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_classifier` 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:

There are **two ways** to use the package:

- **UNIX users (native installation)** can run *sabinaHSBM* locally **if** their system includes:
  - R (version $\leq$ 4.0.4) with all required R packages (listed below)
  - Python $\leq$ with the `graph-tool` library (version $\leq$ 2.45)
- **All other users**, or those who prefer to avoid manual setup, can use the **ready-to-use Docker image**, which includes everything needed:
  - All R and Python dependencies
  - The *sabinaHSBM* package pre-installed and ready to use

Loading Required Libraries

Note:

The following instructions are **only required for UNIX users** running *sabinaHSBM* *natively* (outside Docker).

These users must ensure their system includes the required dependencies before proceeding.

```
# 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)
```

Install required R packages if not already available:

```
list.of.packages <- c(
  "dplyr",
  "parallel",
  "reshape2",
  "reticulate",
  "stringr",
  "tidyr",
  "ROCR",
  "data.table"
```

```r
)

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)
}
```

If you're using the **Docker image** (see *Supporting Information S3*) **you do not need to install any packages**, simply start the container and load the package in your R session with:

```r
library(sabinaHSBM)
```

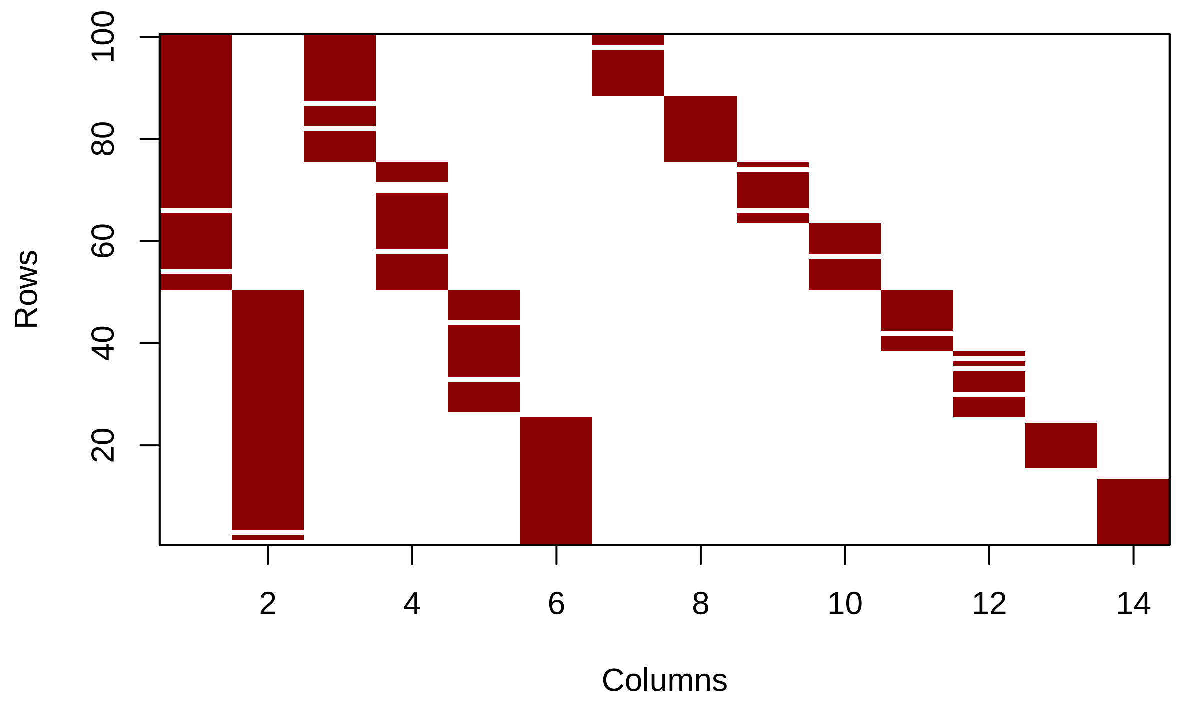
```r
# Record starting time (optional)
start_time <- Sys.time()
```

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 an observed interaction. The network contains gaps, indicating potential missing links.

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

```r
# 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
)
```

```
## 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 0.199"
## [5] "Actual cross-validation rate is 0.199"
```

```
# 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 obtain probabilities for all 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
  save_blocks = TRUE,  # Save group assignments
  save_pickle = FALSE, # Save results as pickle files,
  save_plots = FALSE  # Save hierarchical edge bundling plots
)
```

```
## Computing predictions for fold  1
## Computing predictions for fold  2
## Computing predictions for fold  3
## Computing predictions for fold  4
## Computing predictions for fold  5
```

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 0.0002772821      sp1      SPB reconstructed
## 2  0 103 0.0121795446      sp1      SPD reconstructed
## 3  0 104 0.0001441755      sp1      SPE reconstructed
## 4  0 105 0.0001423910      sp1      SPF reconstructed
## 5  0 107 0.0106033144      sp1      SPH reconstructed
## 6  0 108 0.0051413155      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., nodes 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 names
## 1    100 10 85  0  0   SPA
## 2    101 10 85  0  0   SPB
## 3    102 10 85  0  0   SPC
## 4    103 14 85  0  0   SPD
## 5    104 14 85  0  0   SPE
## 6    105 14 85  0  0   SPF
## 7    106 50 85  0  0   SPG
## 8    107 50 85  0  0   SPH
## 9    108 50 85  0  0   SPI
## 10   109 50 85  0  0   SPJ
## 11   110 50 85  0  0   SPK
## 12   111 50 85  0  0   SPL
## 13   112 50 85  0  0   SPM
## 14   113 91 85  0  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 transformed to binary values 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        288        277              0            11
##
## $`Evaluation Metrics`
##   mean_RLRR  mean_auc mean_aucpr  mean_yPRC mean_prec mean_sens mean_spec
## 1 0.4604545 0.9475875   0.542896 0.04685102 0.6824734 0.4785714 0.9882458
##    mean_ACC   mean_ERR  mean_tss
## 1 0.9643521 0.03564789 0.4668172
```

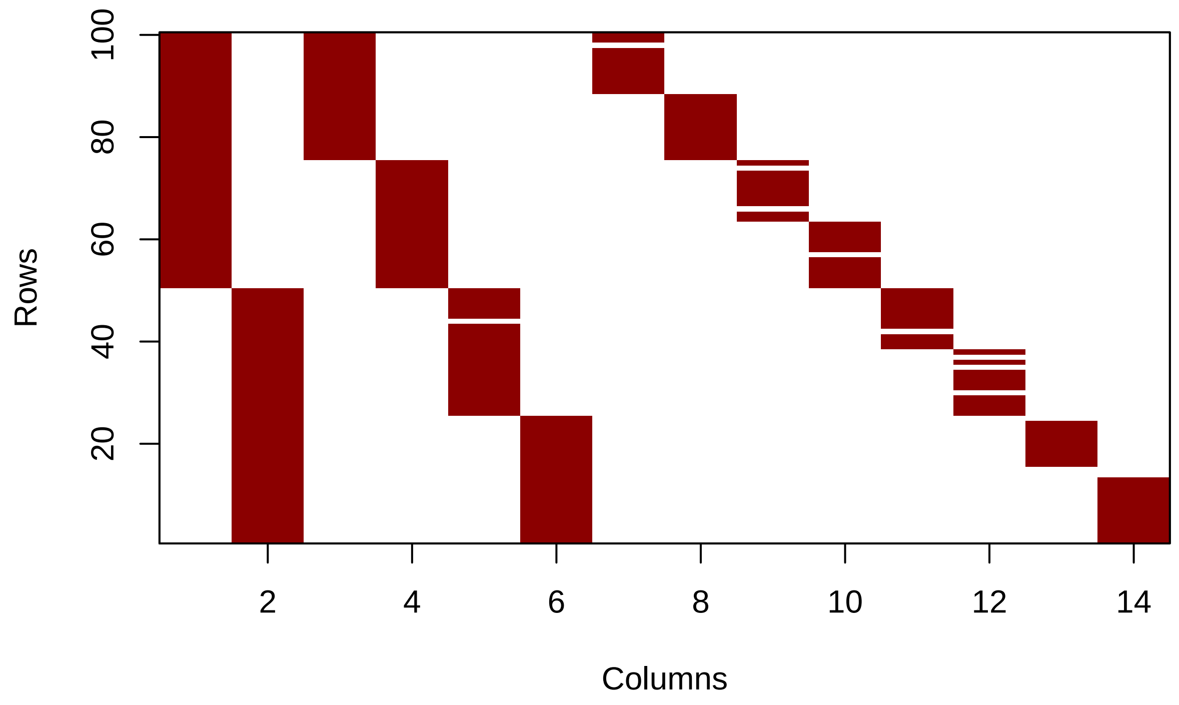
The summary provides the number of missing links, as well as key evaluation metrics, such as the retained link recovery rate (RLRR).

```
# 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 0.1792236 0.07483983
## 2     sp100      SPB 0.1790962 0.07023428
## 3      sp31      SPD 0.1538777 0.09903264
## 4      sp35      SPA 0.1464827 0.03143729
## 5      sp47      SPA 0.1426481 0.05397912
## 6      sp68      SPE 0.1419233 0.04741875
## 7      sp19      SPC 0.1402685 0.07132146
## 8      sp30      SPD 0.1337502 0.11024140
## 9      sp43      SPD 0.1286655 0.13036788
## 10     sp14      SPC 0.1145317 0.06988570
```

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()
```

```r
cat("The processing time of this script took: ", end_time - start_time, "minutes\n")
```

```
## The processing time of this script took:  3.939556 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