

COSMOS tutorial

Aurelien Dugourd

4/14/2020

Copyright (C) 2019 Aurelien Dugourd Contact : aurelien.dugourd@bioquant.uni-heidelberg.de

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Overview

This script is a tutorial to run the COSMOS pipeline. You will need to manually set the working directory to the root COSMOS folder. Then, you may run the script with the provided data or your own if you format it as the same way as the data presented here.

```
library(readr)
library(stringr)
library('org.Hs.eg.db')
library(CARNIVAL) # load CARNIVAL library
library(biomaRt)
library(gsubfn)
library(rpubchem)

setwd("~/Dropbox/COSMOS/")

meta_network_carnival_ready_exch_solved <- read_delim("support/meta_network_carnival_ready_exch_solved.
\t", escape_double = FALSE, trim_ws = TRUE)
```

Preparation of metabolomic inputs

The metabolites in the prior knowledge network are identified as XMetab_*PUBCHEMID***compartment**, for example “XMetab_*1983r*”. The compartment code is the BIGG model standard (r, c, e, x, m, l, n, g). Thus we will first need to map whatever identifier for metabolite the data has to the one of the network.

First, we import the metabolic differential analysis dataframe, the mapping from metabolite name to kegg, the mapping from kegg to pubchem and the PKN.

It is up to the users to find the appropriate mapping tables to map their identifiers to pubchem ones. an example of a useful resource for that is : “<http://csbg.cnb.csic.es/mbrole/conversion.jsp>”

```
ttop_tumour_vs_healthy <- as.data.frame(
  read_csv("results/metabolomic/ttop_tumour_vs_healthy.csv"))
metab_to_kegg <- as.data.frame(
  read_csv("support/metab_to_kegg.txt"))
```

```
kegg_to_pubchem <- as.data.frame(
  read_csv("support/kegg_to_pubchem.txt",
    col_names = FALSE))
meta_network_with_X <- as.data.frame(
  read_delim("support/meta_network_carnival_ready_exch_solved.tsv",
    "\t", escape_double = FALSE, trim_ws = TRUE))
```

Now we will format the pubchem IDs of the kegg to puchem table by adding the appriariate prefixes and suffixes so that they can be mapped on the PKN.

```
kegg_to_pubchem$X2 <- paste("XMetab__",kegg_to_pubchem$X2, sep = "")

compartment_codes <- unique(c(meta_network_with_X$source,meta_network_with_X$target))
compartment_codes <- compartment_codes[grepl("Metab",compartment_codes)]
compartment_codes <- unique(str_match(compartment_codes,"___-____"))

compartment_mapping <- list()
for(i in 1:length(compartment_codes))
{
  df <- kegg_to_pubchem
  df$X2 <- paste(df$X2, compartment_codes[i], sep = "")
  compartment_mapping[[i]] <- df
}

compartment_mapping <- as.data.frame(do.call(rbind, compartment_mapping))

compartment_mapping <- compartment_mapping[
  compartment_mapping$X2 %in% meta_network_with_X$source |
  compartment_mapping$X2 %in% meta_network_with_X$target,
]

kegg_to_pubchem_with_comp <- compartment_mapping
names(kegg_to_pubchem_with_comp) <- c("KEGG","pubchem")
```

Now in this instance, we will combine the mapping table to have the connections from original metabolite name used here to the matching identifiers of the PKN. A consequence of this is that a same metabolites can be mapped at multiple places in the PKN.

```
full_mapping <- merge(metab_to_kegg, kegg_to_pubchem_with_comp, by = "KEGG")

names(ttop_tumour_vs_healthy)[1] <- "metab_name"

ttop_tumour_vs_healthy <- merge(ttop_tumour_vs_healthy, full_mapping, by = "metab_name")
ttop_tumour_vs_healthy <- ttop_tumour_vs_healthy[,c(9,2:7)]
```

Preparation of signaling (kinase/phosphatases and TFs)

Next we need to format the kinases/phosphatases and TF activities into a the proper format for CARNIVAL. First we map the gene symbole identifiers to corresponding entrez IDs and we add an X at the beginning of each of them so that there are not confused for numbers. Then, we combine both kinases/phosphahatases and TFs activities dataframes.

```
##KINASE/PHOSPHATASES
kinase_activities <- as.data.frame(
```

```

read_csv("results/phospho/kinase_activities.csv"))

symbols <- kinase_activities$X1

# use mapIds method to obtain Entrez IDs
mapping_symbol_to_entrez <- mapIds(org.Hs.eg.db, symbols, 'ENTREZID', 'SYMBOL')
for(i in 1:length(kinase_activities[,1]))
{
  kinase_activities[i,1] <- mapping_symbol_to_entrez[kinase_activities[i,1]]
}

kinase_activities <- kinase_activities[complete.cases(kinase_activities),]
kinase_activities$X1 <- paste("X", kinase_activities$X1, sep = "")

##TF
TF_scores <- as.data.frame(
  read_csv("results/transcriptomic/TF_scores.csv"))

symbols <- TF_scores$ID

# use mapIds method to obtain Entrez IDs
mapping_symbol_to_entrez <- mapIds(org.Hs.eg.db, symbols, 'ENTREZID', 'SYMBOL')
for(i in 1:length(TF_scores[,1]))
{
  TF_scores[i,1] <- mapping_symbol_to_entrez[TF_scores[i,1]]
}

TF_scores <- TF_scores[complete.cases(TF_scores),]
TF_scores$ID <- paste("X", TF_scores$ID, sep = "")

##Combine TF and kinase
names(TF_scores) <- c("X1", "NES")
kinase_activities <- as.data.frame(rbind(kinase_activities, TF_scores))

```

Now we need to convert the dataframe of activities and metabolomic abundance changes into a single row dataframe with column names as genes and metabolite identifiers. We also keep only significant activities and abundance changes as input (threshold at users discretion, here we use 0.05 p-value for metabolites and $\text{abs}(\text{NES}) > 1.7$ for TFs, kinases and phosphatases). We also convert the continuous values into 1 and -1.

Input formatting

```

metab_input_carnival <- ttop_tumour_vs_healthy[ttop_tumour_vs_healthy$P.Value < 0.05, c(1,4)]
metab_input_carnival <- metab_input_carnival[metab_input_carnival$pubchem %in% meta_network_with_X$sour
metabs <- metab_input_carnival$pubchem
metab_input_carnival <- as.data.frame(sign(t(metab_input_carnival[,2])))
names(metab_input_carnival) <- metabs

signaling_input_carnival <- kinase_activities[abs(kinase_activities$NES) > 1.7,]
signaling_input_carnival <- signaling_input_carnival[signaling_input_carnival$X1 %in% meta_network_with
signaling_enzymes <- signaling_input_carnival$X1
signaling_input_carnival <- as.data.frame(sign(t(signaling_input_carnival[,2])))
names(signaling_input_carnival) <- signaling_enzymes

write_csv(metab_input_carnival, "results/metabolomic/metab_input_carnival.csv")

```

```
write_tsv(metab_input_carnival, "results/metabolomic/metab_input_carnival.tsv")

write_csv(signaling_input_carnival, "results/phospho/signaling_input_carnival.csv")
write_tsv(signaling_input_carnival, "results/phospho/signaling_input_carnival.tsv")
```

Finally, we do some cleaning on the PKN, removing characters that are not compatible with carnival as well as duplicated interations.

```
meta_network_with_X$source <- gsub("[+{};,() ]", "_____", meta_network_with_X$source)
meta_network_with_X$target <- gsub("[+{};,() ]", "_____", meta_network_with_X$target)

isdup <- rep(FALSE, length(meta_network_with_X[,1]))
for(i in 1:length(meta_network_with_X[,1]))
{
  isdup[i] <- meta_network_with_X[i,1] == meta_network_with_X[i,3]
}
meta_network_with_X <- meta_network_with_X[!isdup,]
meta_network_with_X <- unique(meta_network_with_X)

write_tsv(meta_network_with_X, "support/meta_network_with_X_nobadchar.tsv")
```

Once the inputs are ready, we can run CARNIVAL. Here we already ran it a priori and saved the output as it can take a significant amount of time. You should uncomment this chunk of code in order to run it with your data. For more information on the CARNIVAL parameters, please check the carnival page :<https://saezlab.github.io/CARNIVAL/>. Briefly, the time limit can be adjusted if the gap is still too high at the end of the run, and you should make sure that the gap is smaller than the mipGAP parameter at the end of the run.

```
# CARNIVAL_Result <- runCARNIVAL(CplexPath=~"/Documents/cplex",
#                               Result_dir="results/multi_omic/carnival/",
#                               CARNIVAL_example=NULL,
#                               UP2GS=F,
#                               netFile = "support/meta_network_with_X_nobadchar.tsv",
#                               measFile = "results/metabolomic/metab_input_carnival.tsv",
#                               inputFile = "results/phospho/signaling_input_carnival.tsv",
#                               weightFile = NULL,
#                               timelimit = 7200,
#                               mipGAP = 0.12
# ) #11.24
#
# CARNIVAL_Result2 <- runCARNIVAL(CplexPath=~"/Documents/cplex",
#                                Result_dir="results/multi_omic/carnival/",
#                                CARNIVAL_example=NULL,
#                                UP2GS=F,
#                                netFile = "support/meta_network_with_X_nobadchar.tsv",
#                                measFile = "results/phospho/signaling_input_carnival.tsv",
#                                inputFile = "results/metabolomic/metab_input_carnival.tsv",
#                                weightFile = NULL,
#                                timelimit = 7200
# ) # 2.44
#
# save(CARNIVAL_Result, file = "results/multi_omic/CARNIVAL_result_forward.RData")
# save(CARNIVAL_Result2, file = "results/multi_omic/CARNIVAL_result_backward.RData")
```

IF you run with you own data, you should skip this chunk.

```
load("results/multi_omic/CARNIVAL_result_forward.RData")
load("results/multi_omic/CARNIVAL_result_backward.RData")
```

Once the carnival results are available, we can map them back to user friendly identifiers and annotate the nodes with appropriate informations. You need to use again your own metabolite name to pubchem mapping table (same you used when preparing the inputs). This should lead to a sif and attribute dataframes that are ready to be imported by cytoscape in csv format.

```
signaling_input_carnival <- as.data.frame(
  read_delim("results/phospho/signaling_input_carnival.tsv",
    "\t", escape_double = FALSE, trim_ws = TRUE))

metab_input_carnival <- as.data.frame(
  read_delim("results/metabolomic/metab_input_carnival.tsv",
    "\t", escape_double = FALSE, trim_ws = TRUE))

sif <- as.data.frame(CARNIVAL_Result[[1]]$weightedSIF)
sif2 <- as.data.frame(CARNIVAL_Result2[[1]]$weightedSIF)

sif$Node1 <- gsub("^X","",sif$Node1)
sif$Node2 <- gsub("^X","",sif$Node2)

sif2$Node1 <- gsub("^X","",sif2$Node1)
sif2$Node2 <- gsub("^X","",sif2$Node2)

att <- as.data.frame(CARNIVAL_Result[[1]]$attributesAll)#[,c(1,2)]
att$measured <- ifelse(att$Nodes %in% names(signaling_input_carnival) | att$Nodes %in% names(metab_input_carnival),
  att$Nodes <- gsub("^X","",att$Nodes)

att2 <- as.data.frame(CARNIVAL_Result2[[1]]$attributesAll)#[,c(1,2)]
att2$measured <- ifelse(att2$Nodes %in% names(signaling_input_carnival) | att2$Nodes %in% names(metab_input_carnival),
  att2$Nodes <- gsub("^X","",att2$Nodes)

att$type <- ifelse(grepl("Metab",att$Nodes), "metabolite","protein")
att2$type <- ifelse(grepl("Metab",att2$Nodes), "metabolite","protein")

#####
prots <- unique(c(att$Nodes, att2$Nodes))
prots <- prots[!(grepl("Metab",prots))]
prots <- gsub("Gene[0-9]+_","",prots)
prots <- gsub("_reverse","",prots)
prots <- gsub("EXCHANGE.*","",prots)
prots <- unique(prots)

ensembl = useEnsembl(biomart="ensembl", dataset="hsapiens_gene_ensembl")

G_list <- getBM(filters = "entrezgene_id",
  attributes = c('hgnc_symbol','entrezgene_id', "description"),
  values = prots, mart = ensembl)

gene_mapping <- G_list[,1]
names(gene_mapping) <- G_list[,2]
```

```

sif$Node1 <- gsub("_reverse","",sif$Node1)
sif$Node2 <- gsub("_reverse","",sif$Node2)
sif2$Node1 <- gsub("_reverse","",sif2$Node1)
sif2$Node2 <- gsub("_reverse","",sif2$Node2)
att$Nodes <- gsub("_reverse","",att$Nodes)
att2$Nodes <- gsub("_reverse","",att2$Nodes)

sif$Node1 <- gsub("EXCHANGE.*","",sif$Node1)
sif$Node2 <- gsub("EXCHANGE.*","",sif$Node2)
sif2$Node1 <- gsub("EXCHANGE.*","",sif2$Node1)
sif2$Node2 <- gsub("EXCHANGE.*","",sif2$Node2)
att$Nodes <- gsub("EXCHANGE.*","",att$Nodes)
att2$Nodes <- gsub("EXCHANGE.*","",att2$Nodes)

metabs <- unique(c(att$Nodes, att2$Nodes))
metabs <- metabs[(grepl("Metab",metabs))]

metab_to_pubchem <- as.data.frame(read_csv("support/metab_to_pubchem.csv"))
metab_to_pubchem_vec <- metab_to_pubchem$name
names(metab_to_pubchem_vec) <- metab_to_pubchem$pubchem

for(i in 1:length(sif$Node1))
{
  if(gsub("Gene[0-9]+_","",sif[i,1]) %in% names(gene_mapping))
  {
    if(grepl("Gene",sif[i,1]))
    {
      prefix <- gsub("_.*", "",sif[i,1])
      sif[i,1] <- paste(prefix,gene_mapping[gsup("Gene[0-9]+_","",sif[i,1])], sep = "_")
    }
    else
    {
      sif[i,1] <- gene_mapping[gsup("Gene[0-9]+_","",sif[i,1])]
    }
  }
  if(gsub("Gene[0-9]+_","",sif[i,3]) %in% names(gene_mapping))
  {
    if(grepl("Gene",sif[i,3]))
    {
      prefix <- gsub("_.*", "",sif[i,3])
      sif[i,3] <- paste(prefix,gene_mapping[gsup("Gene[0-9]+_","",sif[i,3])], sep = "_")
    }
    else
    {
      sif[i,3] <- gene_mapping[gsup("Gene[0-9]+_","",sif[i,3])]
    }
  }
  if(gsub("Metab_","",gsup("___[a-z]___","",sif[i,1])) %in% names(metab_to_pubchem_vec))
  {
    suffix <- str_match(sif[i,1],"___.*___")
    metab <- metab_to_pubchem_vec[gsup("Metab_","",gsup("___[a-z]___","",sif[i,1]))]
    sif[i,1] <- paste(metab,suffix,sep = "")
  }
}

```

```

if(gsub("Metab__","",gsub("___[a-z]___","",sif[i,3])) %in% names(metab_to_pubchem_vec))
{
  suffix <- str_match(sif[i,3],"____.____")
  metab <- metab_to_pubchem_vec[gsub("Metab__","",gsub("___[a-z]___","",sif[i,3]))]
  sif[i,3] <- paste(metab,suffix,sep = "")
}
}

for(i in 1:length(sif2$Node1))
{
  if(gsub("Gene[0-9]+__","",sif2[i,1]) %in% names(gene_mapping))
  {
    if(grepl("Gene",sif2[i,1]))
    {
      prefix <- gsub("__.","",sif2[i,1])
      sif2[i,1] <- paste(prefix,gene_mapping[gsub("Gene[0-9]+__","",sif2[i,1])], sep = "__")
    }
    else
    {
      sif2[i,1] <- gene_mapping[gsub("Gene[0-9]+__","",sif2[i,1])]
    }
  }
  if(gsub("Gene[0-9]+__","",sif2[i,3]) %in% names(gene_mapping))
  {
    if(grepl("Gene",sif2[i,3]))
    {
      prefix <- gsub("__.","",sif2[i,3])
      sif2[i,3] <- paste(prefix,gene_mapping[gsub("Gene[0-9]+__","",sif2[i,3])], sep = "__")
    }
    else
    {
      sif2[i,3] <- gene_mapping[gsub("Gene[0-9]+__","",sif2[i,3])]
    }
  }
  if(gsub("Metab__","",gsub("___[a-z]___","",sif2[i,1])) %in% names(metab_to_pubchem_vec))
  {
    suffix <- str_match(sif2[i,1],"____.____")
    metab <- metab_to_pubchem_vec[gsub("Metab__","",gsub("___[a-z]___","",sif2[i,1]))]
    sif2[i,1] <- paste(metab,suffix,sep = "")
  }
  if(gsub("Metab__","",gsub("___[a-z]___","",sif2[i,3])) %in% names(metab_to_pubchem_vec))
  {
    suffix <- str_match(sif2[i,3],"____.____")
    metab <- metab_to_pubchem_vec[gsub("Metab__","",gsub("___[a-z]___","",sif2[i,3]))]
    sif2[i,3] <- paste(metab,suffix,sep = "")
  }
}

for(i in 1:length(att$Nodes))
{
  if(gsub("Gene[0-9]+__","",att[i,1]) %in% names(gene_mapping))
  {
    if(grepl("Gene",att[i,1]))

```

```

{
  prefix <- gsub("__.*", "", att[i,1])
  att[i,1] <- paste(prefix, gene_mapping[gsub("Gene[0-9]+__", "", att[i,1])], sep = "__")
}
else
{
  att[i,1] <- gene_mapping[gsub("Gene[0-9]+__", "", att[i,1])]
}
}
if(gsub("Metab__", "", gsub("___[a-z]___", "", att[i,1])) %in% names(metab_to_pubchem_vec))
{
  suffix <- str_match(att[i,1], "___.*___")
  metab <- metab_to_pubchem_vec[gsub("Metab__", "", gsub("___[a-z]___", "", att[i,1]))]
  att[i,1] <- paste(metab, suffix, sep = "")
}
}

for(i in 1:length(att2$Nodes))
{
  if(gsub("Gene[0-9]+__", "", att2[i,1]) %in% names(gene_mapping))
  {
    if(grepl("Gene", att2[i,1]))
    {
      prefix <- gsub("__.*", "", att2[i,1])
      att2[i,1] <- paste(prefix, gene_mapping[gsub("Gene[0-9]+__", "", att2[i,1])], sep = "__")
    }
    else
    {
      att2[i,1] <- gene_mapping[gsub("Gene[0-9]+__", "", att2[i,1])]
    }
  }
  if(gsub("Metab__", "", gsub("___[a-z]___", "", att2[i,1])) %in% names(metab_to_pubchem_vec))
  {
    suffix <- str_match(att2[i,1], "___.*___")
    metab <- metab_to_pubchem_vec[gsub("Metab__", "", gsub("___[a-z]___", "", att2[i,1]))]
    att2[i,1] <- paste(metab, suffix, sep = "")
  }
}

#####
url <- paste0(
  'http://omnipathdb.org/ptms?',
  'fields=sources,references&genesymbols=1'
)

download_omnipath <- function(){
  read.table(url, sep = '\t', header = TRUE)
}

omnipath_ptm <- download_omnipath()
omnipath_ptm <- omnipath_ptm[omnipath_ptm$modification %in% c("dephosphorylation", "phosphorylation"),]

```



```

KSN <- omnipath_ptm[,c(4,3)]
KSN$substrate_genesymbol <- paste(KSN$substrate_genesymbol,omnipath_ptm$residue_type, sep="_")
KSN$substrate_genesymbol <- paste(KSN$substrate_genesymbol,omnipath_ptm$residue_offset, sep="")
KSN$sign <- ifelse(omnipath_ptm$modification == "phosphorylation", 1, -1)

att$type <- ifelse(att$Nodes %in% KSN$enzyme_genesymbol, "Kinase",att$type)

url <- paste0(
  'http://omnipathdb.org/interactions?',
  'datasets=tfregulons&tfregulons_levels=A,B,C&genesymbols=1&fields=sources,tfregulons_level'
)

dorothea <- download_omnipath()
dorothea <- dorothea[,c(4,3,6,7)]
dorothea$sign <- dorothea$is_stimulation - dorothea$is_inhibition
dorothea$sign <- ifelse(dorothea$sign == 0, 1, dorothea$sign)
dorothea <- dorothea[,c(1,2,5)]

att$type <- ifelse(att$Nodes %in% dorothea$source_genesymbol, "TF",att$type)

i <- 1
for(node in att$Nodes) #check if a node name is basename appears more than once in nodes. If so, it is
{
  if(sum(gsub("Gene.*_", "", node) == gsub("Gene.*_", "", att$Nodes)) > 1)
  {
    att[i,4] <- "metab_enzyme"
  }
  i <- i+1
}

att$type <- ifelse(grepl("Gene.*_", att$Nodes), "metab_enzyme", att$type)

att2$type <- ifelse(att2$Nodes %in% KSN$enzyme_genesymbol, "Kinase", att2$type)

att2$type <- ifelse(att2$Nodes %in% dorothea$source_genesymbol, "TF", att2$type)

i <- 1
for(node in att2$Nodes) #check if a node name is basename appears more than once in nodes. If so, it is
{
  if(sum(gsub("Gene.*_", "", node) == gsub("Gene.*_", "", att2$Nodes)) > 1)
  {
    att2[i,4] <- "metab_enzyme"
  }
  i <- i+1
}

att2$type <- ifelse(grepl("Gene.*_", att2$Nodes), "metab_enzyme", att2$type)

att$Activity <- sign(as.numeric(as.character(att$Activity)))
att2$Activity <- sign(as.numeric(as.character(att2$Activity)))

sif$Node1 <- gsub("Gene[0-9]+_", "", sif$Node1)
sif$Node2 <- gsub("Gene[0-9]+_", "", sif$Node2)

```

```

att$Nodes <- gsub("Gene[0-9]+__","",att$Nodes)

sif2$Node1 <- gsub("Gene[0-9]+__","",sif2$Node1)
sif2$Node2 <- gsub("Gene[0-9]+__","",sif2$Node2)
att2$Nodes <- gsub("Gene[0-9]+__","",att2$Nodes)

sif <- sif[sif$Node1 != sif$Node2,]
sif2 <- sif2[sif2$Node1 != sif2$Node2,]

sif$Node1 <- gsub("[_]{3,5}","",sif$Node1)
sif$Node2 <- gsub("[_]{3,5}","",sif$Node2)
att$Nodes <- gsub("[_]{3,5}","",att$Nodes)

sif2$Node1 <- gsub("[_]{3,5}","",sif2$Node1)
sif2$Node2 <- gsub("[_]{3,5}","",sif2$Node2)
att2$Nodes <- gsub("[_]{3,5}","",att2$Nodes)

sif3 <- as.data.frame(rbind(sif[as.numeric(as.character(sif$Weight)) >= 80,], sif2[as.numeric(as.character(sif2$Weight)) >= 80,]))

```