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.

# Loading Required Libraries

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)  
}

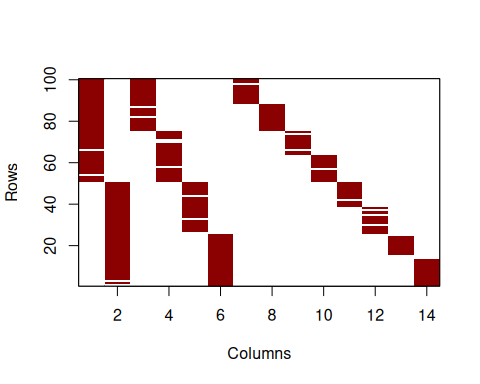
# Record starting time (optional)  
start\_time <- Sys.time()

# 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.

# 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 10-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  
)

## Warning in distribute\_zero\_sum\_folds(folds\_lst$com, folds\_lst$held\_pairs, :  
## Nothing to do in distribute\_zero\_sum\_folds. Return original held\_pairs.

## [1] "Actual cross-validation rate is 0.202"  
## [2] "Actual cross-validation rate is 0.199"  
## [3] "Actual cross-validation rate is 0.199"  
## [4] "Actual cross-validation rate is 0.202"  
## [5] "Actual cross-validation rate is 0.195"

# 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 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 2 101 0.0003951254 sp3 SPB reconstructed  
## 2 2 103 0.0041172083 sp3 SPD reconstructed  
## 3 2 104 0.0001921929 sp3 SPE reconstructed  
## 4 2 105 0.0003009598 sp3 SPF reconstructed  
## 5 2 106 0.0487234830 sp3 SPG reconstructed  
## 6 2 107 0.1591915255 sp3 SPH 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 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 24 9 2 1 3 SPA  
## 2 101 24 9 2 1 3 SPB  
## 3 102 24 9 2 1 3 SPC  
## 4 103 26 9 2 1 3 SPD  
## 5 104 30 9 2 1 3 SPE  
## 6 105 30 9 2 1 3 SPF  
## 7 106 30 9 2 1 3 SPG  
## 8 107 45 22 4 1 3 SPH  
## 9 108 45 22 4 1 3 SPI  
## 10 109 45 22 4 1 3 SPJ  
## 11 110 73 22 4 1 3 SPK  
## 12 111 104 22 4 1 3 SPL  
## 13 112 104 22 4 1 3 SPM  
## 14 113 104 22 4 1 3 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 evaluation and binarization  
 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 reconstruction summary and evaluation metrics  
summary(myReconst)

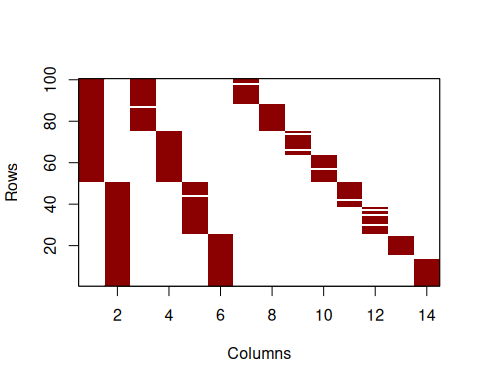
## $`Reconstructed Network Metrics`  
## obs\_links unobs\_links pred\_links kept\_links spurious\_links missing\_links  
## 1 277 1123 287 277 0 10  
##   
## $`Evaluation Metrics`  
## mean\_RLRR mean\_auc mean\_aucpr mean\_yPRC mean\_prec mean\_sens mean\_spec  
## 1 0.4606013 0.9390401 0.5189387 0.04685074 0.6260901 0.4787205 0.9841496  
## mean\_ACC mean\_ERR mean\_tss  
## 1 0.9604525 0.03954749 0.4628701

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 links  
# top\_links\_df <- top\_links(myReconst, n = 10, edge\_type = "undocumented")  
# print(top\_links\_df)

The top-links functions identifies the most probable 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

end\_time <- Sys.time()  
cat("The processing time of this script took: ", end\_time - start\_time, "mins\n")

## The processing time of this script took: 5.221989 mins

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