Using sabinaHSBM for link prediction and network reconstruction using Hierarchical Stochastic Block Models

Example on how to use it to identify missing links with the binary\_classifier method

The *sabinaHSBM* package provides tools for link prediction and network reconstruction using hierarchical stochastic block models (HSBM). This document demonstrates a simple use case based on the binary\_classified method to identify missing links. We will use the example dataset dat included in the package to show key functionalities, including data preparation, link prediction, and network reconstruction.

**Important note**:

The *sabinaHSBM* package depends on various R and Python libraries, as well as system-level components — especially the graph-tool Python package.

* **UNIX users** can run *sabinaHSBM* natively if their system includes:
  + R (version ≥ 4.3.3) with all required R packages (listed below)
  + Python ≥ 3.12 with the graph-tool library (version 2.59)
* For all other users — or to avoid configuration issues — a **ready-to-use Docker image** is available. It includes all dependencies pre-installed, including the *sabinaHSBM* package. See *Supporting Information S3* for instructions on how to use.

# Loading Required Libraries

After ensuring the required system dependencies are available (see note above), we begin by loading the required R packages. Missing packages are automatically installed.

# If the package is not installed, install it from GitHub  
# if (!requireNamespace("sabinaHSBM", quietly = TRUE)) {  
# library(remotes)  
# remotes::install\_github("h-lima/sabinaHSBM")  
# }  
  
# Load the sabinaHSBM package  
#library(sabinaHSBM)  
devtools::load\_all("/home/project\_hsbm/graph\_tool/sabinaHSBM/")

If the required packages are not already installed, install them from CRAN

list.of.packages <- c(  
 "dplyr",  
 "parallel",  
 "reshape2",  
 "reticulate",  
 "stringr",  
 "tidyr",  
 "ROCR",  
 "data.table"  
)  
  
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[, "Package"])]  
if (length(new.packages) > 0) {  
 install.packages(new.packages, dependencies = TRUE)  
}  
  
for (package in list.of.packages) {  
 library(package, character.only = TRUE)  
}

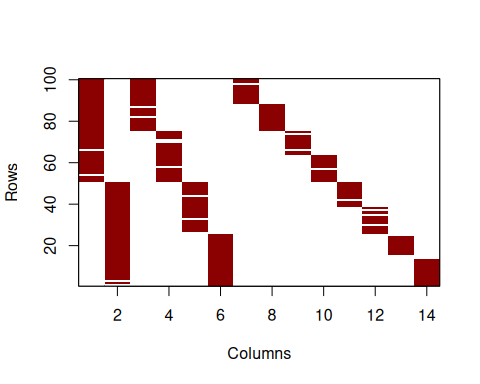
# Record starting time (optional)  
start\_time <- Sys.time()

# Load data

The dataset dat is a binary bipartite matrix representing a hypothetical species interactions network. Columns and rows correspond to two different types of nodes (e.g., hosts and parasites), and links (values of 1, in red) represent interaction between them while 0 (in black) represent lack of observed interaction. The network contains gaps, indicating potential missing links.

# Load the dataset  
data(dat, package = "sabinaHSBM")

# Plot the simulated matrix  
plot\_interaction\_matrix(dat, order\_mat = FALSE)



# Preparing Input data for HSBM

The hsbm.input function pre-processes the dataset, creating cross-validation folds and edge lists required for modeling. Here, we use 5-fold cross-validation.

# Prepare input object  
myInput <- hsbm.input(  
 dat, # Binary bipartite matrix of observed links  
 n\_folds = 5 # Number of folds for cross-validation  
)

## [1] "Actual cross-validation rate is 0.199"  
## [2] "Actual cross-validation rate is 0.199"  
## [3] "Actual cross-validation rate is 0.199"  
## [4] "Actual cross-validation rate is 0.199"  
## [5] "Actual cross-validation rate is 0.202"

# Summarizes network characteristics  
summary(myInput)

## n\_rows n\_cols n\_links rows\_single\_link cols\_single\_link possible\_links  
## 1 100 14 277 1 0 1400

# Predicting Missing Links

The hsbm.predict function applies HSBM to predict link probabilities and group assignments in a network. The function works directly with the processed input created by hsbm.input. Here, we use the binary\_classifier method. This method focuses on predicting probabilities for currently unobserved links (0s). Use this method if you want to identify missing links (unobserved links likely to exist) in partially incomplete networks.

# Predict missing links using HSBM  
myPred <- hsbm.predict(  
 myInput, # Input data processed by hsbm.input()  
 iter = 1000, # Number of iterations ...  
 wait = 1000, # Number of iterations for MCMC equilibration  
 method = "binary\_classifier" # Prediction method  
)

## Computing predictions for fold 1   
## using binary\_classifier  
## Collect marginals  
## Gather data  
## Computing predictions for fold 2   
## using binary\_classifier  
## Collect marginals  
## Gather data  
## Computing predictions for fold 3   
## using binary\_classifier  
## Collect marginals  
## Gather data  
## Computing predictions for fold 4   
## using binary\_classifier  
## Collect marginals  
## Gather data  
## Computing predictions for fold 5   
## using binary\_classifier  
## Collect marginals  
## Gather data

Predicted link probabilities and group assignments are stored for each fold. Below, we extract the probabilities (p) and groups for fold 1.

# View probabilities for fold 1  
probabilities\_fold1 <- myPred$probs[[1]]  
head(probabilities\_fold1)

## v1 v2 p v1\_names v2\_names edge\_type  
## 1 0 101 0.0006322399 sp1 SPB reconstructed  
## 2 0 103 0.0376532629 sp1 SPD reconstructed  
## 3 0 104 0.0003646542 sp1 SPE reconstructed  
## 4 0 105 0.0004307817 sp1 SPF reconstructed  
## 5 0 107 0.0133247957 sp1 SPH reconstructed  
## 6 0 108 0.0151266385 sp1 SPI reconstructed

The group assignments provide the hierarchical clustering structure of nodes for each fold. Let’s extract and examine the group assignments for fold 1:

# View the group/block assignments for fold 1  
groups\_fold1 <- myPred$groups[[1]]  
  
# Filter one type of nodes (e.g., hosts, in columns)  
vnames <- colnames(myPred$data)  
groups\_fold1 <- groups\_fold1 %>% filter(names %in% vnames)  
g\_cols <- grep("^G", names(groups\_fold1))  
groups\_fold1[g\_cols] <- lapply(groups\_fold1[g\_cols], sort)  
  
print(groups\_fold1)

## nodes G1 G2 G3 G4 G5 names  
## 1 100 40 21 5 4 0 SPA  
## 2 101 40 21 5 4 0 SPB  
## 3 102 40 21 5 4 0 SPC  
## 4 103 81 21 5 4 0 SPD  
## 5 104 83 21 5 4 0 SPE  
## 6 105 83 21 5 4 0 SPF  
## 7 106 83 21 5 4 0 SPG  
## 8 107 94 21 5 4 0 SPH  
## 9 108 94 21 5 4 0 SPI  
## 10 109 94 21 5 4 0 SPJ  
## 11 110 94 21 5 4 0 SPK  
## 12 111 94 21 5 4 0 SPL  
## 13 112 94 21 5 4 0 SPM  
## 14 113 94 21 5 4 0 SPN

Hierarchical group assignments provide insight into how nodes (e.g., hosts) are organized across multiple levels. At the first level (G1), nodes are divided into specific groups, reflecting fine-scale patterns. Moving to higher levels (G2, G3, G4), these groups are progressively aggregated, revealing broader patterns and relationships or communities.

# Network Reconstruction

The hsbm.reconstructed function generates a reconstructed binary interaction matrix by combining predictions from all folds. The predicted matrix is binarized using a user-specified threshold.

# Network reconstruction  
myReconst <- hsbm.reconstructed(  
 myPred, # Predictions processed by hsbm.predict  
 rm\_documented = TRUE, # Use of documented entries during validation  
 threshold = "prc\_closest\_topright", # Binarization threshold  
 new\_matrix\_method = "average\_thresholded" # Combine fold predictions  
)

This output includes the averaged probability matrix, the final reconstructed matrix, and evaluation metrics. These results highlight the predicted interactions, showcasing the method’s ability to detect missing links effectively. Let’s explore some of the key outputs in detail.

# View the reconstructed network summary and evaluation metrics  
summary(myReconst)

## $`Reconstructed Network Metrics`  
## obs\_links unobs\_links pred\_links kept\_links spurious\_links missing\_links  
## 1 277 1123 284 277 0 7  
##   
## $`Evaluation Metrics`  
## mean\_RLRR mean\_auc mean\_aucpr mean\_yPRC mean\_prec mean\_sens mean\_spec  
## 1 0.4385714 0.9412932 0.5458532 0.04685102 0.7147156 0.4566883 0.9894924  
## mean\_ACC mean\_ERR mean\_tss  
## 1 0.9645219 0.03547811 0.4461807

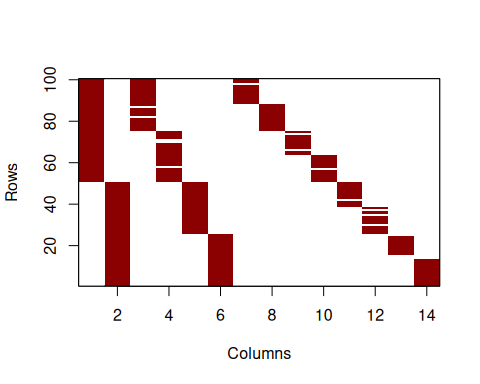
The summary provides the number of missing links, as well as key evaluation metrics, such as the retained link recovery rate (RLRR).

# View the top potential missing links  
top\_links\_df <- top\_links(myReconst,   
 n = 10,   
 edge\_type = "undocumented") # Type of edge to rank  
print(top\_links\_df)

## v1\_names v2\_names p sd  
## 1 sp98 SPB 0.17684724 0.10089205  
## 2 sp35 SPA 0.16673147 0.04770987  
## 3 sp47 SPA 0.12519538 0.07234850  
## 4 sp100 SPB 0.11086634 0.03863053  
## 5 sp57 SPE 0.10707457 0.08342920  
## 6 sp68 SPE 0.10070475 0.08920249  
## 7 sp75 SPE 0.09377506 0.08989643  
## 8 sp43 SPD 0.07024827 0.03467733  
## 9 sp78 SPN 0.06919414 0.05289722  
## 10 sp19 SPC 0.06377444 0.07026501

The top-links functions identifies the undocumented links most likely to be missing links in a network predicted by HSBM.

# View the reconstructed binary matrix  
plot\_interaction\_matrix(myReconst$new\_mat, order\_mat = FALSE)



This document demonstrates the use of the *sabinaHSBM* package for network reconstruction. By applying the binary\_classifier method, we showcased how missing links can be effectively identified and addressed in incomplete networks.

# Computing characteristics

# Show processing time and computer characteristics (optional)  
end\_time <- Sys.time()  
cat("The processing time of this script took: ", end\_time - start\_time, "minutes\n")

## The processing time of this script took: 5.188137 minutes

This analysis was performed on a Dynabook with the following characteristics:

* Processor (CPU): Intel Core i7-1165G7 @ 2.80GHz (11th Gen)
* Memory (RAM): 32 GB
* Operating System: Windows 11 Pro, Version 24H2
* R Version: 4.3.3