Supplemental Material for Environmental connectivity influences long-term evolutionary outcomes

2025-08-11

Contents

Chapter 1

Introduction

This is the supplemental material our 2025 Artificial Life Conference paper, "Environmental connectivity influences long-term evolutionary outcomes". This is not intended as a stand-alone document, but as a companion to our main manuscript.

1.1 About our supplemental material

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)
- TODO

1.2 Contributing authors

- Grant Gordon
- Austin J. Ferguson
- Emily Dolson
- Alexander Lalejini

Chapter 2

Data availability

2.1 Source code

The source code for his work is publicly accessible on GitHub: https://github.com/amlalejini/alife-2025-env-conn. This repository has also been archived on Zenodo: https://doi.org/10.5281/zenodo.16795777

2.2 Experiment results

Data generated from our experiments used in analyses are available online, archived in an OSF repository: https://osf.io/ahs6m/

On OSF, the following compressed archives contain the data presented in our manuscript:

- `2025-04-17-squished-lattice-longer-avida.tar.gz`
- `2025-04-17-vary-structs-avida.tar.gz`
- `squished-lattice-mabe.tar.gz`
- `vary-structs-mabe.tar.gz`

Chapter 3

Compilation instructions

Instructions for compiling and running the software used in this study on your local machine. All experiments were run on Mac or Linux-based operating systems.

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

You will also need Python to run graph generation and analysis. Python dependencies are listed in the `requirements.txt` at the root of this repository.

Statistical analyses and data visualizations were conducted using R.

Experiments in our simplified model used the MABE2 software, and experiments with digital organisms (self-replicating computer programs) used a modified version of the Avida software platform.

3.1 Instructions

First, clone the `alife-2025-env-conn` repository (https://github.com/amlalejini/alife-2025-env-conn.git) to your machine. Then, initialize and update git submodule inside the repository. From inside the repository on your machine, run:

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

This will download and update the following dependencies:

- `avida-empirical` (commit hash: `266f95f8fcb452655330dab55caa9f1408b49ffa`): A modified implementation of the Avida software that supports the capacity to configure environmental connectivity.

- `evo_spatial_discoveries` (commit hash: `2c384e93df231125bae83fc6c38d8dc8c64eb6ee`): Contains configurations for MABE2 experiments.
- `MABE2` (commit hash: `4f8eb86f997ee89f6d0e0b1144c5be162f4d8d1b`): MABE = "Modular agent-based evolver", which is a software platform deigned to empower developers to easily build and customize software for evolutionary computation or artificial life. We used this platform to implement our non-avida experiments.
- `network_correlation` (commit hash: `9d9a07f7436c3569d10eb3b03c6b30e1238c74ef`): Third-party python implementations of various graph statistics and analyses.

To compile Avida, navigate into the `third-party/avida-empirical/` directory and run `./build_avida/`. The compiled executable will be created in the `third-party/avida-empirical/cbuild/work/` directory.

To compile MABE2, navigate into the `third-party/MABE2/build` directory and run `make native`. The compiled executable will be created in the `third-party/MABE2/build` directory.

Configuration files used Avida experiments can be found in the `experiments/` directory (within the `hpc/config` subdirectory for any given experiment). Configuration files used for MABE2 experiments can be found in `third-party/evo_spatial_discoveries/experiments/`.

Chapter 4

Simple model - Varied spatial structure experiment analyses

4.1 Dependencies and setup

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

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

```
experiment_slug <- "vg-experiments"
working_directory <- paste(
  "experiments",
  "mabe2-exps",
  experiment_slug,
  sep = "/"
)
```

```
# Adjust working directory if being knitted for bookdown build.
if (bookdown) {
  working_directory <- paste0(
    bookdown_wd_prefix,
    working_directory
  )
}
```

```
# Configure our default graphing theme
theme_set(theme_cowplot())
# Create a directory to store plots
plot_dir <- paste(
  working_directory,
  "rmd_plots",
  sep = "/"
)

dir.create(
  plot_dir,
  showWarnings = FALSE
)
```

4.2 Max organism data analyses

```
max_generation <- 100000
max_org_data_path <- paste(
  working_directory,
  "data",
  "combined_max_org_data.csv",
  sep = "/"
)
# Data file has time series
max_org_data_ts <- read_csv(max_org_data_path)
max_org_data_ts <- max_org_data_ts %>%
  mutate(
    landscape = as.factor(landscape),
    structure = as.factor(structure),
  ) %>%
  mutate(
    valleys_crossed = case_when(
      landscape == "Valley crossing" ~ round(log(fitness, base = 1.5)),
      .default = 0
```

```
    )
  )
# Get tibble with just final generation
max_org_data <- max_org_data_ts %>%
  filter(generation == max_generation)
```

Check that replicate count for each condition matches expectations.

```
run_summary <- max_org_data %>%
  group_by(landscape, structure) %>%
  summarize(
    n = n()
  )
print(run_summary, n = 30)
```

```
## # A tibble: 30 x 3
## # Groups:   landscape [3]
##    landscape       structure         n
##    <fct>           <fct>         <int>
##  1 Multipath       clique_ring      50
##  2 Multipath       comet_kite       50
##  3 Multipath       cycle            50
##  4 Multipath       lattice          50
##  5 Multipath       linear_chain     50
##  6 Multipath       random_waxman    50
##  7 Multipath       star             50
##  8 Multipath       well_mixed       50
##  9 Multipath       wheel            50
## 10 Multipath       windmill         50
## 11 Single gradient clique_ring      50
## 12 Single gradient comet_kite       50
## 13 Single gradient cycle            50
## 14 Single gradient lattice          50
## 15 Single gradient linear_chain     50
## 16 Single gradient random_waxman    50
## 17 Single gradient star             50
## 18 Single gradient well_mixed       50
## 19 Single gradient wheel            50
## 20 Single gradient windmill         50
## 21 Valley crossing clique_ring      50
## 22 Valley crossing comet_kite       50
## 23 Valley crossing cycle            50
## 24 Valley crossing lattice          50
## 25 Valley crossing linear_chain     50
```

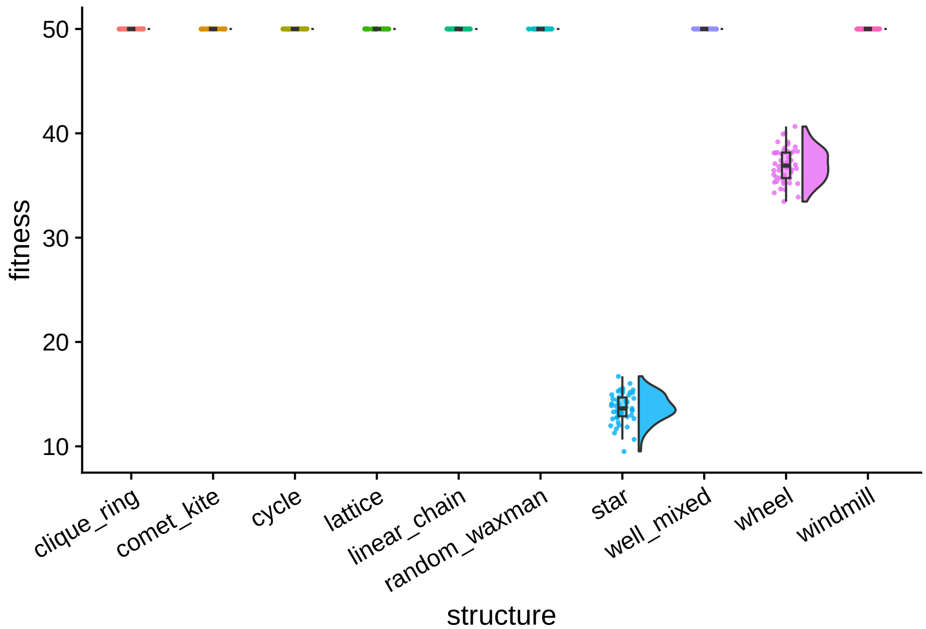
```
## 26 Valley crossing random_waxman    50
## 27 Valley crossing star             50
## 28 Valley crossing well_mixed       50
## 29 Valley crossing wheel            50
## 30 Valley crossing windmill         50
```

4.2.1 Fitness in smooth gradient landscape

Maximum fitness

```
single_gradient_final_fitness_plt <- ggplot(
    data = filter(max_org_data, landscape == "Single gradient"),
    mapping = aes(
      x = structure,
      y = fitness,
      fill = structure
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping = aes(color = structure),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/single_gradient_final_fitness.pdf"),
  plot = single_gradient_final_fitness_plt,
  width = 15,
```

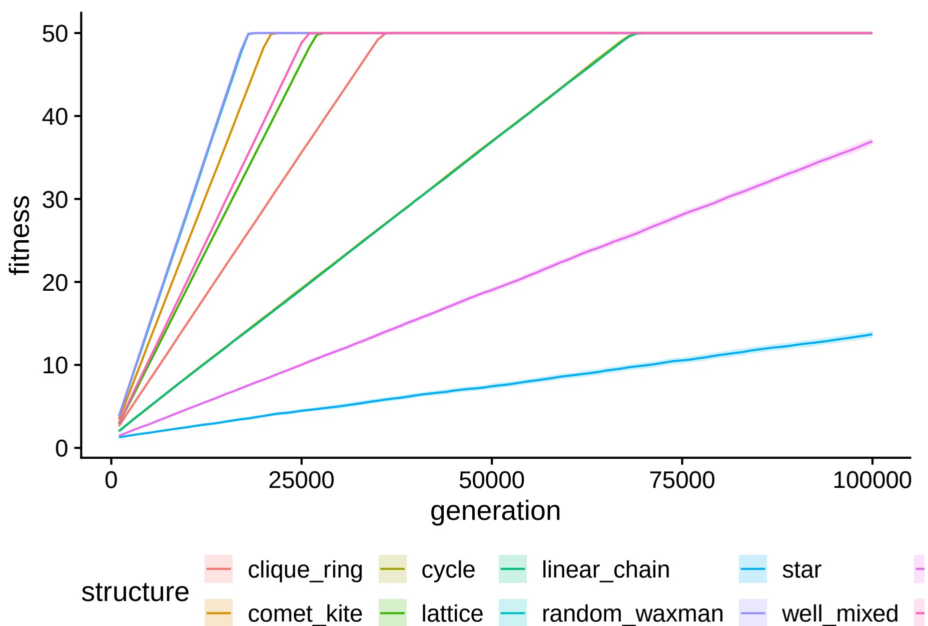
```
  height = 10
)

single_gradient_final_fitness_plt
```



Maximum fitness over time

```
single_gradient_fitness_ts_plt <- ggplot(
    data = filter(max_org_data_ts, landscape == "Single gradient"),
    mapping = aes(
      x = generation,
      y = fitness,
      color = structure,
      fill = structure
    )
  ) +
  stat_summary(fun = "mean", geom = "line") +
  stat_summary(
    fun.data = "mean_cl_boot",
    fun.args = list(conf.int = 0.95),
    geom = "ribbon",
    alpha = 0.2,
    linetype = 0
  ) +
  theme(legend.position = "bottom")
```

```
ggsave(
  plot = single_gradient_fitness_ts_plt,
  filename = paste0(
    plot_dir,
    "/single_gradient_fitness_ts.pdf"
  ),
  width = 15,
  height = 10
)

single_gradient_fitness_ts_plt
```



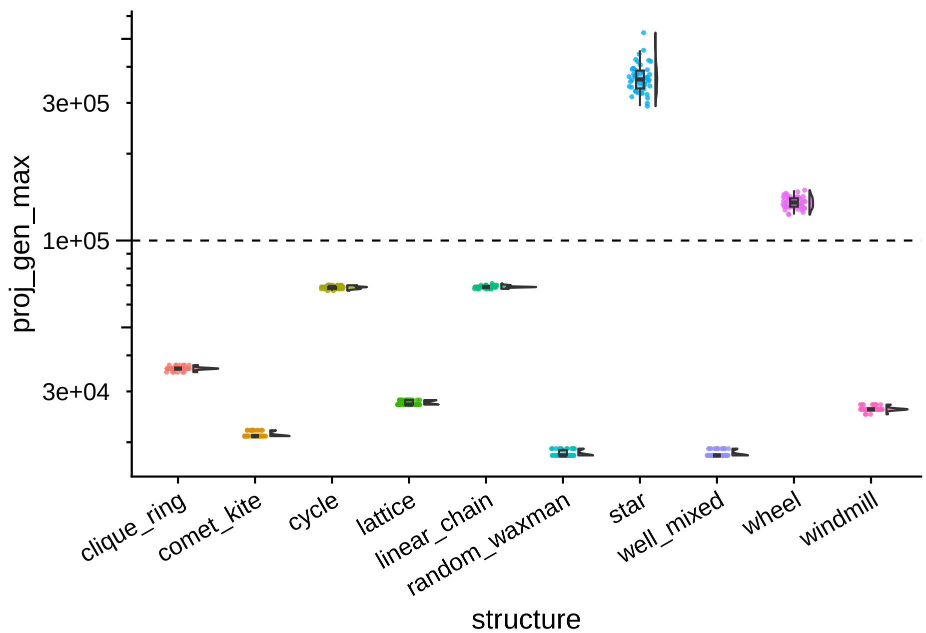
Time to maximum fitness

```
# Find all rows with maximum fitness value, then get row with minimum generation,
#  then project out expected generation to max (for runs that didn't finish)
max_possible_fit = 50
time_to_max_single_gradient <- max_org_data_ts %>%
  filter(landscape == "Single gradient") %>%
  group_by(rep, structure) %>%
  slice_max(
    fitness,
    n = 1
  ) %>%
  slice_min(
```

```
    generation,
    n = 1
  ) %>%
  mutate(
    proj_gen_max = (max_possible_fit / fitness) * generation
  )
```

```
single_gradient_gen_max_proj_plt <- ggplot(
    data = time_to_max_single_gradient,
    mapping = aes(
      x = structure,
      y = proj_gen_max,
      fill = structure
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping = aes(color = structure),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_y_log10(
    guide = "axis_logticks"
  ) +
  # scale_y_continuous(
  #   trans="pseudo_log",
  #   breaks = c(10, 100, 1000, 10000, 100000, 1000000)
  #   ,limits = c(10, 100, 1000, 10000, 100000, 1000000)
  # ) +
  geom_hline(
    yintercept = max_generation,
    linetype = "dashed"
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
```

```
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/single_gradient_gen_max_proj.pdf"),
  plot = single_gradient_gen_max_proj_plt,
  width = 15,
  height = 10
)

single_gradient_gen_max_proj_plt
```



Rank ordering of time to max fitness values

```
time_to_max_single_gradient %>%
  group_by(structure) %>%
  summarize(
    reps = n(),
    median_proj_gen = median(proj_gen_max),
    mean_proj_gen = mean(proj_gen_max)
  ) %>%
  arrange(
    mean_proj_gen
```

```
)
```

```
## # A tibble: 10 x 4
##    structure      reps median_proj_gen mean_proj_gen
##    <fct>         <int>           <dbl>         <dbl>
##  1 well_mixed       50           18000         18240
##  2 random_waxman    50           18000         18260
##  3 comet_kite       50           21000         21220
##  4 windmill         50           26000         26100
##  5 lattice          50           27000         27460
##  6 clique_ring      50           36000         36020
##  7 cycle            50           69000         68840
##  8 linear_chain     50           69000         69080
##  9 wheel            50          135481.       135502.
## 10 star             50          361785.       366603.
```

```
kruskal.test(
  formula = proj_gen_max ~ structure,
  data = time_to_max_single_gradient
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  proj_gen_max by structure
## Kruskal-Wallis chi-squared = 490.93, df = 9, p-value < 2.2e-16
```

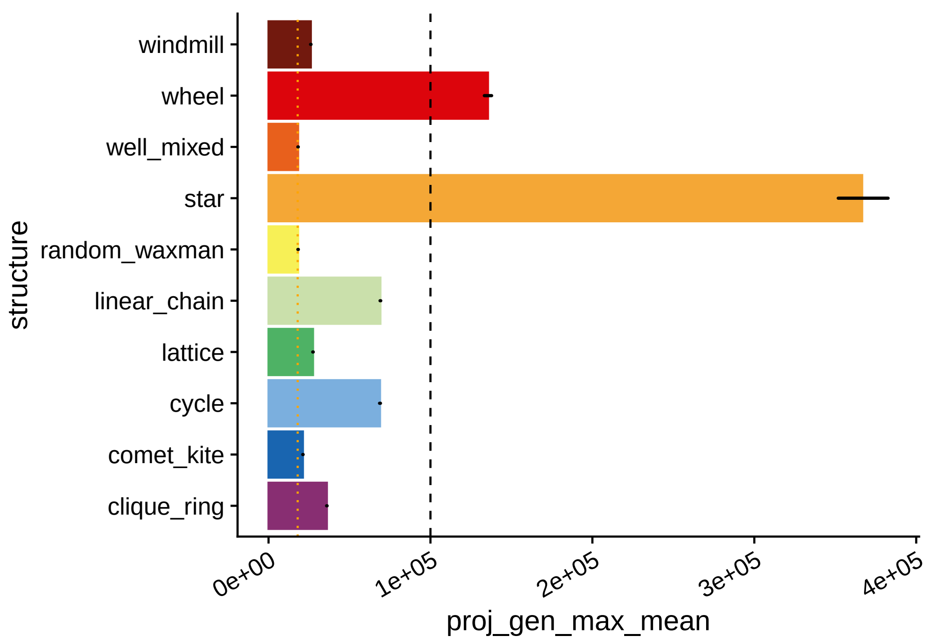
```
wc_results <- pairwise.wilcox.test(
  x = time_to_max_single_gradient$proj_gen_max,
  g = time_to_max_single_gradient$structure,
  p.adjust.method   = "holm",
  exact = FALSE
)

single_gradient_proj_gen_max_wc_table <- kbl(wc_results$p.value) %>%
  kable_styling()

save_kable(
  single_gradient_proj_gen_max_wc_table,
  paste0(plot_dir, "/single_gradient_proj_gen_max_wc_table.pdf")
)

single_gradient_proj_gen_max_wc_table
```

	clique_ring	comet_kite	cycle	lattice	linear_chain	random_waxma
comet_kite	0	NA	NA	NA	NA	N
cycle	0	0	NA	NA	NA	N
lattice	0	0	0.0000000	NA	NA	N
linear_chain	0	0	0.2915242	0	NA	N
random_waxman	0	0	0.0000000	0	0	N
star	0	0	0.0000000	0	0	0.000000
well_mixed	0	0	0.0000000	0	0	0.821833
wheel	0	0	0.0000000	0	0	0.000000
windmill	0	0	0.0000000	0	0	0.000000

```
library(boot)
# Define sample mean function
samplemean <- function(x, d) {
  return(mean(x[d]))
}

summary_gen_to_max <- tibble(
  structure = character(),
  proj_gen_max_mean = double(),
  proj_gen_max_mean_ci_low = double(),
  proj_gen_max_mean_ci_high = double()
)

structures <- levels(time_to_max_single_gradient$structure)
for (struct in structures) {
  boot_result <- boot(
    data = filter(
      time_to_max_single_gradient,
      structure == struct
    )$proj_gen_max,
    statistic = samplemean,
    R = 10000
  )
  result_ci <- boot.ci(boot_result, conf = 0.99, type = "perc")
  m <- result_ci$t0
  low <- result_ci$percent[4]
  high <- result_ci$percent[5]

  summary_gen_to_max <- summary_gen_to_max %>%
    add_row(
      structure = struct,
      proj_gen_max_mean = m,
      proj_gen_max_mean_ci_low = low,
```

```
      proj_gen_max_mean_ci_high = high
    )
}

wm_median <- median(
  filter(time_to_max_single_gradient, structure == "well_mixed")$proj_gen_max
)

simple_time_to_max_plt <- ggplot(
    data = summary_gen_to_max,
    mapping = aes(
      x = structure,
      y = proj_gen_max_mean,
      fill = structure,
      color = structure
    )
  ) +
  # geom_point() +
  geom_col() +
  geom_linerange(
    aes(
      ymin = proj_gen_max_mean_ci_low,
      ymax = proj_gen_max_mean_ci_high
    ),
    color = "black",
    linewidth = 0.75,
    lineend = "round"
  ) +
  # scale_y_log10(
  #   guide = "axis_logticks"
  # ) +
  geom_hline(
    yintercept = max_generation,
    linetype = "dashed"
  ) +
  geom_hline(
    yintercept = wm_median,
    linetype = "dotted",
    color = "orange"
  ) +
  scale_color_discreterainbow() +
  scale_fill_discreterainbow() +
  coord_flip() +
  theme(
    legend.position = "none",
```

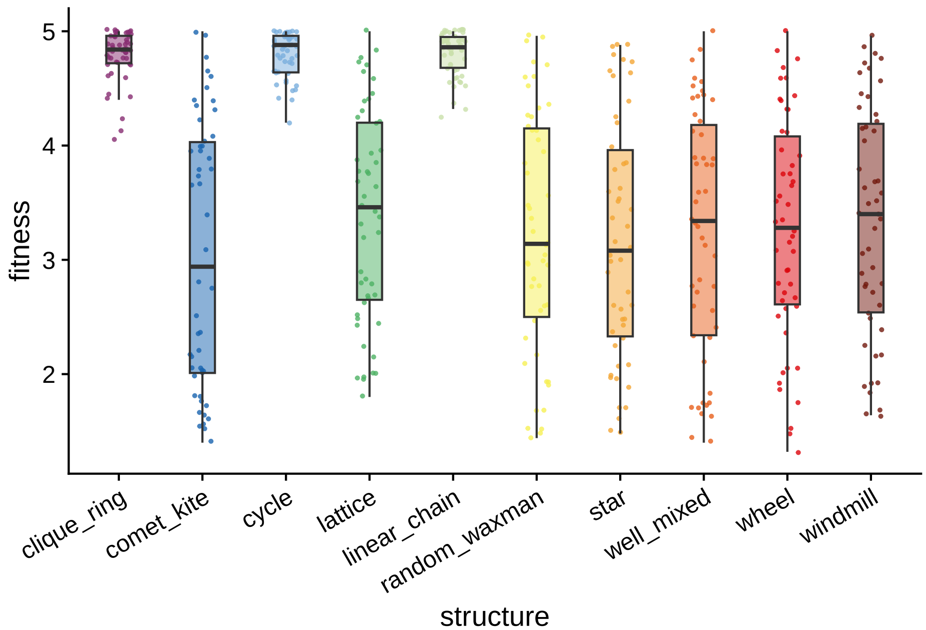
```
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/simple_time_to_max.pdf"),
  plot = simple_time_to_max_plt,
  width = 8,
  height = 4
)

simple_time_to_max_plt
```



4.2.2 Fitness in multi-path landscape

```
multipath_final_fitness_plt <- ggplot(
    data = filter(max_org_data, landscape == "Multipath"),
    mapping = aes(
      x = structure,
      y = fitness,
```

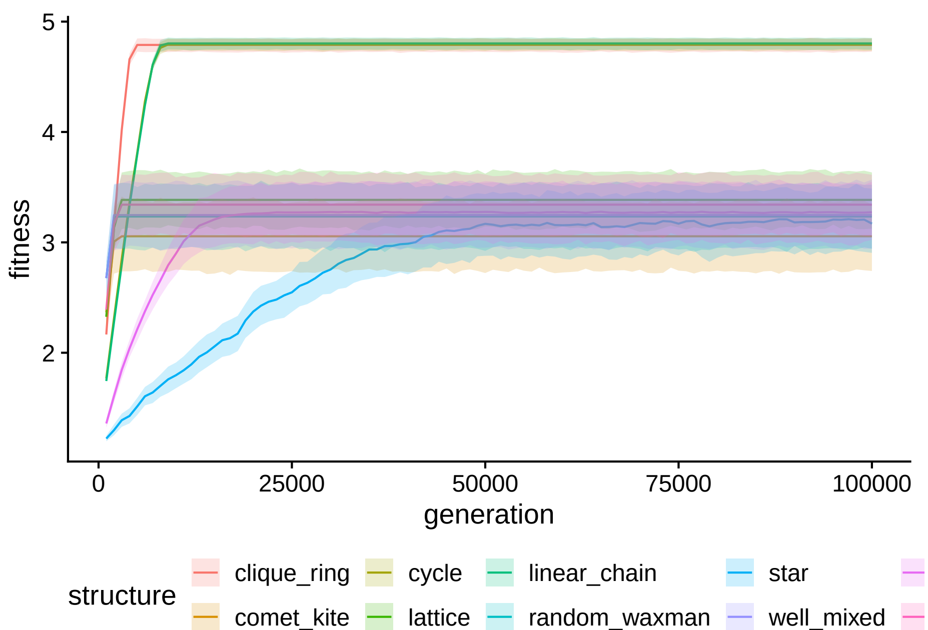
```
      fill = structure
    )
  ) +
  # geom_flat_violin(
  #   position = position_nudge(x = .2, y = 0),
  #   alpha = .8
  # ) +
  geom_point(
    mapping = aes(color = structure),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .3,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_color_discreterainbow() +
  scale_fill_discreterainbow() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/multipath_final_fitness.pdf"),
  plot = multipath_final_fitness_plt,
  width = 6,
  height = 4
)

multipath_final_fitness_plt
```



Max fitness over time

```
multipath_fitness_ts_plt <- ggplot(
    data = filter(max_org_data_ts, landscape == "Multipath"),
    mapping = aes(
      x = generation,
      y = fitness,
      color = structure,
      fill = structure
    )
  ) +
  stat_summary(fun = "mean", geom = "line") +
  stat_summary(
    fun.data = "mean_cl_boot",
    fun.args = list(conf.int = 0.95),
    geom = "ribbon",
    alpha = 0.2,
    linetype = 0
  ) +
  theme(legend.position = "bottom")

ggsave(
  plot = multipath_fitness_ts_plt,
  filename = paste0(
    plot_dir,
```

```
    "/multipath_fitness_ts.pdf"
  ),
  width = 15,
  height = 10
)

multipath_fitness_ts_plt
```



Rank ordering of fitness values

```
max_org_data %>%
  filter(landscape == "Multipath") %>%
  group_by(structure) %>%
  summarize(
    reps = n(),
    median_fitness = median(fitness),
    mean_fitness = mean(fitness)
  ) %>%
  arrange(
    desc(mean_fitness)
  )
```

```
## # A tibble: 10 x 4
##    structure      reps median_fitness mean_fitness
```

```
##    <fct>         <int>          <dbl>        <dbl>
##  1 linear_chain     50           4.86         4.80
##  2 cycle            50           4.88         4.79
##  3 clique_ring      50           4.84         4.79
##  4 lattice          50           3.46         3.38
##  5 windmill         50           3.4          3.34
##  6 wheel            50           3.28         3.27
##  7 well_mixed       50           3.34         3.25
##  8 random_waxman    50           3.14         3.23
##  9 star             50           3.08         3.17
## 10 comet_kite       50           2.94         3.06
```

```
kruskal.test(
  formula = fitness ~ structure,
  data = filter(max_org_data, landscape == "Multipath")
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  fitness by structure
## Kruskal-Wallis chi-squared = 246.11, df = 9, p-value < 2.2e-16
```

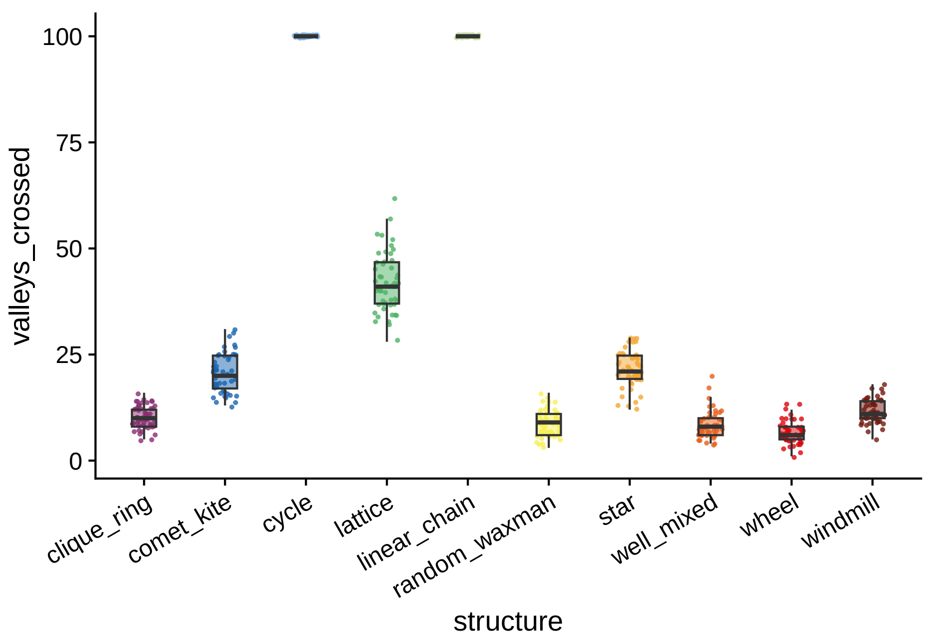
```
wc_results <- pairwise.wilcox.test(
  x = filter(max_org_data, landscape == "Multipath")$fitness,
  g = filter(max_org_data, landscape == "Multipath")$structure,
  p.adjust.method   = "holm",
  exact = FALSE
)

mp_fitness_wc_table <- kbl(wc_results$p.value) %>%
  kable_styling()

save_kable(
  mp_fitness_wc_table,
  paste0(plot_dir, "/multipath_fitness_wc_table.pdf")
)

mp_fitness_wc_table
```

4.2.3 Valleys crossed in valley-crossing landscape

	clique_ring	comet_kite	cycle	lattice	linear_chain	random_waxman	star	well_r
comet_kite	0	NA	NA	NA	NA	NA	NA	
cycle	1	0	NA	NA	NA	NA	NA	
lattice	0	1	0	NA	NA	NA	NA	
linear_chain	1	0	1	0	NA	NA	NA	
random_waxman	0	1	0	1	0	NA	NA	
star	0	1	0	1	0	1	NA	
well_mixed	0	1	0	1	0	1	1	
wheel	0	1	0	1	0	1	1	
windmill	0	1	0	1	0	1	1	

```r
valleycrossing_valleys_plt <- ggplot(
    data = filter(max_org_data, landscape == "Valley crossing"),
    mapping = aes(
      x = structure,
      y = valleys_crossed,
      fill = structure
    )
  ) +
  # geom_flat_violin(
  #   position = position_nudge(x = .2, y = 0),
  #   alpha = .8
  # ) +
  geom_point(
    mapping = aes(color = structure),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .3,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_color_discreterainbow() +
  scale_fill_discreterainbow() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )
```

```
ggsave(
  filename = paste0(plot_dir, "/valleycrossing_valleys_crossed.pdf"),
  plot = valleycrossing_valleys_plt,
  width = 6,
  height = 4
)

valleycrossing_valleys_plt
```



Rank ordering of fitness values

```
vc <- max_org_data %>%
  filter(landscape == "Valley crossing") %>%
  group_by(structure) %>%
  summarize(
    reps = n(),
    median_valleys_crossed = median(valleys_crossed),
    mean_valleys_crossed = mean(valleys_crossed),
    min_valleys_crossed = min(valleys_crossed)
  ) %>%
  arrange(
    desc(mean_valleys_crossed)
  )
vc
```

```
## # A tibble: 10 x 5
##    structure      reps median_valleys_crossed mean_valleys_crossed
##    <fct>         <int>                  <dbl>                <dbl>
##  1 cycle            50                    100                100
##  2 linear_chain     50                    100                100
##  3 lattice          50                     41                 41.9
##  4 star             50                     21                 21.5
##  5 comet_kite       50                     20                 20.5
##  6 windmill         50                     11                 11.6
##  7 clique_ring      50                     10                 10.3
##  8 random_waxman    50                      9                  8.76
##  9 well_mixed       50                      8                  8.46
## 10 wheel            50                      6                  6.6
## # i 1 more variable: min_valleys_crossed <dbl>
```

```r
vc$min_valleys_crossed
```

```
##  [1] 100 100  28  12  13   5   5   3   4   1
```

```r
kruskal.test(
  formula = valleys_crossed ~ structure,
  data = filter(max_org_data, landscape == "Valley crossing")
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  valleys_crossed by structure
## Kruskal-Wallis chi-squared = 444.04, df = 9, p-value < 2.2e-16
```

```r
wc_results <- pairwise.wilcox.test(
  x = filter(max_org_data, landscape == "Valley crossing")$valleys_crossed,
  g = filter(max_org_data, landscape == "Valley crossing")$structure,
  p.adjust.method   = "holm",
  exact = FALSE
)

vc_valleys_crossed_wc_table <- kbl(wc_results$p.value) %>%
  kable_styling()

save_kable(
  vc_valleys_crossed_wc_table,
  paste0(plot_dir, "/valley_crossing_valleys_wc_table.pdf")
)
```

	clique_ring	comet_kite	cycle	lattice	linear_chain	random_waxman	s
comet_kite	0.0000000	NA	NA	NA	NA	NA	
cycle	0.0000000	0.0000000	NA	NA	NA	NA	
lattice	0.0000000	0.0000000	0	NA	NA	NA	
linear_chain	0.0000000	0.0000000	NaN	0	NA	NA	
random_waxman	0.0414336	0.0000000	0	0	0	NA	
star	0.0000000	0.4016992	0	0	0	0.0000000	
well_mixed	0.0028498	0.0000000	0	0	0	0.4620430	
wheel	0.0000001	0.0000000	0	0	0	0.0029961	
windmill	0.0895493	0.0000000	0	0	0	0.0001323	

```
vc_valleys_crossed_wc_table
```

Chapter 5

Simple model - Squished toroid experiment analyses

5.1 Setup and Dependencies

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

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

```
experiment_slug <- "lattice-experiments"
working_directory <- paste(
  "experiments",
  "mabe2-exps",
  experiment_slug,
  sep = "/"
)
# Adjust working directory if being knitted for bookdown build.
if (bookdown) {
  working_directory <- paste0(
```

```
    bookdown_wd_prefix,
    working_directory
  )
}
```

```
# Configure our default graphing theme
theme_set(theme_cowplot())
# Create a directory to store plots
plot_dir <- paste(
  working_directory,
  "rmd_plots",
  sep = "/"
)

dir.create(
  plot_dir,
  showWarnings = FALSE
)
```

5.2 Max organism data analyses

```
max_generation <- 100000
max_org_data_path <- paste(
  working_directory,
  "data",
  "combined_max_org_data.csv",
  sep = "/"
)
# Data file has time series
max_org_data_ts <- read_csv(max_org_data_path)
max_org_data_ts <- max_org_data_ts %>%
  mutate(
    landscape = as.factor(landscape),
    structure = factor(
      structure,
      levels = c(
        "1_3600",
        "2_1800",
        "3_1200",
        "4_900",
        "15_240",
        "30_120",
```

```
        "60_60"
      )
    ),
  ) %>%
  mutate(
    valleys_crossed = case_when(
      landscape == "Valley crossing" ~ round(log(fitness, base = 1.5)),
      .default = 0
    )
  )
# Get tibble with just final generation
max_org_data <- max_org_data_ts %>%
  filter(generation == max_generation)
```

Check that replicate count for each condition matches expectations.

```
run_summary <- max_org_data %>%
  group_by(landscape, structure) %>%
  summarize(
    n = n()
  )
print(run_summary, n = 30)
```

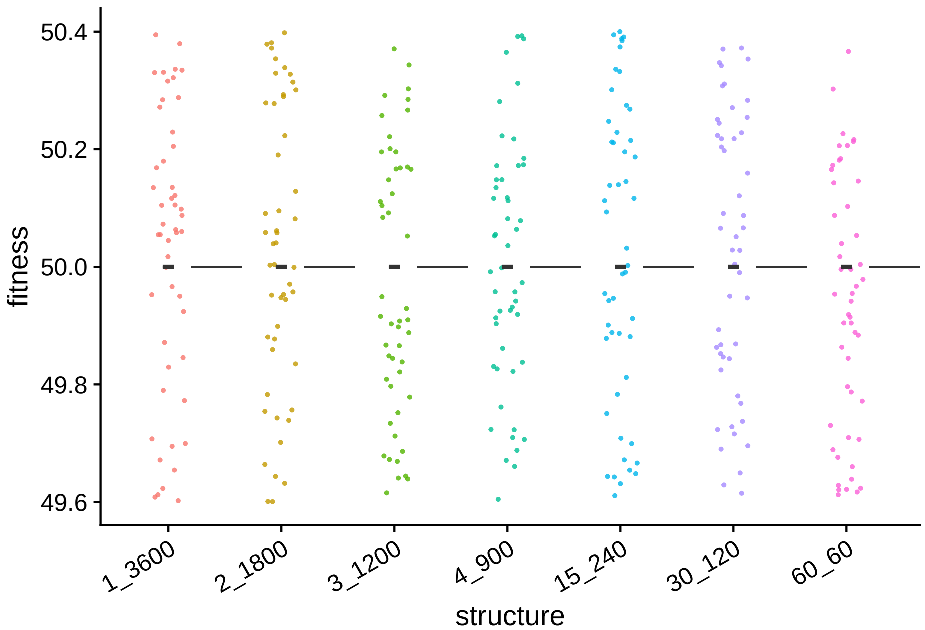
```
## # A tibble: 21 x 3
## # Groups:   landscape [3]
##    landscape       structure     n
##    <fct>           <fct>     <int>
##  1 Multipath       1_3600       50
##  2 Multipath       2_1800       50
##  3 Multipath       3_1200       50
##  4 Multipath       4_900        50
##  5 Multipath       15_240       50
##  6 Multipath       30_120       50
##  7 Multipath       60_60        50
##  8 Single gradient 1_3600       50
##  9 Single gradient 2_1800       50
## 10 Single gradient 3_1200       50
## 11 Single gradient 4_900        50
## 12 Single gradient 15_240       50
## 13 Single gradient 30_120       50
## 14 Single gradient 60_60        50
## 15 Valley crossing 1_3600       50
## 16 Valley crossing 2_1800       50
## 17 Valley crossing 3_1200       50
```

```
## 18 Valley crossing 4_900        50
## 19 Valley crossing 15_240       50
## 20 Valley crossing 30_120       50
## 21 Valley crossing 60_60        50
```

5.2.1 Fitness in smooth gradient landscape

```
single_gradient_final_fitness_plt <- ggplot(
    data = filter(max_org_data, landscape == "Single gradient"),
    mapping = aes(
      x = structure,
      y = fitness,
      fill = structure
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping = aes(color = structure),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/single_gradient_final_fitness.pdf"),
  plot = single_gradient_final_fitness_plt,
  width = 15,
  height = 10
)
```

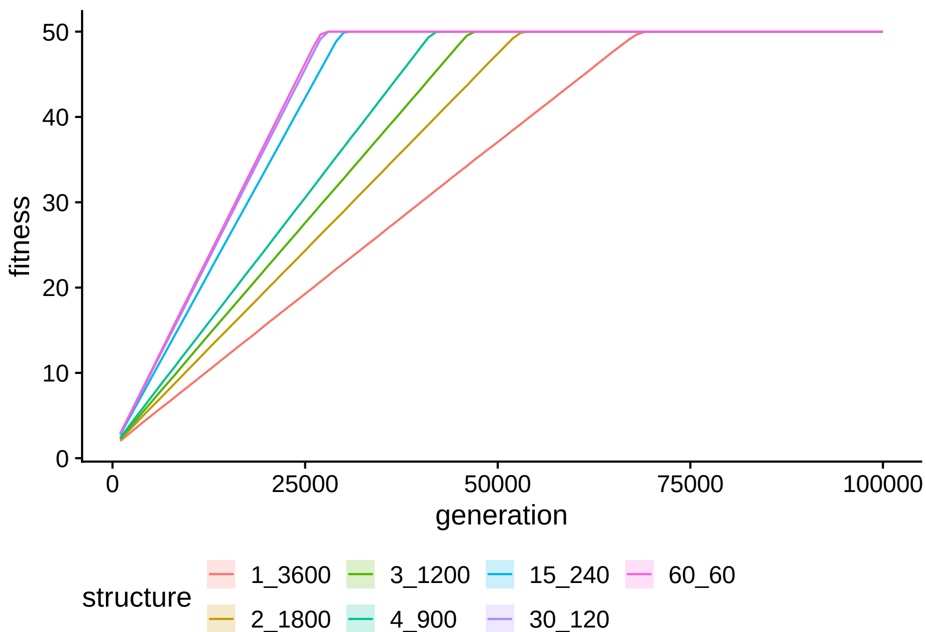
```
single_gradient_final_fitness_plt
```



Max fitness over time

```
single_gradient_fitness_ts_plt <- ggplot(
    data = filter(max_org_data_ts, landscape == "Single gradient"),
    mapping = aes(
      x = generation,
      y = fitness,
      color = structure,
      fill = structure
    )
  ) +
  stat_summary(fun = "mean", geom = "line") +
  stat_summary(
    fun.data = "mean_cl_boot",
    fun.args = list(conf.int = 0.95),
    geom = "ribbon",
    alpha = 0.2,
    linetype = 0
  ) +
  theme(legend.position = "bottom")

ggsave(
```

```
  plot = single_gradient_fitness_ts_plt,
  filename = paste0(
    plot_dir,
    "/single_gradient_fitness_ts.pdf"
  ),
  width = 15,
  height = 10
)

single_gradient_fitness_ts_plt
```



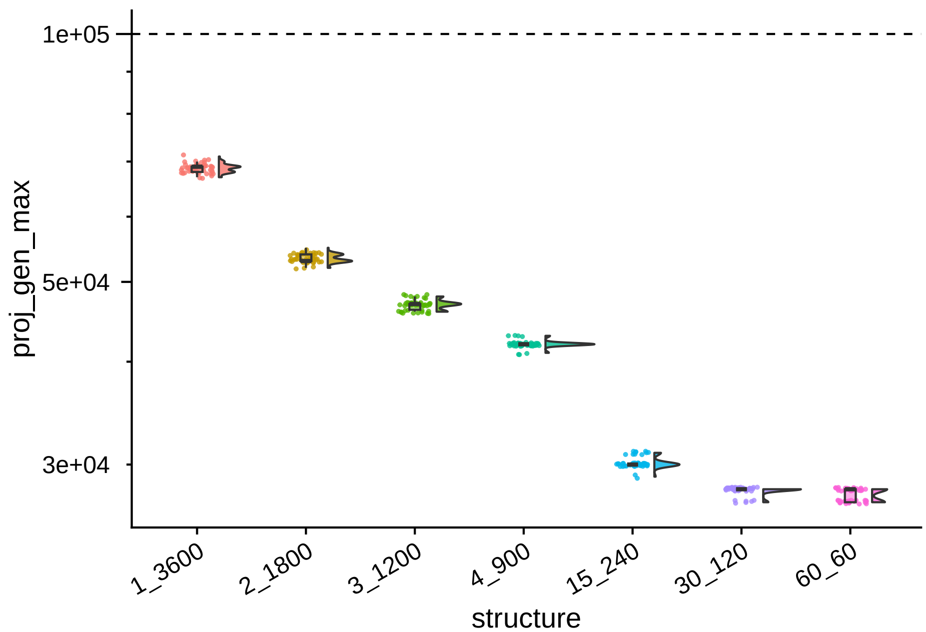
Time to maximum fitness

```
# Find all rows with maximum fitness value, then get row with minimum generation,
#  then project out expected generation to max (for runs that didn't finish)
max_possible_fit = 50
time_to_max_single_gradient <- max_org_data_ts %>%
  filter(landscape == "Single gradient") %>%
  group_by(rep, structure) %>%
  slice_max(
    fitness,
    n = 1
  ) %>%
  slice_min(
    generation,
```

```
    n = 1
  ) %>%
  mutate(
    proj_gen_max = (max_possible_fit / fitness) * generation
  )
```

```
single_gradient_gen_max_proj_plt <- ggplot(
    data = time_to_max_single_gradient,
    mapping = aes(
      x = structure,
      y = proj_gen_max,
      fill = structure
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping = aes(color = structure),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_y_log10(
    guide = "axis_logticks"
  ) +
  # scale_y_continuous(
  #   trans="pseudo_log",
  #   breaks = c(10, 100, 1000, 10000, 100000, 1000000)
  #   ,limits = c(10, 100, 1000, 10000, 100000, 1000000)
  # ) +
  geom_hline(
    yintercept = max_generation,
    linetype = "dashed"
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
```

```
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/single_gradient_gen_max_proj.pdf"),
  plot = single_gradient_gen_max_proj_plt,
  width = 15,
  height = 10
)

single_gradient_gen_max_proj_plt
```



Rank ordering of time to max fitness values

```
time_to_max_single_gradient %>%
  group_by(structure) %>%
  summarize(
    reps = n(),
    median_proj_gen = median(proj_gen_max),
    mean_proj_gen = mean(proj_gen_max)
  ) %>%
  arrange(
    mean_proj_gen
  )
```

```
## # A tibble: 7 x 4
##   structure  reps median_proj_gen mean_proj_gen
##   <fct>     <int>           <dbl>         <dbl>
## 1 60_60        50           28000         27540
## 2 30_120       50           28000         27880
## 3 15_240       50           30000         30160
## 4 4_900        50           42000         42020
## 5 3_1200       50           47000         46900
## 6 2_1800       50           53000         53340
## 7 1_3600       50           69000         68700
```

```r
kruskal.test(
  formula = proj_gen_max ~ structure,
  data = time_to_max_single_gradient
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  proj_gen_max by structure
## Kruskal-Wallis chi-squared = 341.17, df = 6, p-value < 2.2e-16
```

```r
wc_results <- pairwise.wilcox.test(
  x = time_to_max_single_gradient$proj_gen_max,
  g = time_to_max_single_gradient$structure,
  p.adjust.method   = "holm",
  exact = FALSE
)

single_gradient_proj_gen_max_wc_table <- kbl(wc_results$p.value) %>%
  kable_styling()

save_kable(
  single_gradient_proj_gen_max_wc_table,
  paste0(plot_dir, "/single_gradient_proj_gen_max_wc_table.pdf")
)
single_gradient_proj_gen_max_wc_table
```

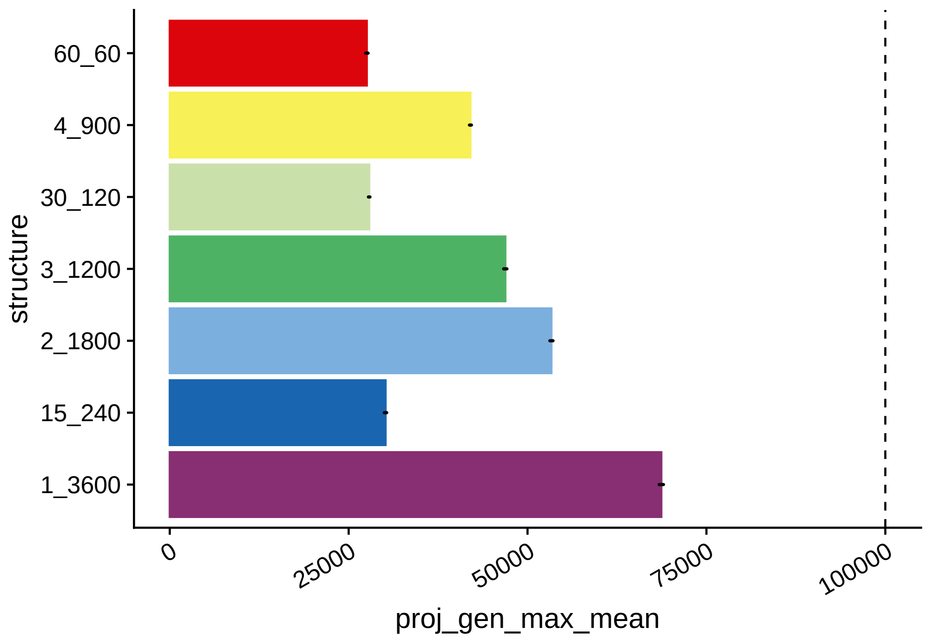
```r
library(boot)
# Define sample mean function
samplemean <- function(x, d) {
  return(mean(x[d]))
}
```

	1_3600	2_1800	3_1200	4_900	15_240	30_120
2_1800	0	NA	NA	NA	NA	NA
3_1200	0	0	NA	NA	NA	NA
4_900	0	0	0	NA	NA	NA
15_240	0	0	0	0	NA	NA
30_120	0	0	0	0	0	NA
60_60	0	0	0	0	0	0.0001966

```
summary_gen_to_max <- tibble(
  structure = character(),
  proj_gen_max_mean = double(),
  proj_gen_max_mean_ci_low = double(),
  proj_gen_max_mean_ci_high = double()
)

structures <- levels(time_to_max_single_gradient$structure)
for (struct in structures) {
  boot_result <- boot(
    data = filter(
      time_to_max_single_gradient,
      structure == struct
    )$proj_gen_max,
    statistic = samplemean,
    R = 10000
  )
  result_ci <- boot.ci(boot_result, conf = 0.99, type = "perc")
  m <- result_ci$t0
  low <- result_ci$percent[4]
  high <- result_ci$percent[5]

  summary_gen_to_max <- summary_gen_to_max %>%
    add_row(
      structure = struct,
      proj_gen_max_mean = m,
      proj_gen_max_mean_ci_low = low,
      proj_gen_max_mean_ci_high = high
    )
}

wm_median <- median(
  filter(time_to_max_single_gradient, structure == "well_mixed")$proj_gen_max
)

simple_time_to_max_plt <- ggplot(
```

```
    data = summary_gen_to_max,
    mapping = aes(
      x = structure,
      y = proj_gen_max_mean,
      fill = structure,
      color = structure
    )
  ) +
  # geom_point() +
  geom_col() +
  geom_linerange(
    aes(
      ymin = proj_gen_max_mean_ci_low,
      ymax = proj_gen_max_mean_ci_high
    ),
    color = "black",
    linewidth = 0.75,
    lineend = "round"
  ) +
  # scale_y_log10(
  #   guide = "axis_logticks"
  # ) +
  geom_hline(
    yintercept = max_generation,
    linetype = "dashed"
  ) +
  geom_hline(
    yintercept = wm_median,
    linetype = "dotted",
    color = "orange"
  ) +
  scale_color_discreterainbow() +
  scale_fill_discreterainbow() +
  coord_flip() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/simple_time_to_max.pdf"),
  plot = simple_time_to_max_plt,
```

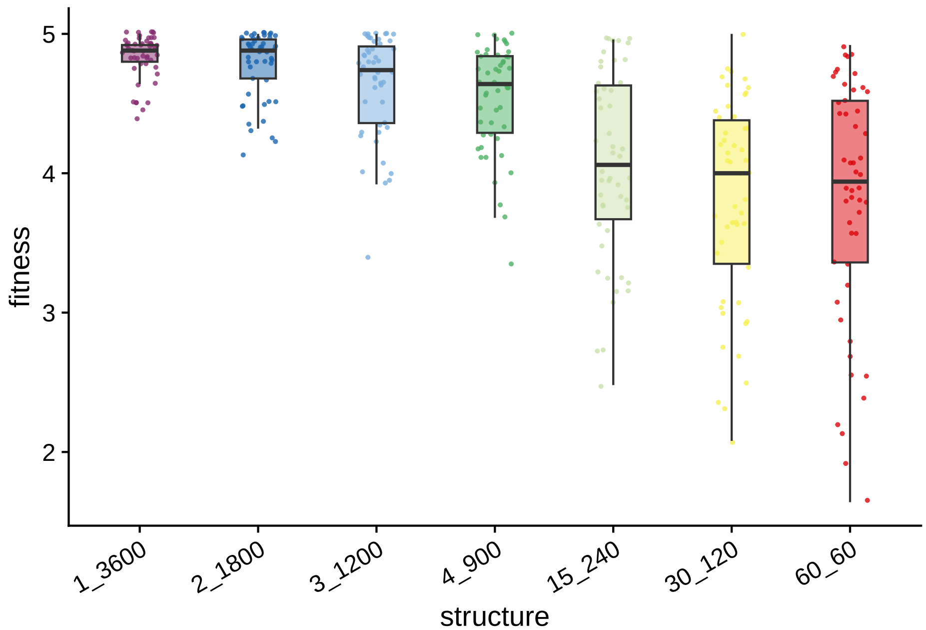
```
  width = 8,
  height = 4
)

simple_time_to_max_plt
```



5.2.2 Fitness in multi-path landscape

```
multipath_final_fitness_plt <- ggplot(
    data = filter(max_org_data, landscape == "Multipath"),
    mapping = aes(
      x = structure,
      y = fitness,
      fill = structure
    )
  ) +
  # geom_flat_violin(
  #   position = position_nudge(x = .2, y = 0),
  #   alpha = .8
  # ) +
  geom_point(
    mapping = aes(color = structure),
```

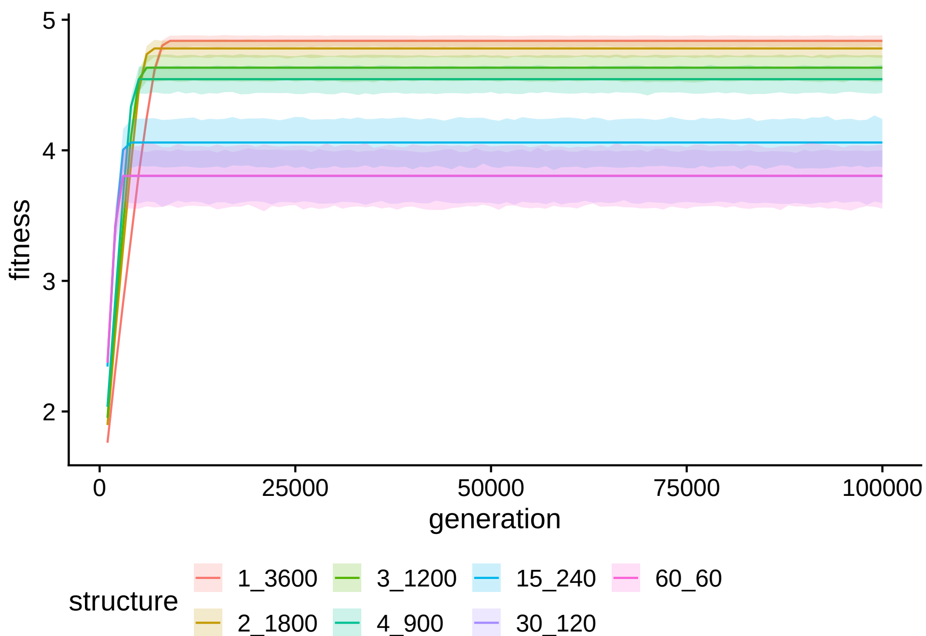
```
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .3,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_color_discreterainbow() +
  scale_fill_discreterainbow() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/multipath_final_fitness.pdf"),
  plot = multipath_final_fitness_plt,
  width = 6,
  height = 4
)

multipath_final_fitness_plt
```



Max fitness over time

```
multipath_fitness_ts_plt <- ggplot(
    data = filter(max_org_data_ts, landscape == "Multipath"),
    mapping = aes(
      x = generation,
      y = fitness,
      color = structure,
      fill = structure
    )
  ) +
  stat_summary(fun = "mean", geom = "line") +
  stat_summary(
    fun.data = "mean_cl_boot",
    fun.args = list(conf.int = 0.95),
    geom = "ribbon",
    alpha = 0.2,
    linetype = 0
  ) +
  theme(legend.position = "bottom")

ggsave(
  plot = multipath_fitness_ts_plt,
  filename = paste0(
    plot_dir,
```

```
    "/multipath_fitness_ts.pdf"
  ),
  width = 15,
  height = 10
)

multipath_fitness_ts_plt
```



Rank ordering of fitness values

```
max_org_data %>%
  filter(landscape == "Multipath") %>%
  group_by(structure) %>%
  summarize(
    reps = n(),
    median_fitness = median(fitness),
    mean_fitness = mean(fitness)
  ) %>%
  arrange(
    desc(mean_fitness)
  )
```

```
## # A tibble: 7 x 4
##   structure  reps median_fitness mean_fitness
```

```
##   <fct>     <int>          <dbl>        <dbl>
## 1 1_3600       50           4.88         4.84
## 2 2_1800       50           4.88         4.78
## 3 3_1200       50           4.74         4.63
## 4 4_900        50           4.64         4.54
## 5 15_240       50           4.06         4.06
## 6 60_60        50           3.94         3.81
## 7 30_120       50           4            3.80
```

```
kruskal.test(
  formula = fitness ~ structure,
  data = filter(max_org_data, landscape == "Multipath")
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  fitness by structure
## Kruskal-Wallis chi-squared = 144.73, df = 6, p-value < 2.2e-16
```

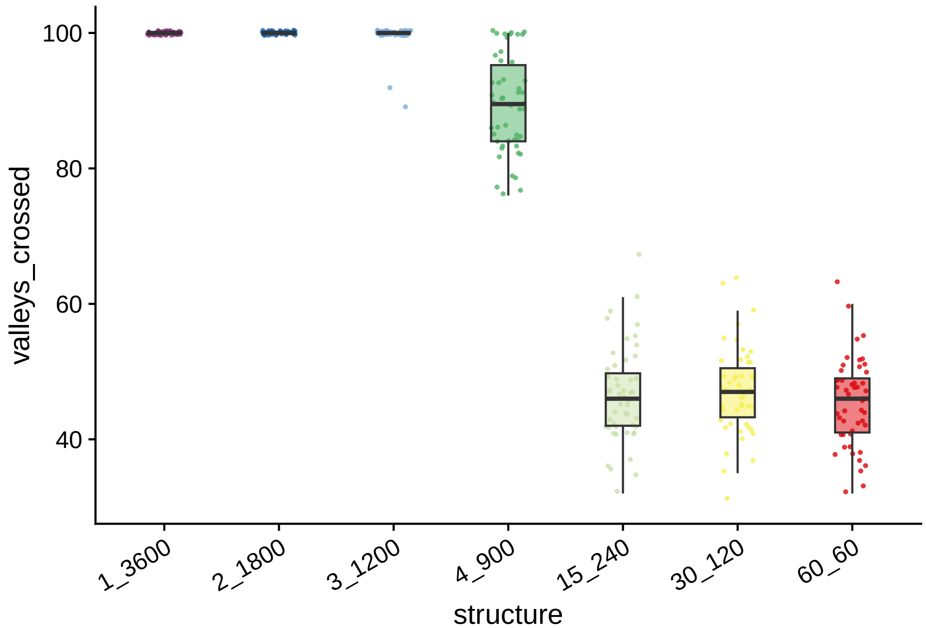
```
wc_results <- pairwise.wilcox.test(
  x = filter(max_org_data, landscape == "Multipath")$fitness,
  g = filter(max_org_data, landscape == "Multipath")$structure,
  p.adjust.method   = "holm",
  exact = FALSE
)

mp_fitness_wc_table <- kbl(wc_results$p.value) %>%
  kable_styling()

save_kable(
  mp_fitness_wc_table,
  paste0(plot_dir, "/multipath_fitness_wc_table.pdf")
)
mp_fitness_wc_table
```

5.2.3 Valleys crossed in valley-crossing landscape

```
valleycrossing_valleys_plt <- ggplot(
    data = filter(max_org_data, landscape == "Valley crossing"),
    mapping = aes(
      x = structure,
```

	1_3600	2_1800	3_1200	4_900	15_240	30_120
2_1800	1.0000000	NA	NA	NA	NA	NA
3_1200	0.0389539	0.2309342	NA	NA	NA	NA
4_900	0.0000552	0.0022081	0.6036094	NA	NA	NA
15_240	0.0000000	0.0000001	0.0000387	0.0022081	NA	NA
30_120	0.0000000	0.0000000	0.0000000	0.0000003	0.4456978	NA
60_60	0.0000000	0.0000000	0.0000002	0.0000094	0.6036094	1

```
      y = valleys_crossed,
      fill = structure
    )
  ) +
  # geom_flat_violin(
  #   position = position_nudge(x = .2, y = 0),
  #   alpha = .8
  # ) +
  geom_point(
    mapping = aes(color = structure),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .3,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_color_discreterainbow() +
  scale_fill_discreterainbow() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )
ggsave(
  filename = paste0(plot_dir, "/valleycrossing_valleys_crossed.pdf"),
  plot = valleycrossing_valleys_plt,
  width = 6,
  height = 4
)

valleycrossing_valleys_plt
```




```
vc <- max_org_data %>%
  filter(landscape == "Valley crossing") %>%
  group_by(structure) %>%
  summarize(
    reps = n(),
    median_valleys_crossed = median(valleys_crossed),
    mean_valleys_crossed = mean(valleys_crossed),
    min_valleys_crossed = min(valleys_crossed)
  ) %>%
  arrange(
    desc(mean_valleys_crossed)
  )
vc
```

```
## # A tibble: 7 x 5
##   structure  reps median_valleys_crossed mean_valleys_crossed
##   <fct>     <int>                  <dbl>                <dbl>
## 1 1_3600       50                  100                  100
## 2 2_1800       50                  100                  100
## 3 3_1200       50                  100                   99.6
## 4 4_900        50                   89.5                 89.3
## 5 30_120       50                   47                   47.2
## 6 15_240       50                   46                   46.6
## 7 60_60        50                   46                   45.5
## # i 1 more variable: min_valleys_crossed <dbl>
```

	1__3600	2__1800	3__1200	4__900	15__240	30__120
2__1800	NaN	NA	NA	NA	NA	NA
3__1200	0.796952	0.796952	NA	NA	NA	NA
4__900	0.000000	0.000000	0	NA	NA	NA
15__240	0.000000	0.000000	0	0	NA	NA
30__120	0.000000	0.000000	0	0	0.9787605	NA
60__60	0.000000	0.000000	0	0	0.9787605	0.796952

```
vc$min_valleys_crossed
```

```
## [1] 100 100  89  76  31  32  32
```

```
kruskal.test(
  formula = valleys_crossed ~ structure,
  data = filter(max_org_data, landscape == "Valley crossing")
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  valleys_crossed by structure
## Kruskal-Wallis chi-squared = 309.49, df = 6, p-value < 2.2e-16
```

```
wc_results <- pairwise.wilcox.test(
  x = filter(max_org_data, landscape == "Valley crossing")$valleys_crossed,
  g = filter(max_org_data, landscape == "Valley crossing")$structure,
  p.adjust.method   = "holm",
  exact = FALSE
)

vc_valleys_crossed_wc_table <- kbl(wc_results$p.value) %>%
  kable_styling()

save_kable(
  vc_valleys_crossed_wc_table,
  paste0(plot_dir, "/valley_crossing_valleys_wc_table.pdf")
)
vc_valleys_crossed_wc_table
```

Chapter 6

Avida - Varied graph structure experiment analyses

6.1 Dependencies and setup

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

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

```
experiment_slug <- "2025-04-17-vary-structs"
working_directory <- paste(
  "experiments",
  experiment_slug,
  "analysis",
  sep = "/"
)
# Adjust working directory if being knitted for bookdown build.
```

```r
if (bookdown) {
  working_directory <- paste0(
    bookdown_wd_prefix,
    working_directory
  )
}
```

```r
# Configure our default graphing theme
theme_set(theme_cowplot())
# Create a directory to store plots
plot_dir <- paste(
  working_directory,
  "plots",
  sep = "/"
)

dir.create(
  plot_dir,
  showWarnings = FALSE
)
```

```r
focal_graphs <- c(
  "star",
  "random-waxman",
  "comet-kite",
  "linear-chain",
  "cycle",
  "clique-ring",
  "toroidal-lattice",
  "well-mixed",
  "wheel",
  "windmill"
)

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

data <- data %>%
  mutate(
```

```
    graph_type = factor(
      graph_type,
      levels = c(
        "star",
        "random-waxman",
        "comet-kite",
        "linear-chain",
        "cycle",
        "clique-ring",
        "toroidal-lattice",
        "well-mixed",
        "wheel",
        "windmill"
      )
    ),
    ENVIRONMENT_FILE = as.factor(ENVIRONMENT_FILE)
  )
data <- data %>% filter(
  graph_type %in% focal_graphs
)
data <- data %>% filter(reached_target_update)
```

```
time_series_path <- paste(
  working_directory,
  "data",
  "time_series.csv",
  sep = "/"
)
time_series_data <- read_csv(time_series_path)

time_series_data <- time_series_data %>%
  mutate(
    graph_type = factor(
      graph_type,
      levels = c(
        "star",
        "random-waxman",
        "comet-kite",
        "linear-chain",
        "cycle",
        "clique-ring",
        "toroidal-lattice",
        "well-mixed",
        "wheel",
        "windmill"
```

```
      )
    ),
    ENVIRONMENT_FILE = as.factor(ENVIRONMENT_FILE),
    seed = as.factor(seed)
  )
time_series_data <- time_series_data %>% filter(seed %in% data$seed)
time_series_data <- time_series_data %>% filter(
  graph_type %in% focal_graphs
)
```

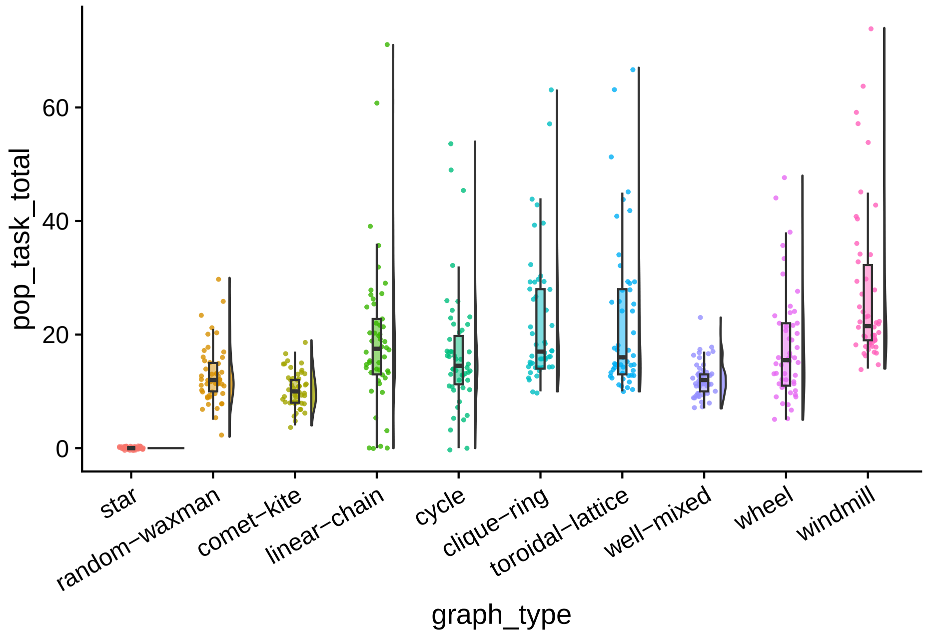
```
# Check that all runs completed
data %>%
  filter(update == 400000) %>%
  group_by(graph_type) %>%
  summarize(
    n = n()
  )
```

```
## # A tibble: 10 x 2
##    graph_type           n
##    <fct>            <int>
##  1 star                50
##  2 random-waxman       50
##  3 comet-kite          50
##  4 linear-chain        50
##  5 cycle               50
##  6 clique-ring         50
##  7 toroidal-lattice    50
##  8 well-mixed          50
##  9 wheel               50
## 10 windmill            50
```

6.2 Number of tasks completed

```
pop_tasks_total_plt <- ggplot(
  data = data,
  mapping = aes(
    x = graph_type,
    y = pop_task_total,
    fill = graph_type
  )
) +
```

```
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/pop_tasks_total.pdf"),
  plot = pop_tasks_total_plt,
  width = 15,
  height = 10
)

pop_tasks_total_plt
```




```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_pop_tasks = median(pop_task_total),
    mean_pop_tasks = mean(pop_task_total)
  ) %>%
  arrange(
    desc(mean_pop_tasks)
  )
```

```
## # A tibble: 10 x 4
##    graph_type        reps median_pop_tasks mean_pop_tasks
##    <fct>            <int>            <dbl>          <dbl>
##  1 windmill            50             21.5           27.4
##  2 toroidal-lattice    50             16             22.4
##  3 clique-ring         50             17             22.1
##  4 linear-chain        50             17.5           19.1
##  5 wheel               50             15.5           17.8
##  6 cycle               50             14.5           16.6
##  7 random-waxman       50             12             12.8
##  8 well-mixed          50             12             12.2
##  9 comet-kite          50             10             10.3
## 10 star                50              0              0
```

	star	random-waxman	comet-kite	linear-chain	cycle	clique-ring	toroidal-lattice
random-waxman	0	NA	NA	NA	NA	NA	NA
comet-kite	0	0.0755715	NA	NA	NA	NA	NA
linear-chain	0	0.0040473	0.0000027	NA	NA	NA	NA
cycle	0	0.1530200	0.0002750	1.0000000	NA	NA	NA
clique-ring	0	0.0000008	0.0000000	1.0000000	0.0702670	NA	NA
toroidal-lattice	0	0.0000297	0.0000000	1.0000000	0.3032752	1.0000000	NA
well-mixed	0	1.0000000	0.0542973	0.0002739	0.0505844	0.0000000	0.0000018
wheel	0	0.0681554	0.0000297	1.0000000	1.0000000	0.2644489	0.6033308
windmill	0	0.0000000	0.0000000	0.0058779	0.0000035	0.0302955	0.0302955

```
kruskal.test(
  formula = pop_task_total ~ graph_type,
  data = data
)
```

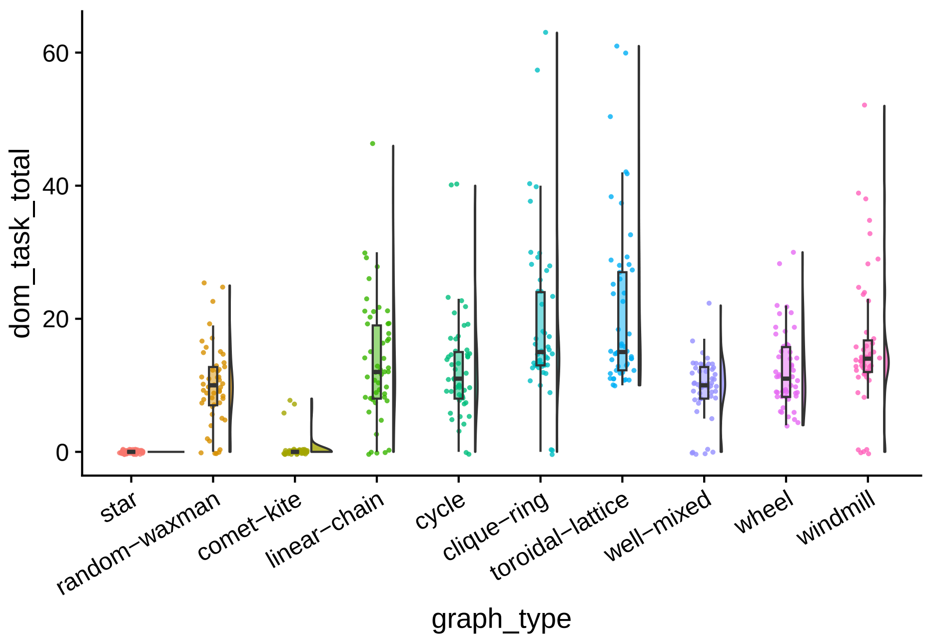
```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_task_total by graph_type
## Kruskal-Wallis chi-squared = 252.46, df = 9, p-value < 2.2e-16
```

```
wc_results <- pairwise.wilcox.test(
  x = data$pop_task_total,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

pop_task_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(pop_task_wc_table, paste0(plot_dir, "/pop_task_wc_table.pdf"))
pop_task_wc_table
```

6.3 Dominant tasks

```
dom_tasks_total_plt <- ggplot(
  data = data,
  mapping = aes(
    x = graph_type,
    y = dom_task_total,
```

```
    fill = graph_type
  )
) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/dom_tasks_total.pdf"),
  plot = dom_tasks_total_plt,
  width = 15,
  height = 10
)

dom_tasks_total_plt
```




```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_dom_task_total = median(dom_task_total),
    mean_dom_task_total = mean(dom_task_total)
  ) %>%
  arrange(
    desc(mean_dom_task_total)
  )
```

```
## # A tibble: 10 x 4
##    graph_type        reps median_dom_task_total mean_dom_task_total
##    <fct>            <int>                 <dbl>               <dbl>
##  1 toroidal-lattice    50                    15               21.4
##  2 clique-ring         50                    15               19.2
##  3 windmill            50                    14               16.1
##  4 linear-chain        50                    12               13.6
##  5 cycle               50                    11               12.6
##  6 wheel               50                    11               12.4
##  7 random-waxman       50                    10                9.92
##  8 well-mixed          50                    10                9.62
##  9 comet-kite          50                     0                0.42
## 10 star                50                     0                0
```

	star	random-waxman	comet-kite	linear-chain	cycle	clique-ring
random-waxman	0.0000000	NA	NA	NA	NA	NA
comet-kite	0.9646146	0.0000000	NA	NA	NA	NA
linear-chain	0.0000000	0.4710663	0	NA	NA	NA
cycle	0.0000000	0.9646146	0	1.0000000	NA	NA
clique-ring	0.0000000	0.0000042	0	0.0907079	0.0083936	NA
toroidal-lattice	0.0000000	0.0000001	0	0.0112805	0.0003268	1.0000000
well-mixed	0.0000000	1.0000000	0	0.5076566	0.9646146	0.0000000
wheel	0.0000000	0.9646146	0	1.0000000	1.0000000	0.0033520
windmill	0.0000000	0.0009903	0	0.9865397	0.4321933	0.9865397

```
kruskal.test(
  formula = dom_task_total ~ graph_type,
  data = data
)
```

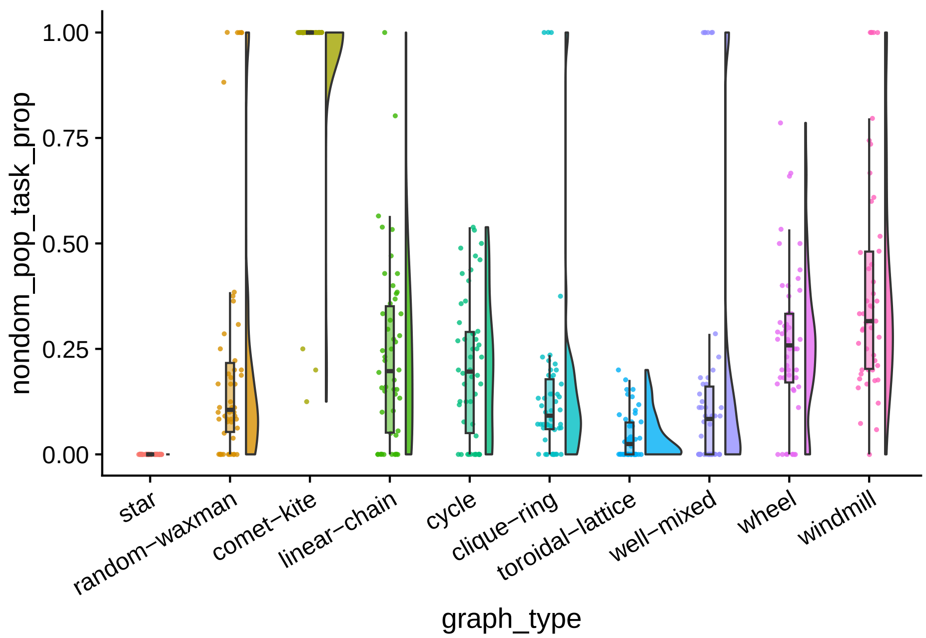
```
##
##  Kruskal-Wallis rank sum test
##
## data:  dom_task_total by graph_type
## Kruskal-Wallis chi-squared = 262.43, df = 9, p-value < 2.2e-16
```

```
wc_results <- pairwise.wilcox.test(
  x = data$dom_task_total,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

dom_task_total_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(dom_task_total_wc_table, paste0(plot_dir, "/dom_task_total_wc_table.pdf"))
dom_task_total_wc_table
```

Tasks done by organisms not in dominant taxon:

```
data <- data %>%
  mutate(
    nondom_pop_task_prop = case_when(
      pop_task_total == 0 ~ 0,
      .default = (pop_task_total - dom_task_total) / (pop_task_total)
    )
  )
```

```
nondom_tasks_total_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = nondom_pop_task_prop,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/non_dom_tasks_total.pdf"),
  plot = nondom_tasks_total_plt,
  width = 15,
  height = 10
)

nondom_tasks_total_plt
```




```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_nondom_pop_task_prop = median(nondom_pop_task_prop),
    mean_nondom_pop_task_prop = mean(nondom_pop_task_prop)
  ) %>%
  arrange(
    desc(mean_nondom_pop_task_prop)
  )
```

```
## # A tibble: 10 x 4
##    graph_type        reps median_nondom_pop_task_prop mean_nondom_pop_task_prop
##    <fct>            <int>                       <dbl>                     <dbl>
##  1 comet-kite          50                      1                         0.952
##  2 windmill            50                      0.316                     0.398
##  3 wheel               50                      0.258                     0.264
##  4 linear-chain        50                      0.197                     0.234
##  5 random-waxman       50                      0.106                     0.222
##  6 cycle               50                      0.196                     0.206
##  7 well-mixed          50                      0.0839                    0.177
##  8 clique-ring         50                      0.0917                    0.154
##  9 toroidal-lattice    50                      0.0245                    0.0432
## 10 star                50                      0                         0
```

	star	random-waxman	comet-kite	linear-chain	cycle	clique-ring	toroidal-lattic
random-waxman	0e+00	NA	NA	NA	NA	NA	N
comet-kite	0e+00	0.0000000	NA	NA	NA	NA	N
linear-chain	0e+00	0.9560919	0	NA	NA	NA	N
cycle	0e+00	1.0000000	0	1.0000000	NA	NA	N
clique-ring	0e+00	1.0000000	0	0.1142983	0.1337364	NA	N
toroidal-lattice	2e-07	0.0001655	0	0.0000044	0.0000102	0.0019354	N
well-mixed	0e+00	0.4737074	0	0.0370381	0.0501364	1.0000000	0.473707
wheel	0e+00	0.0560536	0	1.0000000	0.8350382	0.0002916	0.000000
windmill	0e+00	0.0000660	0	0.0129065	0.0022312	0.0000000	0.000000

```
kruskal.test(
  formula = nondom_pop_task_prop ~ graph_type,
  data = data
)
```

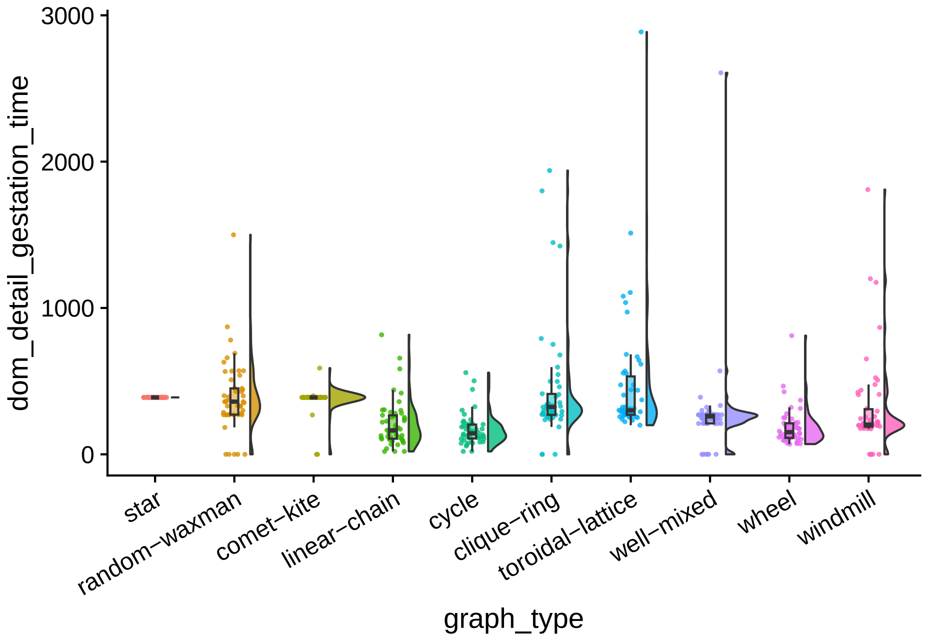
```
##
##  Kruskal-Wallis rank sum test
##
## data:  nondom_pop_task_prop by graph_type
## Kruskal-Wallis chi-squared = 256.7, df = 9, p-value < 2.2e-16
```

```
wc_results <- pairwise.wilcox.test(
  x = data$nondom_pop_task_prop,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

nondom_pop_task_prop_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(nondom_pop_task_prop_wc_table, paste0(plot_dir, "/nondom_pop_task_prop_wc_table.pdf"))
nondom_pop_task_prop_wc_table
```

```
# kruskal.test(
#   formula = dom_task_total ~ graph_type,
#   data = filter(completed_runs_data)
# )
```

6.4 Dominant gestation time

```
dom_gestation_time_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = dom_detail_gestation_time,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/dom_gestation_time.pdf"),
  plot = dom_gestation_time_plt,
  width = 15,
  height = 10
)

dom_gestation_time_plt
```




```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_dom_detail_gestation_time = median(dom_detail_gestation_time),
    mean_dom_detail_gestation_time = mean(dom_detail_gestation_time)
  ) %>%
  arrange(
    desc(mean_dom_detail_gestation_time)
  )
```

```
## # A tibble: 10 x 4
##    graph_type        reps median_dom_detail_gestation_t~1 mean_dom_detail_gest~2
##    <fct>            <int>                           <dbl>                  <dbl>
##  1 toroidal-lattice    50                            300.                   480.
##  2 clique-ring         50                            324.                   440.
##  3 random-waxman       50                            360                    393.
##  4 star                50                            389                    389
##  5 comet-kite          50                            389                    375.
##  6 windmill            50                            204.                   315.
##  7 well-mixed          50                            260.                   281.
##  8 linear-chain        50                            164                    208.
##  9 wheel               50                            152.                   187.
## 10 cycle               50                            144.                   169.
```

	star	random-waxman	comet-kite	linear-chain	cycle	clique-ring
random-waxman	1.0000000	NA	NA	NA	NA	NA
comet-kite	1.0000000	1.0000000	NA	NA	NA	NA
linear-chain	0.0000000	0.0000161	0.0000000	NA	NA	NA
cycle	0.0000000	0.0000001	0.0000000	1.0000000	NA	NA
clique-ring	0.0039456	1.0000000	0.0375642	0.0000080	0.0000000	NA
toroidal-lattice	0.1400273	1.0000000	0.5320337	0.0000001	0.0000000	1.0000000
well-mixed	0.0000000	0.0000135	0.0000000	0.1875321	0.0000565	0.0001086
wheel	0.0000000	0.0000003	0.0000000	1.0000000	1.0000000	0.0000000
windmill	0.0000430	0.0038095	0.0005370	0.2673588	0.0015818	0.0015818

```
## # i abbreviated names: 1: median_dom_detail_gestation_time,
## #   2: mean_dom_detail_gestation_time
```

```
kruskal.test(
  formula = dom_detail_gestation_time ~ graph_type,
  data = data
)
```

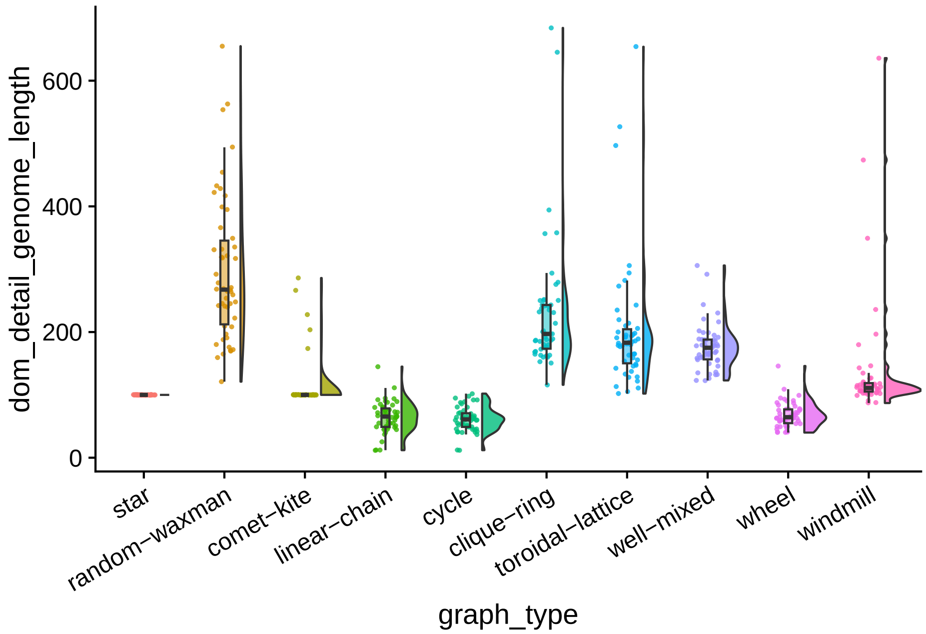
```
##
##  Kruskal-Wallis rank sum test
##
## data:  dom_detail_gestation_time by graph_type
## Kruskal-Wallis chi-squared = 197.97, df = 9, p-value < 2.2e-16
```

```
wc_results <- pairwise.wilcox.test(
  x = data$dom_detail_gestation_time,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

dom_detail_gestation_time_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(dom_detail_gestation_time_wc_table, paste0(plot_dir, "/dom_detail_gestation_
dom_detail_gestation_time_wc_table
```

6.5 Dominant genome length

```
dom_genome_length_plt <- ggplot(
    data = data,
```

```
    mapping = aes(
      x = graph_type,
      y = dom_detail_genome_length,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/dom_genome_length.pdf"),
  plot = dom_genome_length_plt,
  width = 15,
  height = 10
)

dom_genome_length_plt
```




```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_dom_detail_genome_length = median(dom_detail_genome_length),
    mean_dom_detail_genome_length = mean(dom_detail_genome_length)
  ) %>%
  arrange(
    desc(mean_dom_detail_genome_length)
  )
```

```
## # A tibble: 10 x 4
##    graph_type        reps median_dom_detail_genome_length mean_dom_detail_geno~1
##    <fct>            <int>                           <dbl>                  <dbl>
##  1 random-waxman       50                           268.                   298.
##  2 clique-ring         50                           197                    230.
##  3 toroidal-lattice    50                           183                    203.
##  4 well-mixed          50                           175                    177.
##  5 windmill            50                           111                    139.
##  6 comet-kite          50                           100                    113.
##  7 star                50                           100                    100
##  8 wheel               50                            64.5                   68.2
##  9 linear-chain        50                            65.5                   64.4
## 10 cycle               50                            61                     61.2
## # i abbreviated name: 1: mean_dom_detail_genome_length
```

	star	random-waxman	comet-kite	linear-chain	cycle	clique-ring	toroidal-l
random-waxman	0.0000000	NA	NA	NA	NA	NA	
comet-kite	0.1373564	0.0000000	NA	NA	NA	NA	
linear-chain	0.0000000	0.0000000	0	NA	NA	NA	
cycle	0.0000000	0.0000000	0	0.85562	NA	NA	
clique-ring	0.0000000	0.0007957	0	0.00000	0.0000000	NA	
toroidal-lattice	0.0000000	0.0000044	0	0.00000	0.0000000	0.1373564	
well-mixed	0.0000000	0.0000000	0	0.00000	0.0000000	0.0016198	0.
wheel	0.0000000	0.0000000	0	0.85562	0.4828684	0.0000000	0.0
windmill	0.0000000	0.0000000	0	0.00000	0.0000000	0.0000000	0.0

```
kruskal.test(
  formula = dom_detail_genome_length ~ graph_type,
  data = data
)
```

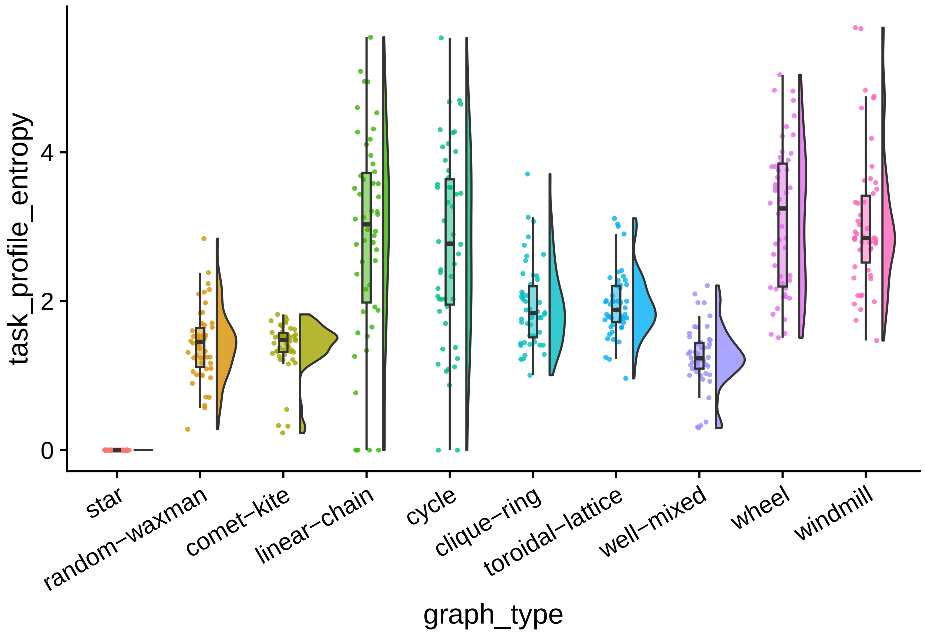
```
##
##  Kruskal-Wallis rank sum test
##
## data:  dom_detail_genome_length by graph_type
## Kruskal-Wallis chi-squared = 423.75, df = 9, p-value < 2.2e-16
```

```
wc_results <- pairwise.wilcox.test(
  x = data$dom_detail_genome_length,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

dom_detail_genome_length_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(dom_detail_genome_length_wc_table, paste0(plot_dir, "/dom_detail_genome_length_wc_tabl
dom_detail_genome_length_wc_table
```

6.6 Task profile entropy

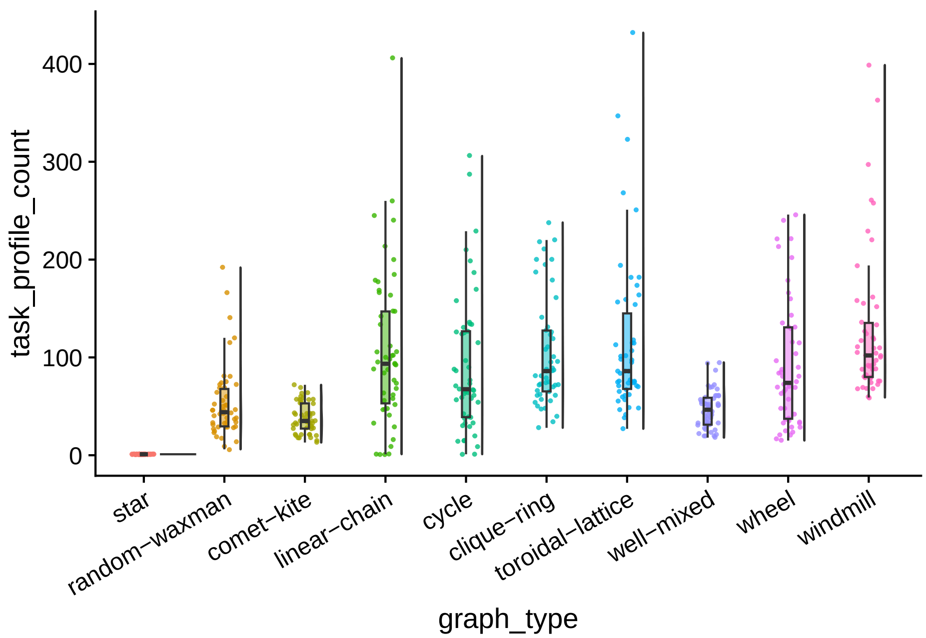
```
task_profile_entropy_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = task_profile_entropy,
```

```
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/task_profile_entropy.pdf"),
  plot = task_profile_entropy_plt,
  width = 15,
  height = 10
)

task_profile_entropy_plt
```




```
task_profile_count_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = task_profile_count,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
```

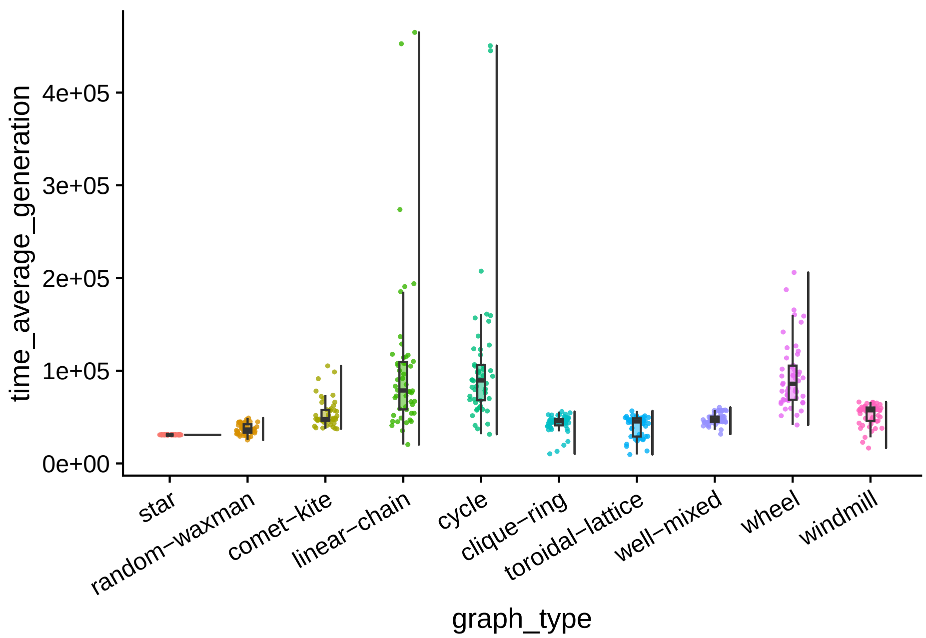
```
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/task_profile_count.pdf"),
  plot = task_profile_count_plt,
  width = 15,
  height = 10
)

task_profile_count_plt
```



6.7 Average generation

```
avg_generation_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
```

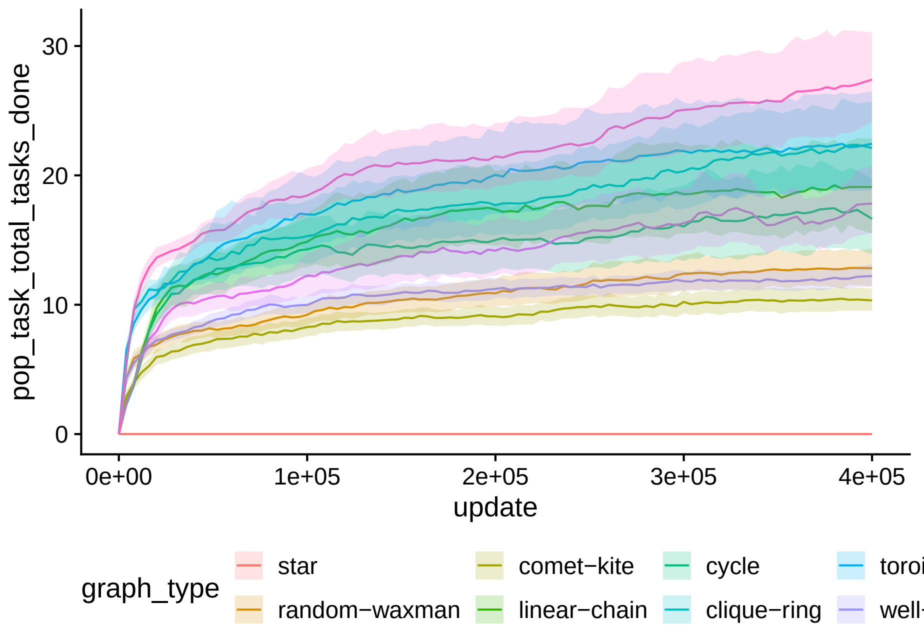
```
      y = time_average_generation,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/avg_generation.pdf"),
  plot = avg_generation_plt,
  width = 15,
  height = 10
)

avg_generation_plt
```



6.8 Population task count over time

```
pop_task_cnt_ts <- ggplot(
    data = time_series_data,
    mapping = aes(
      x = update,
      y = pop_task_total_tasks_done,
      color = graph_type,
      fill = graph_type
    )
  ) +
  stat_summary(fun = "mean", geom = "line") +
  stat_summary(
    fun.data = "mean_cl_boot",
    fun.args = list(conf.int = 0.95),
    geom = "ribbon",
    alpha = 0.2,
    linetype = 0
  ) +
  theme(legend.position = "bottom")

ggsave(
```

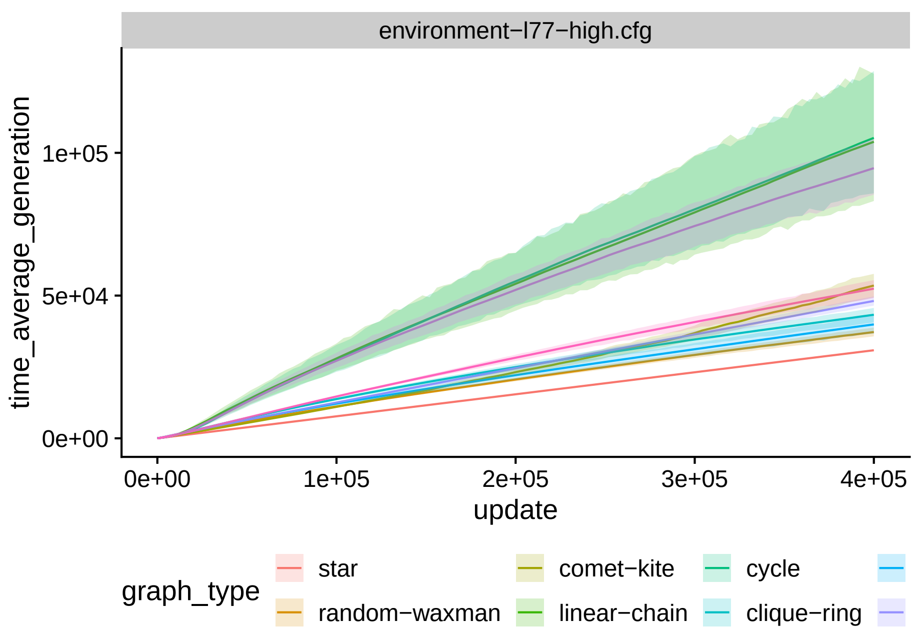
```
  plot = pop_task_cnt_ts,
  filename = paste0(
    working_directory,
    "/plots/pop_tasks_ts.pdf"
  ),
  width = 15,
  height = 10
)

pop_task_cnt_ts
```



6.9 Average generation over time

```
time_average_generation_ts <- ggplot(
    data = time_series_data,
    mapping = aes(
      x = update,
      y = time_average_generation,
      color = graph_type,
      fill = graph_type
    )
  ) +
```

```
  stat_summary(fun = "mean", geom = "line") +
  stat_summary(
    fun.data = "mean_cl_boot",
    fun.args = list(conf.int = 0.95),
    geom = "ribbon",
    alpha = 0.2,
    linetype = 0
  ) +
  facet_wrap(~ENVIRONMENT_FILE) +
  theme(legend.position = "bottom")

ggsave(
  plot = time_average_generation_ts,
  filename = paste0(
    working_directory,
    "/plots/time_average_generation_ts.pdf"
  ),
  width = 15,
  height = 10
)

time_average_generation_ts
```



6.10 Graph location info

Analyze graph_birth_info_annotated.csv

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

graph_loc_data <- graph_loc_data %>%
  mutate(
    graph_type = factor(
      graph_type,
      levels = c(
        "star",
        "random-waxman",
        "comet-kite",
        "linear-chain",
        "cycle",
        "clique-ring",
        "toroidal-lattice",
        "well-mixed",
        "wheel",
        "windmill"
      )
    ),
    seed = as.factor(seed)
  ) %>%
  filter(
    graph_type %in% focal_graphs
  )
```

Summarize by seed

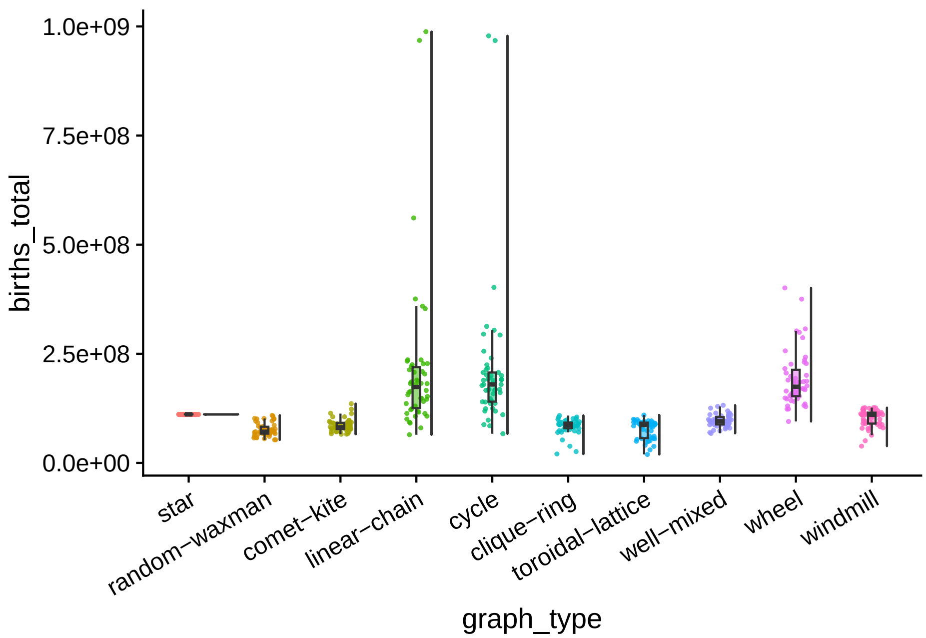
```
graph_loc_data_summary <- graph_loc_data %>%
  group_by(seed, graph_type) %>%
  summarize(
    births_var = var(births),
    births_total = sum(births),
    task_apps_total = sum(task_appearances),
    task_apps_var = var(task_appearances)
```

```
  ) %>%
  ungroup()
```

6.10.1 Total birth Counts

```
birth_counts_total_plt <- ggplot(
    data = graph_loc_data_summary,
    mapping = aes(
      x = graph_type,
      y = births_total,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/birth_counts_total.pdf"),
  plot = birth_counts_total_plt,
  width = 15,
  height = 10
)
```

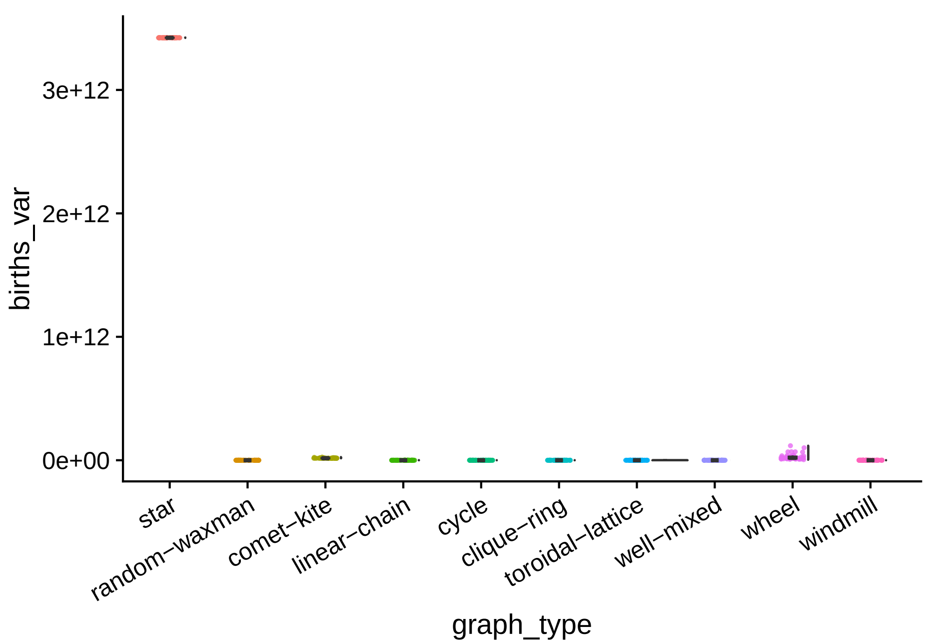
```
birth_counts_total_plt
```



6.10.2 Variance birth Counts

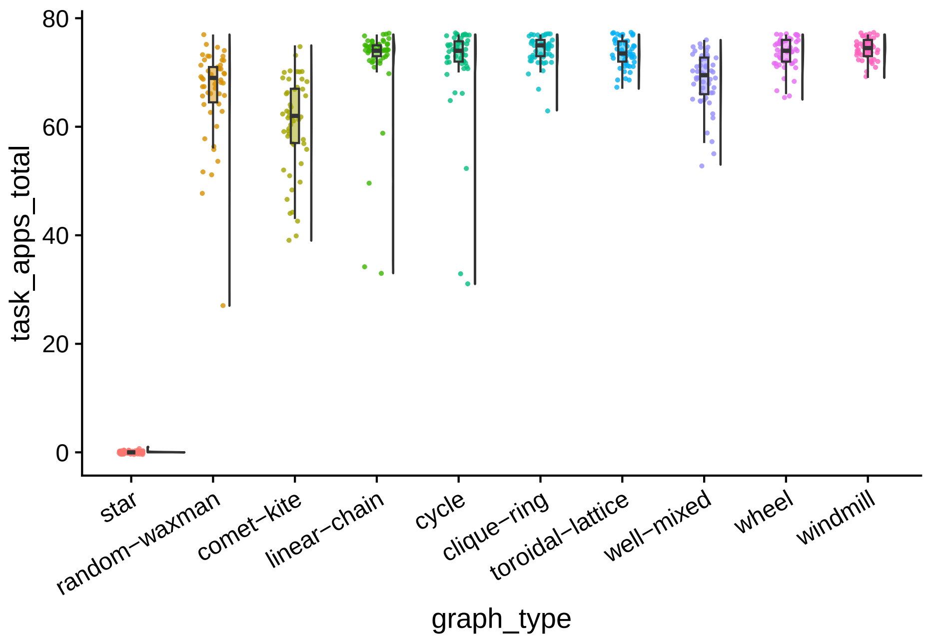
```
birth_counts_var_plt <- ggplot(
    data = graph_loc_data_summary,
    mapping = aes(
      x = graph_type,
      y = births_var,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
```

```
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/birth_counts_var.pdf"),
  plot = birth_counts_var_plt,
  width = 15,
  height = 10
)

birth_counts_var_plt
```



6.10.3 Task appearances total

```
task_apps_total_plt <- ggplot(
    data = graph_loc_data_summary,
    mapping = aes(
      x = graph_type,
      y = task_apps_total,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/task_apps_total.pdf"),
  plot = task_apps_total_plt,
  width = 15,
  height = 10
)

task_apps_total_plt
```




```r
kruskal.test(
  formula = task_apps_total ~ graph_type,
  data = graph_loc_data_summary
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  task_apps_total by graph_type
## Kruskal-Wallis chi-squared = 288.67, df = 9, p-value < 2.2e-16
```

```r
wc_results <- pairwise.wilcox.test(
  x = graph_loc_data_summary$task_apps_total,
  g = graph_loc_data_summary$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE

)
wc_results
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  graph_loc_data_summary$task_apps_total and graph_loc_data_summary$graph_type
```

```
##
##                   star    random-waxman comet-kite linear-chain cycle
## random-waxman     < 2e-16 -             -          -            -
## comet-kite        < 2e-16 0.00089       -          -            -
## linear-chain      < 2e-16 2.8e-08       2.7e-11    -            -
## cycle             < 2e-16 8.3e-07       6.3e-11    1.00000      -
## clique-ring       < 2e-16 4.4e-10       2.0e-14    1.00000      1.00000
## toroidal-lattice  < 2e-16 1.2e-08       3.7e-14    1.00000      1.00000
## well-mixed        < 2e-16 1.00000       3.5e-06    8.8e-07      2.1e-05
## wheel             < 2e-16 1.4e-08       8.1e-14    1.00000      1.00000
## windmill          < 2e-16 2.7e-11       4.3e-15    1.00000      1.00000
##                   clique-ring toroidal-lattice well-mixed wheel
## random-waxman     -           -                -          -
## comet-kite        -           -                -          -
## linear-chain      -           -                -          -
## cycle             -           -                -          -
## clique-ring       -           -                -          -
## toroidal-lattice  1.00000     -                -          -
## well-mixed        2.3e-08     7.7e-07          -          -
## wheel             1.00000     1.00000          5.0e-07    -
## windmill          1.00000     1.00000          1.2e-09    1.00000
##
## P value adjustment method: holm
```

Chapter 7

Avida - Squished lattice experiment analyses

7.1 Dependencies and setup

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

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

```
experiment_slug <- "2025-04-17-squished-lattice-longer"
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
  )
}
```

```
# Configure our default graphing theme
theme_set(theme_cowplot())
# Create a directory to store plots
plot_dir <- paste(
  working_directory,
  "plots",
  sep = "/"
)

dir.create(
  plot_dir,
  showWarnings = FALSE
)
```

```
focal_graphs <- c(
  "toroidal-lattice_60x60",
  "toroidal-lattice_15x240",
  "toroidal-lattice_2x1800",
  "cycle"
)

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

data <- data %>%
  filter(graph_type %in% focal_graphs) %>%
  mutate(
    graph_type = factor(
      graph_type,
      levels = focal_graphs
    ),
    ENVIRONMENT_FILE = as.factor(ENVIRONMENT_FILE)
  )
```

```r
time_series_path <- paste(
  working_directory,
  "data",
  "time_series.csv",
  sep = "/"
)
time_series_data <- read_csv(time_series_path)

time_series_data <- time_series_data %>%
  filter(graph_type %in% focal_graphs) %>%
  mutate(
    graph_type = factor(
      graph_type,
      levels = focal_graphs
    ),
    ENVIRONMENT_FILE = as.factor(ENVIRONMENT_FILE),
    seed = as.factor(seed)
  )
time_series_data <- time_series_data %>% filter(seed %in% data$seed)
```

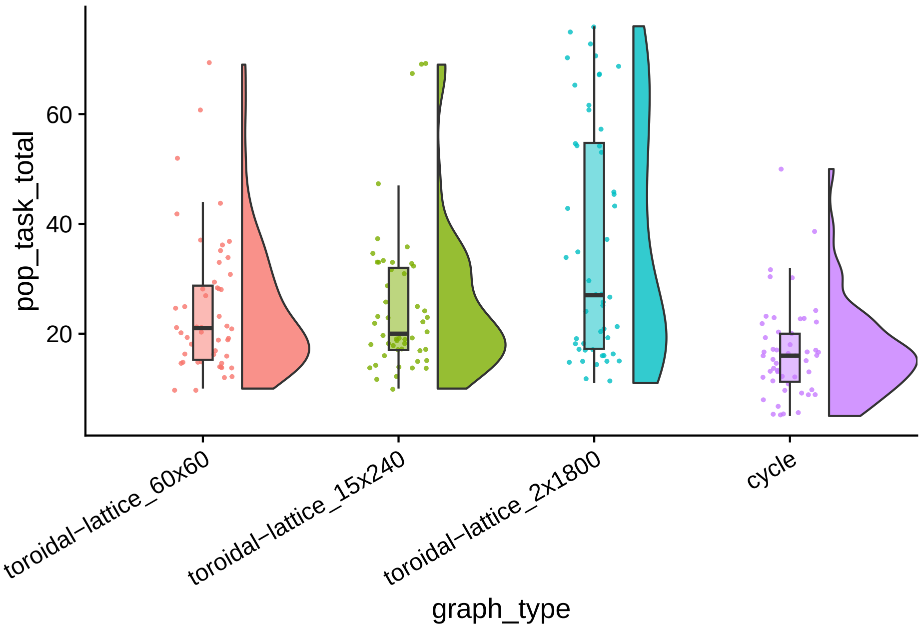
```r
# Check that all runs completed
data %>%
  filter(update == 400000) %>%
  group_by(graph_type) %>%
  summarize(
    n = n()
  )
```

```
## # A tibble: 4 x 2
##   graph_type                  n
##   <fct>                   <int>
## 1 toroidal-lattice_60x60     50
## 2 toroidal-lattice_15x240    50
## 3 toroidal-lattice_2x1800    50
## 4 cycle                      50
```

7.2 Number of tasks completed

```r
pop_tasks_total_plt <- ggplot(
  data = data,
  mapping = aes(
    x = graph_type,
```

```
    y = pop_task_total,
    fill = graph_type
  )
) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/pop_tasks_total.pdf"),
  plot = pop_tasks_total_plt,
  width = 15,
  height = 10
)

pop_tasks_total_plt
```




```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_pop_tasks = median(pop_task_total),
    mean_pop_tasks = mean(pop_task_total)
  ) %>%
  arrange(
    desc(mean_pop_tasks)
  )
```

```
## # A tibble: 4 x 4
##   graph_type               reps median_pop_tasks mean_pop_tasks
##   <fct>                   <int>            <dbl>          <dbl>
## 1 toroidal-lattice_2x1800    50               27           36.6
## 2 toroidal-lattice_15x240    50               20           25.2
## 3 toroidal-lattice_60x60     50               21           24.6
## 4 cycle                      50               16           16.7
```

```
kruskal.test(
  formula = pop_task_total ~ graph_type,
  data = data
)
```

```
##
```

	toroidal-lattice_60x60	toroidal-lattice_15x240	toroidal-lattice_2x1800
toroidal-lattice_15x240	0.9011082	NA	NA
toroidal-lattice_2x1800	0.0313929	0.0386732	NA
cycle	0.0010892	0.0002773	8e-07

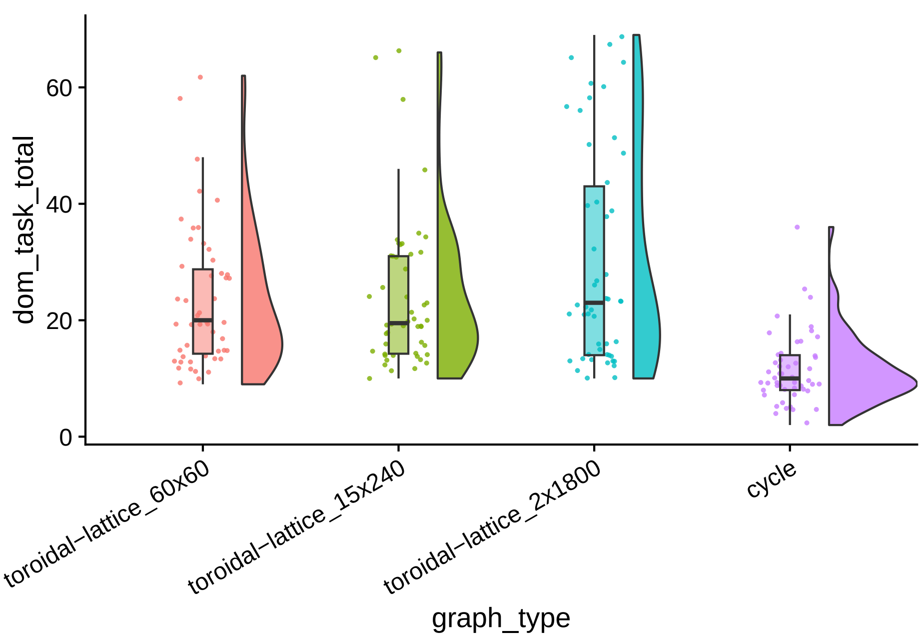
```
##  Kruskal-Wallis rank sum test
##
## data:  pop_task_total by graph_type
## Kruskal-Wallis chi-squared = 34.696, df = 3, p-value = 1.412e-07
```

```
wc_results <- pairwise.wilcox.test(
  x = data$pop_task_total,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE

)

pop_task_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(pop_task_wc_table, paste0(plot_dir, "/pop_task_wc_table.pdf"))

pop_task_wc_table
```

7.3 Dominant tasks

```
dom_tasks_total_plt <- ggplot(
  data = data,
  mapping = aes(
    x = graph_type,
    y = dom_task_total,
    fill = graph_type
  )
) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
```

```
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/dom_tasks_total.pdf"),
  plot = dom_tasks_total_plt,
  width = 15,
  height = 10
)

dom_tasks_total_plt
```



```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_dom_task_total = median(dom_task_total),
    mean_dom_task_total = mean(dom_task_total)
  ) %>%
  arrange(
    desc(mean_dom_task_total)
  )
```

```
## # A tibble: 4 x 4
##   graph_type               reps median_dom_task_total mean_dom_task_total
##   <fct>                   <int>                 <dbl>               <dbl>
## 1 toroidal-lattice_2x1800    50                  23                  30.1
## 2 toroidal-lattice_15x240    50                  19.5                24.1
## 3 toroidal-lattice_60x60     50                  20                  23.5
## 4 cycle                      50                  10                  11.5
```

```
kruskal.test(
  formula = dom_task_total ~ graph_type,
  data = data
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  dom_task_total by graph_type
## Kruskal-Wallis chi-squared = 62.705, df = 3, p-value = 1.553e-13
```

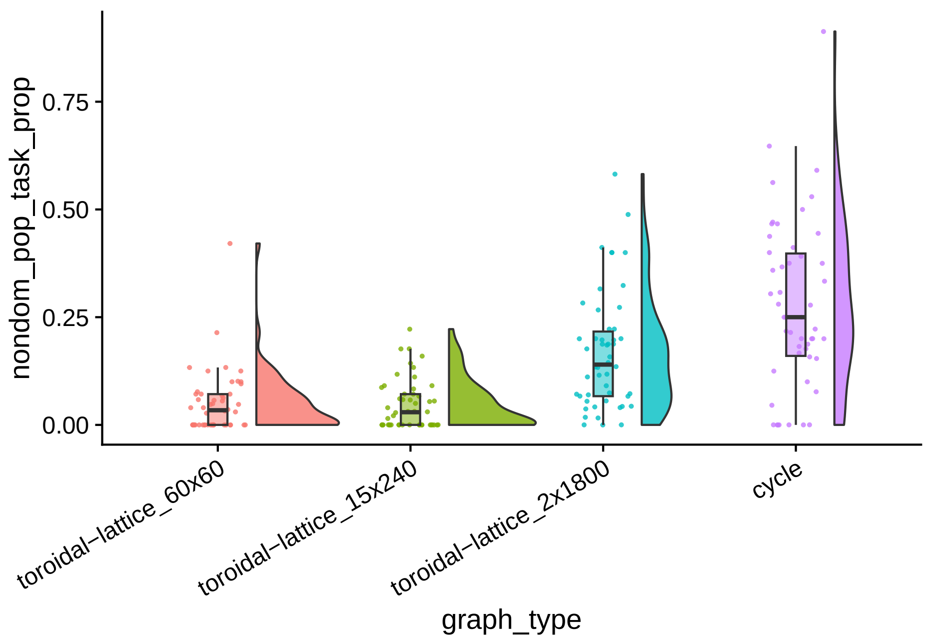
```
wc_results <- pairwise.wilcox.test(
  x = data$dom_task_total,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

dom_task_total_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(dom_task_total_wc_table, paste0(plot_dir, "/dom_task_total_wc_table.pdf"))

dom_task_total_wc_table
```

Tasks done by organisms not in dominant taxon:

	toroidal-lattice_60x60	toroidal-lattice_15x240	toroidal-lattice_2x1800
toroidal-lattice_15x240	0.8224738	NA	NA
toroidal-lattice_2x1800	0.4960864	0.5387756	NA
cycle	0.0000000	0.0000000	0

```
data <- data %>%
  mutate(
    nondom_pop_task_prop = case_when(
      pop_task_total == 0 ~ 0,
      .default = (pop_task_total - dom_task_total) / (pop_task_total)
    )
  )
nondom_tasks_total_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = nondom_pop_task_prop,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
```

```
  filename = paste0(plot_dir, "/non_dom_tasks_total.pdf"),
  plot = nondom_tasks_total_plt,
  width = 15,
  height = 10
)

nondom_tasks_total_plt
```




```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_nondom_pop_task_prop = median(nondom_pop_task_prop),
    mean_nondom_pop_task_prop = mean(nondom_pop_task_prop)
  ) %>%
  arrange(
    desc(mean_nondom_pop_task_prop)
  )
```

```
## # A tibble: 4 x 4
##   graph_type               reps median_nondom_pop_task_~1 mean_nondom_pop_task~2
##   <fct>                   <int>                     <dbl>                  <dbl>
## 1 cycle                      50                     0.25                   0.276
## 2 toroidal-lattice_2x1800    50                     0.140                  0.168
```

	toroidal-lattice__60x60	toroidal-lattice__15x240	toroidal-lattice__2x1800
toroidal-lattice__15x240	0.9884975	NA	NA
toroidal-lattice__2x1800	0.0000002	2e-07	NA
cycle	0.0000000	0e+00	0.0066588

```
## 3 toroidal-lattice_60x60     50                     0.0340                 0.0498
## 4 toroidal-lattice_15x240    50                     0.0294                 0.0467
## # i abbreviated names: 1: median_nondom_pop_task_prop,
## #   2: mean_nondom_pop_task_prop
```

```
kruskal.test(
  formula = nondom_pop_task_prop ~ graph_type,
  data = data
)
```

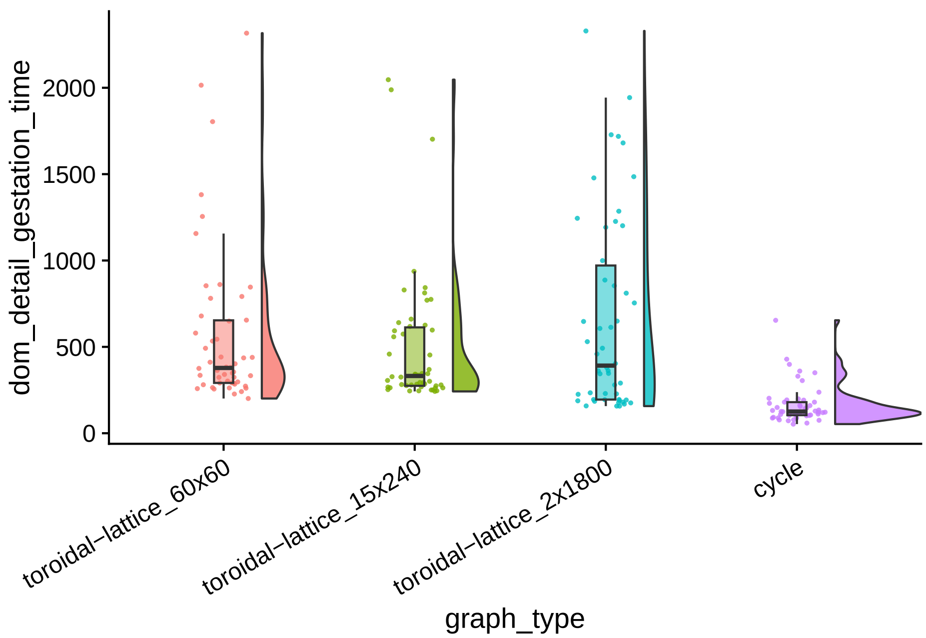
```
##
##  Kruskal-Wallis rank sum test
##
## data:  nondom_pop_task_prop by graph_type
## Kruskal-Wallis chi-squared = 72.727, df = 3, p-value = 1.112e-15
```

```
wc_results <- pairwise.wilcox.test(
  x = data$nondom_pop_task_prop,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

nondom_pop_task_prop_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(nondom_pop_task_prop_wc_table, paste0(plot_dir, "/nondom_pop_task_prop_wc_table.pdf"))

nondom_pop_task_prop_wc_table
```

7.4 Dominant gestation time

```
dom_gestation_time_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = dom_detail_gestation_time,
```

```
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/dom_gestation_time.pdf"),
  plot = dom_gestation_time_plt,
  width = 15,
  height = 10
)

dom_gestation_time_plt
```




```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_dom_detail_gestation_time = median(dom_detail_gestation_time),
    mean_dom_detail_gestation_time = mean(dom_detail_gestation_time)
  ) %>%
  arrange(
    desc(mean_dom_detail_gestation_time)
  )
```

```
## # A tibble: 4 x 4
##   graph_type               reps median_dom_detail_gesta~1 mean_dom_detail_gest~2
##   <fct>                   <int>                     <dbl>                  <dbl>
## 1 toroidal-lattice_2x1800    50                      392.                   660.
## 2 toroidal-lattice_60x60     50                      378                    569.
## 3 toroidal-lattice_15x240    50                      332.                   508.
## 4 cycle                      50                      126.                   166.
## # i abbreviated names: 1: median_dom_detail_gestation_time,
## #   2: mean_dom_detail_gestation_time
```

```
kruskal.test(
  formula = dom_detail_gestation_time ~ graph_type,
  data = data
)
```

	toroidal-lattice__60x60	toroidal-lattice__15x240	toroidal-lattice__2x1800
toroidal-lattice__15x240	0.8010941	NA	NA
toroidal-lattice__2x1800	1.0000000	1	NA
cycle	0.0000000	0	0

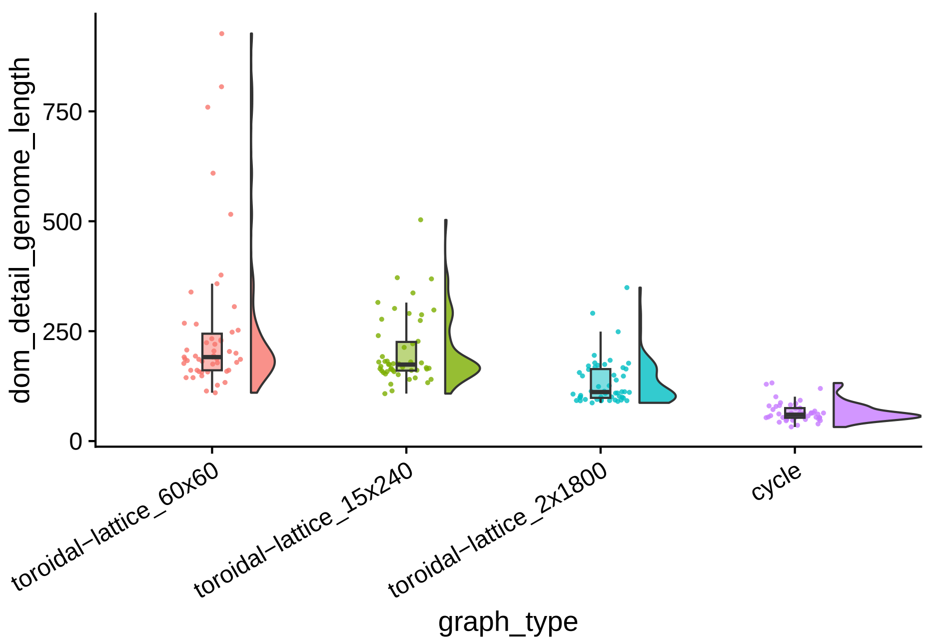
```
##
##  Kruskal-Wallis rank sum test
##
## data:  dom_detail_gestation_time by graph_type
## Kruskal-Wallis chi-squared = 77.537, df = 3, p-value < 2.2e-16
```

```
wc_results <- pairwise.wilcox.test(
  x = data$dom_detail_gestation_time,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

dom_detail_gestation_time_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(dom_detail_gestation_time_wc_table, paste0(plot_dir, "/dom_detail_gestation_

dom_detail_gestation_time_wc_table
```

7.5 Dominant genome length

```
dom_genome_length_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = dom_detail_genome_length,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
```

```
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/dom_genome_length.pdf"),
  plot = dom_genome_length_plt,
  width = 15,
  height = 10
)

dom_genome_length_plt
```



```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_dom_detail_genome_length = median(dom_detail_genome_length),
    mean_dom_detail_genome_length = mean(dom_detail_genome_length)
  ) %>%
  arrange(
    desc(mean_dom_detail_genome_length)
  )
```

```
## # A tibble: 4 x 4
##   graph_type               reps median_dom_detail_genom~1 mean_dom_detail_geno~2
##   <fct>                   <int>                     <dbl>                  <dbl>
## 1 toroidal-lattice_60x60     50                      191                   252.
## 2 toroidal-lattice_15x240    50                      174                   203.
## 3 toroidal-lattice_2x1800    50                      112.                  134.
## 4 cycle                      50                       60                    65.4
## # i abbreviated names: 1: median_dom_detail_genome_length,
## #   2: mean_dom_detail_genome_length
```

```
kruskal.test(
  formula = dom_detail_genome_length ~ graph_type,
  data = data
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  dom_detail_genome_length by graph_type
## Kruskal-Wallis chi-squared = 133.54, df = 3, p-value < 2.2e-16
```

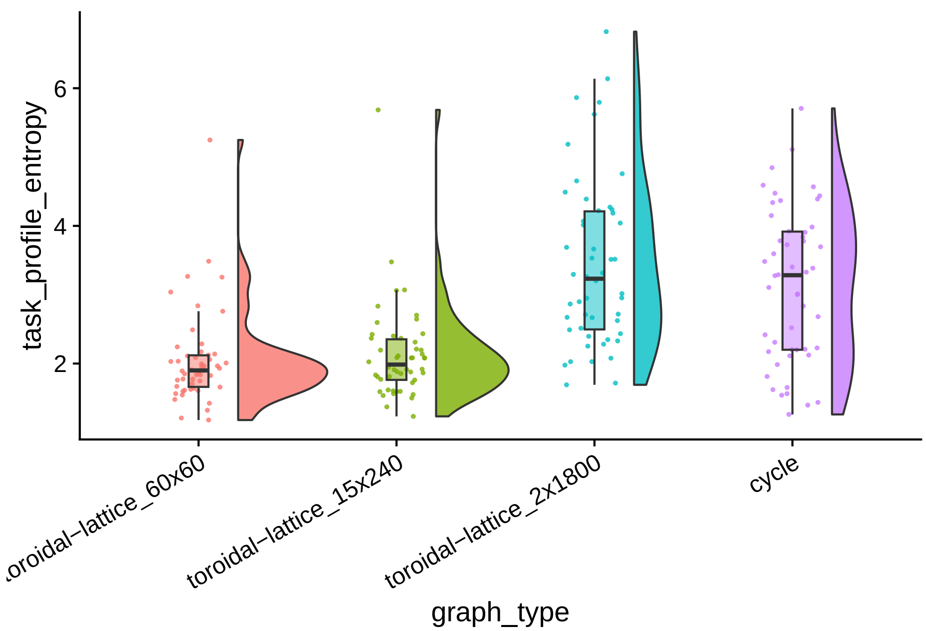
```
wc_results <- pairwise.wilcox.test(
  x = data$dom_detail_genome_length,
  g = data$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE
)

dom_detail_genome_length_wc_table <- kbl(wc_results$p.value) %>% kable_styling()
save_kable(dom_detail_genome_length_wc_table, paste0(plot_dir, "/dom_detail_genome_leng

dom_detail_genome_length_wc_table
```

	toroidal-lattice_60x60	toroidal-lattice_15x240	toroidal-lattice_2x1800
toroidal-lattice_15x240	0.0972676	NA	NA
toroidal-lattice_2x1800	0.0000000	1e-07	NA
cycle	0.0000000	0e+00	0

7.6 Task profile entropy

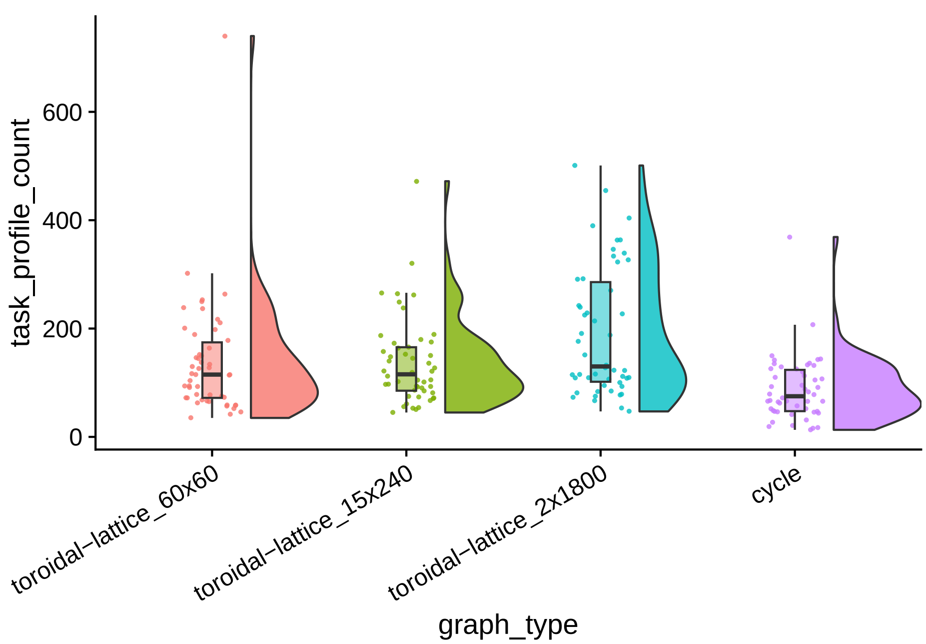
```
task_profile_entropy_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = task_profile_entropy,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/task_profile_entropy.pdf"),
  plot = task_profile_entropy_plt,
  width = 15,
  height = 10
```

```
)

task_profile_entropy_plt
```



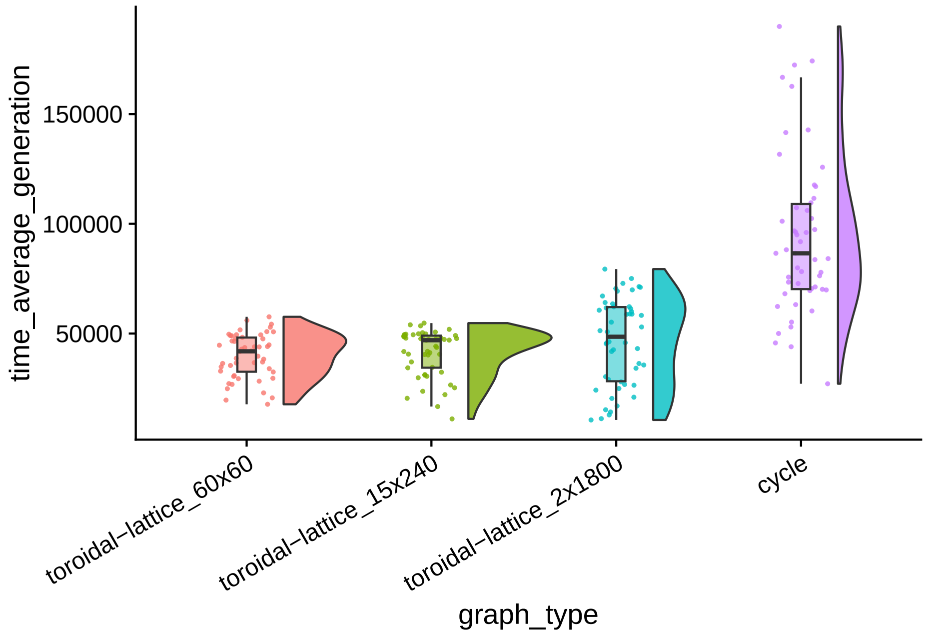

```
task_profile_count_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = task_profile_count,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
```

```
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/task_profile_count.pdf"),
  plot = task_profile_count_plt,
  width = 15,
  height = 10
)

task_profile_count_plt
```



7.7 Average generation

```
avg_generation_plt <- ggplot(
    data = data,
    mapping = aes(
      x = graph_type,
      y = time_average_generation,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/avg_generation.pdf"),
  plot = avg_generation_plt,
  width = 15,
  height = 10
)

avg_generation_plt
```

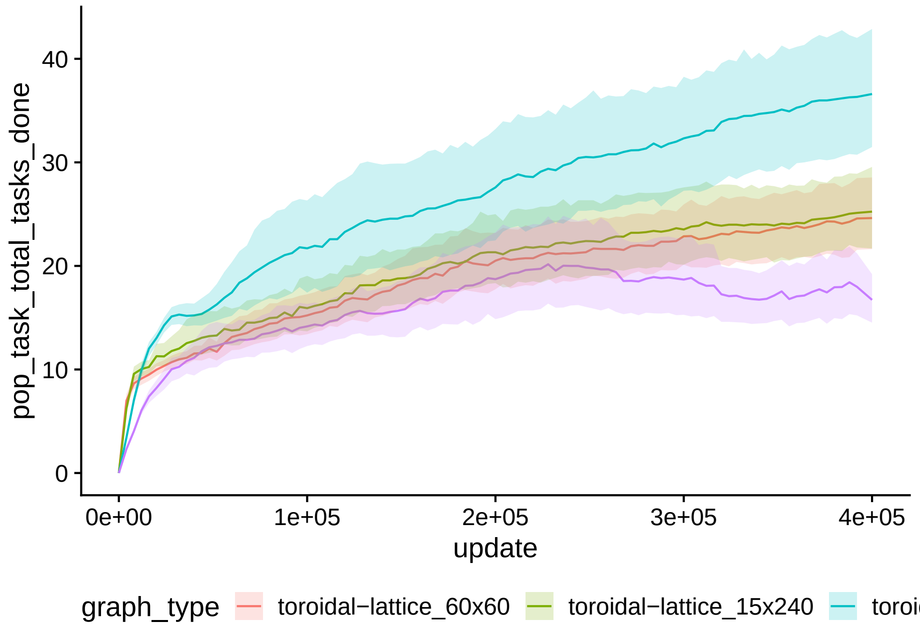



```
data %>%
  group_by(graph_type) %>%
  summarize(
    reps = n(),
    median_time_average_generation = median(time_average_generation),
    mean_time_average_generation = mean(time_average_generation)
  ) %>%
  arrange(
    desc(mean_time_average_generation)
  )
```

```
## # A tibble: 4 x 4
##   graph_type                reps median_time_average_gen~1 mean_time_average_ge~2
##   <fct>                    <int>                     <dbl>                  <dbl>
## 1 cycle                       50                    86561.                 93949.
## 2 toroidal-lattice_2x1800     50                    48531.                 46331.
## 3 toroidal-lattice_15x240     50                    46955.                 41302.
## 4 toroidal-lattice_60x60      50                    41905.                 39813.
## # i abbreviated names: 1: median_time_average_generation,
## #   2: mean_time_average_generation
```

7.8 Population task count over time

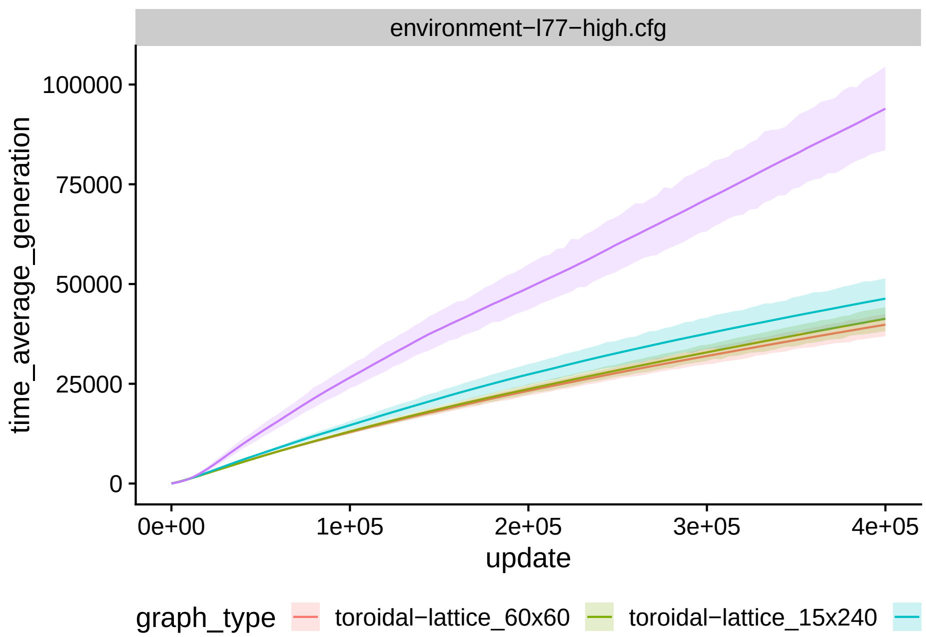
```
pop_task_cnt_ts <- ggplot(
    data = time_series_data,
    mapping = aes(
      x = update,
      y = pop_task_total_tasks_done,
      color = graph_type,
      fill = graph_type
    )
  ) +
  stat_summary(fun = "mean", geom = "line") +
  stat_summary(
    fun.data = "mean_cl_boot",
    fun.args = list(conf.int = 0.95),
    geom = "ribbon",
    alpha = 0.2,
    linetype = 0
  ) +
  theme(legend.position = "bottom")

ggsave(
  plot = pop_task_cnt_ts,
  filename = paste0(
    working_directory,
    "/plots/pop_tasks_ts.pdf"
  ),
  width = 15,
  height = 10
)

pop_task_cnt_ts
```



7.9 Average generation over time

```
time_average_generation_ts <- ggplot(
    data = time_series_data,
    mapping = aes(
      x = update,
      y = time_average_generation,
      color = graph_type,
      fill = graph_type
    )
  ) +
  stat_summary(fun = "mean", geom = "line") +
  stat_summary(
    fun.data = "mean_cl_boot",
    fun.args = list(conf.int = 0.95),
    geom = "ribbon",
    alpha = 0.2,
    linetype = 0
  ) +
  facet_wrap(~ENVIRONMENT_FILE) +
  theme(legend.position = "bottom")
```

```
ggsave(
  plot = time_average_generation_ts,
  filename = paste0(
    working_directory,
    "/plots/time_average_generation_ts.pdf"
  ),
  width = 15,
  height = 10
)

time_average_generation_ts
```



7.10 Graph location info

Analyze graph_birth_info_annotated.csv

```
# Load summary data from final update
graph_loc_data_path <- paste(
  working_directory,
  "data",
  "graph_birth_info_annotated.csv",
  sep = "/"
```

```
)
graph_loc_data <- read_csv(graph_loc_data_path)

graph_loc_data <- graph_loc_data %>%
  mutate(
    graph_type = factor(
      graph_type,
      levels = c(
        "toroidal-lattice_60x60",
        "toroidal-lattice_30x120",
        "toroidal-lattice_15x240",
        "toroidal-lattice_4x900",
        "toroidal-lattice_3x1200",
        "toroidal-lattice_2x1800",
        "cycle"
      )
    ),
    seed = as.factor(seed)
  ) %>%
  filter(
    graph_type %in% focal_graphs
  )
```

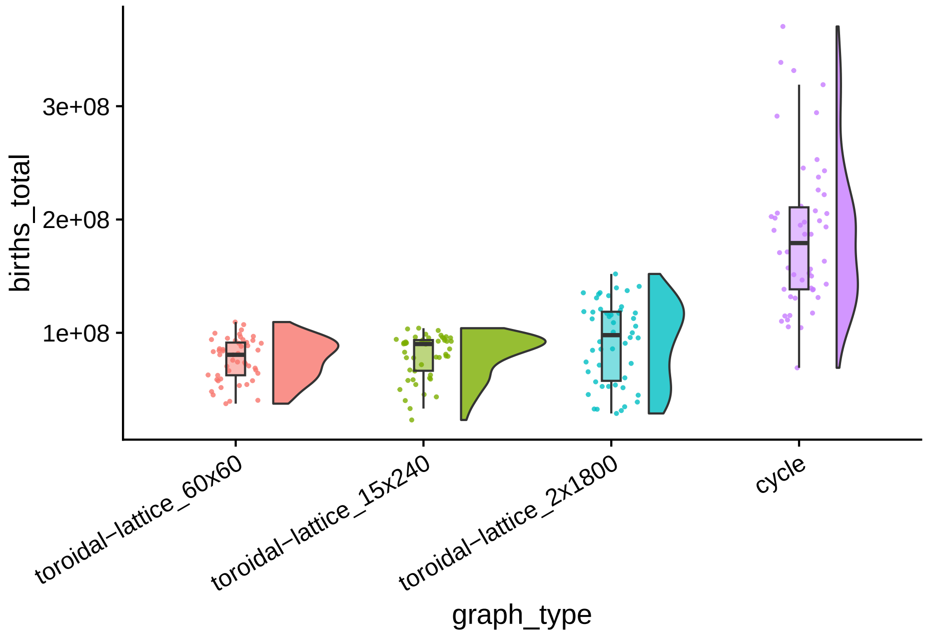
Summarize by seed

```
graph_loc_data_summary <- graph_loc_data %>%
  group_by(seed, graph_type) %>%
  summarize(
    births_var = var(births),
    births_total = sum(births),
    task_apps_total = sum(task_appearances),
    task_apps_var = var(task_appearances)
  ) %>%
  ungroup()
```

7.10.1 Total birth Counts

```
birth_counts_total_plt <- ggplot(
    data = graph_loc_data_summary,
    mapping = aes(
      x = graph_type,
      y = births_total,
      fill = graph_type
```

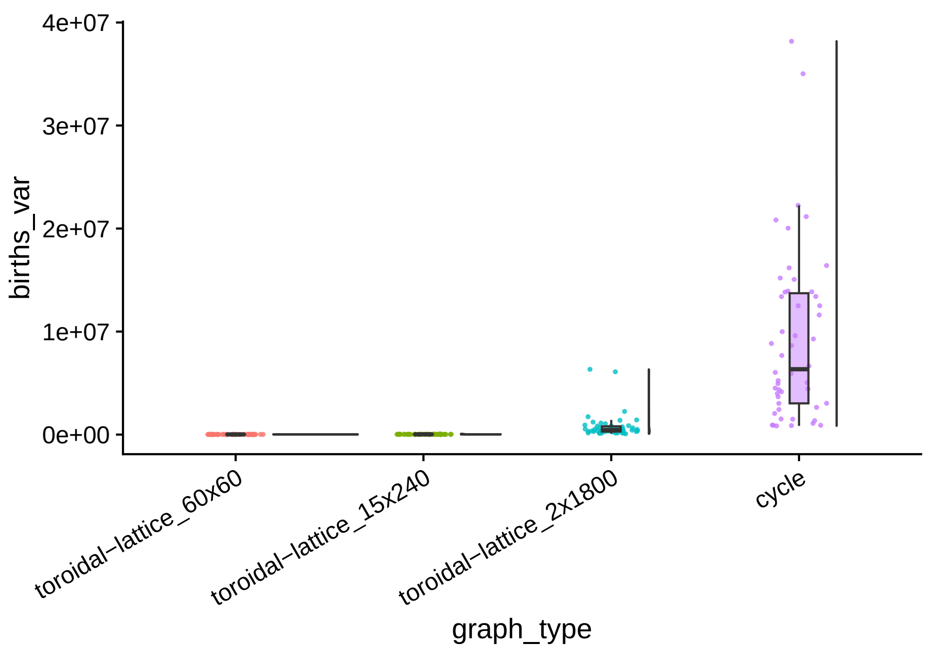
```
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/birth_counts_total.pdf"),
  plot = birth_counts_total_plt,
  width = 15,
  height = 10
)

birth_counts_total_plt
```



7.10.2 Variance birth Counts

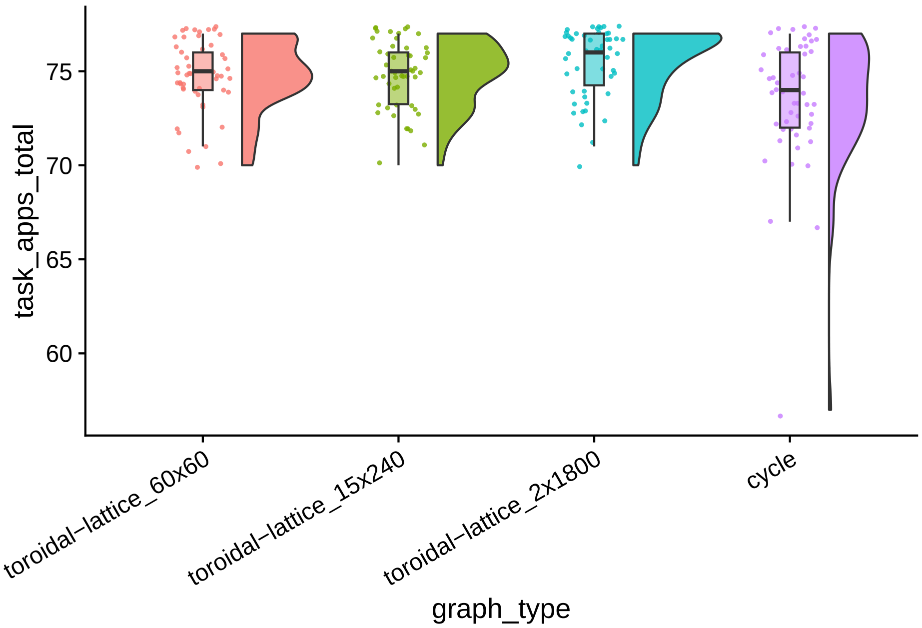
```
birth_counts_var_plt <- ggplot(
    data = graph_loc_data_summary,
    mapping = aes(
      x = graph_type,
      y = births_var,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
```

```
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/birth_counts_var.pdf"),
  plot = birth_counts_var_plt,
  width = 15,
  height = 10
)

birth_counts_var_plt
```



7.10.3 Task appearances total

```
task_apps_total_plt <- ggplot(
    data = graph_loc_data_summary,
    mapping = aes(
      x = graph_type,
      y = task_apps_total,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/task_apps_total.pdf"),
  plot = task_apps_total_plt,
  width = 15,
  height = 10
)

task_apps_total_plt
```




```
kruskal.test(
  formula = task_apps_total ~ graph_type,
  data = graph_loc_data_summary
)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  task_apps_total by graph_type
## Kruskal-Wallis chi-squared = 15.09, df = 3, p-value = 0.001742
```

```
wc_results <- pairwise.wilcox.test(
  x = graph_loc_data_summary$task_apps_total,
  g = graph_loc_data_summary$graph_type,
  p.adjust.method   = "holm",
  exact = FALSE

)
wc_results
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  graph_loc_data_summary$task_apps_total and graph_loc_data_summary$graph_type
```

```
##
##                          toroidal-lattice_60x60 toroidal-lattice_15x240
## toroidal-lattice_15x240 0.771                  -
## toroidal-lattice_2x1800 0.125                  0.125
## cycle                    0.128                  0.125
##                          toroidal-lattice_2x1800
## toroidal-lattice_15x240 -
## toroidal-lattice_2x1800 -
## cycle                    0.002
##
## P value adjustment method: holm
```

7.11 Moran's I results

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

morans_i_data <- morans_i_data %>%
  mutate(
    graph_type = str_split_i(
      graph_file,
      pattern = ".mat",
      1
    )
  ) %>%
  mutate(
    graph_type = factor(
      graph_type,
      levels = c(
        "toroidal-lattice_60x60",
        "toroidal-lattice_30x120",
        "toroidal-lattice_15x240",
        "toroidal-lattice_4x900",
        "toroidal-lattice_3x1200",
        "toroidal-lattice_2x1800",
        "cycle"
      )
```

```
    ),
    seed = as.factor(seed)
  ) %>%
  filter(
    graph_type %in% focal_graphs
  )
```

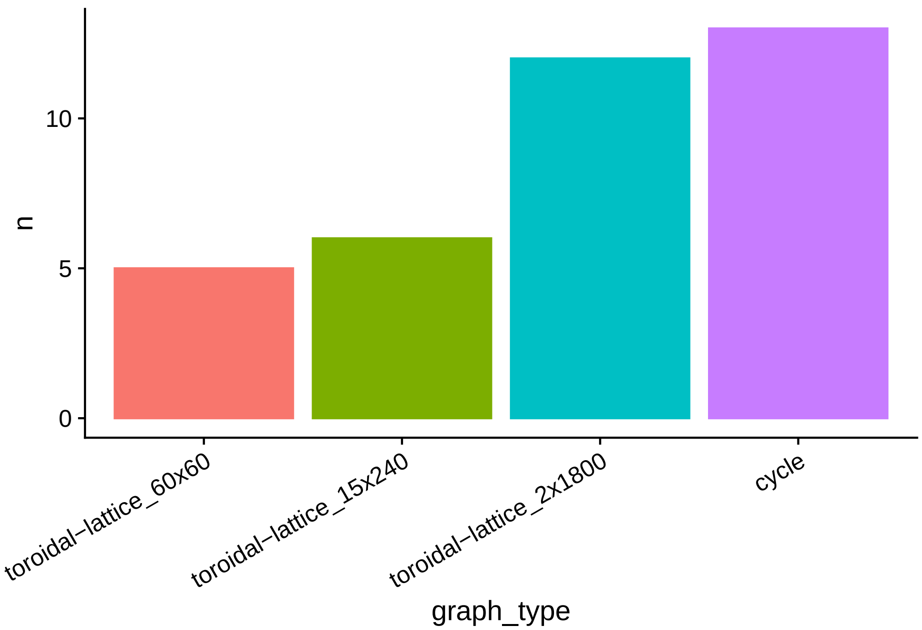
7.11.1 Clustered task appearances

Summarize statistically significant runs where I > 0.

```
# Identify number of runs where distribution of task appearances is more
# clustered than we would expect by chance.
clustered_counts <- morans_i_data %>%
  filter(
    (task_morans_i > 0) & (task_p_val <= 0.05)
  ) %>%
  group_by(graph_type) %>%
  summarize(
    n = n()
  )
```

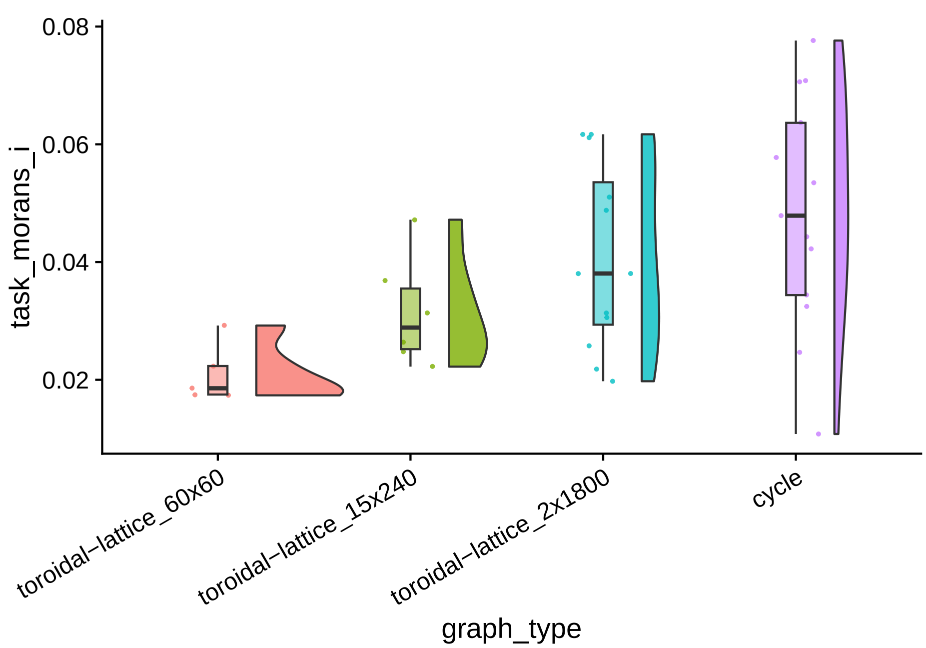
```
tasks_clustered_plt <- clustered_counts %>%
  ggplot(
    aes(
      x = graph_type,
      y = n,
      color = graph_type,
      fill = graph_type
    )
  ) +
  geom_col() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/tasks_clustered_plt.pdf"),
  plot = tasks_clustered_plt,
  width = 15,
```

```
  height = 10
)

tasks_clustered_plt
```



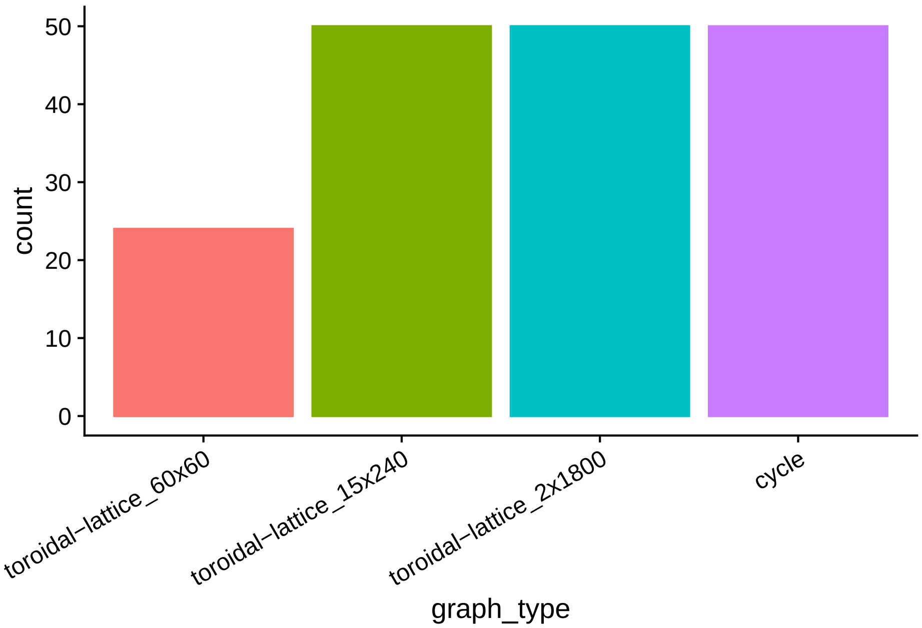

```
tasks_clustered_i_vals_plt <- morans_i_data %>%
  filter((task_morans_i > 0) & (task_p_val <= 0.05)) %>%
  ggplot(
    mapping = aes(
      x = graph_type,
      y = task_morans_i,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
```

```
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/tasks_clustered_i_vals_plt.pdf"),
  plot = tasks_clustered_i_vals_plt,
  width = 15,
  height = 10
)

tasks_clustered_i_vals_plt
```



7.11.2 Clustered birth counts

```r
births_clustered_plot <- morans_i_data %>%
  filter((birth_morans_i > 0) & (birth_p_val <= 0.05)) %>%
  ggplot(
    aes(
      x = graph_type,
      color = graph_type,
      fill = graph_type
    )
  ) +
  geom_bar() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/births_clustered_plot.pdf"),
  plot = births_clustered_plot,
  width = 15,
  height = 10
)

births_clustered_plot
```




```
birth_clustered_i_vals_plt <- morans_i_data %>%
  filter((birth_morans_i > 0) & (birth_p_val <= 0.05)) %>%
  ggplot(
    mapping = aes(
      x = graph_type,
      y = birth_morans_i,
      fill = graph_type
    )
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color = graph_type),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  theme(
```

```
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    )
  )

ggsave(
  filename = paste0(plot_dir, "/birth_clustered_i_vals_plt.pdf"),
  plot = birth_clustered_i_vals_plt,
  width = 15,
  height = 10
)

birth_clustered_i_vals_plt
```