ggNetView manual documentation

Yue Liu

2026-01-09

Contents

Chapter 1

ggNetView manual documentation

1.1 Introdution

Network analysis has been widely applied in the **life sciences, microbiology, ecology, and agronomy** to quantify interactions among **genes, proteins, metabolites, microorganisms, and environmental factors**. These interactions collectively shape the functioning of biological and ecological systems. Despite its broad adoption, existing tools often face limitations in flexibility, customization, reproducibility, and the generation of publication-ready figures.

To address these challenges, **ggNetView** was developed as an **R package** that integrates **ggplot2, ggraph, and tidygraph** within the **Grammar of Graphics framework**, enabling fully reproducible and highly customizable network visualizations. The package provides deterministic layout algorithms, comprehensive topological analyses, modular coloring schemes, hierarchical annotations, and consistent theming, ensuring standardized and interpretable graphical output.

Although initially designed for applications in **soil science and microbial ecology**, **ggNetView** is broadly applicable to network analyses in `molecular biology`, including **WGCNA and protein–protein interaction (PPI) networks**. By lowering technical barriers in network construction and visualization, **ggNetView** enables researchers across disciplines to efficiently produce reproducible, publication-quality network figures.

1.2 Installation

First, install the required dependencies

```
# install.packages("BiocManager")
BiocManager::install("WGCNA")

# install.packages("remotes")
remotes::install_github("alserglab/mascarade")
remotes::install_github("zdk123/SpiecEasi")
```

and then install ggNetView.

```
# install.packages("devtools")
devtools::install_github("Jiawang1209/ggNetView")

# install.packages("pak")
pak::pak("Jiawang1209/ggNetView")
```

1.3 Citation

If you use ggNetView in your research, please cite:

```
Yue Liu, Chao Wang (2025). ggNetView: An R package for complex biological and ecologica
https://github.com/Jiawang1209/ggNetView
```

1.4 Source Code

The source code for ggNetView is available in the ggNetView repository.

https://github.com/Jiawang1209/ggNetView

1.5 Contact

- Email: Jiawang1209@163.com

Chapter 2

Build graph object

Basic workflow of **ggNetView**

1. Build a **graph object**
2. Understand and manipulate the **graph object**
3. Visualize the network using **layout algorithms**
4. Retrieve **topology** of network

Load R Package

```
library(tidyverse)
library(ggNetView)
```

2.1 Build graph from matrix

Example data

```
# Access built-in example datasets in ggNetView

# Raw ASV or OTU table
data("otu_tab")
dim(otu_tab)
```

```
## [1] 2859   18
```

```
otu_tab[1:5, 1:5]
```

```
##        KO1  KO2  KO3  KO4  KO5
## ASV_1 1113 1968  816 1372 1062
## ASV_2 1922 1227 2355 2218 2885
## ASV_3  568  460  899  902 1226
## ASV_4 1433  400  535  759 1287
## ASV_6  882  673  819  888 1475
```

```
# Rarefied ASV or OTU table
data("otu_rare")
dim(otu_rare)
```

```
## [1] 2859   18
```

```
otu_rare[1:5, 1:5]
```

```
##        KO1  KO2  KO3  KO4  KO5
## ASV_1  992 1636  604 1084  806
## ASV_2 1725 1018 1814 1743 2196
## ASV_3  520  389  687  701  932
## ASV_4 1280  328  425  580 1004
## ASV_6  794  557  633  706 1142
```

```
# Relative abundance table of rarefied ASVs or OTUs
data("otu_rare_relative")
dim(otu_rare_relative)
```

```
## [1] 2859   18
```

```
otu_rare_relative[1:5, 1:5]
```

```
##              KO1        KO2        KO3        KO4        KO5
## ASV_1 0.03306667 0.05453333 0.02013333 0.03613333 0.02686667
## ASV_2 0.05750000 0.03393333 0.06046667 0.05810000 0.07320000
## ASV_3 0.01733333 0.01296667 0.02290000 0.02336667 0.03106667
## ASV_4 0.04266667 0.01093333 0.01416667 0.01933333 0.03346667
## ASV_6 0.02646667 0.01856667 0.02110000 0.02353333 0.03806667
```

```
# Taxonomic annotation table of ASVs or OTUs
data("tax_tab")
dim(tax_tab)
```

```
## [1] 2859    8
```

```
tax_tab[1:5, 1:5]
```

```
## # A tibble: 5 x 5
##   OTUID  Kingdom  Phylum          Class          Order
##   <chr>  <chr>    <chr>           <chr>          <chr>
## 1 ASV_2  Archaea  Thaumarchaeota  Unassigned     Nitrososphaerales
## 2 ASV_3  Bacteria Verrucomicrobia Spartobacteria Unassigned
## 3 ASV_31 Bacteria Actinobacteria  Actinobacteria Actinomycetales
## 4 ASV_27 Archaea  Thaumarchaeota  Unassigned     Nitrososphaerales
## 5 ASV_9  Bacteria Unassigned      Unassigned     Unassigned
```

Build graph object

```
graph_obj <- build_graph_from_mat(
  mat = otu_rare_relative,
  transfrom.method = "none", #  based your input data
  r.threshold = 0.7,
  p.threshold = 0.05,
  method = "WGCNA",
  cor.method = "pearson",
  proc = "Bonferroni",
  module.method = "Fast_greedy",
  node_annotation = tax_tab,
  top_modules = 15,
  seed = 1115
)

graph_obj
```

```
## # A tbl_graph: 213 nodes and 844 edges
## #
## # An undirected simple graph with 29 components
## #
## # Node Data: 213 x 14 (active)
##    name    modularity modularity2 modularity3 Modularity Degree Strength Kingdom
##    <chr>   <fct>      <ord>       <chr>       <ord>       <dbl>    <dbl> <chr>
##  1 ASV_649 5          5           5           5              27     26.5 Bacter~
##   2 ASV_705 5          5           5           5              27      26.5 Bacter~
##   3 ASV_12~ 5          5           5           5              27      26.5 Bacter~
##   4 ASV_13~ 5          5           5           5              27      26.5 Bacter~
##   5 ASV_14~ 5          5           5           5              27      26.5 Bacter~
##   6 ASV_14~ 5          5           5           5              27      26.5 Bacter~
##   7 ASV_24~ 5          5           5           5              27      26.5 Bacter~
##   8 ASV_25~ 5          5           5           5              27      26.4 Bacter~
##   9 ASV_28~ 5          5           5           5              27      26.5 Bacter~
##  10 ASV_28~ 5          5           5           5              27      26.5 Bacter~
## # i 203 more rows
## # i 6 more variables: Phylum <chr>, Class <chr>, Order <chr>, Family <chr>,
## #   Genus <chr>, Species <chr>
## #
## # Edge Data: 844 x 5
##    from    to weight correlation corr_direction
##   <int> <int>  <dbl>       <dbl> <chr>
## 1   194   195  0.959       0.959 Positive
## 2   185   208  0.954       0.954 Positive
## 3   185   213  0.957       0.957 Positive
## # i 841 more rows
```

2.2 Build graph from data frame

Example data

```
# Access built-in example datasets in ggNetView
data("ppi_example")
df = ppi_example$ppi
head(df)
```

```
##   from  to    weight
## 1   A1 D40  9.306533
## 2   A2 D39 11.783920
## 3   A3 D38 23.005025
## 4   A4 D37  7.412060
## 5   A5 D36 18.778894
## 6   A6 D35 16.592965
```

```
node_annotation = ppi_example$annotation
head(node_annotation)
```

```
##   node group
## 1   A1     A
## 2   A2     A
## 3   A3     A
## 4   A4     A
## 5   A5     A
## 6   A6     A
```

Build graph object

```
graph_obj_from_df <- build_graph_from_df(
  df = df,
  node_annotation = node_annotation,
  directed = F,
  module.method = "Fast_greedy",
  top_modules = 15,
  seed = 1115
)

graph_obj_from_df
```

```
## # A tbl_graph: 100 nodes and 50 edges
## #
## # An unrooted forest with 50 trees
## #
## # Node Data: 100 x 9 (active)
##    name  group modularity modularity2 modularity3 Modularity Degree Segree
##    <chr> <chr> <fct>      <fct>       <chr>       <fct>       <dbl>  <dbl>
##  1 C13   C     1          1           1           1               1      1
##  2 C28   C     1          1           1           1               1      1
##  3 C2    C     10         10          10          10              1      1
##  4 D9    D     10         10          10          10              1      1
##  5 A3    A     11         11          11          11              1      1
##  6 D38   D     11         11          11          11              1      1
##  7 B12   B     12         12          12          12              1      1
##  8 D19   D     12         12          12          12              1      1
##  9 A1    A     13         13          13          13              1      1
## 10 D40   D     13         13          13          13              1      1
## # i 90 more rows
## # i 1 more variable: Strength <dbl>
## #
## # Edge Data: 50 x 4
##    from    to weight correlation
##   <int> <int>  <dbl>       <dbl>
## 1     9    10   45.2        45.2
## 2    15    16   50.6         50.6
## 3     5     6   37.8         37.8
## # i 47 more rows
```

```
graph_obj_from_df2 <- build_graph_from_df(
  df = df,
  node_annotation = NULL,
  directed = F,
  module.method = "Fast_greedy",
  top_modules = 15,
  seed = 1115
)

graph_obj_from_df2
```

```
## # A tbl_graph: 100 nodes and 50 edges
## #
## # An unrooted forest with 50 trees
## #
## # Node Data: 100 x 8 (active)
##    name  modularity modularity2 modularity3 Modularity Degree Segree Strength
##    <chr> <fct>      <fct>       <chr>       <fct>       <dbl>  <dbl>    <dbl>
##  1 C13   1          1           1           1               1      1     26.7
##  2 C28   1          1           1           1               1      1     26.7
##  3 C2    10         10          10          10              1      1     37.4
##  4 D9    10         10          10          10              1      1     37.4
##  5 A3    11         11          11          11              1      1     37.8
##  6 D38   11         11          11          11              1      1     37.8
##  7 B12   12         12          12          12              1      1     41.3
##  8 D19   12         12          12          12              1      1     41.3
##  9 A1    13         13          13          13              1      1     45.2
## 10 D40   13         13          13          13              1      1     45.2
## # i 90 more rows
## #
## # Edge Data: 50 x 4
##    from    to weight correlation
##   <int> <int>  <dbl>       <dbl>
## 1     9    10   45.2        45.2
## 2    15    16   50.6        50.6
## 3     5     6   37.8        37.8
## # i 47 more rows
```

2.3 Build graph from data frame with module

Example data

```
# Access built-in example datasets in ggNetView
data("ppi_module")
df = ppi_module$ppi
head(df)
```

```
##   from  to    weight
## 1   A1 D40  9.306533
## 2   A2 D39 11.783920
## 3   A3 D38 23.005025
## 4   A4 D37  7.412060
## 5   A5 D36 18.778894
## 6   A6 D35 16.592965
```

```
node_annotation = ppi_module$annotation
head(node_annotation)
```

```
##   node Modularity
## 1   A1          A
## 2   A2          A
## 3   A3          A
## 4   A4          A
## 5   A5          A
## 6   A6          A
```

Build graph object

```
graph_obj_from_module <- build_graph_from_module(
  df = df,
  node_annotation = node_annotation,
  directed = F,
  top_modules = 15,
  seed = 1115
)

graph_obj_from_module
```

```
## # A tbl_graph: 100 nodes and 50 edges
## #
## # An unrooted forest with 50 trees
## #
## # Node Data: 100 x 7 (active)
##    name  Modularity modularity2 modularity3 Degree Segree Strength
##    <chr> <ord>      <ord>       <chr>        <dbl>  <dbl>    <dbl>
##  1 D1    D          D           D                1      1     37.1
##  2 D2    D          D           D                1      1     63.9
##  3 D3    D          D           D                1      1     54.6
##  4 D4    D          D           D                1      1     61.7
##  5 D5    D          D           D                1      1     36.1
##  6 D6    D          D           D                1      1     71.0
##  7 D7    D          D           D                1      1     27.9
##  8 D8    D          D           D                1      1     34.6
##  9 D9    D          D           D                1      1     37.4
## 10 D10   D          D           D                1      1     63.2
## # i 90 more rows
## #
## # Edge Data: 50 x 4
##    from    to weight correlation
##   <int> <int>  <dbl>       <dbl>
## 1    40    91   45.2        45.2
## 2    39    92   50.6        50.6
## 3    38    93   37.8        37.8
## # i 47 more rows
```

2.4 Build graph from adjacency matrix

Example data

```
data("adjacency_matrix_example")
dim(adjacency_matrix_example)
```

```
## [1] 2859 2859
```

```
adjacency_matrix_example[1:5, 1:5]
```

```
##       ASV_1     ASV_2 ASV_3 ASV_4     ASV_6
## ASV_1     0 0.0000000     0     0 0.0000000
## ASV_2     0 0.0000000     0     0 0.8947427
## ASV_3     0 0.0000000     0     0 0.0000000
## ASV_4     0 0.0000000     0     0 0.0000000
## ASV_6     0 0.8947427     0     0 0.0000000
```

```r
data("tax_tab")
dim(tax_tab)
```

```
## [1] 2859    8
```

```r
tax_tab[1:5, 1:5]
```

```
## # A tibble: 5 x 5
##   OTUID  Kingdom  Phylum          Class          Order
##   <chr>  <chr>    <chr>           <chr>          <chr>
## 1 ASV_2  Archaea  Thaumarchaeota  Unassigned     Nitrososphaerales
## 2 ASV_3  Bacteria Verrucomicrobia Spartobacteria Unassigned
## 3 ASV_31 Bacteria Actinobacteria  Actinobacteria Actinomycetales
## 4 ASV_27 Archaea  Thaumarchaeota  Unassigned     Nitrososphaerales
## 5 ASV_9  Bacteria Unassigned      Unassigned     Unassigned
```

Build graph object

```r
graph_obj_adj <- build_graph_from_adj_mat(
  adjacency_matrix = adjacency_matrix_example,
  module.method = "Fast_greedy",
  node_annotation = tax_tab,
  top_modules = 15,
  seed = 1115
)

graph_obj_adj
```

```
## # A tbl_graph: 2049 nodes and 9602 edges
## #
## # An undirected simple graph with 100 components
## #
## # Node Data: 2,049 x 14 (active)
##    name    modularity modularity2 modularity3 Modularity Degree Strength Kingdom
##    <chr>   <fct>      <ord>       <chr>       <ord>       <dbl>    <dbl> <chr>
##  1 ASV_916 1          1           1           1              58     50.5 Bacter~
##  2 ASV_777 1          1           1           1              58     48.7 Bacter~
##  3 ASV_606 1          1           1           1              55     45.8 Bacter~
##  4 ASV_740 1          1           1           1              54     47.2 Bacter~
##  5 ASV_14~ 1          1           1           1              54     44.5 Bacter~
##  6 ASV_23~ 1          1           1           1              54     47.4 Bacter~
##  7 ASV_15~ 1          1           1           1              52     45.3 Bacter~
##  8 ASV_24~ 1          1           1           1              52     43.0 Bacter~
##  9 ASV_19~ 1          1           1           1              52     43.0 Bacter~
## 10 ASV_568 1          1           1           1              51     45.1 Bacter~
## # i 2,039 more rows
## # i 6 more variables: Phylum <chr>, Class <chr>, Order <chr>, Family <chr>,
## #   Genus <chr>, Species <chr>
## #
## # Edge Data: 9,602 x 5
##    from    to weight correlation corr_direction
##   <int> <int>  <dbl>       <dbl> <chr>
## 1  1771  1825  0.793       0.793 Positive
## 2   594   597  0.895       0.895 Positive
## 3   588   597  0.864       0.864 Positive
## # i 9,599 more rows
```

2.5 Build graph from double matrix

Example data

```
data("BASV_tab")
dim(BASV_tab)
```

```
## [1] 50 10
```

```
BASV_tab[1:5, 1:5]
```

```
##         Sample1   Sample2   Sample3   Sample4   Sample5
## BASV1  7.769249 14.676091  6.822246 11.806957 12.086254
## BASV2 10.771846  7.749385 14.966987  7.486507 12.645461
## BASV3 11.746937 10.023547 10.025906 11.967562  8.748440
## BASV4  8.210859  4.408296 10.070862  8.954202 14.639298
## BASV5  9.833271  9.776280 10.465463  8.380012  8.905738
```

```
data("FASV_tab")
dim(FASV_tab)
```

```
## [1] 50 10
```

```
FASV_tab[1:5, 1:5]
```

```
##         Sample1   Sample2   Sample3   Sample4   Sample5
## FASV1  8.195641  7.715472 13.247401 10.419618 11.961410
## FASV2 10.032701 12.427388  7.693471 10.711077 12.157527
## FASV3 10.109549 11.891917  9.607849  8.256637  9.989264
## FASV4  9.817056  8.038730  5.519155  7.601228 12.688531
## FASV5 13.370263  9.795554 13.639041 11.071635  8.176218
```

```
data("double_mat_node_df")
dim(double_mat_node_df)
```

```
## [1] 100   2
```

```
head(double_mat_node_df)
```

```
##    name      type
## 1 BASV1 Bacterial
## 2 BASV2 Bacterial
## 3 BASV3 Bacterial
## 4 BASV4 Bacterial
## 5 BASV5 Bacterial
## 6 BASV6 Bacterial
```

Build graph object

```
graph_obj_double_mat <- build_graph_from_double_mat(
  mat1 = BASV_tab,
  mat2 = FASV_tab,
   module.method = "Fast_greedy",
  node_annotation = double_mat_node_df,
  top_modules = 15,
  seed = 1115
)
```

```
## The max module in network is 4 we use the 4  modules for next analysis
```

```
graph_obj_double_mat
```

```
## # A tbl_graph: 100 nodes and 2500 edges
## #
## # A bipartite simple graph with 1 component
## #
## # Node Data: 100 x 9 (active)
##    name   type      modularity modularity2 modularity3 Modularity Degree Segree
##    <chr>  <chr>     <fct>      <fct>       <chr>       <fct>       <dbl>  <dbl>
##  1 BASV3  Bacterial 1          1           1           1              50     50
##  2 BASV6  Bacterial 1          1           1           1              50     50
##  3 BASV8  Bacterial 1          1           1           1              50     50
##  4 BASV13 Bacterial 1          1           1           1              50     50
##  5 BASV17 Bacterial 1          1           1           1              50     50
##  6 BASV19 Bacterial 1          1           1           1              50     50
##  7 BASV25 Bacterial 1          1           1           1              50     50
##  8 BASV26 Bacterial 1          1           1           1              50     50
##  9 BASV27 Bacterial 1          1           1           1              50     50
## 10 BASV31 Bacterial 1          1           1           1              50     50
## # i 90 more rows
## # i 1 more variable: Strength <dbl>
## #
## # Edge Data: 2,500 x 4
##    from    to weight correlation
##   <int> <int>  <dbl>       <dbl>
## 1    15    57  0.338      -0.338
## 2    57    90  0.648       0.648
## 3    16    57  0.162       0.162
## # i 2,497 more rows
```

Chapter 3

Random Matrix Theory (RMT)–based random network

When a numerical matrix is used as input, it is necessary to filter and screen the resulting correlation matrix. Empirical thresholds are commonly adopted, usually between 0.6 to 0.85; however, these thresholds are largely subjective.

Compared with empirical thresholding, random matrix theory (RMT) provides a data-driven and objective approach for determining correlation thresholds, effectively separating true biological signals from random noise and improving the robustness and comparability of ecological networks.

3.1 Random Matrix Theory (RMT)

Load R Package

```r
library(tidyverse)
library(ggNetView)
```

3.2 Example data

Example data

```
# Access built-in example datasets in ggNetView

# Relative abundance table of rarefied ASVs or OTUs
data("otu_rare_relative")
dim(otu_rare_relative)
```

```
## [1] 2859   18
```

```
otu_rare_relative[1:5, 1:5]
```

```
##              K01        K02        K03        K04        K05
## ASV_1 0.03306667 0.05453333 0.02013333 0.03613333 0.02686667
## ASV_2 0.05750000 0.03393333 0.06046667 0.05810000 0.07320000
## ASV_3 0.01733333 0.01296667 0.02290000 0.02336667 0.03106667
## ASV_4 0.04266667 0.01093333 0.01416667 0.01933333 0.03346667
## ASV_6 0.02646667 0.01856667 0.02110000 0.02353333 0.03806667
```

```
# Taxonomic annotation table of ASVs or OTUs
data("tax_tab")
dim(tax_tab)
```

```
## [1] 2859    8
```

```
tax_tab[1:5, 1:5]
```

```
## # A tibble: 5 x 5
##   OTUID  Kingdom  Phylum          Class          Order
##   <chr>  <chr>    <chr>           <chr>          <chr>
## 1 ASV_2  Archaea  Thaumarchaeota  Unassigned     Nitrososphaerales
## 2 ASV_3  Bacteria Verrucomicrobia Spartobacteria Unassigned
## 3 ASV_31 Bacteria Actinobacteria  Actinobacteria Actinomycetales
## 4 ASV_27 Archaea  Thaumarchaeota  Unassigned     Nitrososphaerales
## 5 ASV_9  Bacteria Unassigned      Unassigned     Unassigned
```

3.3 Compute RMT threshold

```r
out <- ggNetView_RMT(mat = otu_rare_relative,
                     transfrom.method = "none",
                     method = "WGCNA",
                     cor.method = "pearson",
                     unfold.method = "gaussian",
                     bandwidth = "nrd0",
                     nr.fit.points = 51,
                     discard.outliers = TRUE,
                     discard.zeros = TRUE,
                     min.mat.dim = 40,
                     max.ev.spacing = 3,
                     save_plots = FALSE,
                     out_dir = "RMT_plots",
                     verbose = TRUE,
                     seed = 1115
                     )
```

```
## [Info] Matrix dimension: 2859 x 2859

## [Info] #non-zeros: 8173831 | Sparseness: 0.0000

##   [Scan]   1/51 | threshold = 0

##          Effective dimension: 2859

##   [Scan]   2/51 | threshold = 0.02

##          Effective dimension: 2859

##   [Scan]   3/51 | threshold = 0.04

##          Effective dimension: 2859

##   [Scan]   4/51 | threshold = 0.06

##          Effective dimension: 2859

##   [Scan]   5/51 | threshold = 0.08

##          Effective dimension: 2859
##   [Scan]   6/51 | threshold = 0.1

##          Effective dimension: 2859

##   [Scan]   7/51 | threshold = 0.12

##          Effective dimension: 2859

##   [Scan]   8/51 | threshold = 0.14

##          Effective dimension: 2859

##   [Scan]   9/51 | threshold = 0.16

##          Effective dimension: 2859

##   [Scan]  10/51 | threshold = 0.18

##          Effective dimension: 2859

##   [Scan]  11/51 | threshold = 0.2

##          Effective dimension: 2859

##   [Scan]  12/51 | threshold = 0.22

##          Effective dimension: 2859

##   [Scan]  13/51 | threshold = 0.24

##          Effective dimension: 2859

##   [Scan]  14/51 | threshold = 0.26

##          Effective dimension: 2859

##   [Scan]  15/51 | threshold = 0.28

##          Effective dimension: 2859

##   [Scan]  16/51 | threshold = 0.3
##          Effective dimension: 2859
##   [Scan]  17/51 | threshold = 0.32
##          Effective dimension: 2859
##   [Scan]  18/51 | threshold = 0.34
##          Effective dimension: 2859
##   [Scan]  19/51 | threshold = 0.36
##          Effective dimension: 2859
##   [Scan]  20/51 | threshold = 0.38
##          Effective dimension: 2859
##   [Scan]  21/51 | threshold = 0.4
##          Effective dimension: 2859
##   [Scan]  22/51 | threshold = 0.42
##          Effective dimension: 2859
##   [Scan]  23/51 | threshold = 0.44
##          Effective dimension: 2859
##   [Scan]  24/51 | threshold = 0.46
##          Effective dimension: 2859
##   [Scan]  25/51 | threshold = 0.48
##          Effective dimension: 2859
##   [Scan]  26/51 | threshold = 0.5
##          Effective dimension: 2859
##   [Scan]  27/51 | threshold = 0.52

##          Effective dimension: 2859

##   [Scan]  28/51 | threshold = 0.54

##          Effective dimension: 2859

##   [Scan]  29/51 | threshold = 0.56

##          Effective dimension: 2859

##   [Scan]  30/51 | threshold = 0.58

##          Effective dimension: 2859

##   [Scan]  31/51 | threshold = 0.6

##          Effective dimension: 2859

##   [Scan]  32/51 | threshold = 0.62

##          Effective dimension: 2859

##   [Scan]  33/51 | threshold = 0.64

##          Effective dimension: 2857

##   [Scan]  34/51 | threshold = 0.66

##          Effective dimension: 2855

##   [Scan]  35/51 | threshold = 0.68

##          Effective dimension: 2842

##   [Scan]  36/51 | threshold = 0.7

##          Effective dimension: 2800

##   [Scan]  37/51 | threshold = 0.72
##          Effective dimension: 2705
##   [Scan]  38/51 | threshold = 0.74
##          Effective dimension: 2546
##   [Scan]  39/51 | threshold = 0.76
##          Effective dimension: 2333
##   [Scan]  40/51 | threshold = 0.78
##          Effective dimension: 2049
##   [Scan]  41/51 | threshold = 0.8
##          Effective dimension: 1735
##   [Scan]  42/51 | threshold = 0.82
##          Effective dimension: 1396
##   [Scan]  43/51 | threshold = 0.84
##          Effective dimension: 1082
##   [Scan]  44/51 | threshold = 0.86
##          Effective dimension: 825
##   [Scan]  45/51 | threshold = 0.88
##          Effective dimension: 600
##   [Scan]  46/51 | threshold = 0.9
##          Effective dimension: 454
##   [Scan]  47/51 | threshold = 0.92
##          Effective dimension: 324
##   [Scan]  48/51 | threshold = 0.94

##          Effective dimension: 215

##   [Scan]  49/51 | threshold = 0.96

##          Effective dimension: 157

##   [Scan]  50/51 | threshold = 0.98

##          Effective dimension: 79

##   [Scan]  51/51 | threshold = 1

##          Effective dimension: 2

##          Too small. Stop scanning.
```

```
out$chosen_threshold
```

```
## [1] 0.82
```

```
out2 <- ggNetView_RMT(mat = otu_rare_relative,
                     transfrom.method = "none",
                     method = "cor",
                     cor.method = "pearson",
                     unfold.method = "gaussian",
                     bandwidth = "nrd0",
                     nr.fit.points = 51,
                     discard.outliers = TRUE,
                     discard.zeros = TRUE,
                     min.mat.dim = 40,
                     max.ev.spacing = 3,
                     save_plots = FALSE,
                     out_dir = "RMT_plots",
                     verbose = TRUE,
                     seed = 1115
                     )
```

```
## [Info] Matrix dimension: 2859 x 2859

## [Info] #non-zeros: 8173761 | Sparseness: 0.0000
##   [Scan]   1/51 | threshold = 0

##          Effective dimension: 2859

##   [Scan]   2/51 | threshold = 0.02

##          Effective dimension: 2859

##   [Scan]   3/51 | threshold = 0.04

##          Effective dimension: 2859

##   [Scan]   4/51 | threshold = 0.06

##          Effective dimension: 2859

##   [Scan]   5/51 | threshold = 0.08

##          Effective dimension: 2859

##   [Scan]   6/51 | threshold = 0.1

##          Effective dimension: 2859

##   [Scan]   7/51 | threshold = 0.12

##          Effective dimension: 2859

##   [Scan]   8/51 | threshold = 0.14

##          Effective dimension: 2859

##   [Scan]   9/51 | threshold = 0.16

##          Effective dimension: 2859

##   [Scan]  10/51 | threshold = 0.18

##          Effective dimension: 2859

##   [Scan]  11/51 | threshold = 0.2
##          Effective dimension: 2859

##   [Scan]  12/51 | threshold = 0.22

##          Effective dimension: 2859

##   [Scan]  13/51 | threshold = 0.24

##          Effective dimension: 2859

##   [Scan]  14/51 | threshold = 0.26

##          Effective dimension: 2859

##   [Scan]  15/51 | threshold = 0.28

##          Effective dimension: 2859

##   [Scan]  16/51 | threshold = 0.3

##          Effective dimension: 2859

##   [Scan]  17/51 | threshold = 0.32

##          Effective dimension: 2859

##   [Scan]  18/51 | threshold = 0.34

##          Effective dimension: 2859

##   [Scan]  19/51 | threshold = 0.36

##          Effective dimension: 2859

##   [Scan]  20/51 | threshold = 0.38

##          Effective dimension: 2859

##   [Scan]  21/51 | threshold = 0.4

##          Effective dimension: 2859
##   [Scan]  22/51 | threshold = 0.42
##          Effective dimension: 2859
##   [Scan]  23/51 | threshold = 0.44
##          Effective dimension: 2859
##   [Scan]  24/51 | threshold = 0.46
##          Effective dimension: 2859
##   [Scan]  25/51 | threshold = 0.48
##          Effective dimension: 2859
##   [Scan]  26/51 | threshold = 0.5
##          Effective dimension: 2859
##   [Scan]  27/51 | threshold = 0.52
##          Effective dimension: 2859
##   [Scan]  28/51 | threshold = 0.54
##          Effective dimension: 2859
##   [Scan]  29/51 | threshold = 0.56
##          Effective dimension: 2859
##   [Scan]  30/51 | threshold = 0.58
##          Effective dimension: 2859
##   [Scan]  31/51 | threshold = 0.6
##          Effective dimension: 2859
##   [Scan]  32/51 | threshold = 0.62
##          Effective dimension: 2859

##   [Scan]  33/51 | threshold = 0.64

##          Effective dimension: 2857

##   [Scan]  34/51 | threshold = 0.66

##          Effective dimension: 2855

##   [Scan]  35/51 | threshold = 0.68

##          Effective dimension: 2842

##   [Scan]  36/51 | threshold = 0.7

##          Effective dimension: 2800

##   [Scan]  37/51 | threshold = 0.72

##          Effective dimension: 2705

##   [Scan]  38/51 | threshold = 0.74

##          Effective dimension: 2546

##   [Scan]  39/51 | threshold = 0.76

##          Effective dimension: 2333

##   [Scan]  40/51 | threshold = 0.78

##          Effective dimension: 2049

##   [Scan]  41/51 | threshold = 0.8

##          Effective dimension: 1735

##   [Scan]  42/51 | threshold = 0.82

##          Effective dimension: 1396
## [Scan] 43/51 | threshold = 0.84

##          Effective dimension: 1082

## [Scan] 44/51 | threshold = 0.86

##          Effective dimension: 825

## [Scan] 45/51 | threshold = 0.88

##          Effective dimension: 600

## [Scan] 46/51 | threshold = 0.9

##          Effective dimension: 454

## [Scan] 47/51 | threshold = 0.92

##          Effective dimension: 324

## [Scan] 48/51 | threshold = 0.94

##          Effective dimension: 215

## [Scan] 49/51 | threshold = 0.96

##          Effective dimension: 157

## [Scan] 50/51 | threshold = 0.98

##          Effective dimension: 79

## [Scan] 51/51 | threshold = 1

##          Effective dimension: 10

##          Too small. Stop scanning.
```

```
out2$chosen_threshold
```

```
## [1] 0.82
```

3.4 Build graph from matrix based on RMT-threshold

Build graph object

```
graph_obj <- build_graph_from_mat(
  mat = otu_rare_relative,
  transfrom.method = "none", #  based your input data
  r.threshold = out$chosen_threshold,
  p.threshold = 0.05,
  method = "WGCNA",
  cor.method = "pearson",
  proc = "Bonferroni",
  module.method = "Fast_greedy",
  node_annotation = tax_tab,
  top_modules = 15,
  seed = 1115
)

graph_obj
```

```
## # A tbl_graph: 213 nodes and 844 edges
## #
## # An undirected simple graph with 29 components
## #
## # Node Data: 213 x 14 (active)
##    name    modularity modularity2 modularity3 Modularity Degree Strength Kingdom
##    <chr>   <fct>      <ord>       <chr>       <ord>       <dbl>    <dbl> <chr>
##  1 ASV_649 5          5           5           5              27     26.5 Bacter~
##  2 ASV_705 5          5           5           5              27     26.5 Bacter~
##  3 ASV_12~ 5          5           5           5              27     26.5 Bacter~
##  4 ASV_13~ 5          5           5           5              27     26.5 Bacter~
##  5 ASV_14~ 5          5           5           5              27     26.5 Bacter~
##  6 ASV_14~ 5          5           5           5              27     26.5 Bacter~
##  7 ASV_24~ 5          5           5           5              27     26.5 Bacter~
##  8 ASV_25~ 5          5           5           5              27     26.4 Bacter~
##  9 ASV_28~ 5          5           5           5              27     26.5 Bacter~
## 10 ASV_28~ 5          5           5           5              27     26.5 Bacter~
## # i 203 more rows
## # i 6 more variables: Phylum <chr>, Class <chr>, Order <chr>, Family <chr>,
## #   Genus <chr>, Species <chr>
## #
## # Edge Data: 844 x 5
##    from    to weight correlation corr_direction
##   <int> <int>  <dbl>       <dbl> <chr>
## 1   194   195  0.959       0.959 Positive
## 2   185   208  0.954       0.954 Positive
## 3   185   213  0.957       0.957 Positive
## # i 841 more rows
```

Chapter 4

Get network information

Once the graph object is constructed, basic network information can be retrieved, and subnetworks can be extracted.

Load R Package

```r
library(tidyverse)
library(ggNetView)
```

4.1 Full-network information

4.2 Sub-network (modularity) information

4.3 Sub-network (sample) information

Chapter 5

Extract subgraph

5.1 Extract subgraph by module

5.2 Extract subgraph by sample

Chapter 6

Network layout

6.1 Gephi layout

6.2 Fruchterman–Reingold force-directed layout

6.3 Diamond layout

6.4 Kk layout

6.5 Nicley layout

6.6 Multrings layout

6.7 Petal layout

6.8 Circle layout

6.9 Circle outline layout

6.10 Diamond outline layout

6.11 Grid layout

6.12 Heart_centered layout

6.13 Lgl layout

6.14 Randomly layout

6.15 Rectangle layout

Chapter 7

Network topology information

7.1 Get network topology information

7.2 Get network topology information with matrix

7.3 Get network topology information by parallel

7.4 Get network topology information with matrix by parallel

Chapter 8

Network comparison

8.1 Subgraph comparisopn

8.2 Comparison of multi-sample networks

Chapter 9

Network & Environment

9.1 Network Environment

Chapter 10

Multi-omics network analysis

10.1 Multi-omics