Supplemental Material for Environmental connectivity influences the origination of adaptive processes

John Shea, Sydney Leither, Max Foreback, Emily Dolson, and Alexander Lalejini

2024-03-28

Contents

Chapter 1

Introduction

This is the supplemental material for our manuscript submitted to the 2024 Artificial Life Conference. This is not intended as a stand-alone document, but as a companion to our main manuscript.

1.1 About our supplemental material

As you may have noticed (unless you're reading a pdf version of this), our supplemental material is hosted using GitHub pages. We compiled our data analyses and supplemental documentation into this web-accessible book using bookdown.

The source code and configuration files for this supplemental material can be found in this GitHub repository.

Our supplemental material includes the following:

- Data availability (Section 2)
- Local compilation instructions (Section 3)
- Graphs (Section 4)
- Graph properties (Section 5)
- Summary of literature review on how different spatial structures affect evolutionary adaptation (Section 6)
- Transitionability score analyses (Section 7)
- Graph properties correlation analyses (Section 8)

1.2 Contributing authors

- John Shea

- Sydney Leither
- Max Foreback
- Emily Dolson
- Alexander Lalejini

Chapter 2

Data availability

2.1 Source code

The source code for this work is publicly accessible on GitHub: https://github.com/amlalejini/alife-2024-spatial-chem-eco.

2.2 Experimental results

Data generated from our experiments used in analyses are available online, archived in an OSF repository: https://osf.io/k3d8g/)

Chapter 3

Local compilation

You will need a C++ compiler that supports at least C++17. We used g++13 for all local compilations.

First, clone the `alife-2024-spatial-chem-eco` repository, which contains the code needed to run our experiment software: https://github.com/amlalejini/alife-2024-spatial-chem-eco

Once cloned, `cd` into your local repository directory. Then, initialize and update all of the git submodules:

```
git submodule update --init --recursive
```

This will download the correct version of the `chemical-ecology` repository into the `third-party` directory, which contains the implementation of the artificial ecology model that we used in our experiments. Specifically, we used this version of the `chemical-ecology` code base:

- https://github.com/amlalejini/chemical-ecology/tree/2024-01-09-spatial-struct-exp
  - Commit hash: `9a7022c238e04103bad2e399477b7f9bbe2ec9f4`

To compile the model:

```
cd third-party/chemical-ecology
make native
```

This will create an executable `chemical-ecology`.

Once you have an executable, you can generate a configuration file by running:

```
./chemical-ecology --gen chemical-ecology.cfg
```

You may also use the configuration files from any of our experiments, which can be found in the `experiments` directory.

3.1 Python dependencies

Many of the scripts that we used to manage experiments on our computing cluster, aggregate data, and run analyses are written in Python. The dependencies for these scripts are given in the `requirements.txt` file. We recommend setting up a Python virtual environment and installing the dependencies there:

```
python -m venv pyenv
pip install -r requirements
```

Chapter 4

Graph structures

We used ten different graph types to define spatial structures in our experiments. The exact graphs used in our experiments are given in this repository: `experiments/2024-03-08-varied-interaction-matrices/hpc/config/spatial-structures/`

We include descriptions and visualizations of each graph type below. All graph visualizations of our spatial structures can be found in `docs/graph-visualizations/`.

For graphs generated with a stochastic graph generation algorithm, we generated 20 graphs (one per replicate). Each experiment used those 20 graphs for each replicate of that particular spatial structure regime. Below, we include a representative visualize of each.

The code used to generate the graphs used in this work can be found in this repository in `scripts/SpatialStructure.py`.

4.1 Well-mixed (fully connected)

A fully connected graph where each vertex is connected to all other vertices.

4.2 Toroidal lattice

Vertices are organized into a toroidal grid where each vertex is connected to its four neighboring vertices. The vertices in the top and bottom rows and left and right columns are connected, respectively.

This graph type is very commonly used in Artificial Life systems.

4.3 Linear chain

Vertices are organized into a linear chain, where each vertex is connected to its two neighbors.

4.4 Cycle

A linear chain graph, but the vertices at the two ends of the chain are connected.

4.5 Wheel

A single hub vertex is connected to all vertices in a cycle comprising all other vertices in the graph.

4.6 Star

A tree with one internal vertex, and all other vertices are leaves connected to the single internal vertex.

4.7 Windmill

A graph with n size-k cliques that each share a single “hub’ ’ vertex. For this work, n=10 and k=10.

4.8 Comet-kite

A graph comprising a large fully connected set of core nodes with randomly attached “tail’ ’ nodes. To generate a comet-kite graph, we construct a fully connected core, select t random nodes from the core to attach initial tail nodes to, and then sequentially attach additional nodes to randomly chosen tail nodes. In this work, we used a core size of 40 nodes, attached 20 initial tail nodes, and added 40 additional tail nodes.

4.9 Random Barabasi-Albert

A randomly generated, scale-free graph is constructed by sequentially attaching new nodes with m edges, which are preferentially connected to existing nodes with high degree.

4.10 Random Waxman

A randomly generated graph is constructed by placing nodes uniformly at random in a 2-dimensional space. Each pair of nodes distance d from one another are connected with probability $p = \beta e^{-d/\alpha L}$. In this work, we used $\beta = 0.4$ and $\alpha = 0.2$.

Chapter 5

Graph properties

We screened for properties of spatial structures that correlated with transitionability scores. We included the following 21 graph properties in these analyses:

- Density - The density, d, of a graph is given by $d = \frac{2m}{n(n-1)}$ where n is the number of nodes and m is the number of edges in the graph.
- Mean degree - Average degree of all nodes in the graph.
- Median degree - Median degree of all nodes in the graph.
- Variance degree - Variance in degree values for all nodes in the graph.
- Girth - The girth of a graph is the length of the shortest cycle in the graph.
- Degree assortativity coefficient - Also known as assortative mixing. Measures the tendency for a graph's nodes to attach to others with a similar degree.
- Number of bridges - Number of "bridges" in the graph. A bridge is an edge that, if deleted, would increase the graph's number of connected components.
- Max clique size - Size of the largest clique (fully connected component) in the graph.
- Transitivity - The fraction of all possible triangle structures present in the graph.
- Average clustering - Estimate of the graph's clustering coefficient.
- Number of connected components
- Number of articulation points - A node is an articulation point if removing that node and all of its edges would disconnect the graph.
- Average node connectivity - Average local connectivity of nodes in the graph.
- Edge Connectivity - The edge connectivity of the graph is the minimum number of edges that must be removed to disconnect the graph.
- Node Connectivity - The minimum number of nodes that must be removed to disconnect the graph.

- Diameter - Maximum eccentricity of the graph. The eccentricity of each node in the graph is equal to the maximum distance from that node to all other nodes in the graph.
- Radius - Minimum eccentricity of nodes in the graph graph.
- Kemeny constant - The expected number of steps to transition from one node to a random other node in the graph. This measures the time needed for spreading across a graph: low values indicate a closely connected graph, whereas large values indicate a more diffuse graph.
- Global Efficiency - The average efficiency of the graph. The efficiency of a pair of nodes is the multiplicative inverse of the shortest path distance between the nodes.
- Wiener index - Sum of the shortest-path distances between each pair of reachable nodes.
- Longest shortest path - the maximum path length among all shortest paths between all pairs of nodes in the graph.

The code used to compute these properties for each graph structure used in this work can be found in this repository `scripts/graph-properties.py`. The majority of these properties were computed using the networkx library.

Chapter 6

Summary of spatial structure effects on evolutionary adaptation

This table summarizes the effects of different spatial structures from evolutionary graph theory literature.

This browser does not support PDFs. Please download the PDF to view it: Download PDF.

Chapter 7

Community transitionability analyses

7.1 Dependencies and setup

```
library(tidyverse)
library(cowplot)
library(RColorBrewer)
library(khroma)
library(rstatix)
library(knitr)
library(kableExtra)
library(ggh4x)
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce9
```

```
# Check if Rmd is being compiled using bookdown
bookdown <- exists("bookdown_build")
```

```
experiment_slug <- "2024-03-08-varied-interaction-matrices"
working_directory <- paste(
  "experiments",
  experiment_slug,
  "analysis",
  sep = "/"
)
# Adjust working directory if being knitted for bookdown build.
if (bookdown) {
  working_directory <- paste0(
    bookdown_wd_prefix,
    working_directory
  )
}

plot_dir <- paste(
  working_directory,
  "plots",
  sep = "/"
)

# Load summary data from final update
data_path <- paste(
  working_directory,
  "data",
  "world_summary_final_update.csv",
  sep = "/"
)
data <- read_csv(data_path)
```

Set cowplot theme as default plotting theme.

```
theme_set(theme_cowplot())
```

7.2 Data preprocessing

```
data <- data %>%
  mutate(
    interaction_matrix = as.factor(interaction_matrix),
    graph_type = as.factor(graph_type),
    summary_mode = as.factor(summary_mode),
    update = as.numeric(update),
    SEED = as.factor(SEED)
  )

# Separate proof-of-concept runs from other interaction matrics
# (we don't use the proof-of-concept in our analyses)
poc_data <- data %>% filter(interaction_matrix == "orig-pof")
data <- data %>%
  filter(interaction_matrix != "orig-pof") %>%
  mutate(
    im_connectance = case_when(
      str_detect(interaction_matrix, "c25") ~ "25",
      str_detect(interaction_matrix, "c50") ~ "50",
      str_detect(interaction_matrix, "c75") ~ "75"
    ),
    im_pip = case_when(
      str_detect(interaction_matrix, "pip25") ~ "25",
      str_detect(interaction_matrix, "pip50") ~ "50",
      str_detect(interaction_matrix, "pip75") ~ "75"
    )
  ) %>%
  mutate(
    im_connectance = as.factor(im_connectance),
    im_pip = as.factor(im_pip)
  )

# Ensure that we're isolating values from end-of-simulation.
max_update <- max(data$update)
final_update_data <- data %>%
  filter(update == max_update)
```

There are several different summarization methods supported by the chemical ecology model software. Here, we use the ranked-threshold metric.

```
rt_final_data <- final_update_data %>%
  filter(summary_mode == "ranked_threshold")

rt_final_data <- rt_final_data %>%
  mutate(
    Connectance = case_when(
      im_connectance == "25" ~ "0.25",
      im_connectance == "50" ~ "0.50",
      im_connectance == "75" ~ "0.75"
    ),
    PIP = case_when(
      im_pip == "25" ~ "0.25",
      im_pip == "50" ~ "0.50",
      im_pip == "75" ~ "0.75"
    )
  )
```

Calculate the median transitionability score (`logged_mult_score`) for each well-mixed regime.

```r
wm_median <- rt_final_data %>%
  filter(graph_type == "well-mixed") %>%
  dplyr::group_by(interaction_matrix, Connectance, PIP) %>%
  dplyr::summarize(wm_median = median(logged_mult_score))
```

7.3 Final transitionability scores

We visualize the final community transitionability scores (`logged_mult_score`) for each spatial structure regime across for each interaction network. For each interaction matric, we draw a vertical dashed line (black) to indicate median transitionability score achived in the well-mixed regime. This value serves as the baseline expectation in the absence of spatial structure.

Additionally, we draw a solid vertical line (red) to indicate 0 on the transitionability score axis. Transitionability scores greater than zero indicate that a community exhibited dynamics more closely resembling pure adaptive dynamics than pure ecological dynamics. Transitionability scores less than zero indicate that a community exhibited dynamics more closely resembling pure ecological dynamics than pure adaptive dynamics.

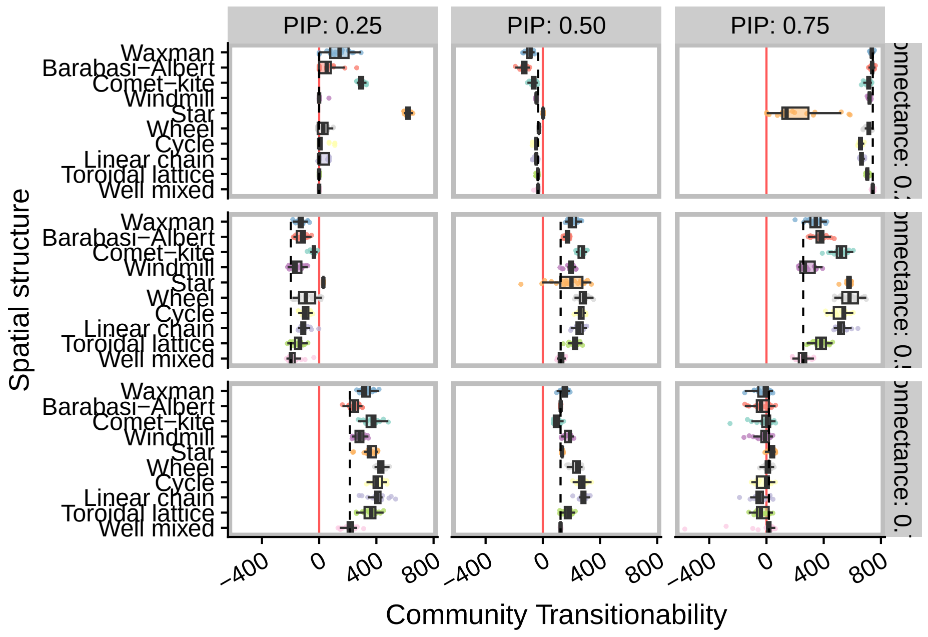
```r
# Provide explicit ordering for graph ticks/labels
graph_ticks <- c(
  "well-mixed",
  "toroidal-lattice",
  "linear-chain",
  "cycle",
  "wheel",
  "star",
  "windmill",
  "comet-kite",
  "random-barabasi-albert",
  "random-waxman"
)
graph_labels <- c(
  "Well mixed",
  "Toroidal lattice",
  "Linear chain",
  "Cycle",
  "Wheel",
  "Star",
  "Windmill",
  "Comet-kite",
  "Barabasi-Albert",
  "Waxman"
```

```
)

plot_final <- ggplot(
    rt_final_data,
    aes(
      x = graph_type,
      y = logged_mult_score,
      fill = graph_type
    )
  ) +
  geom_hline(
    yintercept = 0,
    color = "red",
    linetype = "solid",
    alpha = 0.65
  ) +
  geom_point(
    mapping = aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    outlier.shape = NA,
    alpha = 0.5
  ) +
  geom_hline(
    data = wm_median,
    aes(yintercept = wm_median),
    linetype = "dashed"
  ) +
  scale_color_brewer(palette = "Set3") +
  scale_fill_brewer(palette = "Set3") +
  scale_x_discrete(
    name = "Spatial structure",
    limits = graph_ticks,
    breaks = graph_ticks,
    labels = graph_labels
  ) +
  scale_y_continuous(
    name = "Community Transitionability"
  ) +
  ggh4x::facet_grid2(
    Connectance ~ PIP,
    labeller = label_both
  ) +
  coord_flip() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    ),
    panel.border = element_rect(color = "gray", size = 2)
  )

ggsave(
  paste(
    plot_dir,
    "final_ranked_thresh_logged_mult_score.pdf",
    sep = "/"
  ),
  plot = plot_final,
  width = 8.5,
  height = 8
)

plot_final
```



For reference, we generate a table of mean and median transitionability scores per-regime per-experiment:

```
summary_data <- rt_final_data %>%
  dplyr::group_by(interaction_matrix, graph_type) %>%
  dplyr::summarize(
    score_median = median(logged_mult_score),
    score_mean = mean(logged_mult_score),
    replicates = n()
  ) %>%
  arrange(score_median, .by_group = TRUE)

summary_table <- summary_data %>%
  kable() %>%
  kable_styling(
    latex_options = "striped"
  )
save_kable(
  summary_table,
  paste(
    plot_dir,
    "summary_table.pdf",
    sep = "/"
  )
)
summary_table
```

7.3.1 Statistical analyses

7.3.1.1 Transitionabiliy score distributions - Kruskal-Wallis test results

First, a Kruskal-Wallis test (per-interaction matrix) to test for significant differences in distributions across spatial structure regimes.

```
kw_test <- rt_final_data %>%
  group_by(interaction_matrix) %>%
  kruskal_test(logged_mult_score ~ graph_type) %>%
  mutate(sig = (p < 0.05))

kw_table <- kw_test %>%
  kable() %>%
  kable_styling(
    latex_options = "striped"
  )
```

interaction_matrix	graph_type	score_median	score_mean	replicates
c25pip25	windmill	-0.1552570	3.2755443	20
c25pip25	cycle	-0.1448880	23.2577198	20
c25pip25	linear-chain	-0.1448875	30.0107428	20
c25pip25	toroidal-lattice	-0.1398950	-0.1386467	20
c25pip25	well-mixed	-0.1274125	-0.1274118	20
c25pip25	wheel	29.1941680	29.2538536	20
c25pip25	random-barabasi-albert	52.5468000	58.7830525	20
c25pip25	random-waxman	142.6250000	132.2016943	20
c25pip25	comet-kite	290.7705000	294.3542000	20
c25pip25	star	622.3975000	621.4329500	20
c25pip50	random-barabasi-albert	-123.7055000	-128.4367700	20
c25pip50	random-waxman	-85.6997500	-93.6634050	20
c25pip50	comet-kite	-62.4832500	-64.5371050	20
c25pip50	cycle	-46.8019500	-48.0774750	20
c25pip50	linear-chain	-46.2323500	-47.4541800	20
c25pip50	windmill	-44.0345500	-44.8190350	20
c25pip50	toroidal-lattice	-34.1670000	-36.9710150	20
c25pip50	well-mixed	-32.6162000	-34.8772050	20
c25pip50	wheel	-27.7729000	-27.7023650	20
c25pip50	star	2.4022500	2.3551140	20
c25pip75	star	140.4695000	206.2041715	20
c25pip75	cycle	658.9540000	657.9878500	20
c25pip75	linear-chain	664.6090000	665.7244000	20
c25pip75	toroidal-lattice	703.8705000	705.6194500	20
c25pip75	wheel	715.5445000	713.2512500	20
c25pip75	comet-kite	717.4210000	711.4932000	20
c25pip75	windmill	720.5355000	719.9272500	20
c25pip75	random-waxman	736.0230000	736.1203000	20
c25pip75	random-barabasi-albert	739.2250000	738.5561500	20
c25pip75	well-mixed	743.1275000	742.8270000	20
c50pip25	well-mixed	-198.2860000	-179.9309250	20
c50pip25	windmill	-164.2615000	-158.1420900	20
c50pip25	toroidal-lattice	-136.0975000	-146.1914300	20
c50pip25	random-waxman	-128.2555000	-127.8362350	20
c50pip25	random-barabasi-albert	-119.6675000	-124.3501100	20
c50pip25	linear-chain	-109.6645000	-104.4384365	20
c50pip25	cycle	-96.3967500	-96.4360300	20
c50pip25	wheel	-93.1401500	-80.2479790	20
c50pip25	comet-kite	-35.9154000	-38.5166250	20
c50pip25	star	29.5336000	29.6897650	20
c50pip50	well-mixed	124.5965000	126.7944950	20
c50pip50	random-barabasi-albert	175.8165000	171.1758000	20
c50pip50	windmill	195.5930000	192.2814000	20
c50pip50	random-waxman	196.9120000	207.3342000	20
c50pip50	star	200.4315000	179.9736390	20
c50pip50	toroidal-lattice	224.4375000	221.0585000	20
c50pip50	linear-chain	254.4380000	257.4893000	20
c50pip50	cycle	260.7760000	265.9643500	20
c50pip50	comet-kite	280.7530000	272.8958000	20
c50pip50	wheel	286.9300000	282.9178000	20
c50pip75	well-mixed	256.7610000	254.8647500	20
c50pip75	windmill	268.4115000	284.2451000	20
c50pip75	random-waxman	344.5975000	337.5601000	20

interaction_matrix	.y.	n	statistic	df	p	method	sig
c25pip25	logged_mult_score	200	135.4711	9	0e+00	Kruskal-Wallis	TRUE
c25pip50	logged_mult_score	200	176.1459	9	0e+00	Kruskal-Wallis	TRUE
c25pip75	logged_mult_score	200	172.1451	9	0e+00	Kruskal-Wallis	TRUE
c50pip25	logged_mult_score	200	128.1468	9	0e+00	Kruskal-Wallis	TRUE
c50pip50	logged_mult_score	200	125.9629	9	0e+00	Kruskal-Wallis	TRUE
c50pip75	logged_mult_score	200	166.6052	9	0e+00	Kruskal-Wallis	TRUE
c75pip25	logged_mult_score	200	141.7244	9	0e+00	Kruskal-Wallis	TRUE
c75pip50	logged_mult_score	200	172.4309	9	0e+00	Kruskal-Wallis	TRUE
c75pip75	logged_mult_score	200	48.4034	9	2e-07	Kruskal-Wallis	TRUE

```r
save_kable(
  kw_table,
  paste(
    plot_dir,
    "kw_test_results.pdf",
    sep = "/"
  )
)
kw_table
```

7.3.1.2 Transitionability score distributions - Pairwise Wilcoxon rank-sum test results

Next, we perform pairwise Wilcoxon rank-sum tests for all significant comparison groups. We use a Holm-Bonferroni correction for multiple comparisons.

```r
# Grab group names of significant comparisons
sig_kw_groups <- filter(kw_test, p < 0.05)$interaction_matrix
# Perform pairwise rank-sum tests, adjust for multiple comparisons
wrs_test <- rt_final_data %>%
  filter(
    interaction_matrix %in% sig_kw_groups
  ) %>%
  group_by(interaction_matrix) %>%
  pairwise_wilcox_test(logged_mult_score ~ graph_type) %>%
  adjust_pvalue(method = "holm") %>%
  add_significance("p.adj")
# Build a pretty table
wrs_test_table <- kable(wrs_test) %>%
  kable_styling(
    latex_options = "striped"
  )
```

```
save_kable(
  wrs_test_table,
  paste(
    plot_dir,
    "wrs_test_results.pdf",
    sep = "/"
  )
)
wrs_test_table
```

7.4 Identify amplifiers or suppressors

Next, we categorize each spatial structure as an amplifier or suppressor based on its transitionability score relative to the well-mixed regime. Spatial structure regimes with scores that are greater than well-mixed (statistically significant) are categorized as amplifiers. Spatial structure regimes with scores that are lower than well-mixed (statistically significant) are categorized as suppressors. If we failed to detect a statistically significant difference between a spatial structure and well-mixed, we categorized it as "neither".

First, filter the pairwise tests to just those that involve the well-mixed regime.

```
wm_wrs_test_table <- wrs_test %>%
  filter(group1 == "well-mixed" | group2 == "well-mixed") %>%
  kable() %>%
  kable_styling(
    latex_options = "striped"
  )
save_kable(
  wm_wrs_test_table,
  paste(
    plot_dir,
    "wm_wrs_test_results.pdf",
    sep = "/"
  )
)
wm_wrs_test_table
```

For each spatial structure, identify amplifiers and suppressors.

```
int_matrices <- unique(as.character(wrs_test$interaction_matrix))
well_mixed_comps <- wrs_test %>%
  filter(group1 == "well-mixed" | group2 == "well-mixed") %>%
```

interaction_matrix	.y.	group1	group2	n1	n2	statistic
c25pip25	logged_mult_score	comet-kite	cycle	20	20	400.0
c25pip25	logged_mult_score	comet-kite	linear-chain	20	20	400.0
c25pip25	logged_mult_score	comet-kite	random-barabasi-albert	20	20	399.0
c25pip25	logged_mult_score	comet-kite	random-waxman	20	20	388.0
c25pip25	logged_mult_score	comet-kite	star	20	20	0.0
c25pip25	logged_mult_score	comet-kite	toroidal-lattice	20	20	400.0
c25pip25	logged_mult_score	comet-kite	well-mixed	20	20	400.0
c25pip25	logged_mult_score	comet-kite	wheel	20	20	400.0
c25pip25	logged_mult_score	comet-kite	windmill	20	20	400.0
c25pip25	logged_mult_score	cycle	linear-chain	20	20	207.0
c25pip25	logged_mult_score	cycle	random-barabasi-albert	20	20	104.5
c25pip25	logged_mult_score	cycle	random-waxman	20	20	33.0
c25pip25	logged_mult_score	cycle	star	20	20	0.0
c25pip25	logged_mult_score	cycle	toroidal-lattice	20	20	146.5
c25pip25	logged_mult_score	cycle	well-mixed	20	20	105.5
c25pip25	logged_mult_score	cycle	wheel	20	20	243.0
c25pip25	logged_mult_score	cycle	windmill	20	20	306.0
c25pip25	logged_mult_score	linear-chain	random-barabasi-albert	20	20	131.5
c25pip25	logged_mult_score	linear-chain	random-waxman	20	20	36.0
c25pip25	logged_mult_score	linear-chain	star	20	20	0.0
c25pip25	logged_mult_score	linear-chain	toroidal-lattice	20	20	191.0
c25pip25	logged_mult_score	linear-chain	well-mixed	20	20	181.0
c25pip25	logged_mult_score	linear-chain	wheel	20	20	256.0
c25pip25	logged_mult_score	linear-chain	windmill	20	20	297.0
c25pip25	logged_mult_score	random-barabasi-albert	random-waxman	20	20	86.0
c25pip25	logged_mult_score	random-barabasi-albert	star	20	20	0.0
c25pip25	logged_mult_score	random-barabasi-albert	toroidal-lattice	20	20	314.5
c25pip25	logged_mult_score	random-barabasi-albert	well-mixed	20	20	270.5
c25pip25	logged_mult_score	random-barabasi-albert	wheel	20	20	256.0
c25pip25	logged_mult_score	random-barabasi-albert	windmill	20	20	374.0
c25pip25	logged_mult_score	random-waxman	star	20	20	0.0
c25pip25	logged_mult_score	random-waxman	toroidal-lattice	20	20	385.5
c25pip25	logged_mult_score	random-waxman	well-mixed	20	20	354.5
c25pip25	logged_mult_score	random-waxman	wheel	20	20	354.0
c25pip25	logged_mult_score	random-waxman	windmill	20	20	396.0
c25pip25	logged_mult_score	star	toroidal-lattice	20	20	400.0
c25pip25	logged_mult_score	star	well-mixed	20	20	400.0
c25pip25	logged_mult_score	star	wheel	20	20	400.0
c25pip25	logged_mult_score	star	windmill	20	20	400.0
c25pip25	logged_mult_score	toroidal-lattice	well-mixed	20	20	53.5
c25pip25	logged_mult_score	toroidal-lattice	wheel	20	20	200.0
c25pip25	logged_mult_score	toroidal-lattice	windmill	20	20	357.0
c25pip25	logged_mult_score	well-mixed	wheel	20	20	200.0
c25pip25	logged_mult_score	well-mixed	windmill	20	20	378.0
c25pip25	logged_mult_score	wheel	windmill	20	20	206.0
c25pip50	logged_mult_score	comet-kite	cycle	20	20	89.0
c25pip50	logged_mult_score	comet-kite	linear-chain	20	20	83.0
c25pip50	logged_mult_score	comet-kite	random-barabasi-albert	20	20	396.0
c25pip50	logged_mult_score	comet-kite	random-waxman	20	20	340.0
c25pip50	logged_mult_score	comet-kite	star	20	20	0.0
c25pip50	logged_mult_score	comet-kite	toroidal-lattice	20	20	17.0
c25pip50	logged_mult_score	comet-kite	well-mixed	20	20	14.0
c25pip50	logged_mult_score	comet-kite	wheel	20	20	0.0

interaction_matrix	.y.	group1	group2	n1	n2	statistic
c25pip25	logged_mult_score	comet-kite	well-mixed	20	20	400.0
c25pip25	logged_mult_score	cycle	well-mixed	20	20	105.5
c25pip25	logged_mult_score	linear-chain	well-mixed	20	20	181.0
c25pip25	logged_mult_score	random-barabasi-albert	well-mixed	20	20	270.5
c25pip25	logged_mult_score	random-waxman	well-mixed	20	20	354.5
c25pip25	logged_mult_score	star	well-mixed	20	20	400.0
c25pip25	logged_mult_score	toroidal-lattice	well-mixed	20	20	53.5
c25pip25	logged_mult_score	well-mixed	wheel	20	20	200.0
c25pip25	logged_mult_score	well-mixed	windmill	20	20	378.0
c25pip50	logged_mult_score	comet-kite	well-mixed	20	20	14.0
c25pip50	logged_mult_score	cycle	well-mixed	20	20	28.0
c25pip50	logged_mult_score	linear-chain	well-mixed	20	20	32.0
c25pip50	logged_mult_score	random-barabasi-albert	well-mixed	20	20	0.0
c25pip50	logged_mult_score	random-waxman	well-mixed	20	20	3.0
c25pip50	logged_mult_score	star	well-mixed	20	20	400.0
c25pip50	logged_mult_score	toroidal-lattice	well-mixed	20	20	82.0
c25pip50	logged_mult_score	well-mixed	wheel	20	20	0.0
c25pip50	logged_mult_score	well-mixed	windmill	20	20	380.0
c25pip75	logged_mult_score	comet-kite	well-mixed	20	20	2.0
c25pip75	logged_mult_score	cycle	well-mixed	20	20	0.0
c25pip75	logged_mult_score	linear-chain	well-mixed	20	20	0.0
c25pip75	logged_mult_score	random-barabasi-albert	well-mixed	20	20	159.0
c25pip75	logged_mult_score	random-waxman	well-mixed	20	20	95.0
c25pip75	logged_mult_score	star	well-mixed	20	20	0.0
c25pip75	logged_mult_score	toroidal-lattice	well-mixed	20	20	0.0
c25pip75	logged_mult_score	well-mixed	wheel	20	20	387.0
c25pip75	logged_mult_score	well-mixed	windmill	20	20	399.0
c50pip25	logged_mult_score	comet-kite	well-mixed	20	20	392.0
c50pip25	logged_mult_score	cycle	well-mixed	20	20	367.0
c50pip25	logged_mult_score	linear-chain	well-mixed	20	20	363.0
c50pip25	logged_mult_score	random-barabasi-albert	well-mixed	20	20	345.0
c50pip25	logged_mult_score	random-waxman	well-mixed	20	20	346.0
c50pip25	logged_mult_score	star	well-mixed	20	20	400.0
c50pip25	logged_mult_score	toroidal-lattice	well-mixed	20	20	301.0
c50pip25	logged_mult_score	well-mixed	wheel	20	20	35.0
c50pip25	logged_mult_score	well-mixed	windmill	20	20	129.0
c50pip50	logged_mult_score	comet-kite	well-mixed	20	20	400.0
c50pip50	logged_mult_score	cycle	well-mixed	20	20	400.0
c50pip50	logged_mult_score	linear-chain	well-mixed	20	20	400.0
c50pip50	logged_mult_score	random-barabasi-albert	well-mixed	20	20	379.0
c50pip50	logged_mult_score	random-waxman	well-mixed	20	20	399.0
c50pip50	logged_mult_score	star	well-mixed	20	20	288.0
c50pip50	logged_mult_score	toroidal-lattice	well-mixed	20	20	394.0
c50pip50	logged_mult_score	well-mixed	wheel	20	20	0.0
c50pip50	logged_mult_score	well-mixed	windmill	20	20	23.0
c50pip75	logged_mult_score	comet-kite	well-mixed	20	20	400.0
c50pip75	logged_mult_score	cycle	well-mixed	20	20	400.0
c50pip75	logged_mult_score	linear-chain	well-mixed	20	20	400.0
c50pip75	logged_mult_score	random-barabasi-albert	well-mixed	20	20	388.0
c50pip75	logged_mult_score	random-waxman	well-mixed	20	20	349.0
c50pip75	logged_mult_score	star	well-mixed	20	20	400.0
c50pip75	logged_mult_score	toroidal-lattice	well-mixed	20	20	386.0
c50pip75	logged_mult_score	well-mixed	wheel	20	20	0.0

```
  mutate(
    non_wm_graph = case_when(
      group1 == "well-mixed" ~ group2,
      group2 == "well-mixed" ~ group1
    )
  )
non_wm_graph_types <- unique(as.character(well_mixed_comps$non_wm_graph))

spatial_struct_effects <- data.frame(
  interaction_matrix = character(),
  graph_type = character(),
  effect = character(),
  wm_median_score = numeric(),
  graph_median_score = numeric(),
  sig = logical()
)

# Identify promotors (significant and > well-mixed)
# Identify represssors (significant and < well-mixed)
# Neither (not significant)
# The output of this loop is sanity-checked against statistical results table.
for (interaction_mat in int_matrices) {
  # Get median score for well-mixed
  wm_median_score <- filter(
    summary_data,
    graph_type == "well-mixed" & interaction_matrix == interaction_mat
  )$score_median[[1]]
  # Get relevent wilcoxon rank-sum comparisons
  im_comps <- well_mixed_comps %>%
    filter(interaction_matrix == interaction_mat)
  for (graph in non_wm_graph_types) {
    graph_median_score <- filter(
      summary_data,
      graph_type == graph & interaction_matrix == interaction_mat
    )$score_median[[1]]
    comp_info <- filter(im_comps, non_wm_graph == graph)
    sig <- comp_info$p.adj[[1]] < 0.05
    effect <- "unknown"
    if (sig && graph_median_score < wm_median_score) {
      effect <- "suppressor"
    } else if (sig && graph_median_score > wm_median_score) {
      effect <- "promoter"
    } else {
      effect <- "neither"
    }
    spatial_struct_effects <- add_row(
      spatial_struct_effects,
      interaction_matrix = interaction_mat,
      graph_type = graph,
      effect = effect,
      wm_median_score = wm_median_score,
      graph_median_score = graph_median_score,
      sig = sig
    )
  }
}

effect_table <- spatial_struct_effects %>%
  kable() %>%
  kable_styling(
    latex_options = "striped"
  )

save_kable(
  effect_table,
  paste(
    plot_dir,
    "spatial_struct_effect_table.pdf",
    sep = "/"
  )
)

effect_table
```

Break effects down by interaction matrix (experiment). Arrange in order of effect-size.

```
for (im in int_matrices) {
  max_promoter <- max(
    filter(
      spatial_struct_effects,
      interaction_matrix == im & effect == "promoter"
    )$graph_median_score
  )
  max_suppressor <- min(
    filter(
      spatial_struct_effects,
      interaction_matrix == im & effect == "suppressor"
    )$graph_median_score
  )
```

interaction_matrix	graph_type	effect	wm_median_score	graph_median_score	sig
c25pip25	comet-kite	promoter	-0.1274125	290.7705000	TRUE
c25pip25	cycle	neither	-0.1274125	-0.1448880	FALSE
c25pip25	linear-chain	neither	-0.1274125	-0.1448875	FALSE
c25pip25	random-barabasi-albert	neither	-0.1274125	52.5468000	FALSE
c25pip25	random-waxman	promoter	-0.1274125	142.6250000	TRUE
c25pip25	star	promoter	-0.1274125	622.3975000	TRUE
c25pip25	toroidal-lattice	suppressor	-0.1274125	-0.1398950	TRUE
c25pip25	wheel	neither	-0.1274125	29.1941680	FALSE
c25pip25	windmill	suppressor	-0.1274125	-0.1552570	TRUE
c25pip50	comet-kite	suppressor	-32.6162000	-62.4832500	TRUE
c25pip50	cycle	suppressor	-32.6162000	-46.8019500	TRUE
c25pip50	linear-chain	suppressor	-32.6162000	-46.2323500	TRUE
c25pip50	random-barabasi-albert	suppressor	-32.6162000	-123.7055000	TRUE
c25pip50	random-waxman	suppressor	-32.6162000	-85.6997500	TRUE
c25pip50	star	promoter	-32.6162000	2.4022500	TRUE
c25pip50	toroidal-lattice	neither	-32.6162000	-34.1670000	FALSE
c25pip50	wheel	promoter	-32.6162000	-27.7729000	TRUE
c25pip50	windmill	suppressor	-32.6162000	-44.0345500	TRUE
c25pip75	comet-kite	suppressor	743.1275000	717.4210000	TRUE
c25pip75	cycle	suppressor	743.1275000	658.9540000	TRUE
c25pip75	linear-chain	suppressor	743.1275000	664.6090000	TRUE
c25pip75	random-barabasi-albert	neither	743.1275000	739.2250000	FALSE
c25pip75	random-waxman	neither	743.1275000	736.0230000	FALSE
c25pip75	star	suppressor	743.1275000	140.4695000	TRUE
c25pip75	toroidal-lattice	suppressor	743.1275000	703.8705000	TRUE
c25pip75	wheel	suppressor	743.1275000	715.5445000	TRUE
c25pip75	windmill	suppressor	743.1275000	720.5355000	TRUE
c50pip25	comet-kite	promoter	-198.2860000	-35.9154000	TRUE
c50pip25	cycle	promoter	-198.2860000	-96.3967500	TRUE
c50pip25	linear-chain	promoter	-198.2860000	-109.6645000	TRUE
c50pip25	random-barabasi-albert	promoter	-198.2860000	-119.6675000	TRUE
c50pip25	random-waxman	promoter	-198.2860000	-128.2555000	TRUE
c50pip25	star	promoter	-198.2860000	29.5336000	TRUE
c50pip25	toroidal-lattice	neither	-198.2860000	-136.0975000	FALSE
c50pip25	wheel	promoter	-198.2860000	-93.1401500	TRUE
c50pip25	windmill	neither	-198.2860000	-164.2615000	FALSE
c50pip50	comet-kite	promoter	124.5965000	280.7530000	TRUE
c50pip50	cycle	promoter	124.5965000	260.7760000	TRUE
c50pip50	linear-chain	promoter	124.5965000	254.4380000	TRUE
c50pip50	random-barabasi-albert	promoter	124.5965000	175.8165000	TRUE
c50pip50	random-waxman	promoter	124.5965000	196.9120000	TRUE
c50pip50	star	neither	124.5965000	200.4315000	FALSE
c50pip50	toroidal-lattice	promoter	124.5965000	224.4375000	TRUE
c50pip50	wheel	promoter	124.5965000	286.9300000	TRUE
c50pip50	windmill	promoter	124.5965000	195.5930000	TRUE
c50pip75	comet-kite	promoter	256.7610000	520.2015000	TRUE
c50pip75	cycle	promoter	256.7610000	538.1730000	TRUE
c50pip75	linear-chain	promoter	256.7610000	516.4060000	TRUE
c50pip75	random-barabasi-albert	promoter	256.7610000	378.4345000	TRUE
c50pip75	random-waxman	promoter	256.7610000	344.5975000	TRUE
c50pip75	star	promoter	256.7610000	580.5000000	TRUE
c50pip75	toroidal-lattice	promoter	256.7610000	382.7610000	TRUE
c50pip75	wheel	promoter	256.7610000	580.5095000	TRUE

```
  im_effects <- spatial_struct_effects %>%
    filter(interaction_matrix == im) %>%
    mutate(
      max_promoter = graph_median_score == max_promoter,
      max_suppressor = graph_median_score == max_suppressor
    ) %>%
    arrange(effect)

  # Identify biggest suppressor / promoter
  table <- im_effects %>%
    kable() %>%
    kable_styling(
      latex_options = "striped"
    )

  save_kable(
    table,
    paste(
      plot_dir,
      paste0("spatial_struct_effect_table_", im, ".pdf"),
      sep = "/"
    )
  )
}
```

7.4.1 Distribution of effects for each spatial structure type

Count the distribution of effects each graph type is categorized with.

```
effect_counts <- spatial_struct_effects %>%
  mutate(
    effect = as.factor(effect),
    graph_type = as.factor(graph_type)
  ) %>%
  group_by(graph_type, effect) %>%
  dplyr::summarize(
    n = n()
  )

table <- effect_counts %>%
  kable() %>%
  kable_styling(
    latex_options = "striped"
  )
```

```
save_kable(
  table,
  paste(
    plot_dir,
    "spatial_struct_effect_counts.pdf",
    sep = "/"
  )
)
```

Visualize:

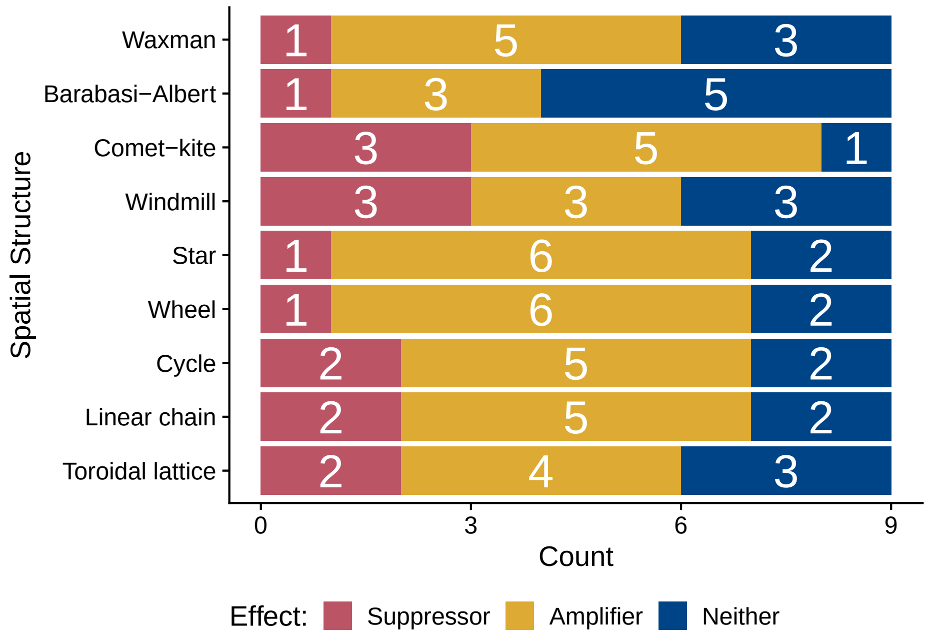
```
# Manually set ordering for plot:
graph_ticks <- c(
  "toroidal-lattice",
  "linear-chain",
  "cycle",
  "wheel",
  "star",
  "windmill",
  "comet-kite",
  "random-barabasi-albert",
  "random-waxman"
)
graph_labels <- c(
  "Toroidal lattice",
  "Linear chain",
  "Cycle",
  "Wheel",
  "Star",
  "Windmill",
  "Comet-kite",
  "Barabasi-Albert",
  "Waxman"
)

effect_counts_fig <- effect_counts %>%
  ggplot(
    aes(
      fill = effect,
      x = graph_type,
      y = n
    )
  ) +
  geom_bar(
    position = "stack",
```

```r
    stat = "identity"
  ) +
  geom_text(
    aes(label = n),
    position = position_stack(vjust = 0.5),
    size = 8,
    color = "white"
  ) +
  scale_fill_highcontrast(
    name = "Effect:",
    limits = c("suppressor", "promoter", "neither"),
    labels = c("Suppressor", "Amplifier", "Neither"),
    reverse = TRUE
  ) +
  scale_x_discrete(
    name = "Spatial Structure",
    limits = graph_ticks,
    breaks = graph_ticks,
    labels = graph_labels
  ) +
  scale_y_continuous(
    name = "Count",
    limits = c(0, 9),
    breaks = c(0, 3, 6, 9)
  ) +
  coord_flip() +
  theme(
    legend.position = "bottom"
  )

ggsave(
  paste(
    plot_dir,
    "spatial_structure_effect_distributions.pdf",
    sep = "/"
  ),
  plot = effect_counts_fig,
  width = 6,
  height = 4
)

effect_counts_fig
```


Waxman
Barabasi−Albert
Comet−kite
Windmill
Star
Wheel
Cycle
Linear chain
Toroidal lattice
Spatial Structure
Count
0
3
6
9
1 5 3
1 3 5
3 5 1
3 3 3
1 6 2
1 6 2
2 5 2
2 5 2
2 4 3
Effect:
Suppressor
Amplifier
Neither

Chapter 8

Graph property correlations

We screened for graph properties correlated with community transitionability scores.

8.1 Dependencies and setup

```r
library(tidyverse)
library(Hmisc)
library(broom)
library(knitr)
library(kableExtra)
```

```r
# Check if Rmd is being compiled using bookdown
bookdown <- exists("bookdown_build")
```

```r
experiment_slug <- "2024-03-08-varied-interaction-matrices"
working_directory <- paste(
  "experiments",
  experiment_slug,
  "analysis",
  sep = "/"
)
# Adjust working directory if being knitted for bookdown build.
if (bookdown) {
  working_directory <- paste0(
    bookdown_wd_prefix,
    working_directory
  )
}

plot_dir <- paste(
  working_directory,
  "plots",
  sep = "/"
)

data_path <- paste(
  working_directory,
  "data",
  "world_summary_final_update_with-graph-props.csv",
  sep = "/"
)
data <- read_csv(data_path)
```

Set cowplot theme as default plotting theme.

```
theme_set(theme_cowplot())
```

8.2 Data preprocessing

```
max_update <- max(data$update)
# Ensure that we just have measurements from final update.
data <- data %>%
  filter(update == max_update) %>%
  mutate(
    interaction_matrix = as.factor(interaction_matrix),
    graph_type = as.factor(graph_type),
    summary_mode = as.factor(summary_mode),
    update = as.numeric(update),
    SEED = as.factor(SEED),
    graph_file = str_split_i(DIFFUSION_SPATIAL_STRUCTURE_FILE, "/", -1)
  ) %>%
  mutate(
    graph_file = as.factor(graph_file)
  )
# write_csv(
#   data,
#   "world_summary_final_update.csv"
# )
```

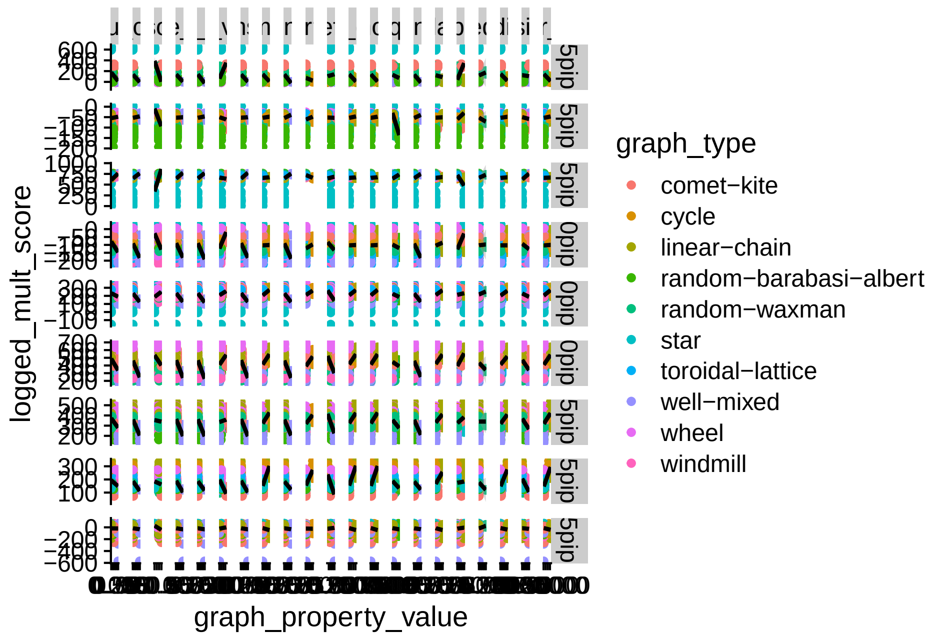
```r
# For each row, assign graph properties
properties <- c(
  "graph_prop_density",
  "graph_prop_degree_mean",
  "graph_prop_degree_median",
  "graph_prop_degree_variance",
  "graph_prop_girth",
  "graph_prop_degree_assortivity_coef",
  "graph_prop_num_bridges",
  "graph_prop_max_clique_size",
  "graph_prop_transitivity",
  "graph_prop_avg_clustering",
  "graph_prop_num_connected_components",
  "graph_prop_num_articulation_points",
  "graph_prop_avg_node_connectivity",
  "graph_prop_edge_connectivity",
  "graph_prop_node_connectivity",
  "graph_prop_diameter",
  "graph_prop_radius",
  "graph_prop_kemeny_constant",
  "graph_prop_global_efficiency",
  "graph_prop_wiener_index",
  "graph_prop_longest_shortest_path"
)

# (3) Pivot longer
long_data <- data %>%
  mutate(
    graph_prop_diameter = case_when(
      graph_prop_diameter == "error" ~ "-1",
      .default = graph_prop_diameter
    ),
    graph_prop_radius = case_when(
      graph_prop_radius == "error" ~ "-1",
      .default = graph_prop_radius
    ),
    graph_prop_kemeny_constant = case_when(
      graph_prop_kemeny_constant == "error" ~ "-1",
      .default = graph_prop_kemeny_constant
    )
  ) %>%
  mutate(
    graph_prop_diameter = as.numeric(graph_prop_diameter),
    graph_prop_radius = as.numeric(graph_prop_radius),
    graph_prop_kemeny_constant = as.numeric(graph_prop_kemeny_constant)
  ) %>%
  select(
    !c(
      DIFFUSION_SPATIAL_STRUCTURE_FILE,
      GROUP_REPRO_SPATIAL_STRUCTURE_FILE,
      INTERACTION_SOURCE
    )
  ) %>%
  filter(
    summary_mode == "ranked_threshold"
  ) %>%
  pivot_longer(
    cols = properties,
    names_to = "graph_property",
    values_to = "graph_property_value"
  ) %>%
  filter(
    (!is.na(graph_property_value)) & graph_property_value != "Inf" &
    (!(graph_property == "graph_prop_diameter" & (graph_property_value == "-1"))) &
    (!(graph_property == "graph_prop_radius" & (graph_property_value == "-1"))) &
    (!(graph_property == "graph_prop_kemeny_constant" & (graph_property_value == "-1")
  ) %>%
  mutate(
    graph_property_value = as.numeric(graph_property_value),
    graph_property = str_remove(graph_property, "graph_prop_")
  ) %>%
  mutate(
    graph_property = as.factor(graph_property)
  )
# write_csv(long_data, "test.csv")
```

8.3 Plot relationships between transitionability and graph properties

```
rel_plot <- long_data %>%
  ggplot(
    aes(
      x = graph_property_value,
      y = logged_mult_score
    )
  ) +
  geom_point(aes(color = graph_type)) +
  geom_smooth(
    method = "lm",
    color = "black"
  ) +
  facet_grid(
    interaction_matrix ~ graph_property,
    scales = "free"
  )

rel_plot
```



```
ggsave(
  plot = rel_plot,
  filename = paste(
    plot_dir,
    "property_relationships.pdf",
    sep = "/"
  ),
  width = 40,
  height = 20
)
```

8.4 Measure correlations

```r
# Reference for running correlations over tidy data:
# https://dominicroye.github.io/en/2019/tidy-correlation-tests-in-r/
cor_fun <- function(data) {
  cor.test(
    data$graph_property_value,
    data$logged_mult_score,
    method = "spearman",
    exact = FALSE
  ) %>% tidy()
}

nested <- long_data %>%
  select(
    c(
      interaction_matrix,
      graph_property,
      graph_property_value,
      logged_mult_score
    )
  ) %>%
  group_by(interaction_matrix, graph_property) %>%
  nest() %>%
  mutate(
    model = map(data, cor_fun)
  )

full_corr <- select(nested, -data) %>% unnest()
full_corr <- full_corr %>%
  mutate(
    abs_estimate = abs(estimate)
  ) %>%
  arrange(
    desc(abs_estimate)
  ) %>%
  group_by(
    interaction_matrix
  ) %>%
  mutate(
    p.value.adj = p.adjust(p.value, method = "holm")
  ) %>%
  filter(
    p.value.adj <= 0.05
  )

full_corr_table <- kable(full_corr) %>%
  kable_styling(latex_options = "striped")
save_kable(
  full_corr_table,
  paste(
    plot_dir,
    "correlation_table.pdf",
    sep = "/"
  )
)
```

8.4.1 Top three significant correlations per interaction matrix

Break correlations down by interaction matrix

```
interaction_matrices <- levels(long_data$interaction_matrix)

for (mat_type in interaction_matrices) {
  mat_data <- filter(long_data, interaction_matrix == mat_type)

  nested <- mat_data %>%
    select(
      c(
        interaction_matrix,
        graph_property,
        graph_property_value,
        logged_mult_score
      )
    ) %>%
    group_by(interaction_matrix, graph_property) %>%
    nest() %>%
    mutate(
      model = map(data, cor_fun)
    )

  im_corr <- select(nested, -data) %>% unnest()
  im_corr <- im_corr %>%
    mutate(
      abs_estimate = abs(estimate)
    ) %>%
    arrange(
      desc(abs_estimate)
    ) %>%
    ungroup() %>%
    group_by(
      interaction_matrix
    ) %>%
    mutate(
      p.value.adj = p.adjust(p.value, method = "holm")
    ) %>%
    filter(
      p.value.adj < 0.05
    )

  im_corr_table <- kable(im_corr) %>%
    kable_styling(latex_options = "striped")
  save_kable(
    im_corr_table,
    paste(
      plot_dir,
      paste0("correlation_table_", mat_type, ".pdf"),
      sep = "/"
    )
  )

  top_corr <- im_corr %>%
    slice_max(
      abs_estimate,
      n = 3
    )
  top_corr_table <- kable(top_corr) %>%
    kable_styling(latex_options = "striped")

  save_kable(
    top_corr_table,
    paste(
      plot_dir,
      paste0("t3_correlation_table_", mat_type, ".pdf"),
      sep = "/"
    )
  )

}
```

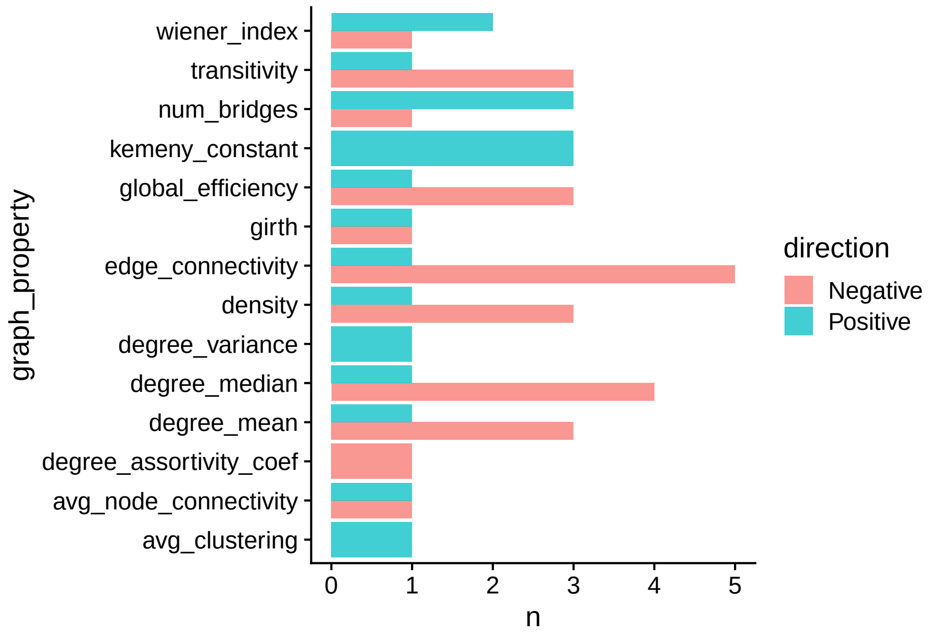
8.4.2 Distrubition of correlations >= 0.5 strength

```
# Look at distribution of correlations >= 0.5 strength
corr_str_thresh <- 0.5
full_corr_thresh <- full_corr %>%
  mutate(
    direction = case_when(
      estimate < 0 ~ "Negative",
      estimate >= 0 ~ "Positive"
    )
  ) %>%
  mutate(
    direction = as.factor(direction)
  ) %>%
  filter(abs_estimate >= corr_str_thresh & p.value.adj <= 0.05)

corr_counts <- full_corr_thresh %>%
  dplyr::group_by(graph_property, direction) %>%
  dplyr::summarize(
    n = n()
  )

# If something has one direction, but not other, fill in 0 for other.
# For property in properties
correlation_dirs_plot <- corr_counts %>%
  ggplot(
    aes(
      x = graph_property,
      fill = direction,
      y = n
    )
  ) +
  geom_bar(stat = "identity", position = position_dodge(), alpha = 0.75) +
  coord_flip()

ggsave(
  plot = correlation_dirs_plot,
  filename = paste(
    plot_dir,
    "most_moderate_correlations.pdf",
    sep = "/"
  )
)
correlation_dirs_plot
```


wiener_index
transitivity
num_bridges
kemeny_constant
global_efficiency
girth
edge_connectivity
density
degree_variance
degree_median
degree_mean
degree_assortivity_coef
avg_node_connectivity
avg_clustering
graph_property
0
1
2
3
4
5
n
direction
Negative
Positive