Modules Detection

Here you can find all the R code and data used for the module detection algorithm described in “Computational identification of co-evolving multi-gene modules in microbial biosynthetic gene clusters”. Before starting, it is necessary to download all the files contained here.

Prerequisites

Several packages needs to be installed and loaded in order to be able to use the code here described. * parallel * foreach * doParallel * Matrix * igraph * Rmpfr

Getting Started

Keep in mind that all intermediate files are saved, so it is not necessary to replicate the entire analysis should something go wrong at a later step.

First it is necessary to load all the necessary packages:

```
library(parallel)
library(foreach)
library(doParallel)
library(Matrix)
library(igraph)
library(Rmpfr)
```

After this has been done, the user should navigate to the folder where all the data and the scripts are stored.

```
setwd("/path/to/chosen/folder/")
```

Trimming

This step starts from the file containing all the BGCs annotated according smCOGs (clustersCOGSnr.csv) and creates the clustersCOGSnr_trimmed.csv where: 1. all clusters containing less than 2 different cogs are removed; 1. if 2 or more clusters have the same smCOG composition, only the shortest one is kept; 1. when the same smCOG is repeated subsequently more than once in a cluster it is replaced with the empty cell “-” and concatenated at the end of the cluster. Even if we are losing some neighbouring interactions and slightly modifying the cluster’s topology, this makes the p-value evaluation much simpler and considerably faster.

To perform this step, the user just needs to paste the following:

```
source("trimming.R")
rm(list = ls())
```

Counting co-localization interactions

This step takes as input the clustersCOGSnr_trimmed.csv file, counts all the coloc interactions between smCOGs and puts them in a matrix. The results are saved both in coloc_count.Rdata and coloc_count.csv.

```
source("counting_coloc.R")
rm(list = ls())
```

Counting adjacency interactions

This step takes as input the clustersCOGSnr_trimmed.csv file, counts all the neigh interactions between smCOGs and puts them in a matrix. The results are saved both in neigh_count.Rdata and neigh_count.csv.

```
source("counting_neigh.R")
rm(list = ls())
```

Computing co-localization p-values

This step takes as input the clustersCOGSnr_trimmed.csv and coloc_count.Rdata files and computes the co-localization p-values in both directions. The computed p-values are saved into coloc_pval.Rdata and coloc_pval.csv.

```
source("coloc_pval.R")
rm(list = ls())
```

Computing adjacency p-values

Computing adjacency p-values

This step takes as input the clustersCOGSnr_trimmed.csv and neigh_count.Rdata files and computes the adjacency p-values in both directions and saves them into neigh_pval.Rdata and neigh_pval.csv.

```
source("neigh_pval.R")
rm(list = ls())
```

Handling co-localization p-values

This step takes as input the coloc_pval.Rdata file. First, it creates a symmetric matrix of p-values where the (i,j) element is substituted by the max((i,j),(j,i)) and saves it into coloc_pval_MAX.Rdata and coloc_pval_MAX.csv.

Second, it takes the last matrix and applies the BY correction to the p-values. This is saved in coloc_pval_adj.Rdata and coloc_pval_adj.csv.

```
source("handling_coloc_pvalues.R")
rm(list = ls())
```

Handling adjacency p-values

This step takes as input the neigh_pval.Rdata file. First, it creates a symmetric matrix of p-values where the (i,j) element is substituted by the max((i,j),(j,i)) and saves it into neigh_pval_MAX.Rdata and neigh_pval_MAX.csv. Second, it takes the last matrix and applies the BY correction to the p-values. This is saved in neigh_pval_adj.Rdata and neigh_pval_adj.csv.

```
source("handling_neigh_pvalues.R")
rm(list = ls())
```

Modules detection

Before running the module detection script, it is necessary to merge the results just obtained in a single Rdata file:

```
load("coloc_pval_adj.Rdata")
load("neigh_pval_adj.Rdata")
save(coloc.pval,neigh.pval, file = "modules_adj_pvalues.Rdata")
rm(list = ls())
```

This step starts takes modules_adj_pvalues.Rdata as input and generates the output Modules.Rdata. Starting from an arbitrary threshold (.1), a symmetric binary matrix is built where the (i,j) element is 1 if at least one of the two p-values (adjacency or co-localization) is lower than the threshold. This binary matrix is then used to create a graph. Next, the igraph package is used to find all the maximal cliques (fully connected sub-graphs that are not subsets of other fully connected sub-graphs). These cliques are considered as potential modules. The procedure just described is repeated considering every possible threshold <.1.

```
source('detect_modules_function.R')
source("detect_modules.R")
rm(list = ls())
```

Computing Metrics

This step takes as inputs: info_clusters.Rdata, Modules.Rdata, modules_adj_pvalues.Rdata, COG_annotation_final.csv. It computes a number of different metrics used to rank and filter the modules: length, Shannon's entropy, N cluster hit, N clusers MIBIG, max pval, N compound classes, % CORE cogs, % CORE/TAILORING cogs, % MIXED cogs, % OTHER cogs, % REGULATOR cogs, % REGULATOR/TAILORING cogs, % TAILORING cogs, % TAILORING/CORE cogs, % TAILORING/REGULATOR cogs, % TAILORING/TRANSPORT cogs and % TRANSPORT cogs. The file all_detected_modules.Rdata contains both module composition and metrics per each of the detected modules. The filtered_modules_2hits.Rdata instead contains all modules found in at least 2 clusters.

```
source("Computing_Metrics.R")
rm(list = ls())
```

Computing MIB score

The MIB score is a weighted rank sum. First all the ranks for the metrics just described are computed.

```
source("compute_ranks.R")
rm(list = ls())
```

It is also necessary to define the weights needed.

```
weights <- c(2, #"length"
             15, #"Shannon's Entropy"
             10, #"N cluster hit"
             5, #"max pvalues"
             0, #"N compound classes"
             0, # "CORE"
             0, # "CORE/TAILORING"
             0, #"MIXED"
             0,  #"OTHER"
             0, #"REGULATOR"
             0, # "REGULATOR/TAILORING"
             10, #"TAILORING"
             0, #"TAILORING/CORE"
             0, #"TAILORING/REGULATOR"
             0, #"TAILORING/TRANSPORT"
             0) #"TRANSPORT"
```

Finally, the MIB score is computed, and the final results are saved in the file final_results_modules.Rdata.

```
source("MIB_score.R")
```

Built With

- R
- RStudio

Authors

- **Francesco Del Carratore**
- **Konrad Zych**
- **Matthew Cummings**
- **Eriko Takano**
- **Marnix Medema**
- **Rainer Breitling**