

Many networks : : CHEAT SHEET

Make, manipulate, and modify many different classes, types, and formats of networks.



🔧 Making networks

Import & export: read_*(), write_*()

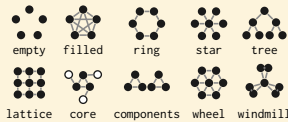
files $\xrightarrow{\text{read}_*()}$ stocnet $\xrightarrow{\text{write}_*()}$ files

Formats: edgelist, nodelist, matrix (xlsx/csv), pajek, ucinet, graphml (all read & write); gml, gdf, dynetml (read only).

read_*() with empty parentheses opens a file browser. See also collect_cran() for CRAN dependency networks and collect_ego() to collect ego networks by interview prompts.

Create deterministic structures

create_*() functions always return the same structure for the same n:



Also create_cycle(), create_degree(), create_motifs(), and create_explicit(A+B) to type out small networks by hand.

Generate stochastic mechanisms

generate_*() runs differ each time – useful for null distributions:

- generate_random(n, p): ties arise with probability p
- generate_configuration(n): given a degree distribution
- generate_man(n): given a Mutual/Asymmetric/Null dyad census
- generate_smallworld(n, p): ring rewired with probability p
- generate_scalefree(n, p): preferential attachment
- generate_fire(), generate_islands(), generate_citations()

Two-mode is easy, give n two integers: create_ring(c(3,2)) $\rightarrow$

Practice on included data

Three families of fully documented, ready-to-analyse datasets:

- ison_* — classic & instructional (ison_karateka, ...)
- fict_* — fictional (fict_thrones, fict_lotr, ...)
- irps_* — international politics (irps_blogs, ...)

table_data() gives an overview of all datasets and their formats.

Describe in prose

describe_*() functions return a readable sentence about a network, its nodes, ties, or changes — the same summaries shown when you print() a stocnet:

describe_network(), describe_nodes(),
describe_ties(), describe_changes()

User interaction

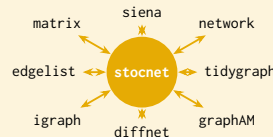
All functions operate with sensible defaults, only asking/expecting user input where necessary.

Decisions taken that affect interpretation are reported in the console. Quieten guidance with options(snet_verbosity = "quiet").

⚙️ Manipulating networks

Translate between classes: as_*()

Lossless coercion from and to many classes:



as_stocnet(), as_igraph(), as_tidygraph(), as_network(), as_matrix(), as_edgelist(), as_siena(), as_diffnet().

Every {manynet} function calls these coercions internally, so all functions work on any of these classes. Extract component tables with as_nodelist(), as_changelist(), as_infolist().

Wrangle with a dplyr-style grammar

Familiar verbs, suffixed by what they act on (_nodes, _ties, _changes, _globals):

	nodes	ties	changes	globals
add_ / delete_	●	●	● ^d	
bind_ / join_	●	●		
mutate_ / rename_	●	●	● ^b	●
filter_ / select_	●	●	●	● ^s
arrange_			●	
summarise_		●		

^d delete_only; ^b bind_only; ^s select_only.

E.g. net |> mutate_ties(weight = 1) |> filter_nodes(mode == 1)

Access & assign attributes

Get vectors out: node_names(), node_attribute(net, "x"), tie_weights(), tie_signs(), tie_attribute(net, "y"); mode_names(), layer_names().

Put vectors in: add_node_attribute(net, "x", vec), add_tie_attribute(), or tidily via mutate_nodes(net, x = vec).

Read off metadata: net_*()

Dimensions and counts as single values: net_nodes(), net_ties(), net_dims(), net_modes(), net_layers().

Names of attributes present: net_node_attributes(), net_tie_attributes(), net_attributes(), net_name().

Mark any network: is_*()

is_*() marks return a single TRUE/FALSE:

- format: is_directed is_weighted is_signed is_multiplex is_twomode is_labelled is_complex is_longitudinal is_dynamic is_changing
- features: is_connected is_acyclic is_aperiodic is_eulerian is_perfect_matching

Node/tie-level variants return logical vectors, e.g. node_is_mode(), tie_is_twomode().

🔧 Modifying networks

Reformat: same nodes, new format

to_*() functions work on any class and return the same class.

- $\bullet \rightarrow \bullet$ to_directed() $\bullet \leftarrow \bullet$ to_undirected()
- also to_redirected() (swap), to_reciprocated(), to_acyclic()
- weights & signs: to_weighted(), to_unweighted(), to_signed(), to_unsigned()
- names: to_named(), to_unnamed(); loops & layers: to_simplex(), to_uniplex(net, tie)
- to_anti() — complement; to_permuted() — shuffled

Transform: new dimensions

Scope in on part of a network: to_ego(net, node), to_giant(), to_no_isolates(), to_subgraph(net, cond), to_blocks(net, membership).

Keep only special tie sets: to_tree(), to_matching(), to_mentoring(), to_dominating(), to_eulerian().

to_correlation()/to_cosine() return nodal similarity matrices.

Join & split: from_*(), to_*(s)

Plural to_*() functions split a network into a list of networks, and from_*() functions join them back up:

list $\xrightarrow{\text{from_egos}()}$ net $\xrightarrow{\text{to_egos}()}$ list

- to_egos() / from_egos(): ego networks
- to_subgraphs(net, attr) / from_subgraphs()
- to_components(): one network per component
- to_waves() / from_waves(): longitudinal panels
- to_slices(net, times) / from_slices(): continuous time
- from_ties(): join tie types into a multiplex network

A typical pipeline

```
ison_southern_women |>
  to_mode1() |>
  mutate_nodes(deg = netrics::node_by_degree(net)) |>
  filter_nodes(deg > 2) |>
  autograph::graphr()
```

Wrangle tidily, every step works on any class, then hand off to {autograph} / {netrics}.

The stocnet suite: Where to go next

- {manynet} is for network wrangling; making, manipulating, and modifying networks
- {autograph} is for network visualization; plotting and theming networks and results
- {netrics} is for network analysis; marks (logical), measures (numeric), and membership (categorical)
- {migraph} imports all of the above and adds some more features
- {RSiena} is for longitudinal network modelling
- {goldfish} is for dynamic network modelling
- {MoNAn} is for mobility network modelling

Build a stocnet with {manynet}, then move on.