metaboGSE - integrating RNA-seq data and metabolic networks

Van Du T. Tran, Marco Pagni

2018-08-07

1 Introduction

The *metaboGSE* package is designed for the integration of transcriptomic data and metabolic networks by constructing condition-specific series metabolic sub-networks by means of RNA-seq data and providing a downstream analysis of RNA-seq data with the aid of such sub-network series (Tran et al., 2018).

2 Installation

metaboGSE depends on the *sybil* package and was evaluated with COIN-OR Clp and GLPK solvers via the *clpAPI* and *glpkAPI* packages, respectively, available in CRAN[1]. The solvers and their R interface API pacakge are not automatically installed along with *metaboGSE*. *clpAPI* with *inibarrier* method seems to be the most consistent in our tests. Other solvers implemented for *sybil* should also work, however have not been tested.

3 Usage

3.1 Sybil settings

```
library(metaboGSE)
SYBIL_SETTINGS("SOLVER", "glpkAPI")
SYBIL_SETTINGS("METHOD", "simplex")
SYBIL_SETTINGS("OPT_DIRECTION", "max")
```

3.2 Data preparation

3.2.1 Metabolic networks

A metabolic network can be imported from tabular or SBML inputs (see *sybil*'s manual for more details). Two *Yarrowia lipolytica* models in normoxic and hypoxic environments are provided in the `iMK735` dataset (Kavšček et al., 2015) and presented in the MetaNetX/MNXref namespace (Moretti et al., 2016). They are identical apart from the bounds of exchange oxygen flux.

```
data(iMK735)
iMK735[1]
```

```
## $hypoxia
## $hypoxia$model
## model name:             No description
## number of compartments  8
##                         Golgi apparatus
##                         cytoplasm
```

[1] `https://cran.r-project.org`

```
##                           endoplasmic reticulum
##                           extracellular region
##                           mitochondrion
##                           nucleus
##                           peroxisome
##                           vacuole
## number of reactions:     1329
## number of metabolites:   1112
## number of unique genes:  709
## objective function:      +1 R1835AC86
##
## $hypoxia$obj
## [1] 4.113629
##
## $hypoxia$comp
## model name:              No description
## number of compartments   9
##                           Golgi apparatus
##                           cytoplasm
##                           endoplasmic reticulum
##                           extracellular region
##                           mitochondrion
##                           nucleus
##                           peroxisome
##                           vacuole
##                           RECO_PUSH_MNXC3
## number of reactions:     818
## number of metabolites:   605
## number of unique genes:  469
## objective function:      +1 R1835AC86
```

3.2.2 RNA-seq data

Normalized (or raw) RNA-seq counts should be provided as a matrix with gene per row and library per column. The `exprMaguire` (Maguire et al., 2014) dataset contains two matrices representing log2 voom-normalized count (`expr`) and RPKM (`pkmExpr`) per library.

```
data(exprMaguire)
names(exprMaguire)
```

```
## [1] "expr"    "pkmExpr"
```

```
str(exprMaguire$expr, vec.len=3)
```

```
##  num [1:469, 1:22] 0.848 7.836 8.244 8.62 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:469] "euk:Q6C902_YARLI" "euk:Q6CDH6_YARLI" "euk:Q6CED5_YARLI" ...
##   ..$ : chr [1:22] "DN2" "DN3" "SH1" ...
```

3.2.3 Gene set annotation

Here we apply `metaboGSE` for Gene Ontology enrichment analysis. A mapping between genes and GO terms should be provided. The `yarli2GO` dataset contains such a mapping in a `list` format. The pre-built *topGOdata* object `yarliGOdata` is also provided.

```
data(yarli2GO)
length(yarli2GO)
```

```
## [1] 4747
```

```
str(head(yarli2GO))
```

```
## List of 6
##  $ euk:3HAO_YARLI : chr [1:7] "GO:0000334" "GO:0005737" "GO:0006569" "GO:0008198" ...
##  $ euk:ACEA_YARLI : chr [1:6] "GO:0004451" "GO:0006097" "GO:0006099" "GO:0009514" ...
##  $ euk:ACEB_YARLI : chr [1:4] "GO:0004451" "GO:0005759" "GO:0019629" "GO:0046421"
##  $ euk:ACH1_YARLI : chr [1:6] "GO:0003986" "GO:0005739" "GO:0005829" "GO:0006083" ...
##  $ euk:ACOX1_YARLI: chr [1:5] "GO:0003995" "GO:0003997" "GO:0005777" "GO:0033540" ...
##  $ euk:ACOX2_YARLI: chr [1:5] "GO:0003995" "GO:0003997" "GO:0005777" "GO:0033540" ...
```

3.2.4 Pre-built data

The tutorial below shows a simple example to be executed in a reasonable computing time. The datasets of pre-built series of metabolic sub-networks and gene set enrichment (Tran et al., 2018) are also provided.

```
data(yarliSubmnets)
names(yarliSubmnets)
```

```
## [1] "DN" "SH" "SN" "UH" "UN" "WH" "WN"
```

```
data(yarliGSE)
length(yarliGSE)
```

```
## [1] 135
```

3.3 Analysis

3.3.1 Model rescuing

Both models of iMK735 grow, nevertheless we should still apply the rescue procedure to produce the rescue reactions for all metabolites in growth and maintenance (if any) reactions (see (Tran et al., 2018)). For instance, we here perform the rescue process while targeting growths of 20% of the initial objective values for the hypoxic model.

```
target.ratio <- 0.2
hmodel <- iMK735$hypoxia$model
hobj <- iMK735$hypoxia$obj
hmodel.rescue <- rescue(model = hmodel,
                        target = c(target.ratio*hobj),
                        react = c(which(obj_coef(hmodel) == 1)))
```

```
## SYBIL_SETTINGS(TOLERANCE) has been set to 1e-12
## SYBIL_SETTINGS(OPT_DIRECTION) has been set to min
```

In the `hmodel.rescue` object, the `rescued` field represents the rescued model $\mathcal{M}''$, which is the same as the initial model $\mathcal{M}$ since the latter grows, as presented in Fig 1. The `rescue` field presents the rescue model $\mathcal{M}'$, which contains rescue reactions on every metabolite in growth and maintenance reactions, and the `coef` field indicates the coefficients of those rescue reactions used for optimization in the rescue process. RECO_PUSH_MNXC3 denotes the compartment of artificial metabolites, e.g. X′. The rescue procecure can also be performed on maintenance reactions, which are restricted to non-empty fixed fluxes.

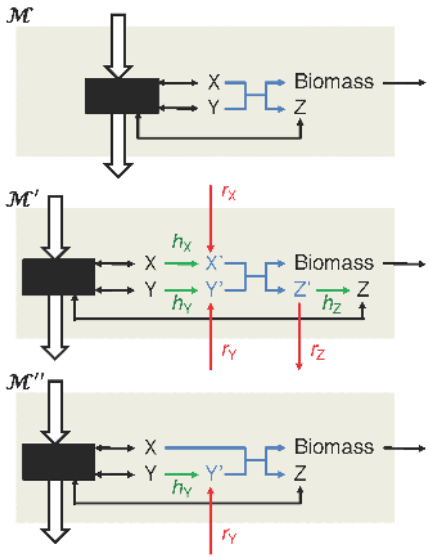

Figure 1: Schema of GSMN rescue process. $\mathcal{M}$, original GSMN with growth reaction X + Y —> Z + Biomass. $\mathcal{M}'$, expanded GSMN with the full set of rescue (r_x) and help (h_x) reactions for every metabolite x in the biomass reaction. $\mathcal{M}''$, example of a minimal rescued GSMN in the particular case where only metabolite Y needs to be rescued. (Tran et al., 2018)

```
hmodel.rescue$rescue
```

```
## model name:             No description
## number of compartments  9
##                         Golgi apparatus
##                         cytoplasm
##                         endoplasmic reticulum
##                         extracellular region
##                         mitochondrion
##                         nucleus
##                         peroxisome
##                         vacuole
##                         RECO_PUSH_MNXC3
## number of reactions:    1419
## number of metabolites:  1157
## number of unique genes: 709
## objective function:     +1 R1835AC86
```

```
head(hmodel.rescue$coef)
```

```
##                              R1835AC86
## RECO_PUSH_MNXM32_MNXC3    3.47947112
## RECO_PUSH_MNXM3_MNXC3     0.06531679
## RECO_PUSH_MNXM1_MNXC3     0.06531679
## RECO_PUSH_MNXM7_MNXC3     0.06531679
## RECO_PUSH_MNXM9_MNXC3     0.06531679
## RECO_PUSH_MNXM89557_MNXC3 5.28960592
```

3.3.2 Model cleaning

We set the TOLERANCE to 1e-8, which indicates that values less than 1e-8 are considered as 0, to deal with numerical imprecision.

```
SYBIL_SETTINGS("TOLERANCE", 1e-08)
```

The blocked reactions from the rescue models are determined by a flux variability analysis via the `fluxVar` function from *sybil*. Those reactions as well as related genes and metabolites are then removed from the models. `mc.cores` can be set appropriately to perform parallel computation of `fluxVar`. A high `mc.cores` is recommended as `fluxVar` is time consuming.

```
## not run
mc.cores <- 10
fva <- multiDel(model=hmodel.rescue$rescue,
                nProc=mc.cores,
                todo="fluxVar",
                fixObjVal=F,
                del1=react_id(hmodel.rescue$rescue))
reacs.blo <- names(which(setNames(unlist(lapply(fva, blReact)),
                   react_id(hmodel.rescue$rescue))))
hmodel.clean <- rmReact(hmodel.rescue$rescue, reacs.blo, rm_met=T)
##
```

`hmodel.clean`, considered as the comprehensive model for hypoxic conditions, can be loaded from the `iMK735` dataset.

```
hmodel.clean <- iMK735$hypoxia$comp
hmodel.clean
```

```
## model name:             No description
## number of compartments  9
##                         Golgi apparatus
##                         cytoplasm
##                         endoplasmic reticulum
##                         extracellular region
##                         mitochondrion
##                         nucleus
##                         peroxisome
##                         vacuole
##                         RECO_PUSH_MNXC3
## number of reactions:    818
## number of metabolites:  605
## number of unique genes: 469
## objective function:     +1 R1835AC86
```

Now we convert the growth objective of the comprehensive model to a weighted objective function on rescue reactions with the determined coefficients. Hereafter, the goal is to minimize this function.

```
SYBIL_SETTINGS("OPT_DIRECTION", "min")
hmodel.weight <- changeObjFunc(hmodel.clean, react=rownames(hmodel.rescue$coef), obj_coef=hmodel.rescue
hmodel.weight
```

```
## model name:             No description
## number of compartments  9
##                         Golgi apparatus
##                         cytoplasm
##                         endoplasmic reticulum
##                         extracellular region
##                         mitochondrion
```

```
##                          nucleus
##                          peroxisome
##                          vacuole
##                          RECO_PUSH_MNXC3
## number of reactions:     818
## number of metabolites:   605
## number of unique genes: 469
## objective function:      +0.0653167864141084 RECO_PUSH_MNXM3_MNXC3 +0.0653167864141084 RECO_PUSH_MNXM
```

```
optimizeProb(hmodel.weight)
```

```
## solver:                                    glpkAPI
## method:                                    simplex
## algorithm:                                 fba
## number of variables:                       818
## number of constraints:                     605
## return value of solver:                    solution process was successful
## solution status:                           solution is optimal
## value of objective function (fba):         0.000000
## value of objective function (model):       0.000000
```

The obtained objective of 0 above indicates that there is no need to rescue the `hmodel.clean` since it grows.

3.3.4 Weighting scheme

We now compute weights for rescue reactions to account for the importance and dependency of metabolites to rescue (see Weighting scheme for model fitness in Tran et al. (2018))

```
mc.cores <- 1
rescue.weight <- (weightReacts(hmodel.weight, mc.cores=mc.cores, gene.num=1))$weight
str(rescue.weight, vec.len=2)
```

```
##  Named num [1:45] 0.03951 0.00152 ...
##  - attr(*, "names")= chr [1:45] "RECO_PUSH_MNXM3_MNXC3" "RECO_PUSH_MNXM1_MNXC3" ...
```

3.3.4 GO annotation

We compute the set of preliminary GO terms in *biological process* category using `topGO` with *fisher* statistic and *weight01* algorithm. The whole GO annotation and gene universe are used. The aim of the following R script is to preliminarily filter the set of GO terms of interest. The resulting 135 GO terms are filtered by p-value < 0.1 and contain at least 3 genes and at most 50 genes from the model.

```
GO2geneID <- inverseList(yarli2GO)
length(GO2geneID)
```

```
## [1] 4629
```

```
str(head(GO2geneID), vec.len=3)
```

```
## List of 6
##  $ GO:0000001: chr [1:13] "euk:GEM1_YARLI" "euk:MDM12_YARLI" "euk:Q6C1G8_YARLI" ...
##  $ GO:0000002: chr [1:27] "euk:B5FVH9_YARLI" "euk:CCM1_YARLI" "euk:DML1_YARLI" ...
##  $ GO:0000007: chr "euk:Q6C0U6_YARLI"
##  $ GO:0000009: chr [1:7] "euk:F2Z612_YARLI" "euk:F2Z6C7_YARLI" "euk:Q6C1C8_YARLI" ...
##  $ GO:0000010: chr "euk:COQ1_YARLI"
##  $ GO:0000011: chr [1:9] "euk:ACT_YARLI" "euk:Q6C134_YARLI" "euk:Q6C1E5_YARLI" ...
```

```
gene.name <- names(yarli2GO)
gene.list <- factor(as.integer(gene.name %in% sybil::allGenes(hmodel.clean)))
names(gene.list) <- gene.name
GOdata <- new("topGOdata",
              ontology = "BP",
              nodeSize = 5,
              allGenes = gene.list,
              annot    = annFUN.gene2GO,
              gene2GO  = yarli2GO
              )
result <- runTest(GOdata, statistic="fisher", algorithm="weight01")
table  <- GenTable(GOdata,
                   weight   = result,
                   orderBy  = "weight",
                   numChar  = 10000,
                   topNodes = result@geneData[4]
                   )
table$weight <- as.numeric(sub("<", "", table$weight))
table <- table[!is.na(table$weight), ]
MINSIG <- 3
MAXSIG <- 50
WCUTOFF <- 0.1
GO.interest <- table[table$Significant >= MINSIG & table$Significant <= MAXSIG &
                         table$weight < WCUTOFF, ]$GO.ID
GO2geneID.interest.proteome <- genesInTerm(GOdata, GO.interest)
GO2geneID.interest <- lapply(GO2geneID.interest.proteome, function(git) {
    intersect(sybil::allGenes(hmodel.clean), git)
})
length(GO.interest)
```

```
## [1] 135
```

```
str(head(GO2geneID.interest), vec.len=3)
```

```
## List of 6
##  $ GO:0035428: chr [1:23] "euk:F2Z653_YARLI" "euk:Q6C0E0_YARLI" "euk:Q6C0K5_YARLI" ...
##  $ GO:0046323: chr [1:23] "euk:F2Z653_YARLI" "euk:Q6C0E0_YARLI" "euk:Q6C0K5_YARLI" ...
##  $ GO:0015986: chr [1:17] "euk:ATP6_YARLI" "euk:ATP8_YARLI" "euk:ATP9_YARLI" ...
##  $ GO:0015991: chr [1:15] "euk:ATP9_YARLI" "euk:Q6CHD8_YARLI" "euk:Q6CH91_YARLI" ...
##  $ GO:0006696: chr [1:19] "euk:Q6C8C2_YARLI" "euk:ERG6_YARLI" "euk:Q6C231_YARLI" ...
##  $ GO:0006099: chr [1:16] "euk:PRPC_YARLI" "euk:Q6C450_YARLI" "euk:Q6C823_YARLI" ...
```

GO.interest contains other GO terms than those in GO2geneID, as `topGO` allows propagating in the gene ontology.

3.3.5 Expression-based gene removal

`step` indicates the difference of gene numbers to remove between consecutive sub-model constructions, then determines numbers of genes to remove in the simulation. Here we set `step = 50` and `draw.num = 4` to reduce the computing time in this tutorial, i.e. the 0, 50, 100, etc. first genes in certain ranking will be successively removed from the comprehensive model, and 4 random removals will be performed. The series of metabolic sub-networks is constructed for the hypoxic *upc*2Δ (UH) condition with various gene rankings as below.

```
cond <- "UH"
step <- 50
draw.num <- 4
reps.i <- grep(cond, colnames(exprMaguire$expr), value=T)
ranks <- mclapply(reps.i, mc.cores=mc.cores, function(ri) {
    data.frame(
        # ranks1. voom-normalized expression
        expr = exprMaguire$expr[, ri, drop=T],
        # ranks2. pkm normalized expression
        pkmExpr  = exprMaguire$pkmExpr[, ri, drop=T],
        # ranks3. relative expression power 1
        relExpr1 = relativeExpr(exprMaguire$expr, power=1)[, ri, drop=T],
        # ranks4. relative expression power 2
        relExpr2 = relativeExpr(exprMaguire$expr, power=2)[, ri, drop=T],
        # ranks5. relative expression power 3
        relExpr3 = relativeExpr(exprMaguire$expr, power=3)[, ri, drop=T],
        # ranks6. reverse expression (the worst)
        revExpr  = 1/(1 + exprMaguire$expr[, ri, drop=T]),
        # ranks7. z-score expression
        zExpr    = zscoreExpr(exprMaguire$expr)[, ri, drop=T]
        )
})
names(ranks) <- reps.i
fitnessUH <- fitness(model         = hmodel.weight,
                     ranks         = ranks,
                     rescue.weight = rescue.weight,
                     step          = step,
                     draw.num      = draw.num,
                     mc.cores      = mc.cores)
submnetsUH <- submnet(model        = hmodel.weight,
                      fn           = fitnessUH,
                      rank.best    = "expr",
                      gene.sets    = GO2geneID.interest,
                      mc.cores     = mc.cores)
```

```
## Warning: 'cBind' is deprecated.
##  Since R version 3.2.0, base's cbind() should work fine with S4 objects
```

```
## Warning: 'rBind' is deprecated.
##  Since R version 3.2.0, base's rbind() should work fine with S4 objects
```

Note that, in a specific application, the evaluation for different rankings is unnecessary. It is sufficient to set `expr` to the expression you want to use, then set `ranks = NULL` and `draw.num = 0`.

```
submnetsUH$condition
```

```
## [1] "UH"
```

```
knitr::kable(submnetsUH$gene.del)
```

	0	50	100	150	200	250	300	350	400	450
gene.del	0	50	100	150	200	250	300	350	400	450

```
knitr::kable(submnetsUH$fitness.random, digits=3)
```

	0	50	100	150	200	250	300	350	400	450
F.random.1	1	0.339	0.314	0.035	0.021	0.007	0.026	0.004	0.007	0.00
F.random.2	1	0.231	0.384	0.044	0.042	0.040	0.001	0.031	0.000	0.01
F.random.3	1	0.457	0.143	0.066	0.028	0.003	0.009	0.004	0.000	0.00
F.random.4	1	0.158	0.287	0.075	0.015	0.026	0.033	0.000	0.010	0.00

```
knitr::kable(submnetsUH$fitness.ranks$UH1, digits=3)
```

	0	50	100	150	200	250	300	350	400	450
expr	1	0.928	0.405	0.405	0.271	0.112	0.029	0.000	0.000	0.000
pkmExpr	1	0.941	0.405	0.404	0.238	0.147	0.000	0.000	0.000	0.000
relExpr1	1	0.536	0.444	0.378	0.128	0.113	0.027	0.027	0.027	0.000
relExpr2	1	0.477	0.405	0.274	0.248	0.043	0.043	0.043	0.003	0.000
relExpr3	1	0.477	0.405	0.274	0.248	0.043	0.034	0.003	0.000	0.000
revExpr	1	0.158	0.084	0.027	0.013	0.013	0.009	0.007	0.006	0.004
zExpr	1	0.549	0.405	0.307	0.127	0.088	0.027	0.027	0.000	0.000

The `yarliSubmnets` dataset contains the series of sub-networks built with `step = 1` and `draw.num = 50`, indicating the gene-by-gene removal.

```
data(yarliSubmnets)
str(yarliSubmnets$UH$gene.del)
```

```
##  num [1, 1:470] 0 1 2 3 4 5 6 7 8 9 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr "gene.del"
##   ..$ : chr [1:470] "0" "1" "2" "3" ...
```

```
dim(yarliSubmnets$UH$fitness.random)
```

```
## [1]  50 470
```

```
str(yarliSubmnets$UH$fitness.ranks)
```

```
## List of 3
##  $ UH1: num [1:7, 1:470] 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:7] "expr" "pkmExpr" "relExpr1" "relExpr2" ...
##   .. ..$ : chr [1:470] "0" "1" "2" "3" ...
##  $ UH2: num [1:7, 1:470] 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:7] "expr" "pkmExpr" "relExpr1" "relExpr2" ...
##   .. ..$ : chr [1:470] "0" "1" "2" "3" ...
##  $ UH3: num [1:7, 1:470] 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:7] "expr" "pkmExpr" "relExpr1" "relExpr2" ...
##   .. ..$ : chr [1:470] "0" "1" "2" "3" ...
```

The sub-network construction can be visualized via `simulateSubmnet` function, which produces a plot *smooth.pdf for each condition.

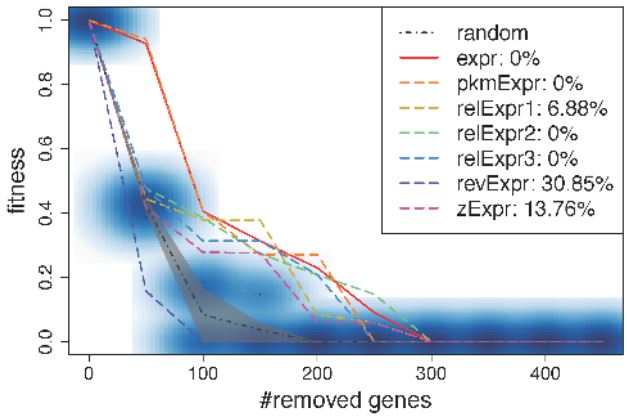

Figure 2: submnetsUH with `step = 50` and `draw.num = 4`

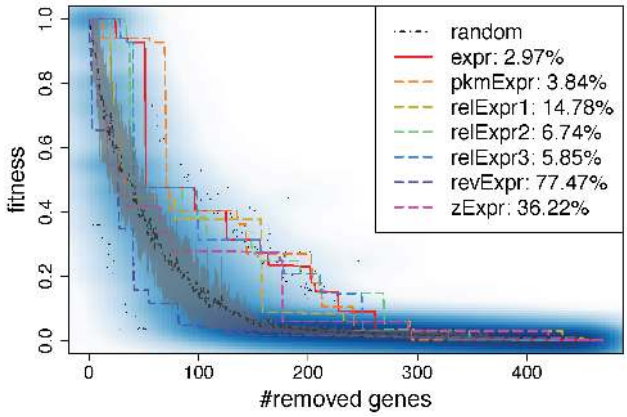

Figure 3: yarliSubmnets with `step = 1` and `draw.num = 50`

```
simulateSubmnet(sgd      = submnetsUH,
                mc.cores = mc.cores)
```

The plots below show the fitness of submodels obtained by removing genes following different rankings for the hypoxic *upc*2Δ condition.

3.3.6 GO term enrichment

We evaluate the significance of given gene sets with the function `metaboGSE` and randomization tests. `nrand = 1000` is used in the test for the significance of the gene sets against random sets in each individual condition. `nperm = 1000` is used in the test for the significance of difference between conditions.

```
## not run
GSE <- metaboGSE(yarliSubmnets, method="perm", nperm=1000, nrand=1000,
                 mc.cores=mc.cores, prefix="/tmp/summary")
##
```

This step is time consuming. The `yarliGSE` dataset can be loaded instead. Note that `yarliGOdata` should be also loaded, since the constructed `GOdata` may not match the pre-built `GOdata` due to `GO.db` and `topGO` version difference.

```
data(yarliGSE)
data(yarliGOdata)
GSE <- yarliGSE
GOdata <- yarliGOdata
str(GSE[["GO:0006696"]], vec.len=2)
```

```
## List of 2
##  $ res :List of 8
##   ..$ GS.ID      : chr "GO:0006696"
##   ..$ Description: chr "ergosterol biosynthetic process"
##   ..$ statistic  : num 0.184
##   ..$ p.Cond     : Named num [1:7] 0.000104 0.002104 ...
##   .. ..- attr(*, "names")= chr [1:7] "DN" "SH" ...
##   ..$ p.Val      : num 0.014
##   ..$ auc        : Named num [1:22] 0.287 0.267 ...
##   .. ..- attr(*, "names")= chr [1:22] "DN2" "DN3" ...
##   ..$ pw.posthoc :'data.frame':  21 obs. of  4 variables:
##   .. ..$ cond1    : chr [1:21] "DN" "DN" ...
##   .. ..$ cond2    : chr [1:21] "SH" "SN" ...
##   .. ..$ statistic: chr [1:21] "0.17" "0.065" ...
##   .. ..$ p.Val    : chr [1:21] "0.174825174825175" "0.0689310689310689" ...
##   ..$ contrast   : Named chr [1:3] "((SH,WH),((UH,WN),(DN,(SN,UN))))" "DN,SN,UH,UN,WN" ...
##   .. ..- attr(*, "names")= chr [1:3] "newick" "group1" ...
##  $ plot:List of 3
##   ..$ gs.fracs         : num [1:470, 1:22] 1 1 1 1 1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:22] "DN2" "DN3" ...
##   ..$ xout             : num [1:44] 0 0.334 ...
##   ..$ gs.fracs.itp.mean: num [1:44, 1:7] 0 0 0 0 0 0 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:7] "DN" "SH" ...
```

```
GSE[["GO:0006696"]]$res$p.Val
```

```
## [1] 0.01398601
```

```
GS.sig.all <- as.data.frame(t(sapply(GSE, function(gsm) {
    c(GS.ID=gsm$res$GS.ID,
      Description=gsm$res$Description,
      Statistic=gsm$res$statistic,
      p.Cond=if (is.null(gsm$res$p.Cond)) NA else min(gsm$res$p.Cond),
      p.Val=gsm$res$p.Val,
      Genes=paste(intersect(genesInTerm(GOdata, gsm$res$GS.ID)[[1]],
                            sybil::allGenes(hmodel.clean)), collapse=','))
})), stringsAsFactors=F)
GS.sig.all$FDR  <- p.adjust(as.numeric(GS.sig.all$p.Val), method="BH")
GS.sig.all <- GS.sig.all[!is.na(GS.sig.all$FDR), ]
dim(GS.sig.all)
```

```
## [1] 135   7
```

The single p-values for each condition and FDR of discrepancy between condition are cut off at 0.01 and 0.05, respectively.

```
GS.sig <- GS.sig.all[as.numeric(GS.sig.all$FDR) < 0.05, , drop=F]
GS.sig <- GS.sig[as.numeric(GS.sig$p.Cond) < 0.01, , drop=F]
dim(GS.sig)
```

```
## [1] 46  7
```

REFERENCES

Kavšček, M., Bhutada, G., Madl, T., Natter, K., 2015. Optimization of lipid production with a genome-scale model of Yarrowia lipolytica. BMC Systems Biology 9, 72.

Maguire, S.L., Wang, C., Holland, L.M., Brunel, F., Neuvéglise, C., Nicaud, J.-M., Zavrel, M., White, T.C., Wolfe, K.H., Butler, G., 2014. Zinc finger transcription factors displaced SREBP proteins as the major Sterol regulators during Saccharomycotina evolution. PLoS genetics 10, e1004076.

Moretti, S., Martin, O., Van Du Tran, T., Bridge, A., Morgat, A., Pagni, M., 2016. MetaNetX/MNXref–reconciliation of metabolites and biochemical reactions to bring together genome-scale metabolic networks. Nucleic Acids Res. 44, D523–526.

Tran, V.D.T., Moretti, S., Coste, A.T., Amorim-Vaz, S., Sanglard, D., Pagni, M., 2018. Condition-specific series of metabolic sub-networks and its application for gene set enrichment analysis. bioRxiv 200964.