}.tsv"
data2="results/source_target_tree_$tlimit.tsv" # "results/source_target_tree_0.4.tsv"
save_class="results/filter_child_class.tsv"
savepath="results/network_facet_$class_pp_limit/"
awk -F $'\t' '
  {
  if(NR==FNR)
```

```
    {
    if(FNR==1)
      {
      for(i=1; i<=NF; i++)
        {
        if($i~/class_nebula_facet/)
          {
          col_class=i
          }
        if($i~/^id$/)
          {
          col_id=i
          }
        }
      }
    if(FNR>=2)
      {
      class[$col_class]=$col_class
      class_id[$col_class,$col_id]=$col_id
      stat_id[$col_class,$col_id]=$col_id
      belong[$col_class,$col_id]=$col_class
      }
    }
  if(NR>FNR)
    {
    if(FNR==1)
      {
      for(i in class)
        {
        printf i"\n" > "'$save_class'"
        printf $0"\t"  "facet\n" > "'$savepath'" i ".tsv"
        }
      }
    if(FNR>=2)
      {
      for(i in class)
        {
        if( class_id[i,$1]==$1 && class_id[i,$2]==$2 )
          {
          printf $0"\t"  i"\n" >> "'$savepath'" i ".tsv"
          delete stat_id[i,$1]
          delete stat_id[i,$2]
          }
        }
      }
    }
  }
  END{
    for(i in stat_id)
      {
      ## source target similarity delta_mz fp fp class
      printf stat_id[i]"\t" stat_id[i]"\t"  "1\t"  "0\t"  "null\t"  "null\t"  belong[i]"\n" \
      \
      >> "'$savepath'" belong[i] ".tsv"
      }
    }' $data1 $data2
#####################
data1="canopus.tsv"
data2="results/filter_child_class.tsv"
data3="results/canopus_pp.tsv"
savepath="results/canopus_pp_filter.tsv"
awk -F $'\t' '
   {
   if(FILENAME~/canopus.tsv/)
       {
       if(FNR==1)
           {
           for(i=1; i<=NF; i++)
               {
               if($i~/absolute/)
                   {
                   col_index=i
                   }
               if($i~/^id/)
                   {
                   col_chemid=i
                   }
               if($i~/name/)
                   {
                   col_name=i
                   }
               if($i~/description/)
```

```
                    {
                    col_des=i
                    }
                }
            }
        if(FNR>2)
            {
            ab_index[$col_name]=$col_index
            chemid[$col_name]=$col_chemid
            des[$col_name]=$col_des
            }
        }
    if(FILENAME~/filter_child_class/)
        {
        class[$1]=$1
        if(FNR==1)
            {
            printf "index\t"  "chem_id\t"  "name\t"  "description\n" > "results/child_class.tsv"
            }
        printf ab_index[$1]"\t"  chemid[$1]"\t"  $1"\t"  des[$1]"\n" >> "results/child_class.tsv"
        }
    if(FILENAME~/canopus_pp/)
        {
        if(FNR==1)
            {
            printf $1 > "'$savepath'"
            for(i=2; i<=NF; i++)
                {
                if(class[$i]!="")
                    {
                    n+=1
                    printf "\tC"ab_index[$i] >> "'$savepath'"
                    col_set[n]=i
                    }
                }
            printf "\n" >> "'$savepath'"
            }
        if(FNR>=2)
            {
            printf $1 >> "'$savepath'"
            for(i=1; i<=n; i++)
```

```
                {
                printf "\t"$col_set[i] >> "'$savepath'"
                }
            printf "\n" >> "'$savepath'"
            }
        }
    }' $data1 $data2 $data3
;;
```

```
####################################
```

```
####################################
compound_idenfication)
echo "Run compound_idenfication."
echo "compound_idenfication "
exit
;;
```

```
####################################
```

```
####################################
double_ion_network)
exit;
;;
```

```
####################################
```

```
####################################
```

```
####################################
```

```
####################################
exit)
echo "The mystery of creation is like the darkness of night--it is great.
Delusions of knowledge are like the fog of the morning."
exit;
;;
```

```
####################################
```

```
####################################
```

```
####################################
```

```
####################################
*)
echo "error"
exit;
```