Ring of Cliques: construction, labels, and sanity checks

Overview

This vignette shows how to create **Ring-of-Cliques (ROC)** test graphs using `make_ring_of_cliques()` with four variants:

- **RC**
- **RC_B** — ring bridges between adjacent cliques
- **RC_C** — one central node connected to each clique
- **RC_BC** — both ring bridges **and** central node

Each clique is a ground-truth community. The function annotates vertices with: - `gt_community` (numeric), `gt_label` (character), - `clique_id`, `within_id`, - role flags (`role`, `is_bridge_endpoint`, `is_center_endpoint`).

Graphs are constructed deterministically: all clique vertices come first, the optional center (if any) is last.

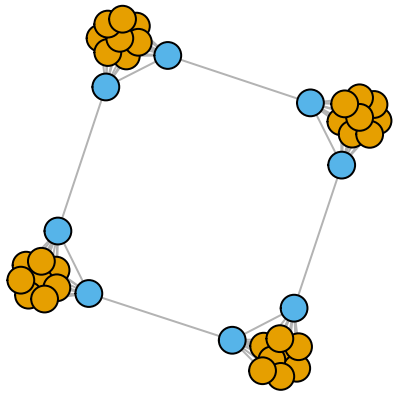
Quick start

```
library(tidyverse)
#> Warning: il pacchetto 'ggplot2' è stato creato con R versione 4.4.3
#> -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
#> v dplyr     1.1.4     v readr     2.1.5
#> v forcats   1.0.0     v stringr   1.5.1
#> v ggplot2   3.5.2     v tibble    3.2.1
#> v lubridate 1.9.4     v tidyr     1.3.1
#> v purrr     1.0.2
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
library(knitr)   # for kable()
library(igraph)
#>
#> Caricamento pacchetto: 'igraph'
#>
#> I seguenti oggetti sono mascherati da 'package:lubridate':
#>
#>     %--%, union
#>
#> I seguenti oggetti sono mascherati da 'package:dplyr':
#>
#>     as_data_frame, groups, union
#>
#> I seguenti oggetti sono mascherati da 'package:purrr':
#>
#>     compose, simplify
```

```
#>
#> Il seguente oggetto è mascherato da 'package:tidyr':
#>
#>     crossing
#>
#> Il seguente oggetto è mascherato da 'package:tibble':
#>
#>     as_data_frame
#>
#> I seguenti oggetti sono mascherati da 'package:stats':
#>
#>     decompose, spectrum
#>
#> Il seguente oggetto è mascherato da 'package:base':
#>
#>     union
library(communities)
```

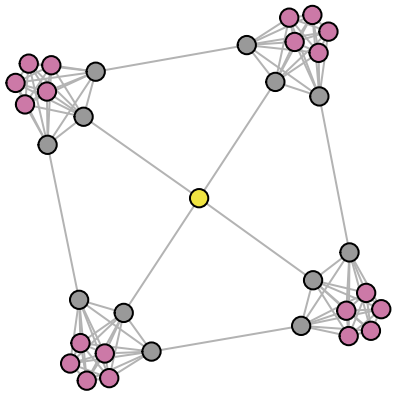
```
g  <- communities::make_ring_of_cliques(
    n_cliques = 4,
    clique_size = 10,
    variant = "RC"
)

plot(
  g,
  vertex.color = degree(g),
  vertex.label = NA,
  vertex.size = 15,
  edge.color = "grey70",
)
```



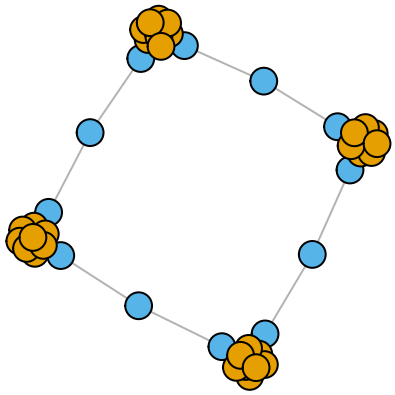
```r
g  <- communities::make_ring_of_cliques(
    n_cliques = 4,
    clique_size = 8,
    variant = "RC_C"
)

plot(
  g,
  vertex.color = degree(g),
  vertex.label = NA,
  vertex.size = 10,
  edge.color = "grey70",
)
```



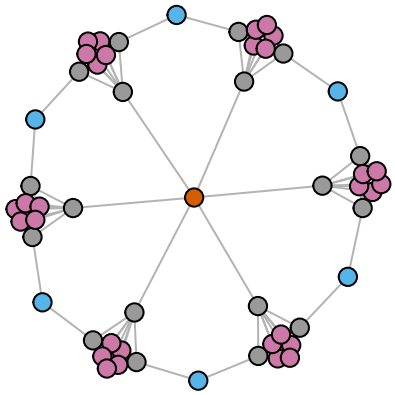
```r
g  <- communities::make_ring_of_cliques(
    n_cliques = 4,
    clique_size = 10,
    variant = "RC_B"
)

plot(
  g,
  vertex.color = degree(g),
  vertex.label = NA,
  vertex.size = 15,
  edge.color = "grey70",
)
```



```r
g  <- communities::make_ring_of_cliques(
    n_cliques = 6,
    clique_size = 8,
    variant = "RC_BC"
)

plot(
  g,
  vertex.color = degree(g),
  vertex.label = NA,
  vertex.size = 10,
  edge.color = "grey70",
)
```



Ground-truth labels

Each clique is assigned to a "truth community"; outliers are assigned to community "0":

```
g  <- communities::make_ring_of_cliques(
    n_cliques = 6,
    clique_size = 8,
    variant = "RC_BC"
)
# Each node belongs to a community identified by its "gt_community" attribute.
# Here we count how many nodes are in each community.
comm_dist <- tibble(
  community = V(g)$gt_community
) %>%
  count(community, name = "n_nodes") %>%
  arrange(desc(n_nodes))

kable(comm_dist, caption = "Distribution of ground-truth communities")
```

Table 1: Distribution of ground-truth communities

community	n_nodes
1	8
2	8
3	8
4	8

community	n_nodes
5	8
6	8
0	7

```
# --- Outlier nodes (if present) ---
# Nodes with gt_community = 0 are considered outliers and are labelled "CENTRAL".
# We extract and display only these nodes for inspection.
outliers <- tibble(
  node = V(g)$name,
  community = V(g)$gt_community
) %>%
  filter(community == 0)

kable(outliers, caption = "List of outlier nodes (community = 0)")
```

Table 2: List of outlier nodes (community = 0)

node	community
bridge1	0
bridge2	0
bridge3	0
bridge4	0
bridge5	0
bridge6	0
center	0

You can inspect roles and edge types:

```
sort(table(V(g)$role))
#>
#> central  bridge  clique
#>       1       6      48
sort(table(E(g)$edge_type))
#>
#> center_spoke  bridge_edge intra_clique
#>            6           12          168
```

Takeaways

- Each clique is annotated with a unique ground-truth label.
- Variants (RC_B, RC_C, RC_BC) change the meso-structure without altering the inner cliques.
- The construction is deterministic and idempotent: given the same parameters, you always obtain the same graph structure and vertex order.

This makes the functions suitable for benchmarking community detection algorithms under controlled conditions.