Ants_Infomap

```
# Install (if not installed) and load necessary packages
package.list=c("attempt", "cowplot", "igraph", "ggalluvial", "magrittr", "metafolio", "tidyverse", "vega

loaded <- package.list %in% .packages()
package.list <- package.list[!loaded]
installed <- package.list %in% .packages(TRUE)
if(!all(installed)) install.packages(package.list[!installed], repose="http://cran.rstudio.com/")

#Install Infomapecology
devtools::install_github('Ecological-Complexity-Lab/infomap_ecology_package', force=T)
```

```
## Downloading GitHub repo Ecological-Complexity-Lab/infomap_ecology_package@master
```

```
##
##      checking for file ‘/tmp/RtmpSWvk6d/remotes58e869dc039b/Ecological-Complexity-Lab-infomap_ecology
##   -  preparing ‘infomapecology’: (628ms)
##      checking DESCRIPTION meta-information ...  v  checking DESCRIPTION meta-information
##   -  checking for LF line-endings in source and make files and shell scripts
##   -  checking for empty or unneeded directories
##    Removed empty directory ‘infomapecology/code_from_paper’
##    Removed empty directory ‘infomapecology/docs’
##   -  building ‘infomapecology_1.0.1.tar.gz’
##      Warning: invalid uid value replaced by that for user 'nobody'
##    Warning: invalid gid value replaced by that for user 'nobody'
##
##
```

```
## Installing package into '/home/sshatzkin/R/x86_64-pc-linux-gnu-library/4.0'
## (as 'lib' is unspecified)
```

```
library(infomapecology)
```

```
## Loading required package: attempt
```

```
## Loading required package: cowplot
```

```
## Loading required package: ggalluvial
```

```
## Loading required package: ggplot2
```

```
## Loading required package: magrittr
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:attempt':
##
##     if_all, if_any, if_else
```

```
## The following objects are masked from 'package:stats':
```

```
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: readr

## Loading required package: stringr

## Loading required package: tibble

## Loading required package: tidyr

##
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':
##
##     extract

## Loading required package: rlang

##
## Attaching package: 'rlang'

## The following object is masked from 'package:magrittr':
##
##     set_names

## Loading required package: igraph

##
## Attaching package: 'igraph'

## The following object is masked from 'package:rlang':
##
##     is_named

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

## Loading required package: vegan

## Loading required package: permute
```

```
##
## Attaching package: 'permute'

## The following object is masked from 'package:igraph':
##
##     permute

## Loading required package: lattice

## This is vegan 2.5-7

##
## Attaching package: 'vegan'

## The following object is masked from 'package:igraph':
##
##     diversity
```

```
# Check the version. Should be at least 0.1.1.1
packageDescription('infomapecology')
```

```
## Package: infomapecology
## Type: Package
## Title: Community Detection using Infomap, Inspired by Ecological
##         Networks
## Version: 1.0.1
## URL:
##         https://github.com/Ecological-Complexity-Lab/infomap_ecology_package
## Date: 2021-05-19
## Author: Shai Pilosof <pilos@post.bgu.ac.il>
## Maintainer: Shai Pilosof <pilos@post.bgu.ac.il>
## BugReports:
##         https://github.com/Ecological-Complexity-Lab/infomap_ecology_package/issues
## Description: Collection of R functions to perform community detection
##         analysis with Infomap. Also includes standardized objects to
##         store and further analyze monolayer and multilayer networks.
##         Inspired by ecological networks but can work for other networks
##         too! Ideas/requests for features are welcome (open an issue
##         here:
##         https://github.com/Ecological-Complexity-Lab/infomap_ecology_package/issues).
## License: GPL
## Encoding: UTF-8
## LazyData: true
## Depends: R (>= 4.0.0), attempt, cowplot, ggalluvial, magrittr, dplyr,
##         readr, ggplot2, stringr, tibble, tidyr, rlang, igraph, vegan
## Suggests: bipartite
## RoxygenNote: 7.1.1
## Comment: Some functions are wrappers for the standalone file Infomap
##         (independently installed) and their examples are set to
##         "dontrun"
## RemoteType: github
## RemoteHost: api.github.com
## RemoteRepo: infomap_ecology_package
## RemoteUsername: Ecological-Complexity-Lab
## RemoteRef: master
## RemoteSha: 412752b53a16aa8d047c26fc80b4d55285b46667
## GithubRepo: infomap_ecology_package
```

```
## GithubUsername: Ecological-Complexity-Lab
## GithubRef: master
## GithubSHA1: 412752b53a16aa8d047c26fc80b4d55285b46667
## NeedsCompilation: no
## Packaged: 2021-06-09 07:56:00 UTC; sshatzkin
## Built: R 4.0.0; ; 2021-06-09 07:56:01 UTC; unix
##
## -- File: /home/sshatzkin/R/x86_64-pc-linux-gnu-library/4.0/infomapecology/Meta/package.rds
```

```
# Install infomap if you have not done so externally (see previous section in this readme)
setwd('/home/sshatzkin/Networks in Biology')
install_infomap()
```

```
## Running make...
```

```
## Finishing...
```

```
## [1] TRUE
```

```
# Check Infomap is running
setwd('/home/sshatzkin/Networks in Biology')
check_infomap() #Make sure file can be run correctly. Should return True
```

```
## [1] TRUE
```

```
library(igraph)
library(bipartite)
```

```
## Loading required package: sna
```

```
## Loading required package: statnet.common
```

```
##
## Attaching package: 'statnet.common'
```

```
## The following object is masked from 'package:base':
##
##     order
```

```
## Loading required package: network
```

```
## network: Classes for Relational Data
## Version 1.16.0 created on 2019-11-30.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Martina Morris, University of Washington
##                     Skye Bender-deMoll, University of Washington
##  For citation information, type citation("network").
##  Type help("network-package") to get started.
```

```
##
## Attaching package: 'network'
```

```
## The following objects are masked from 'package:igraph':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute
```

```
## sna: Tools for Social Network Analysis
## Version 2.5 created on 2019-12-09.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.

##
## Attaching package: 'sna'

## The following objects are masked from 'package:igraph':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census

##  This is bipartite 2.16.
##  For latest changes see versionlog in ?"bipartite-package". For citation see: citation("bipartite").
##  Have a nice time plotting and analysing two-mode networks.

##
## Attaching package: 'bipartite'

## The following object is masked from 'package:vegan':
##
##     nullmodel

## The following object is masked from 'package:igraph':
##
##     strength
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v purrr   0.3.4     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x purrr::%@%()            masks rlang::%@%()
## x igraph::as_data_frame() masks tibble::as_data_frame(), dplyr::as_data_frame()
## x purrr::as_function()    masks rlang::as_function()
## x purrr::compose()        masks igraph::compose()
## x igraph::crossing()      masks tidyr::crossing()
## x tidyr::extract()        masks magrittr::extract()
## x dplyr::filter()         masks stats::filter()
## x purrr::flatten()        masks rlang::flatten()
## x purrr::flatten_chr()    masks rlang::flatten_chr()
## x purrr::flatten_dbl()    masks rlang::flatten_dbl()
## x purrr::flatten_int()    masks rlang::flatten_int()
## x purrr::flatten_lgl()    masks rlang::flatten_lgl()
## x purrr::flatten_raw()    masks rlang::flatten_raw()
## x igraph::groups()        masks dplyr::groups()
## x dplyr::if_all()         masks attempt::if_all()
## x dplyr::if_any()         masks attempt::if_any()
## x dplyr::if_else()        masks attempt::if_else()
## x purrr::invoke()         masks rlang::invoke()
## x igraph::is_named()      masks rlang::is_named()
## x dplyr::lag()            masks stats::lag()
## x purrr::list_along()     masks rlang::list_along()
## x purrr::modify()         masks rlang::modify()
## x purrr::prepend()        masks rlang::prepend()
```

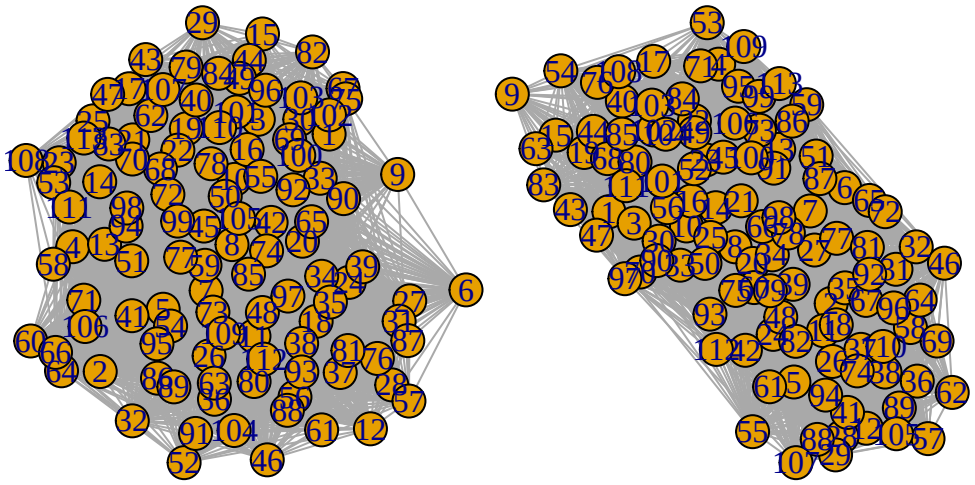
```
## x purrr::set_names()      masks rlang::set_names(), magrittr::set_names()
## x purrr::simplify()       masks igraph::simplify()
## x purrr::splice()         masks rlang::splice()
```

```
library(magrittr)
library(infomapecology)
library(readxl)
library(ggalluvial)
```

1. Infomap Analysis

Importing the Data

```
numFiles <- 41

import_data <- function (string_format, numFiles){
  # Create the file names
  file_names <- sprintf(string_format, 1:numFiles);

  # Create an empty linked list
  colony_data <- vector(mode = "list", length = numFiles)

  # Read in all of the files
  for (i in 1:numFiles){
    colony_data[[i]] <- read_graph(file_names[i], format = "graphml")
  }

  return (colony_data)
}

colony1_data <- import_data('./ants_proximity_weighted/ant_mersch_col1_day%02d_attribute.graphml', numF


for (i in 1:2){
  plot.igraph(colony1_data[[i]]) # This plot is not particularly informative
}
```



Perparing the data.

The data must be converted to an infomap compatible format

Helper functions

```
# Function that takes a vertex list and the number of layers in the multilayer network, and produces a
interlayer_edge_list <- function (vertex_names, num_layers){

  numVerts <- length (vertex_names)
  numRows <- numVerts * (num_layers - 1)

  col_layer_from <- vector(length = numRows)
  col_layer_to <- vector(length = numRows)
  col_node_from <- vector(length = numRows)
  col_node_to <- vector(length = numRows)
  col_weights <- vector(length = numRows)

  for (l in 1:(num_layers-1)){
    for (vertex in 1:numVerts){
      currIndex <- ((l-1) * numVerts) + vertex
      col_layer_from[currIndex] <- l
      col_layer_to[currIndex] <- l + 1
      col_node_from[currIndex] <- vertex_names[vertex]
      col_node_to[currIndex] <- vertex_names[vertex]
      col_weights[currIndex] <- 1
    }
  }

  df <- data.frame("layer_from" = col_layer_from, "node_from" = col_node_from, "layer_to" = col_layer_t

  return (df)
}

# Joins a list of dataframes into a single dataframe
join_edge_dfs <- function(edge_dfs, num_lists){
  df <- edge_dfs[[1]]
  for (i in 2:num_lists){
    df <- rbind(df, edge_dfs[[i]])
  }
  return(df)
}
```

Main datatype conversion function

```
# Now we need to convert this igraph data into data structures that infomap can take

igraphs_to_multilayer_df <- function (igraph_list, num_elements){

  # Create empty lists
  colony_edges <- vector(mode = "list", length = num_elements)
  colony_nodes <- vector(mode = "list", length = num_elements)
```

```
  # Loop over layers
  for (i in 1:num_elements){

    # Pull out the edge lists from each layer
    colony_edges[[i]] <- data.frame(cbind(get.edgelist(igraph_list[[i]]), round (E(igraph_list[[i]])$we

    #Assign column labels
    names(colony_edges[[i]]) <- c('node_from', 'node_to', 'weight')


    #Generate an array of layer labels
    layer_arr <- rep(i, nrow(colony_edges[[i]]))

    colony_edges[[i]]$layer_from <- layer_arr
    colony_edges[[i]]$layer_to <- layer_arr

    # Reorder the columns
    colony_edges[[i]] <- colony_edges[[i]][,c('layer_from', 'node_from', 'layer_to', 'node_to', 'weight



    # Pull out the vertices from each layer
    dayi_vertices <- V(igraph_list[[i]])

    # Prepare the node lists
    colony_nodes[[i]] <- data.frame(as_ids(dayi_vertices))
    names(colony_nodes[[i]]) <- c('node_id')

  }


  # Construct inter-layer edges
  interlayer_edges <- interlayer_edge_list(colony_nodes[[i]]$node_id, num_elements);


  # Now merge these smaller lists to form the full edge and node lists
  colony_edges_joined <- join_edge_dfs(colony_edges, num_elements)
  colony_edges_joined <- rbind(colony_edges_joined, interlayer_edges)


  return (list(colony_edges_joined, colony_nodes[[1]]))

}
```

Plot all 41 days as one multilayer network

```
# Function takes the igraph_list, converts to an infomap data structure, and returns the module results
igraph_to_infomap_modules <- function (igraph_list, num_layers){

  # Convert the igraph list into a dataframe with interlayer edges
  colony_dfs <- igraphs_to_multilayer_df(igraph_list, num_layers)
```

```
  colony_edges <- colony_dfs[[1]]
  colony_nodes <- colony_dfs[[2]]

  # Build layer_map
  layer_map <- tibble(layer_id=1:num_layers, layer_name=1:num_layers)

  # Create the multilary object
  colony <- create_multilayer_object(extended = colony_edges, nodes = colony_nodes, intra_output_extend

  # Run infomap
  colony_modules <- run_infomap_multilayer(M=colony, relax = F, flow_model = 'directed', silent = T, tr

  return (colony_modules)
}
```

```
colony1_modules <- igraph_to_infomap_modules(colony1_data, numFiles)
```

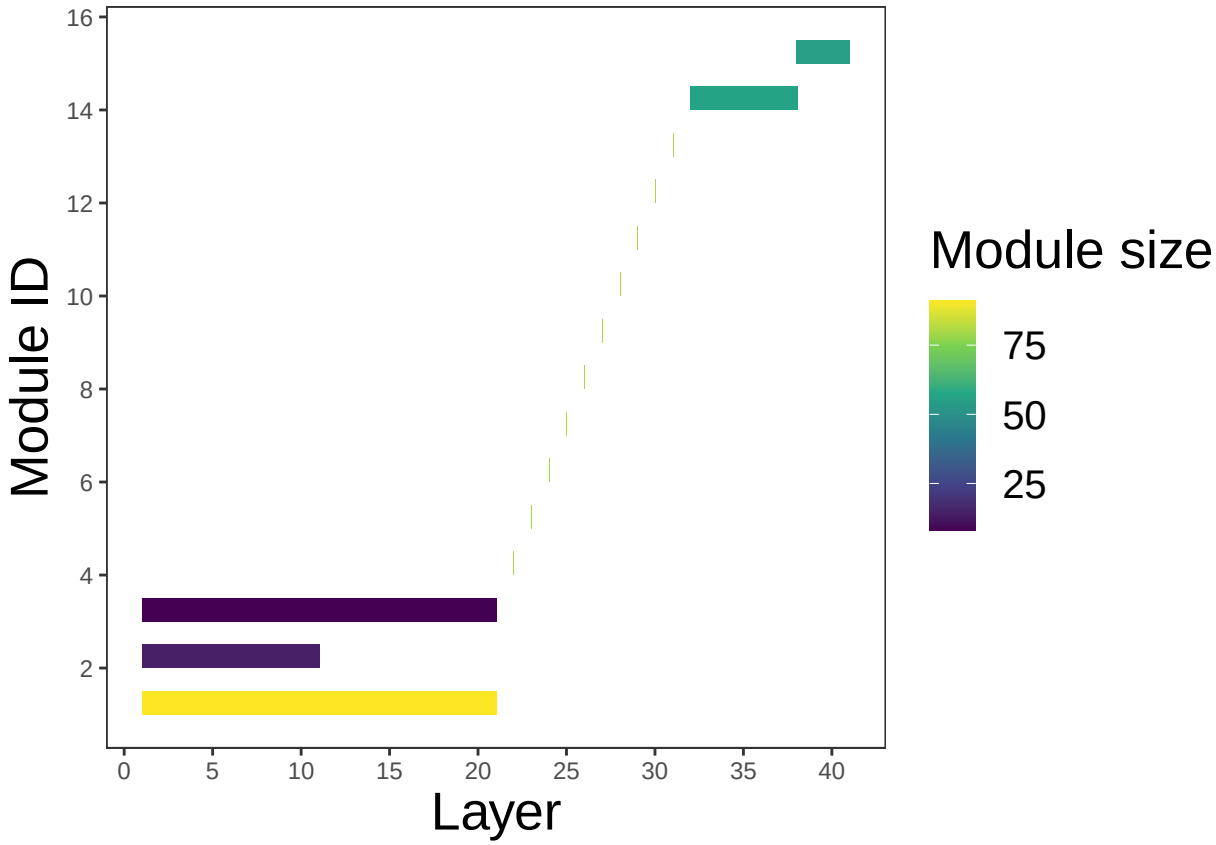
```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --s
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 15 modules."
```

```
colony1_modules
```

```
## $call
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --s
##
## $L
## [1] 5.06991
##
## $m
## [1] 15
##
## $modules
## # A tibble: 3,588 x 3
##    node_id layer_id module
##      <int>    <int>  <int>
##  1       1        2      1
##  2       1        3      1
##  3       1        4      1
##  4       1        5      1
##  5       1        6      1
##  6       1        7      1
##  7       1        8      1
##  8       1        9      1
##  9       1       10      1
## 10       1       11      1
## # ... with 3,578 more rows
##
## attr(,"class")
## [1] "infomap_multilayer"
```

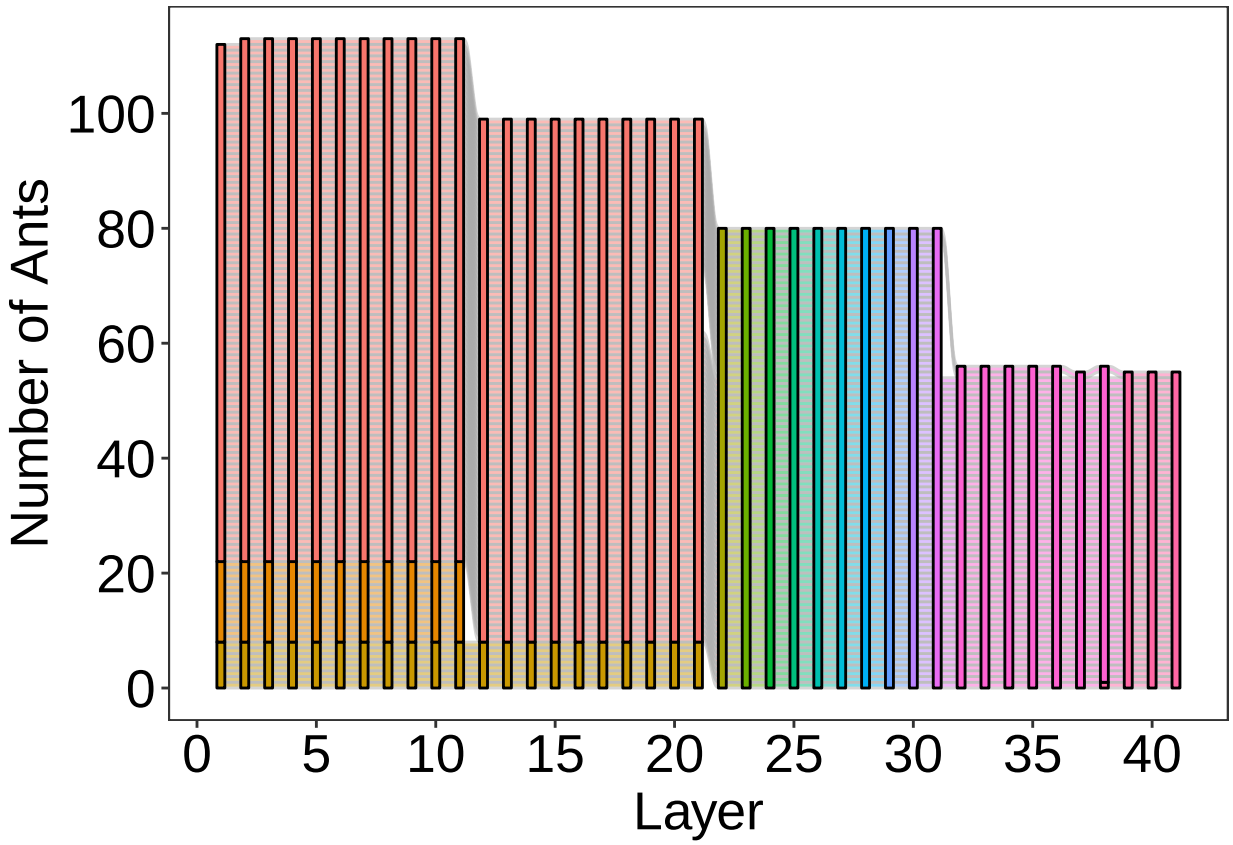
```
plot_persistance <- function (module_data, num_layers, num_modules){
  # Module Persistance
```

```
  module_persistence <- module_data$modules %>%
  group_by(module) %>%
  summarise(b=min(layer_id), d=max(layer_id), persistence=d-b+1)

  # Plot modules' persistence
  plot_multilayer_modules(module_data, type = 'rectangle', color_modules = T)+
    scale_x_continuous(breaks = seq(0, num_layers, 5))+
    scale_y_continuous(breaks = seq(0, num_modules, 2))+
    scale_fill_viridis_c()+
    theme_bw()+
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          axis.title = element_text(size = 20),
          legend.text = element_text(size=15),
          legend.title = element_text(size=20))
}

plot_persistance(colony1_modules, 41, 20)
```




```
plot_alluvial <- function (module_data, num_layers, num_ants){
  # Plot species flow through modules in time
  plot_multilayer_alluvial(module_data, module_labels = F)+
  scale_x_continuous(breaks=seq(0, num_layers, 5))+
  scale_y_continuous(breaks=seq(0,num_ants,20))+
  labs(y='Number of Ants')+
  theme_bw()+
  theme(legend.position = "none",
```

```
        panel.grid = element_blank(),
        axis.text = element_text(color='black', size = 20),
        axis.title = element_text(size=20))
}

plot_alluvial(colony1_modules, 41, 120)
```

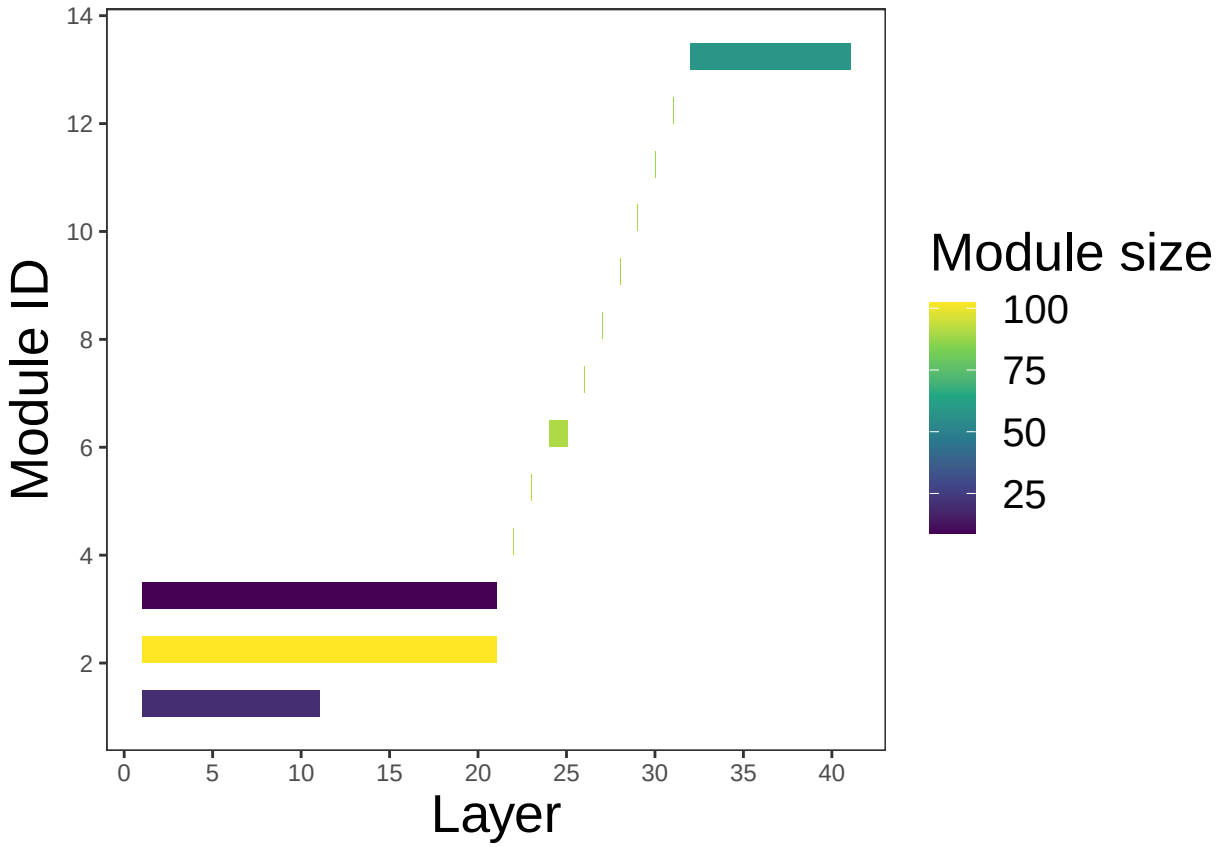


Test with another colony

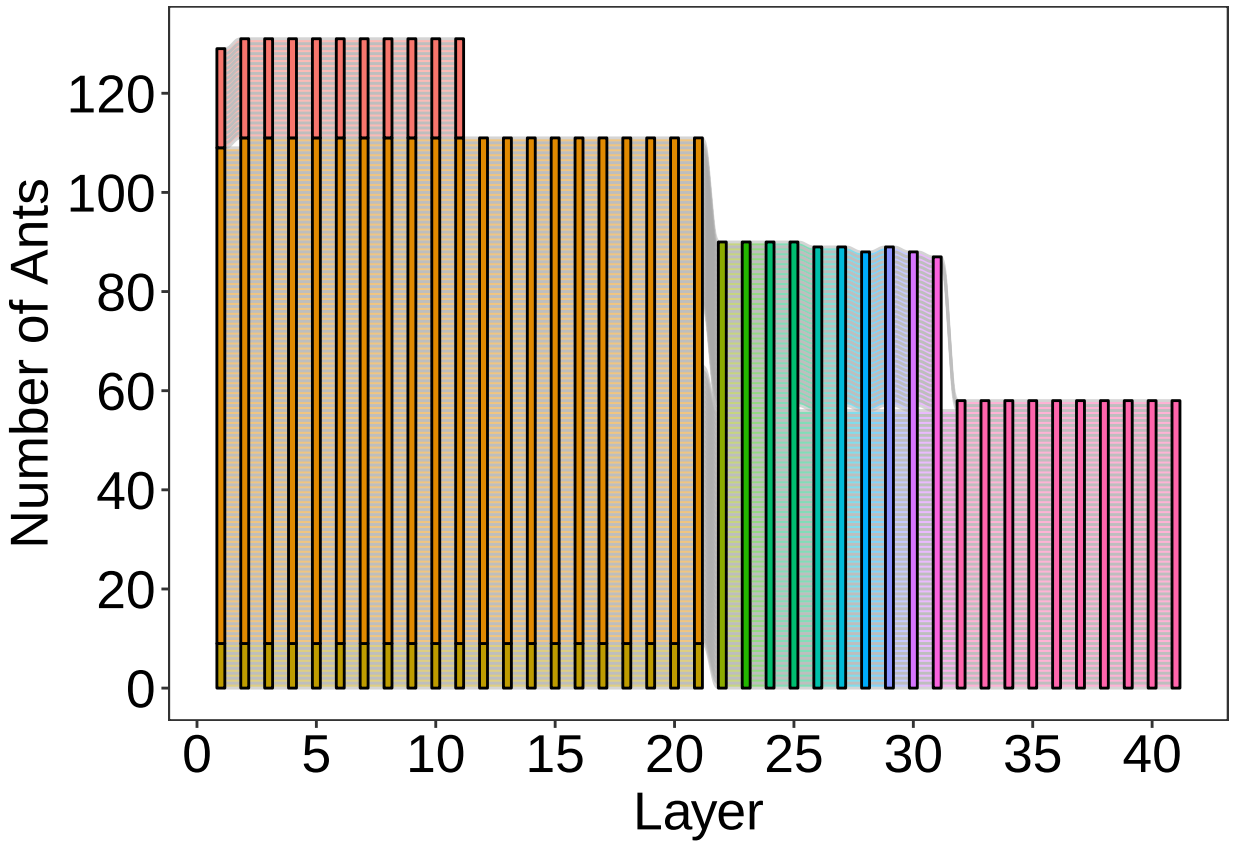
```
# Import all of the colony2 data files
colony2_data <- import_data('./ants_proximity_weighted/ant_mersch_col2_day%02d_attribute.graphml', numF

# Convert a list of igraphs into a processed infomap module
colony2_modules <- igraph_to_infomap_modules(colony2_data, numFiles)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --s
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 13 modules."
```

```
# Plots functions based on just module data
plot_persistance(colony2_modules, 41, 20)
```




```
plot_alluvial(colony2_modules, 41, 120)
```

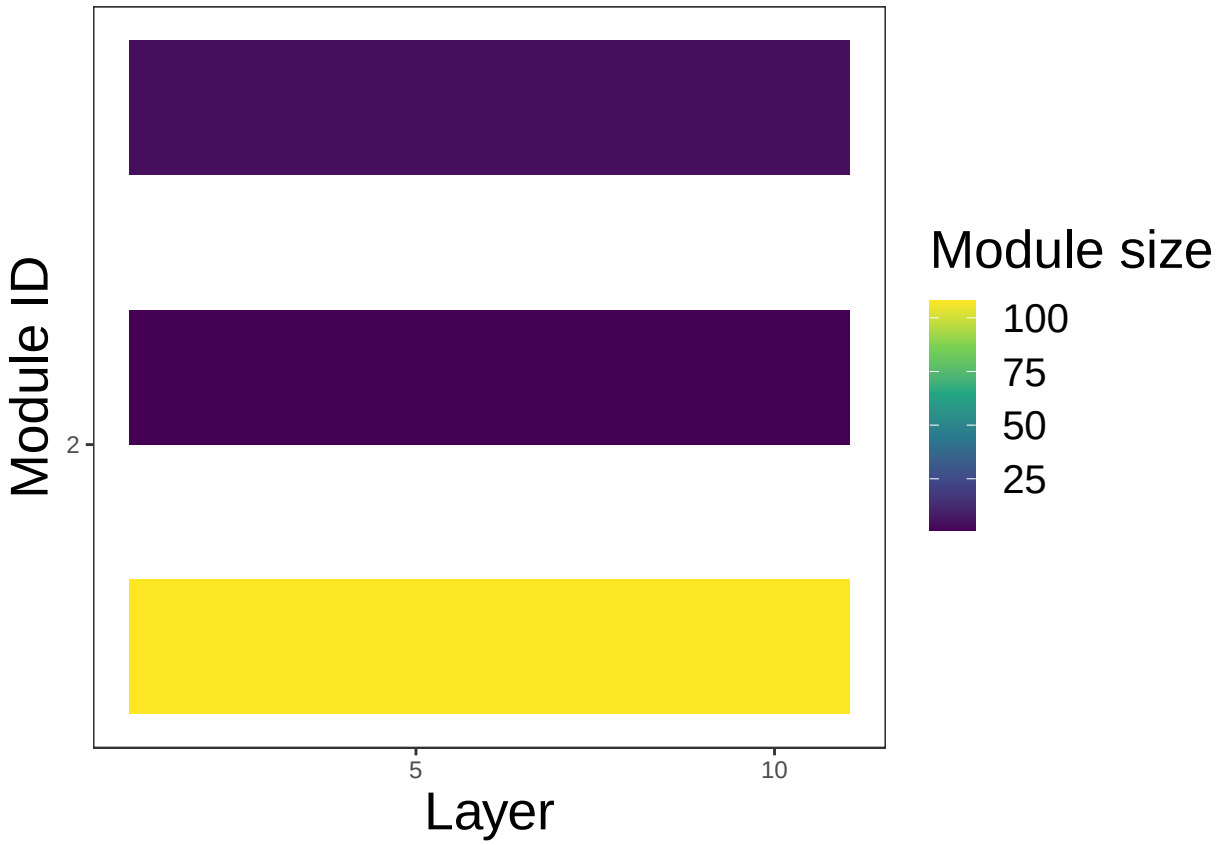


Plot in 10-day increments

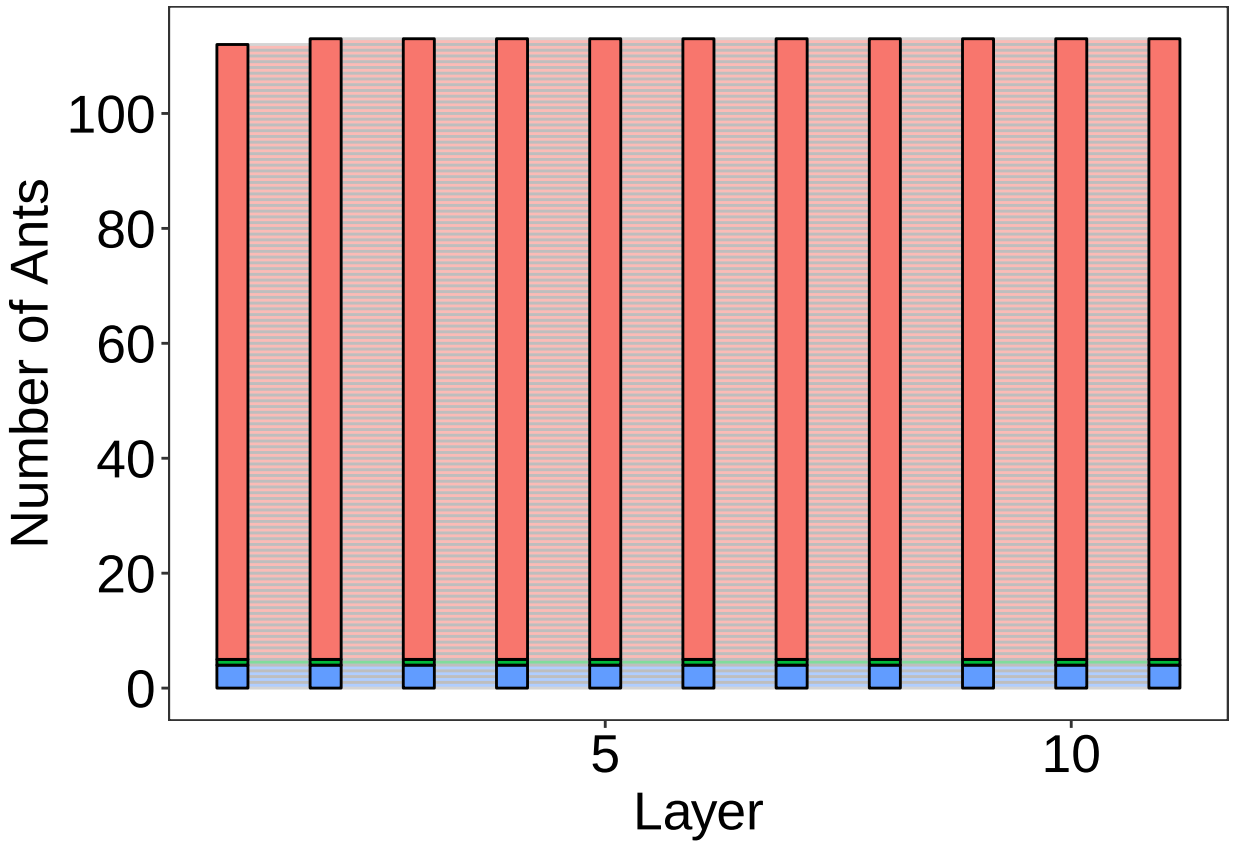
```
divi_modules <- igraph_to_infomap_modules(colony1_data[1:11], 11)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --s
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 3 modules."
```

```
plot_persistance(divi_modules, 11, 20)
```



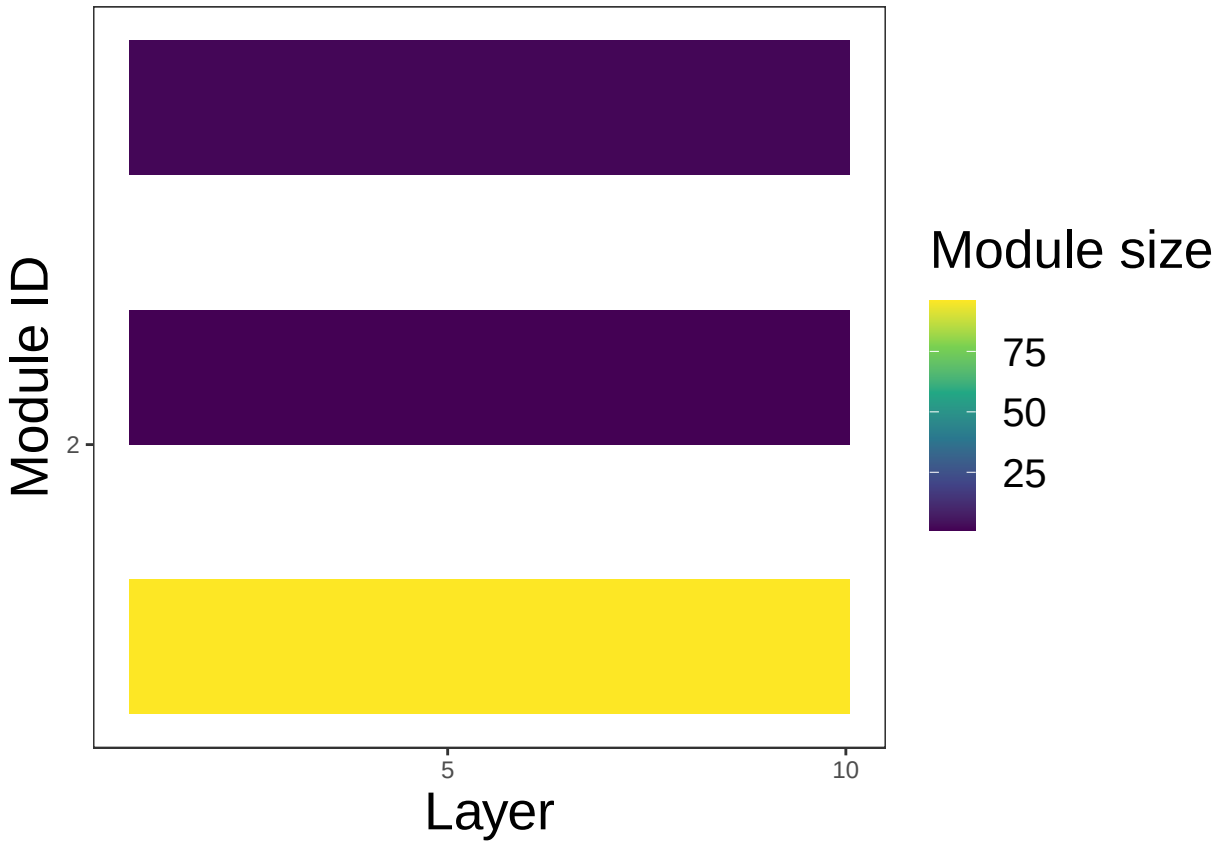

```
plot_alluvial(divi_modules, 11, 120)
```



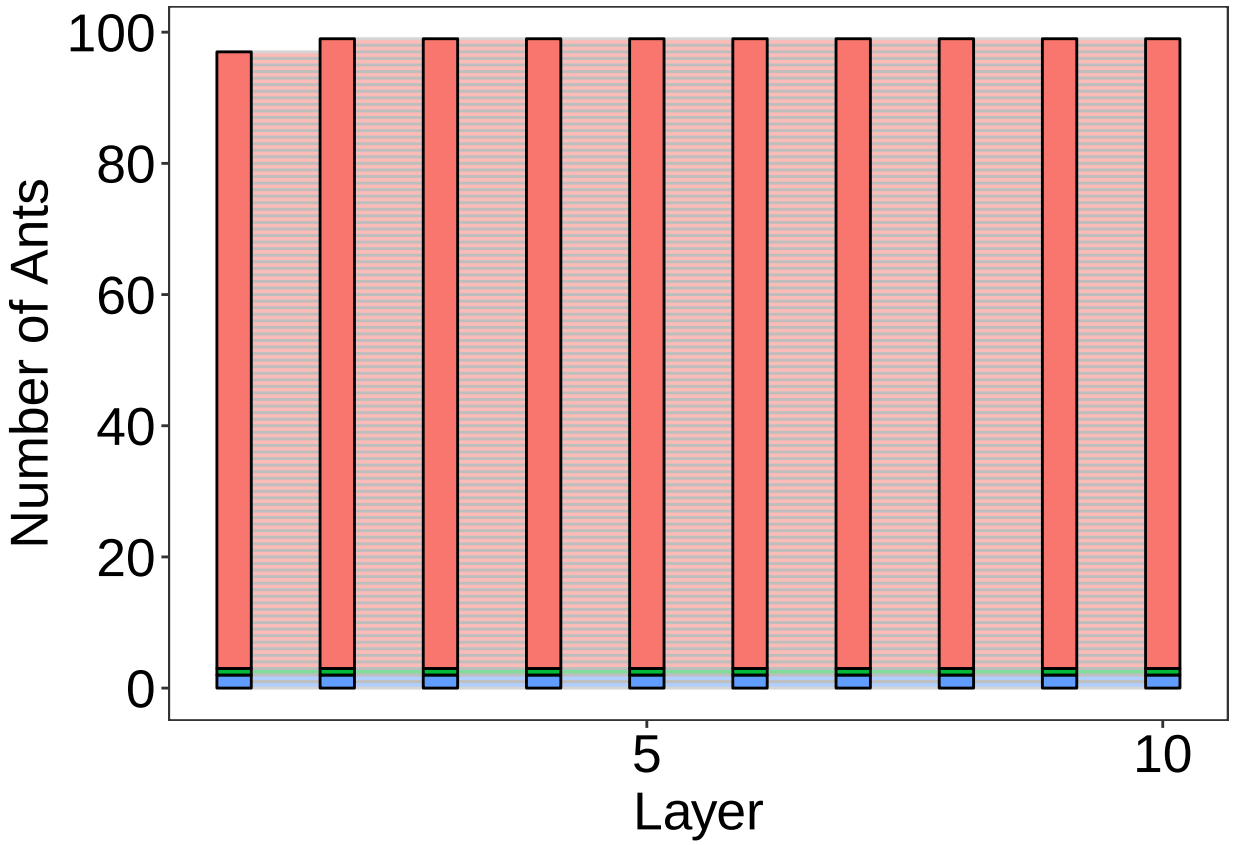

```
divi_modules <- igraph_to_infomap_modules(colony1_data[12:21], 10)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 3 modules."
```

```
plot_persistance(divi_modules, 10, 20)
```



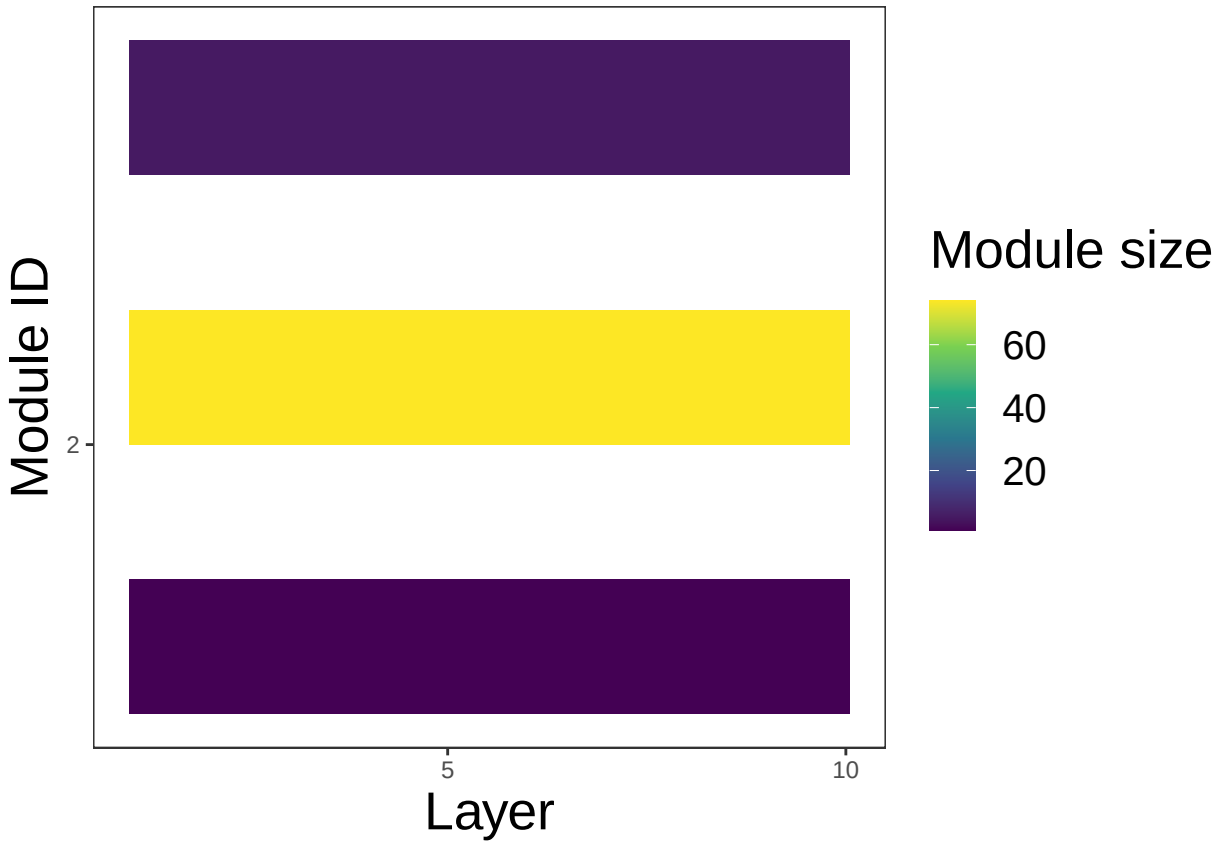

```
plot_alluvial(divi_modules, 10, 120)
```



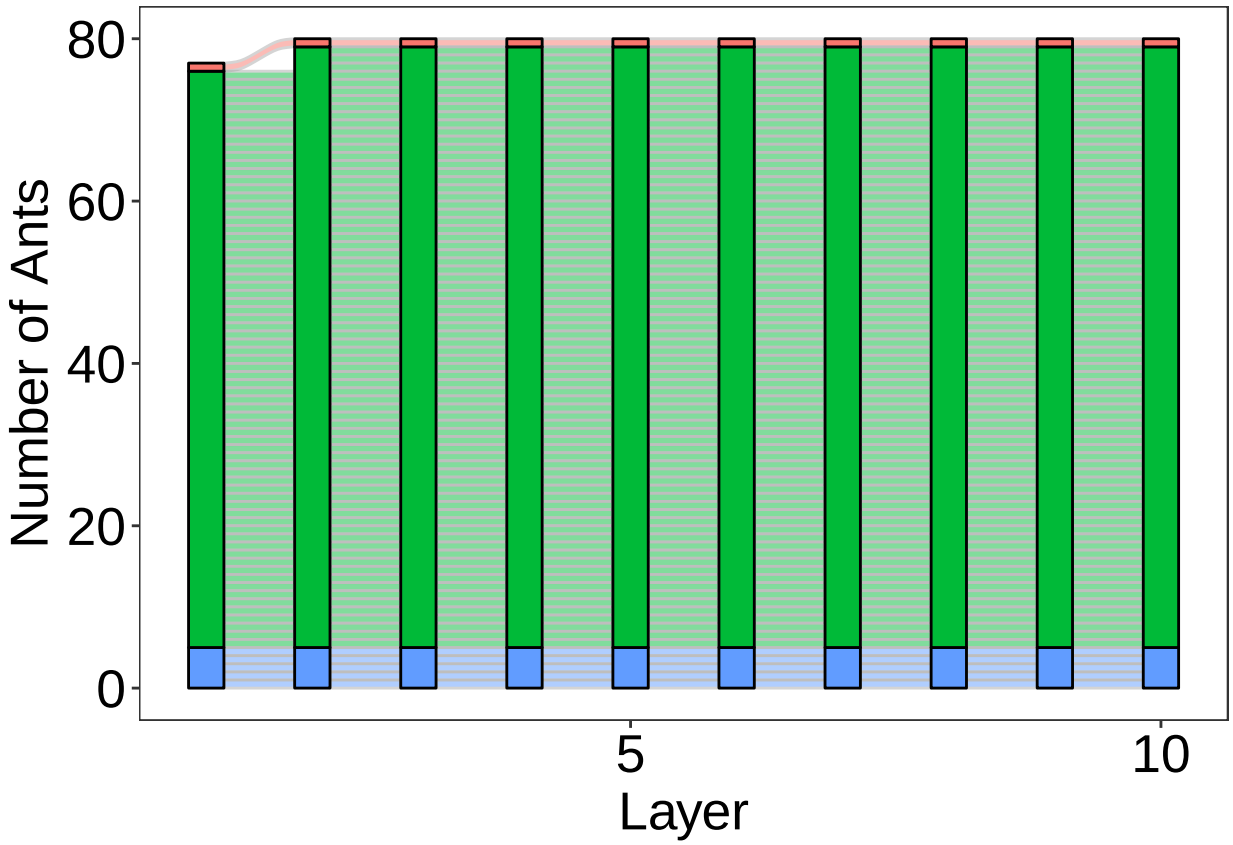

```
divi_modules <- igraph_to_infomap_modules(colony1_data[22:31], 10)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 3 modules."
```

```
plot_persistance(divi_modules, 10, 20)
```



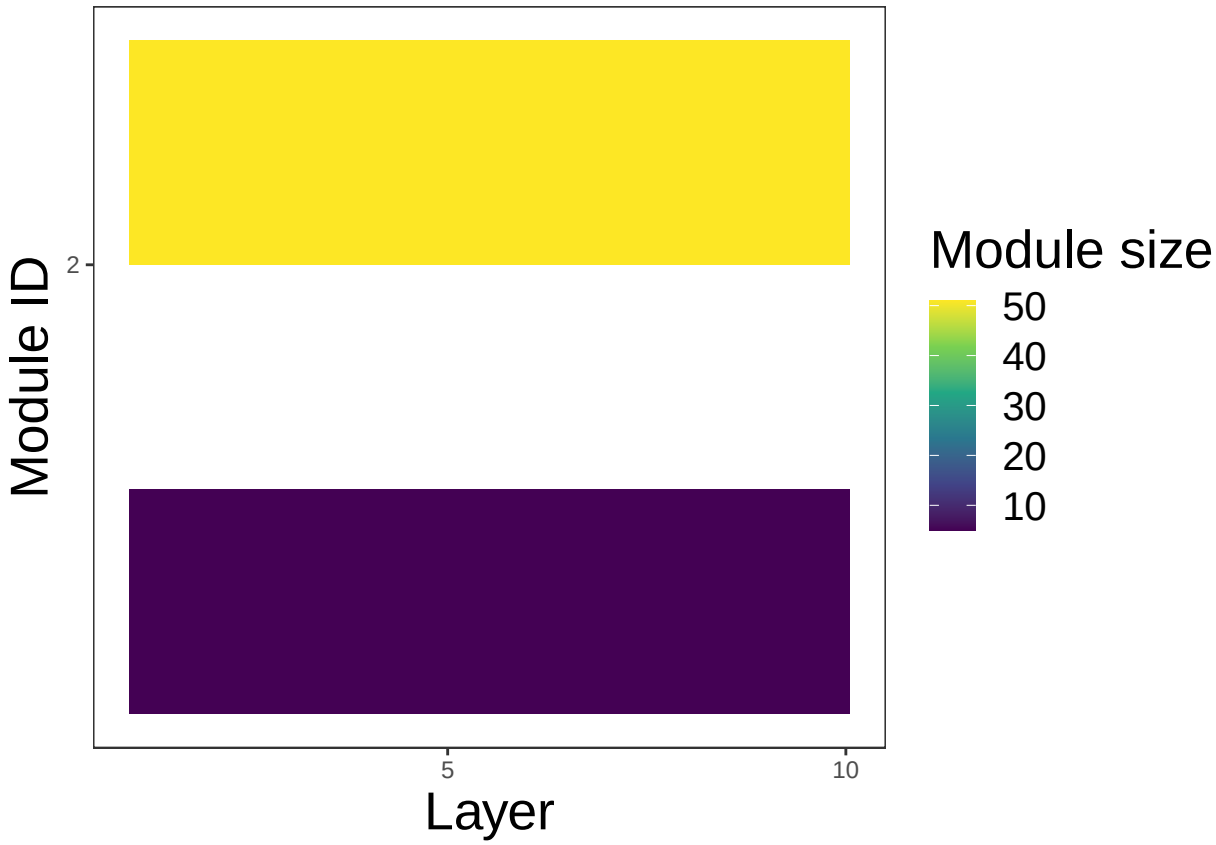

```
plot_alluvial(divi_modules, 10, 120)
```



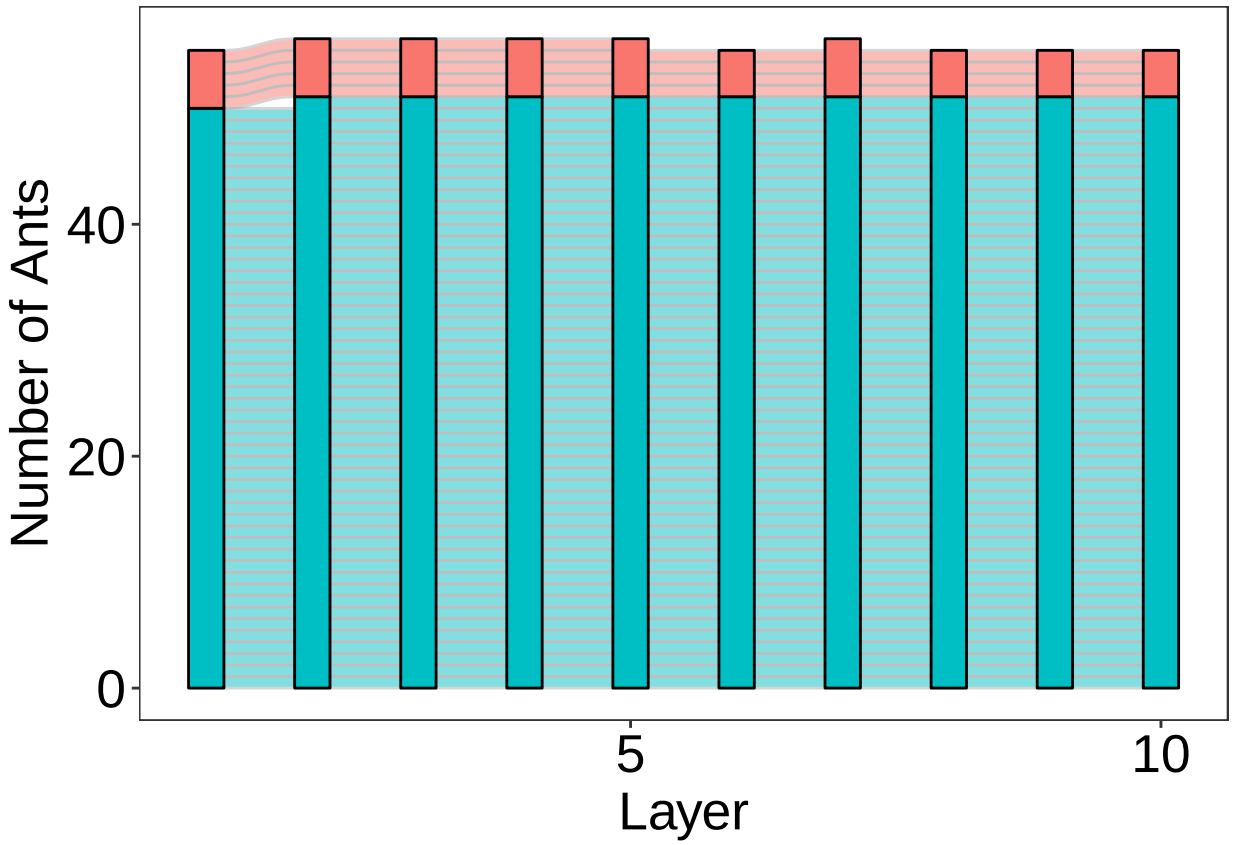

```
divi_modules <- igraph_to_infomap_modules(colony1_data[32:41], 10)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 2 modules."
```

```
plot_persistance(divi_modules, 10, 20)
```



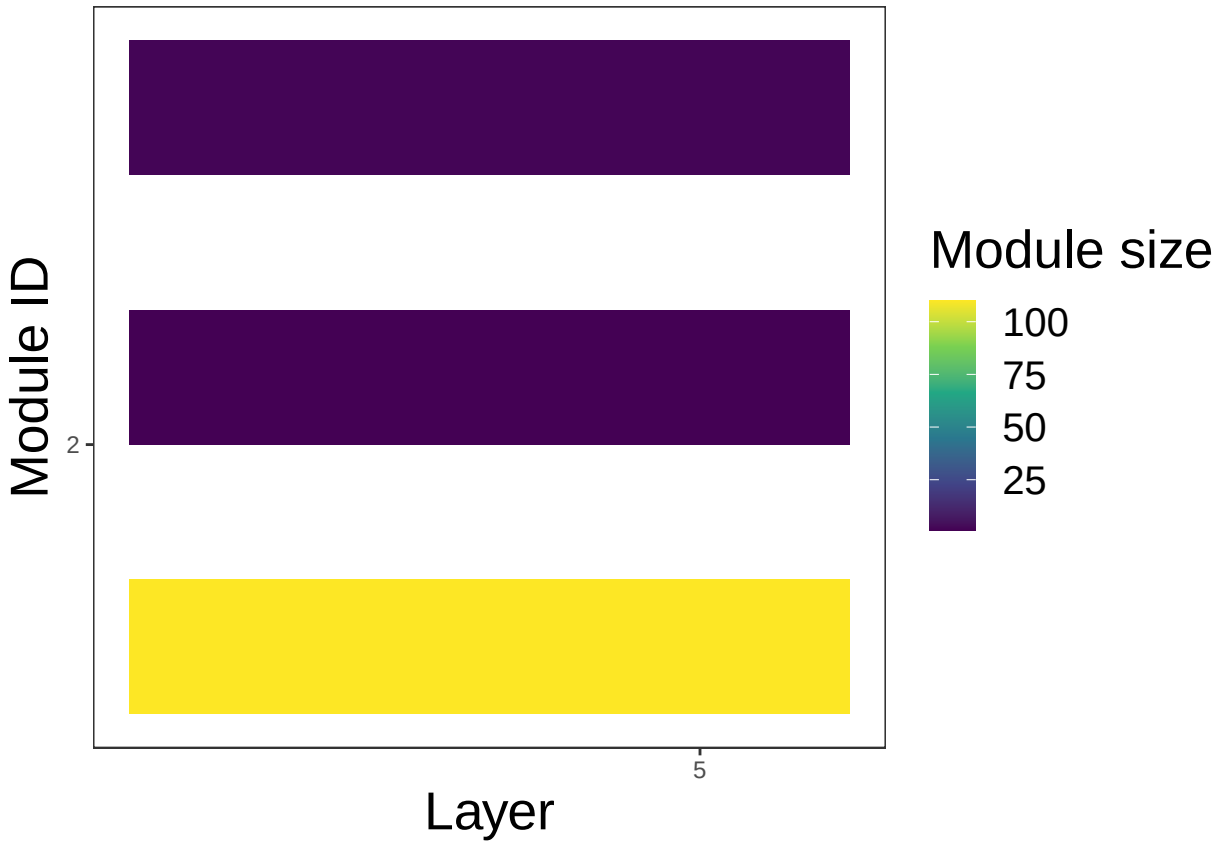

```
plot_alluvial(divi_modules, 10, 120)
```



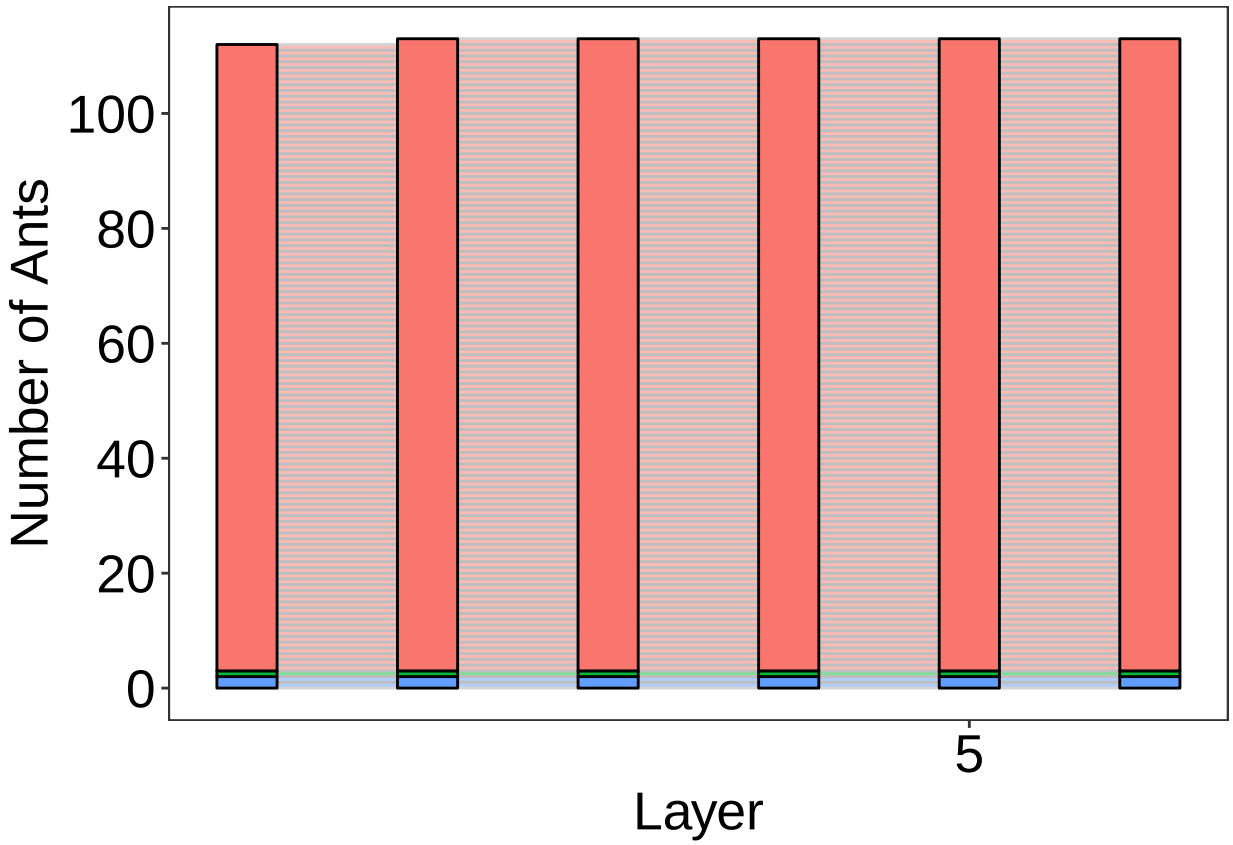

```
divi_modules <- igraph_to_infomap_modules(colony1_data[1:6], 6)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 3 modules."
```

```
plot_persistance(divi_modules, 6, 20)
```



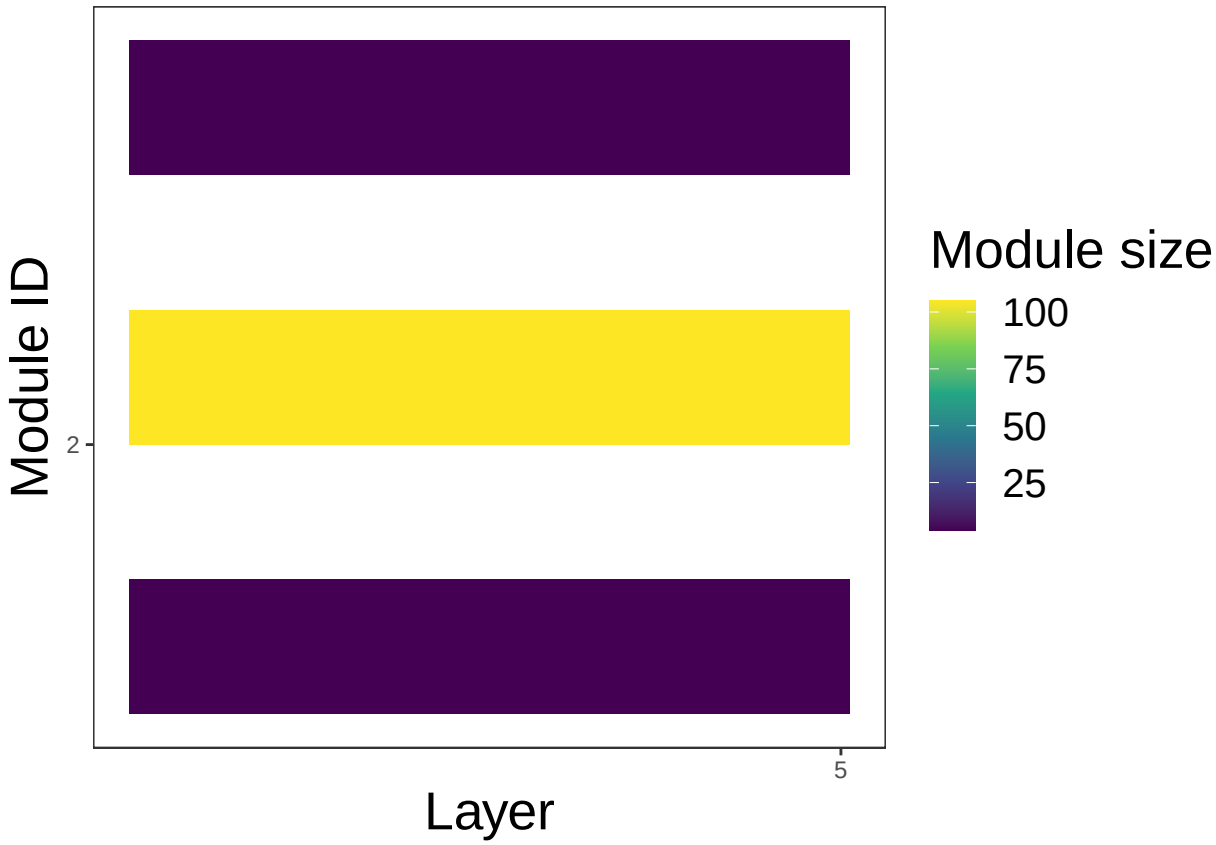

```
plot_alluvial(divi_modules, 6, 120)
```



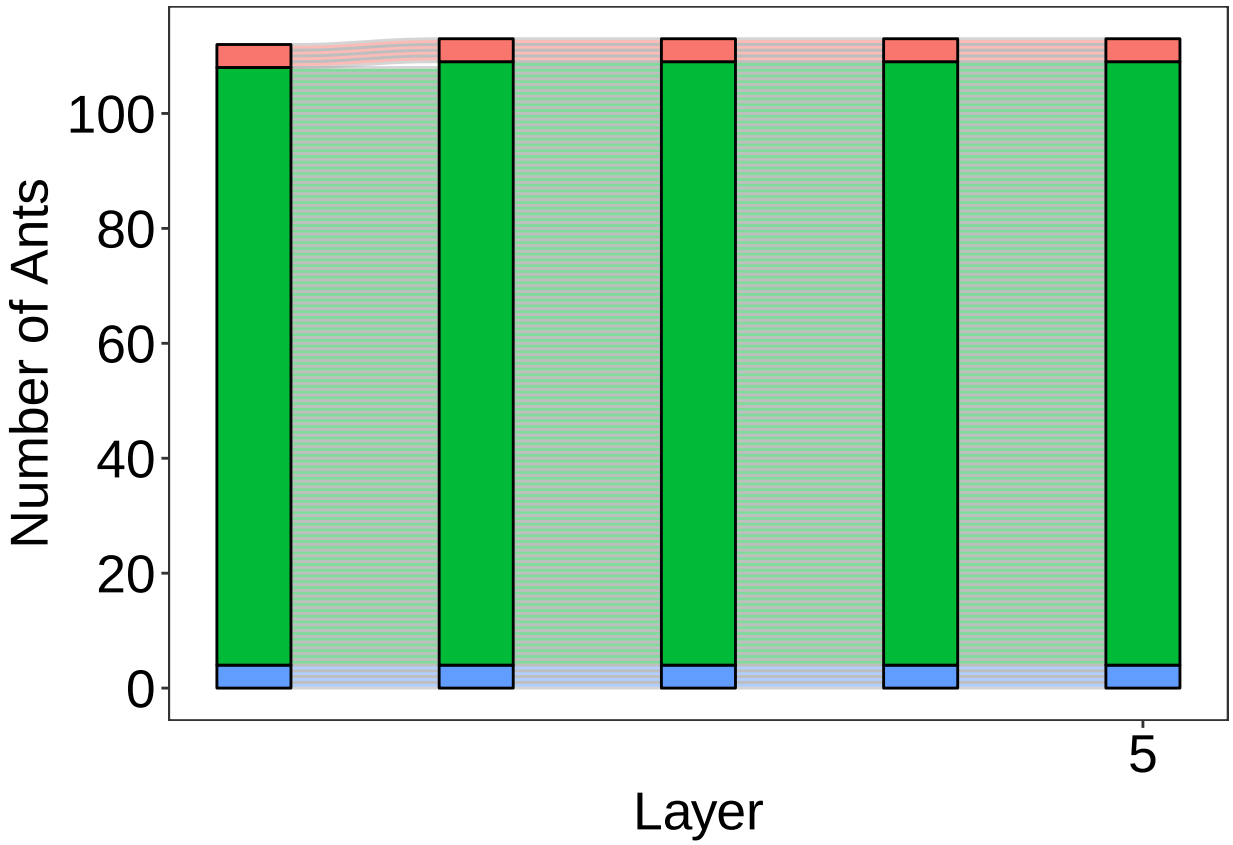

```
divi_modules <- igraph_to_infomap_modules(colony1_data[7:11], 5)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 3 modules."
```

```
plot_persistance(divi_modules, 5, 20)
```



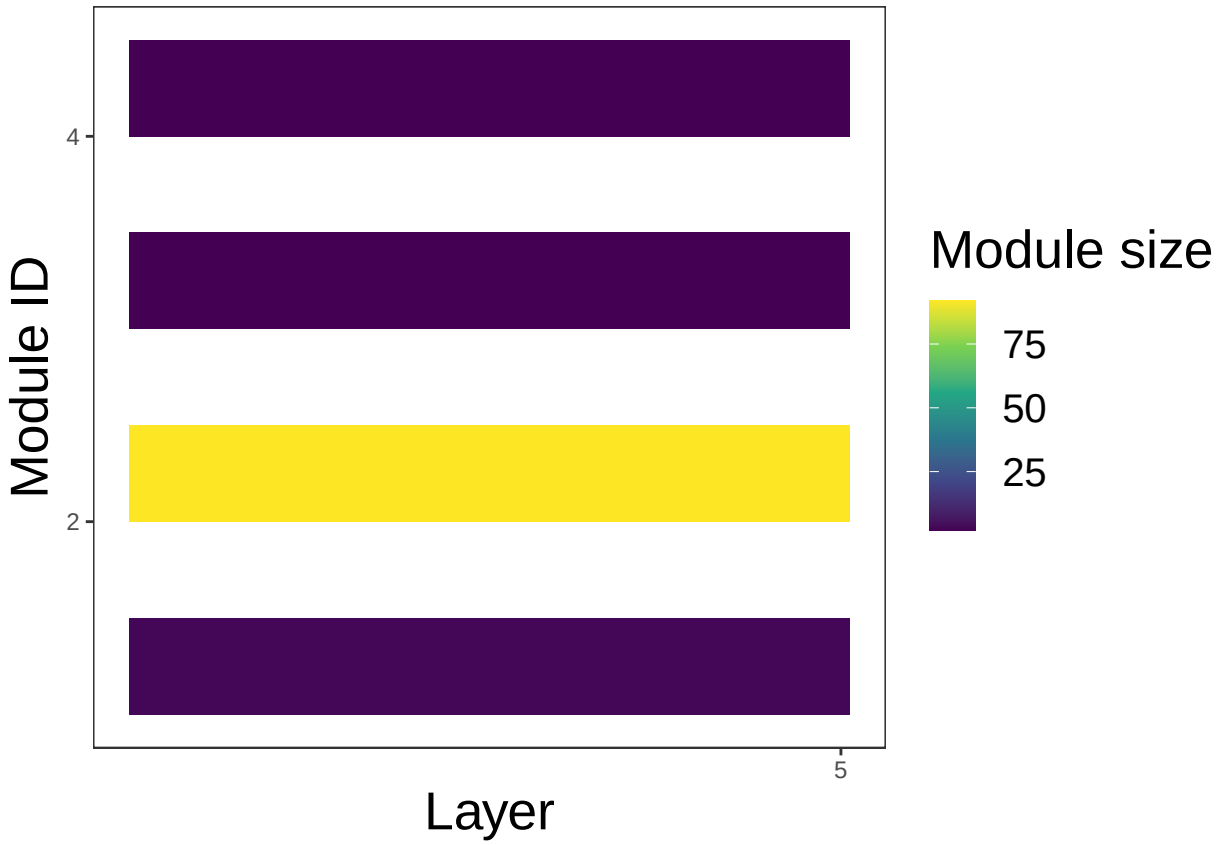

```
plot_alluvial(divi_modules, 5, 120)
```



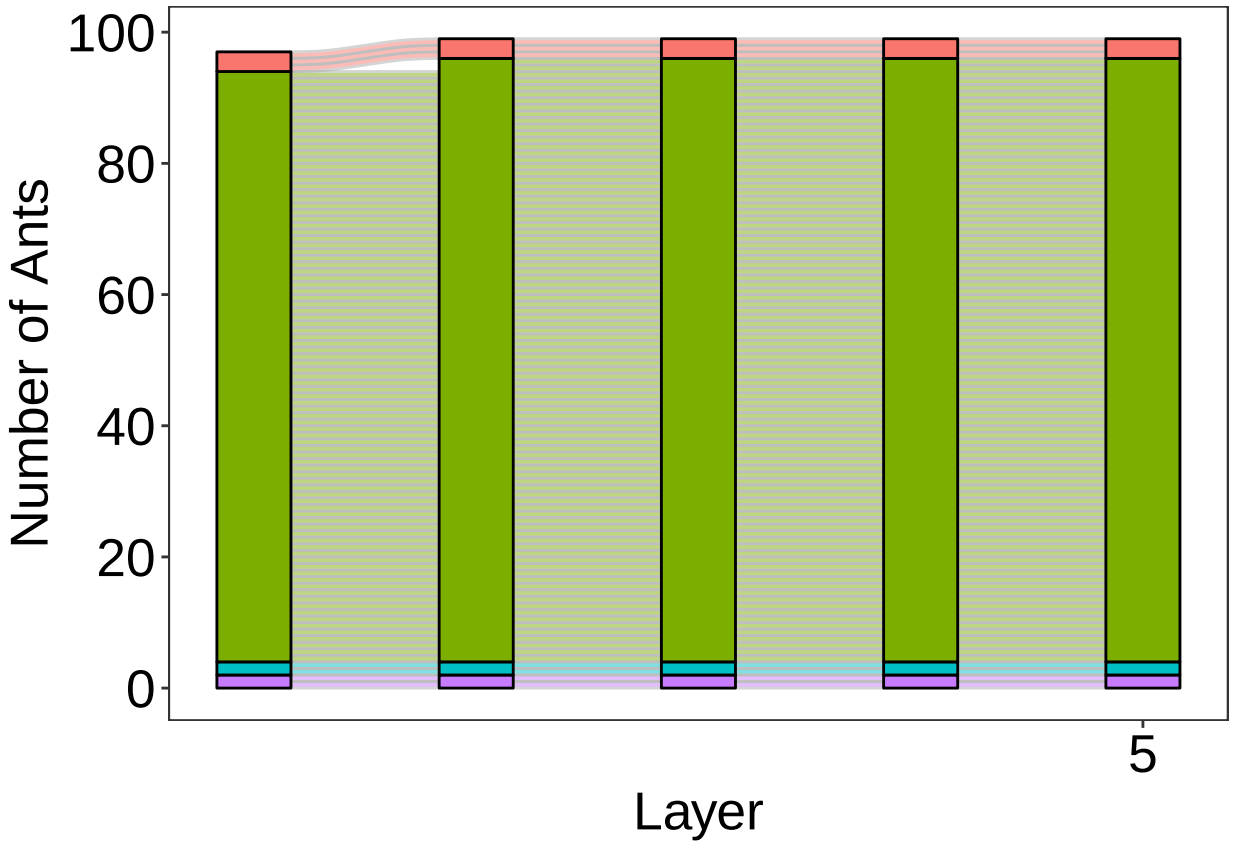

```
divi_modules <- igraph_to_infomap_modules(colony1_data[12:16], 5)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 4 modules."
```

```
plot_persistance(divi_modules, 5, 20)
```



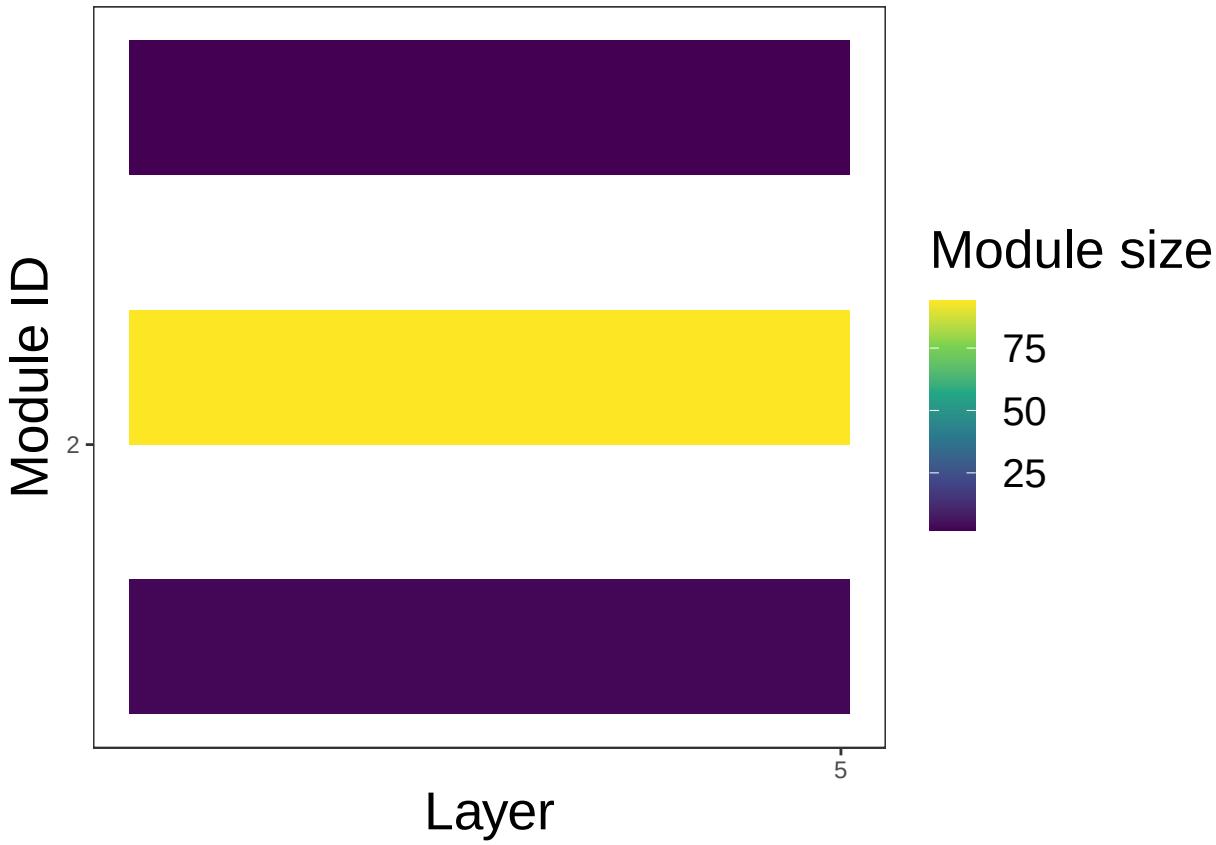

```
plot_alluvial(divi_modules, 5, 120)
```



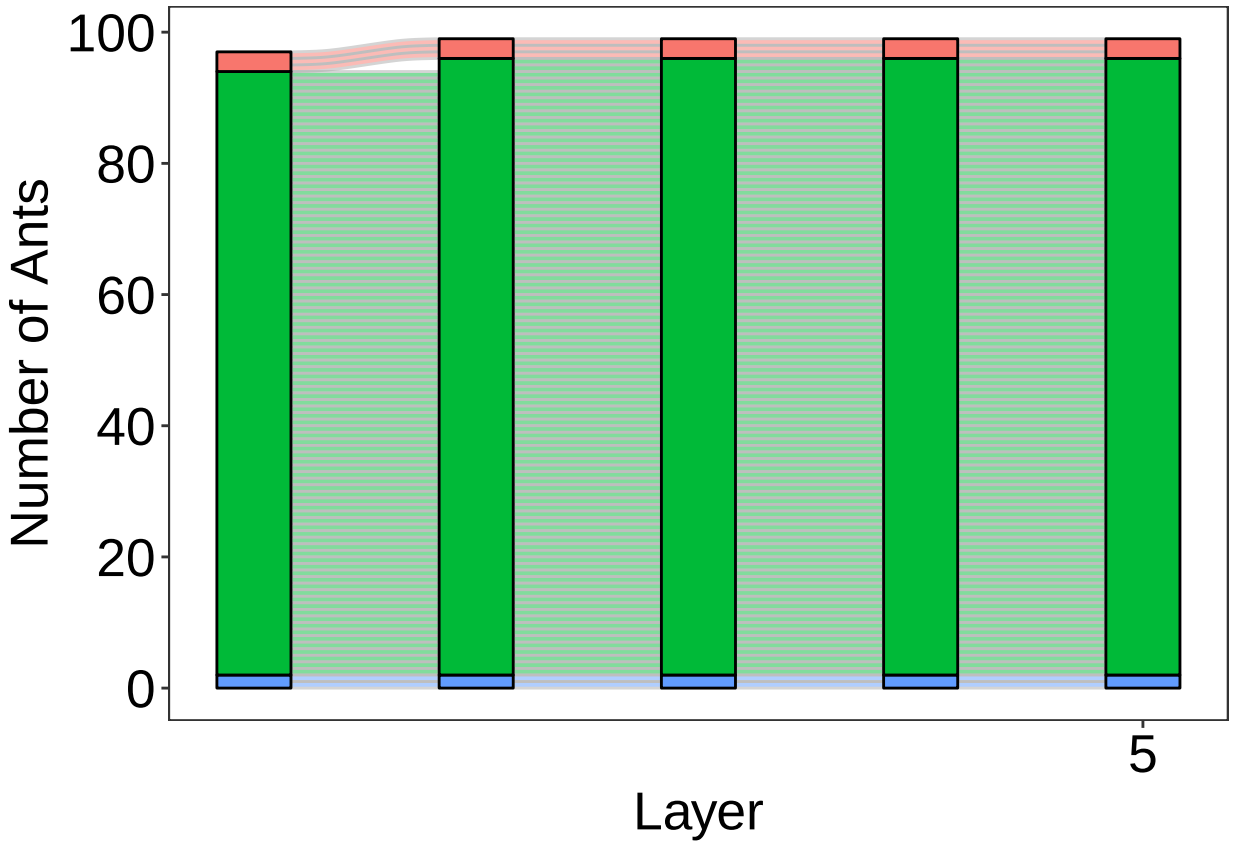

```
divi_modules <- igraph_to_infomap_modules(colony1_data[17:21], 5)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 3 modules."
```

```
plot_persistance(divi_modules, 5, 20)
```



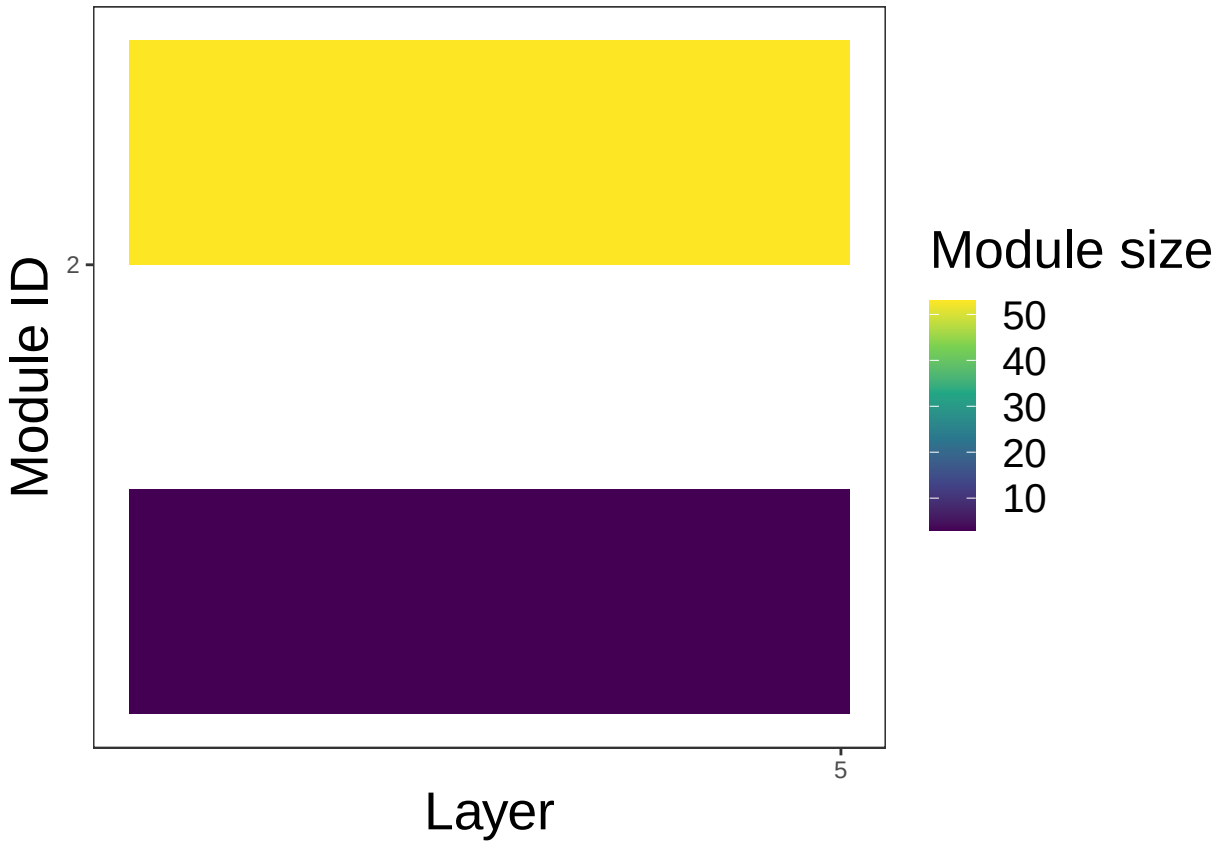

```
plot_alluvial(divi_modules, 5, 120)
```



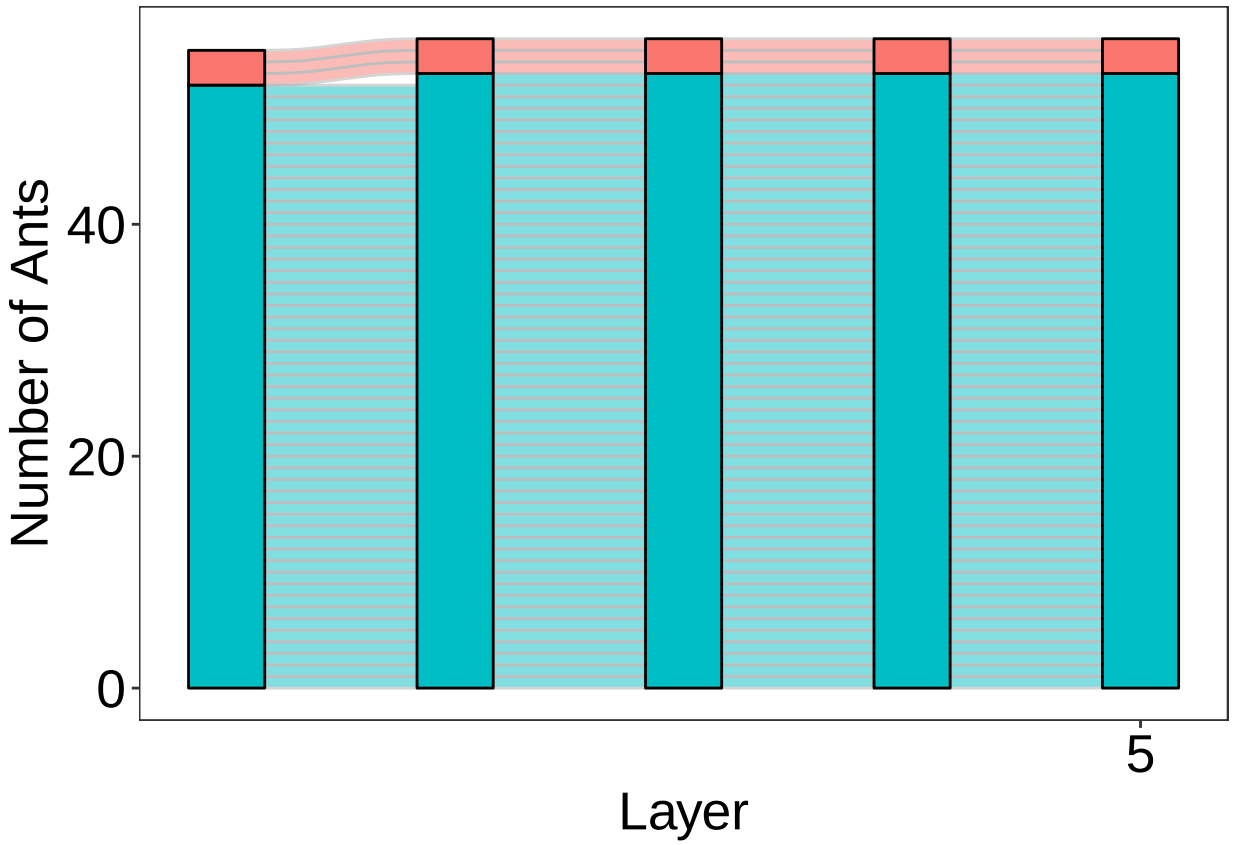

```
divi_modules <- igraph_to_infomap_modules(colony1_data[32:36], 5)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 2 modules."
```

```
plot_persistance(divi_modules, 5, 20)
```



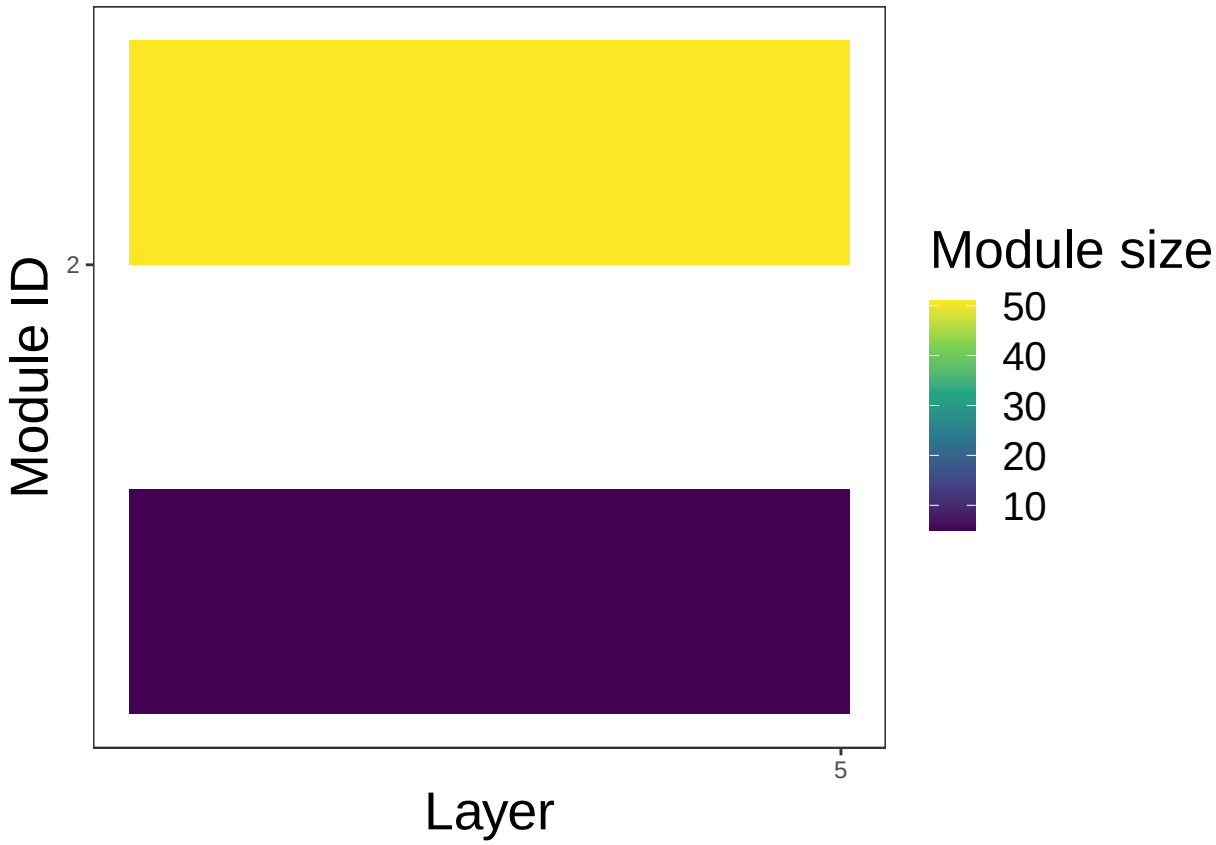

```
plot_alluvial(divi_modules, 5, 120)
```



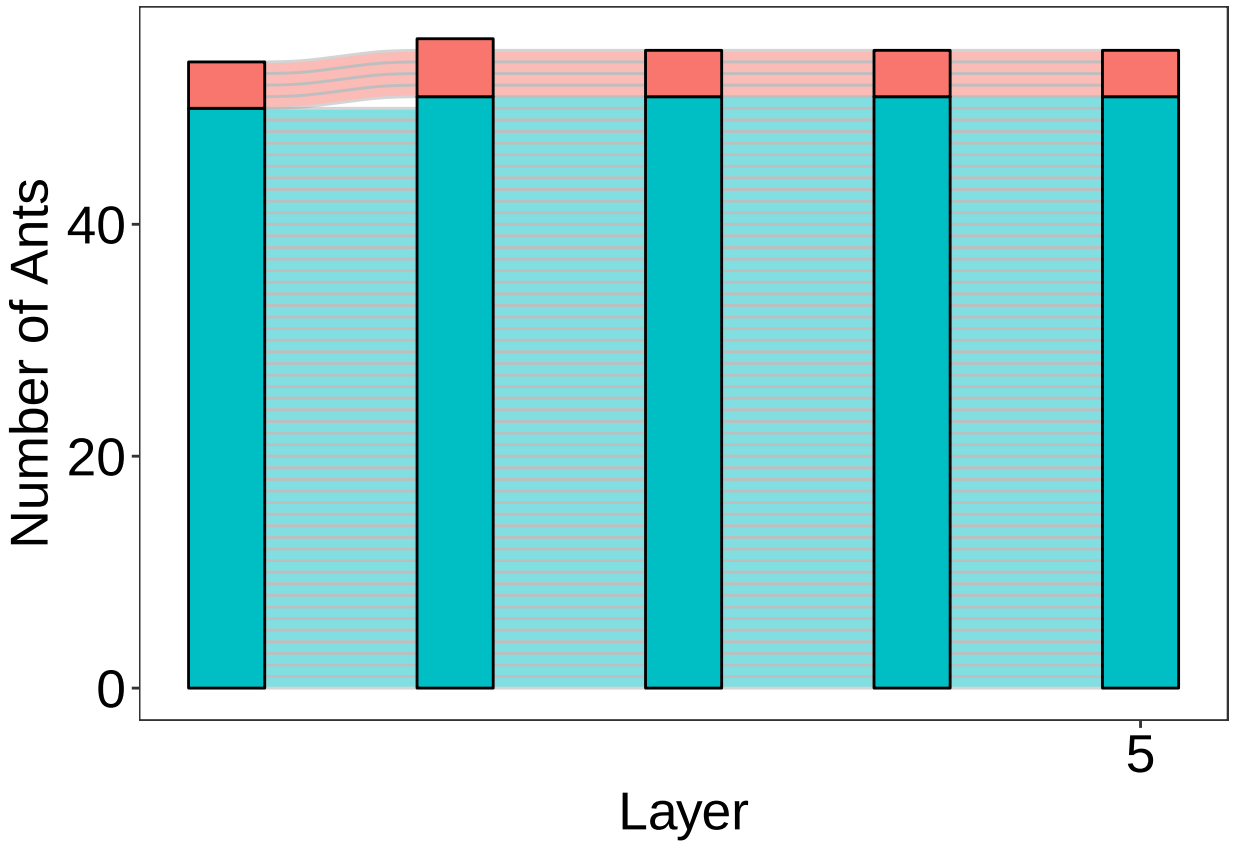

```
divi_modules <- igraph_to_infomap_modules(colony1_data[37:41], 5)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 2 modules."
```

```
plot_persistance(divi_modules, 5, 20)
```




```
plot_alluvial(divi_modules, 5, 120)
```

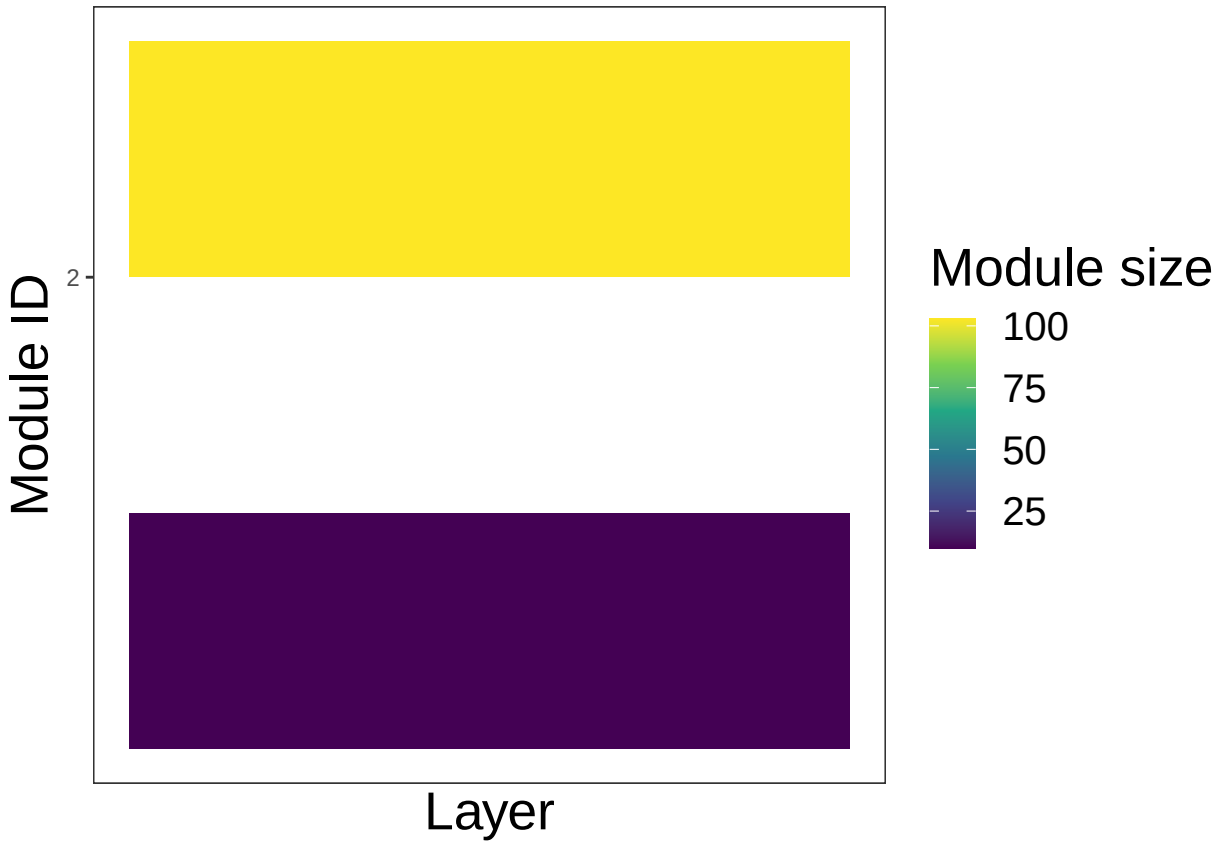


Double Layer Networks

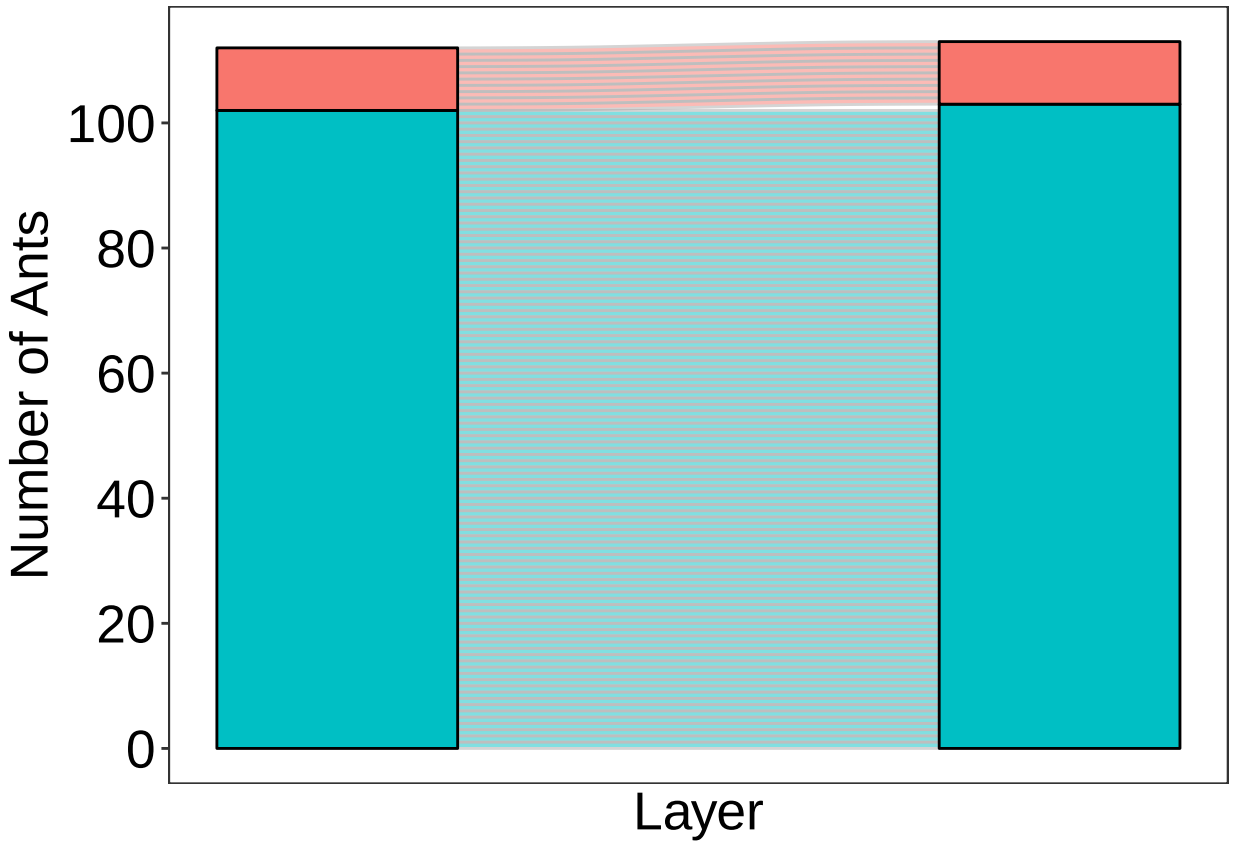
```
divi_modules <- igraph_to_infomap_modules(colony1_data[1:2], 2)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 2 modules."
```

```
plot_persistance(divi_modules, 2, 20)
```



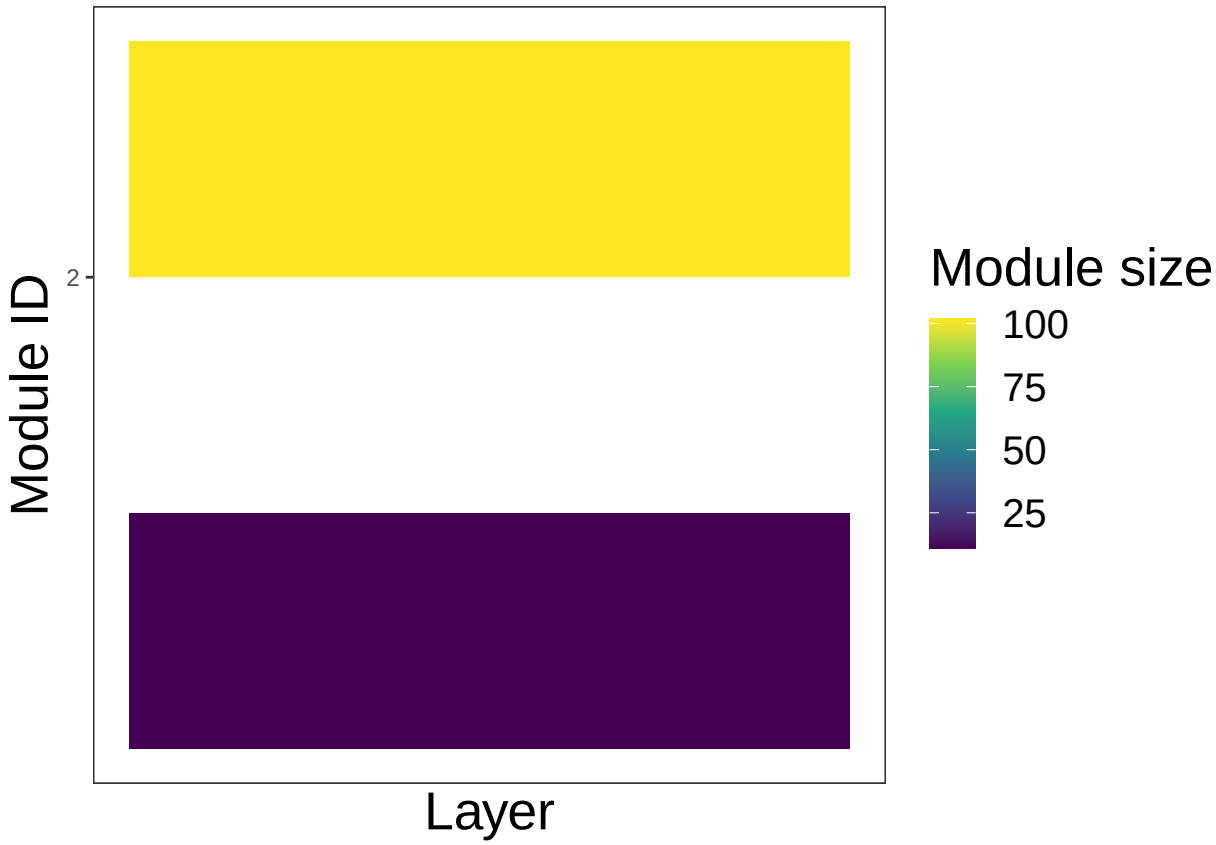

```
plot_alluvial(divi_modules, 2, 120)
```



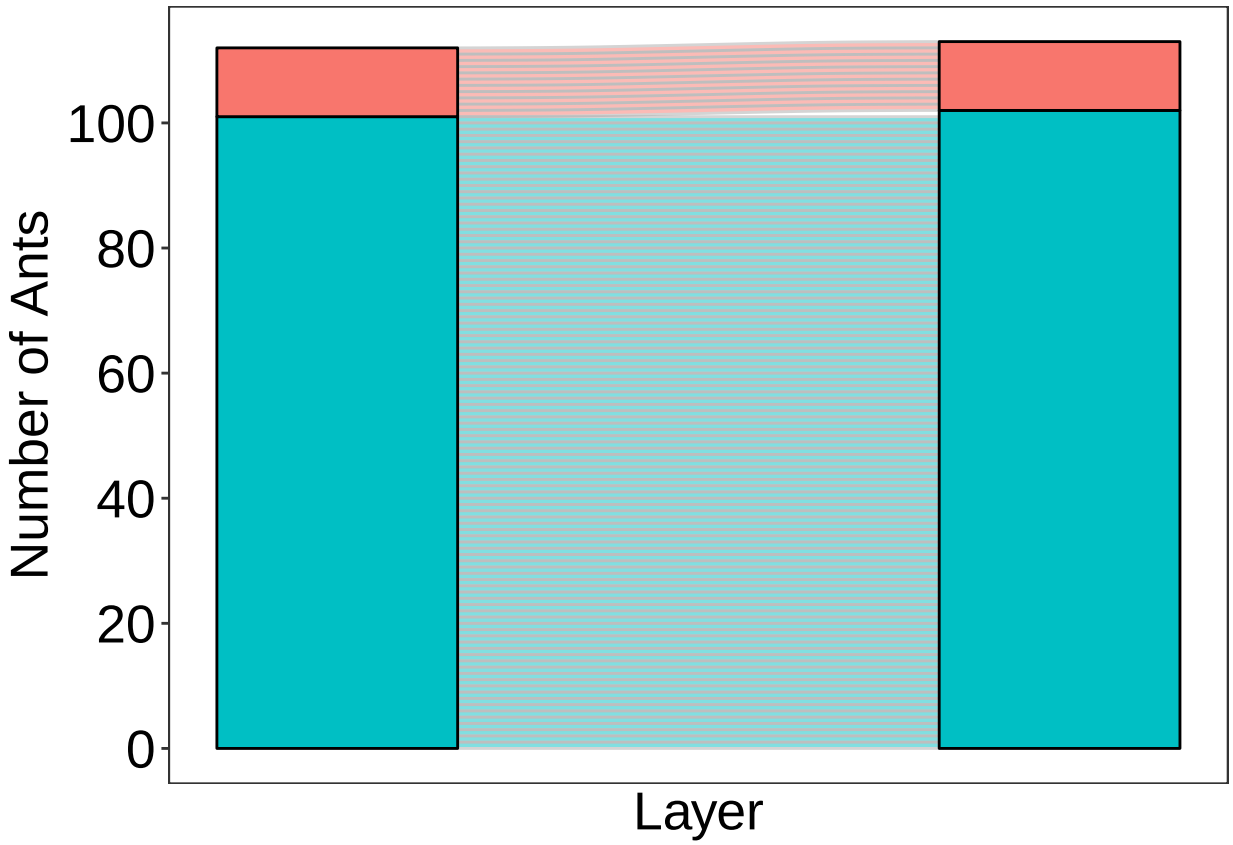

```
divi_modules <- igraph_to_infomap_modules(colony1_data[2:3], 2)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 2 modules."
```

```
plot_persistance(divi_modules, 2, 20)
```



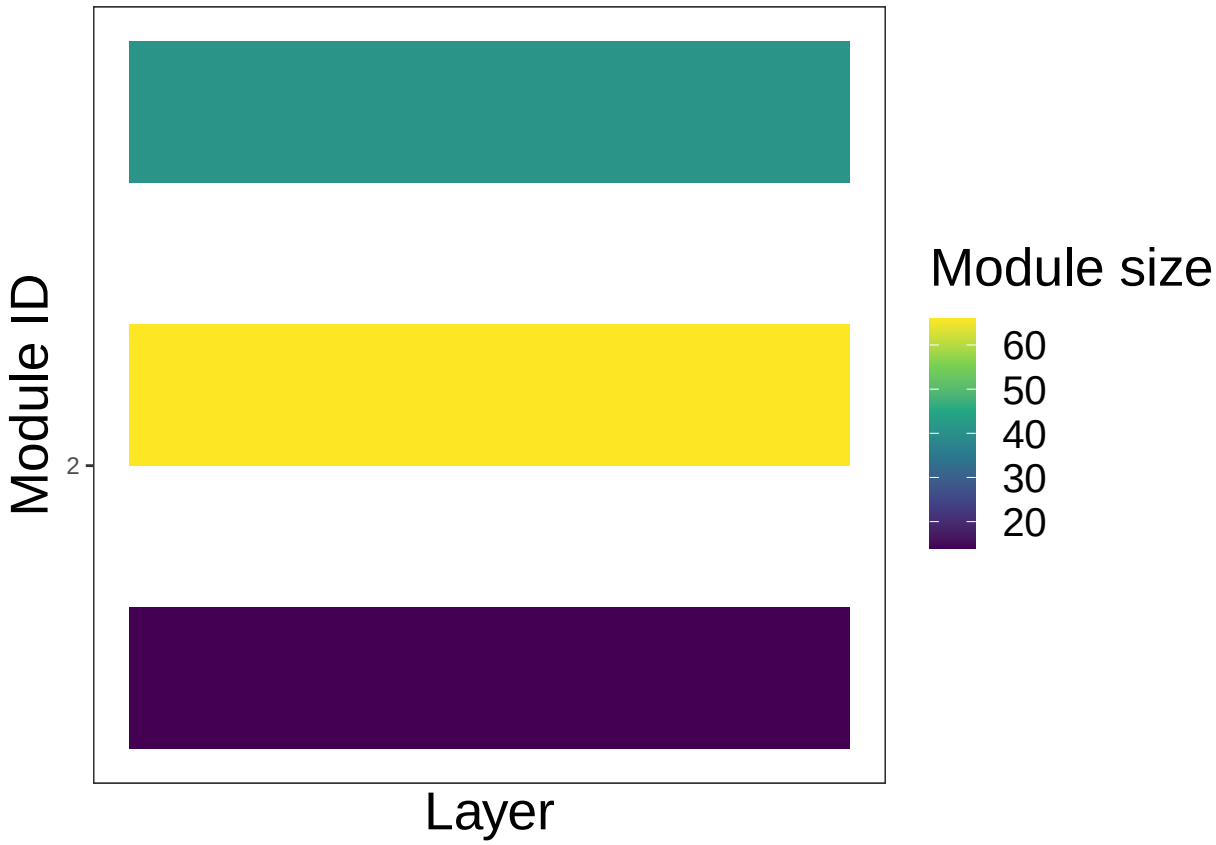

```
plot_alluvial(divi_modules, 2, 120)
```



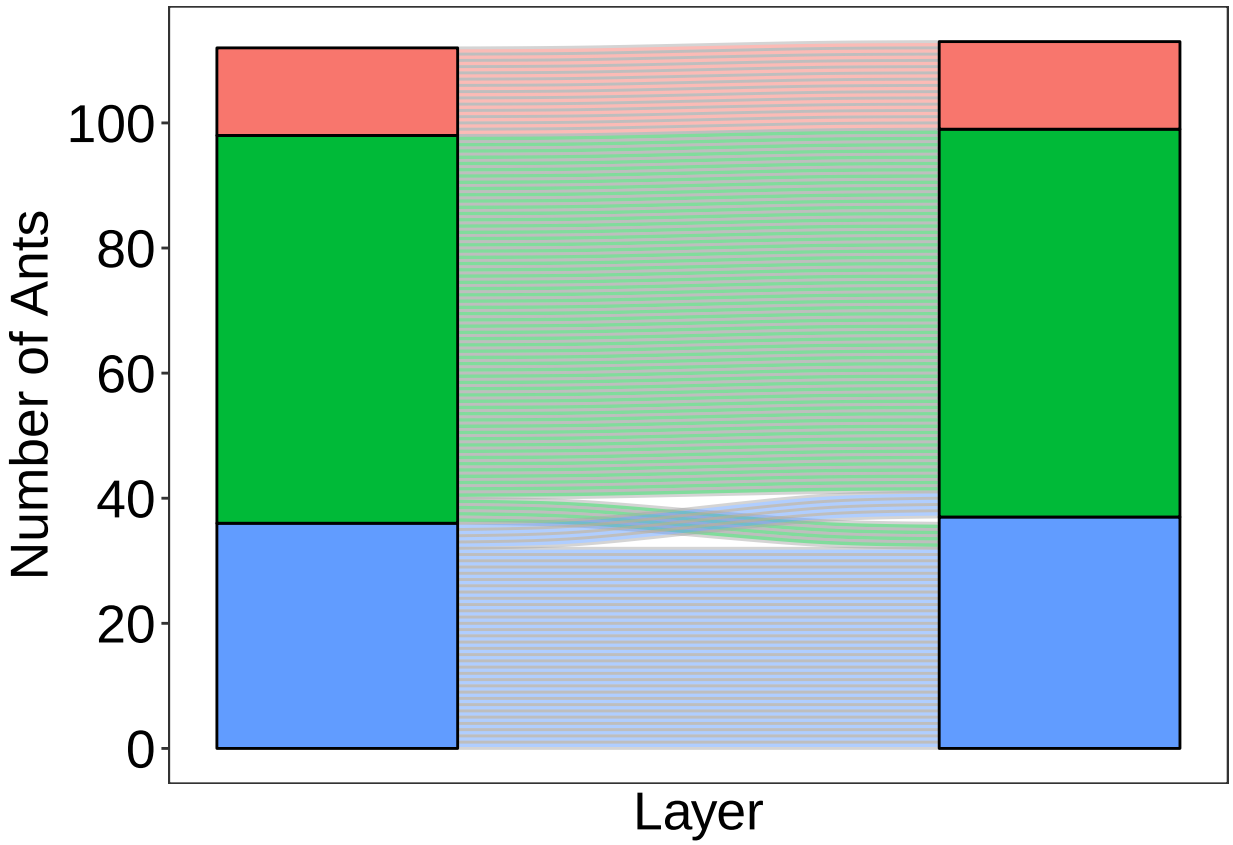

```
divi_modules <- igraph_to_infomap_modules(colony1_data[3:4], 2)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 3 modules."
```

```
plot_persistance(divi_modules, 2, 20)
```



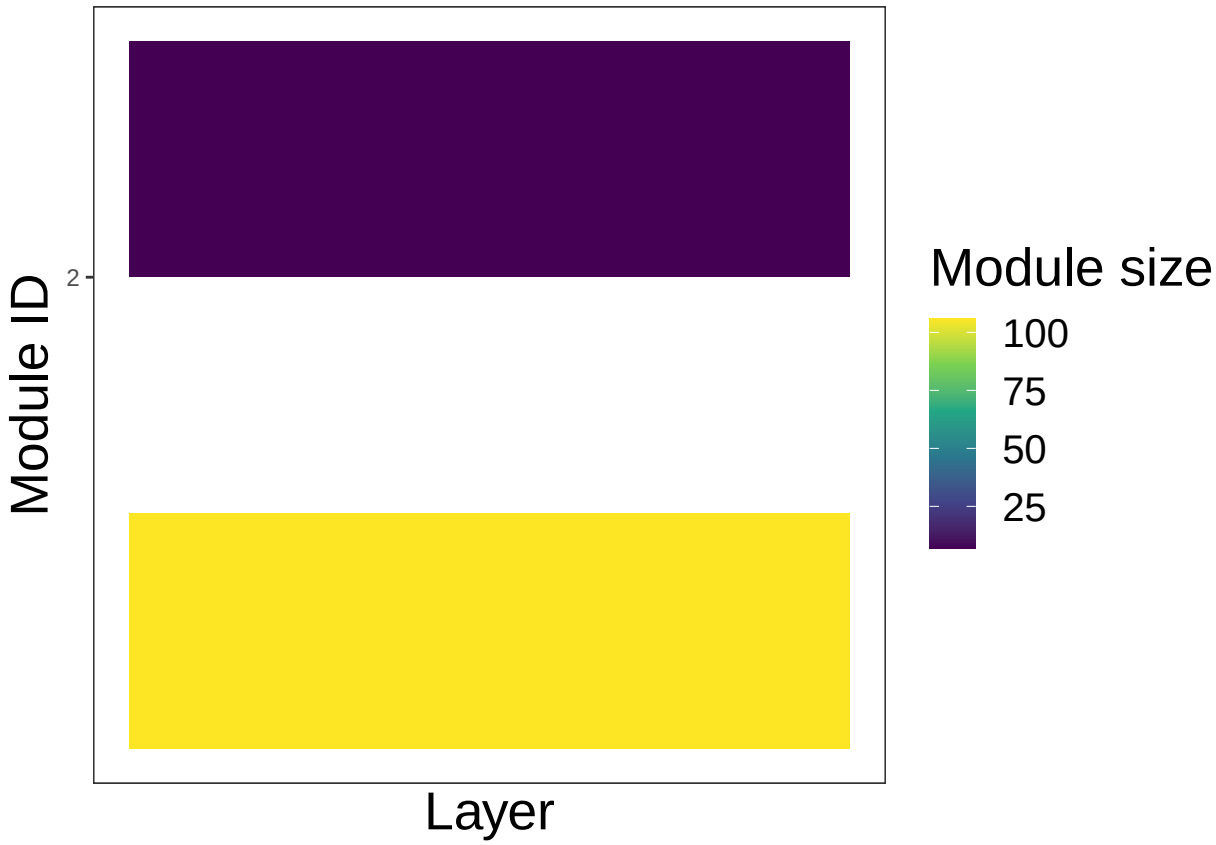

```
plot_alluvial(divi_modules, 2, 120)
```



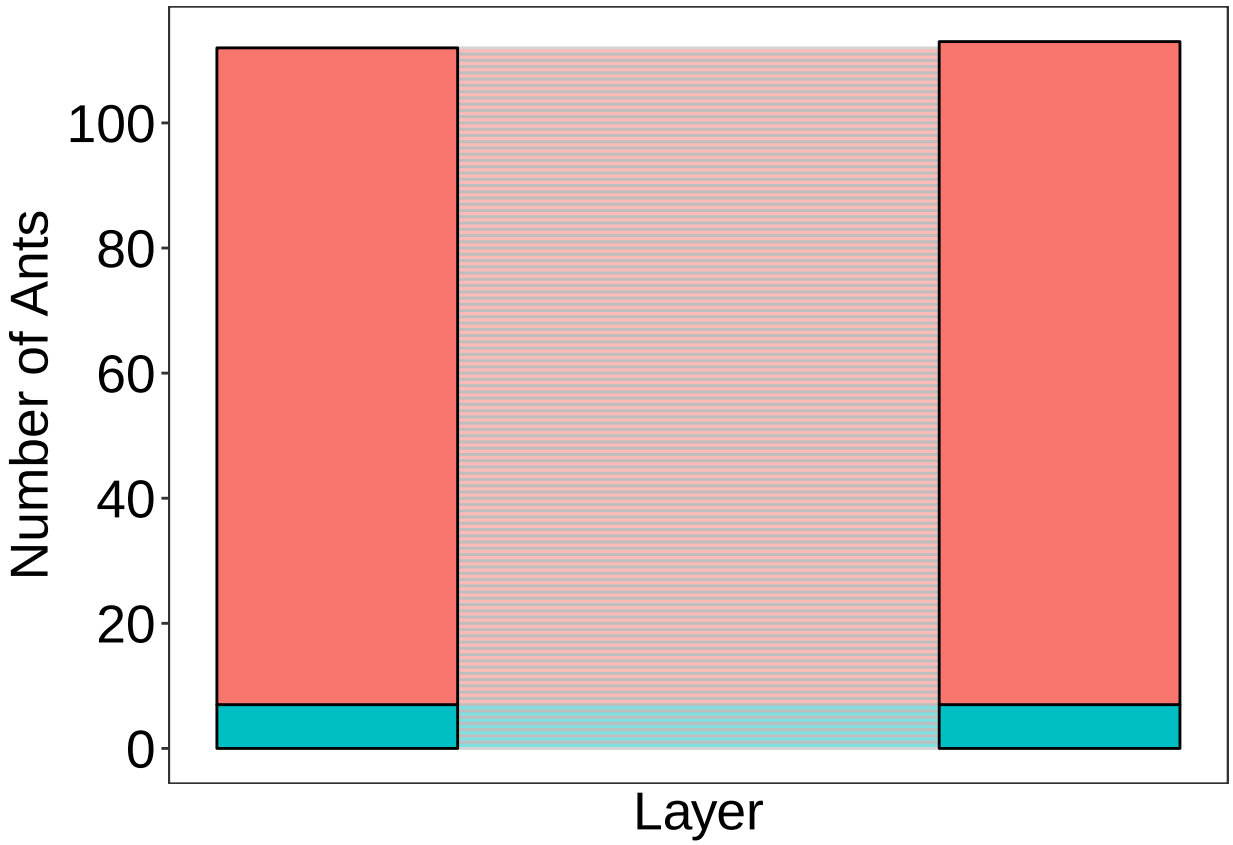

```
divi_modules <- igraph_to_infomap_modules(colony1_data[4:5], 2)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 2 modules."
```

```
plot_persistance(divi_modules, 2, 20)
```



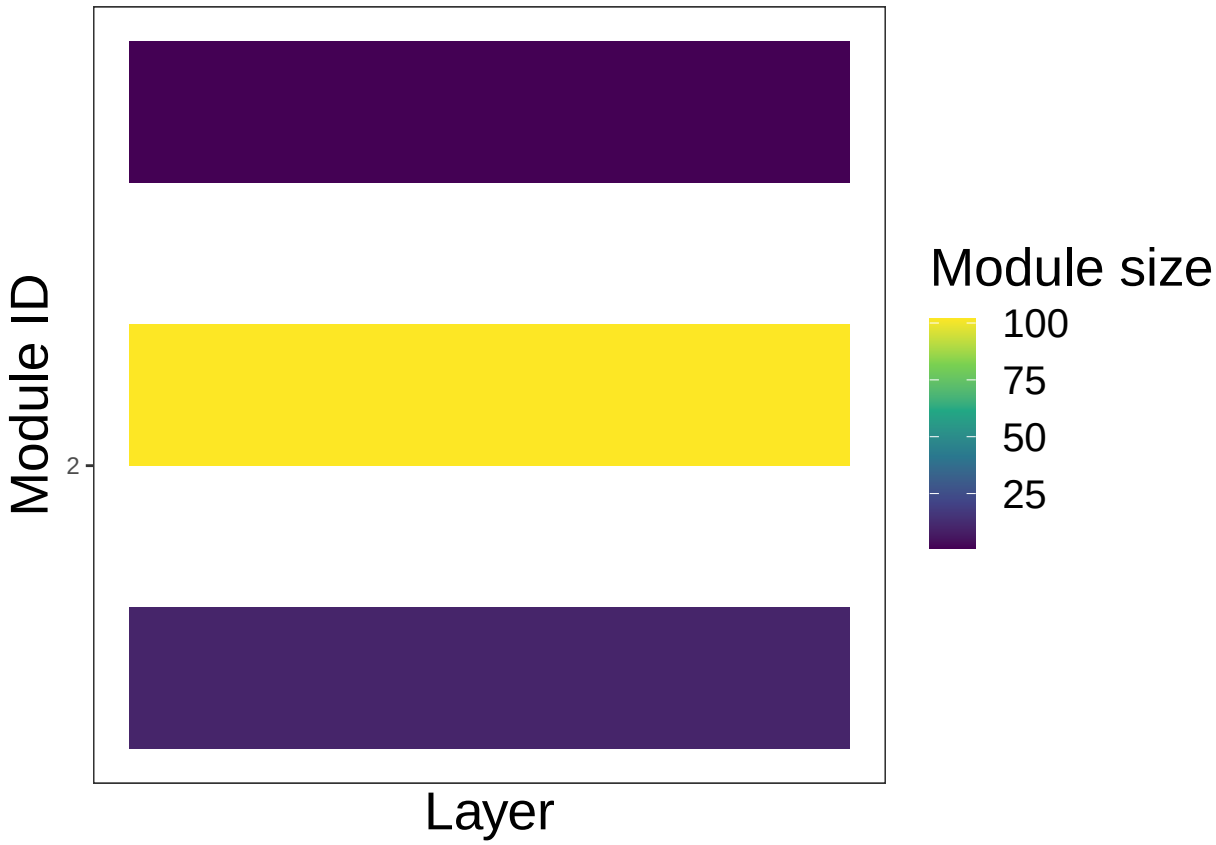

```
plot_alluvial(divi_modules, 2, 120)
```



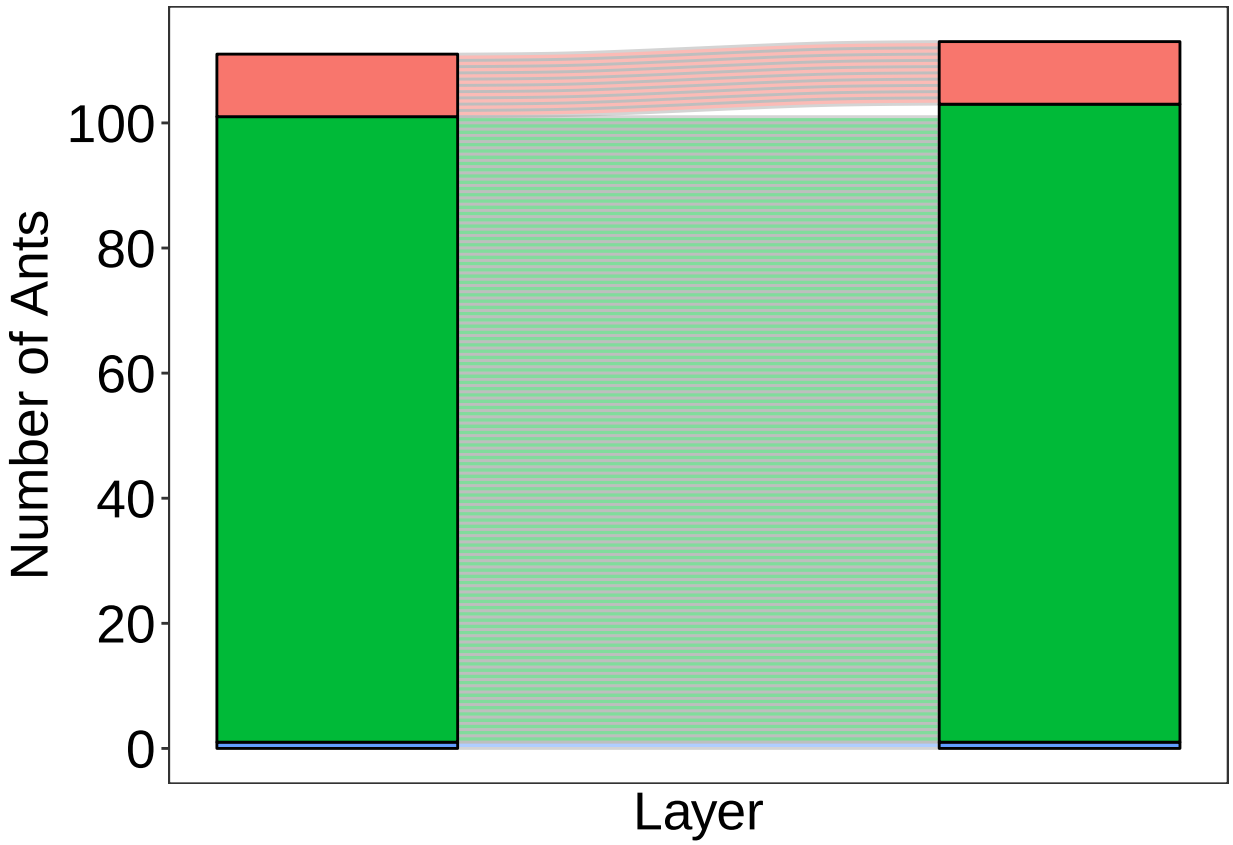

```
divi_modules <- igraph_to_infomap_modules(colony1_data[5:6], 2)
```

```
## [1] "Using interlayer edge values to determine flow between layers."
## [1] "./Infomap infomap_multilayer.txt . -i multilayer --tree -2 -N 100 --seed 497294 -f directed --si
## [1] "Reorganizing modules..."
## [1] "Removing auxilary files..."
## [1] "Partitioned into 3 modules."
```

```
plot_persistance(divi_modules, 2, 20)
```




```
plot_alluvial(divi_modules, 2, 120)
```



Monolayer Approach

Note that this code is a template of the monlayer adaptation of this method, but is not complete.

```
igraphs_to_monolayer_dfs <- function (igraph_list, num_elements){

  # Create empty lists
  #colony_layers <- vector(mode = "list", length = num_elements)

  # Loop over layers
  #for (i in 1:num_elements){
  #  adj_mtrx <- as_adjacency_matrix(igraph_list[[i]])

    # Pull out the edge lists from each layer
    #colony_layers[[i]] <- create_monolayer_object(x = adj_mtrx, directed = F, bipartite = F)

    #Assign column labels
    #names(colony_edges[[i]]) <- c('node_from', 'node_to', 'weight')


  #}

  #return (colony_layers)
}

#colony1_monolayers <- igraphs_to_monolayer_dfs(colony1_data, 41)
```