The Evolution of Cellular Restraint in Multicellular Organisms

Katherine G. Skocelas, Austin J. Ferguson, Clifford Bohm, Katherine Perry, Rosemary Adaji, Cha

2021-03-18

Contents

Chapter 1

Introduction

This document serves as the supplemental material for our ALife 2021 conference submission "The Evolution of Cellular Restraint in Multicellular Organisms".

The document is split into sections. Each section can be accessed via the navigation bar on the left side of the screen. Sections mostly correspond to experiments (some that were discussed at length in the paper, others that were not).

Chapter 2

Baseline: Varying organism size

Here we show all the data for the baseline experiment, where we vary organism size but otherwise all parameters are set to the defaults.

For this experiment (with all default parameters), we also ran size 8x8 and 1024x1024 organisms. In the paper, however, we only included sizes from 16x16 to 512x512. Size 8x8 organisms are quick to run, but these smaller organisms see the most noise in the fitness data. Conversely, size 1024x1024 organisms take so long to run that it was impractical to run them for each experiment.

Here, we show these results for the baseline experiment, including this additional sizes. The configuration script and data for the experiment can be found under `2021_02_26__org_sizes/` in the experiments directory of the git repository.

2.1 Data cleaning

Load necessary libraries

```
library(dplyr)
library(ggplot2)
library(ggridges)
library(scales)
library(khroma)
```

Load the data and trim all the unecessay bits (*e.g.*, we initially ran sizes 8x8, 1024x1024 but cut them from the paper to make plots easier to read).

```
# Load the data
df = read.csv('../experiments/2021_02_26__org_sizes/evolution/data/scraped_evolution_data.csv')
```

```
# Trim off NAs (artifiacts of how we scraped the data) and trim to only have gen 10,00
df2 = df[!is.na(df$MCSIZE) & df$generation == 10000,]
```

We group and summarize the data to make to ensure all replicates are present.

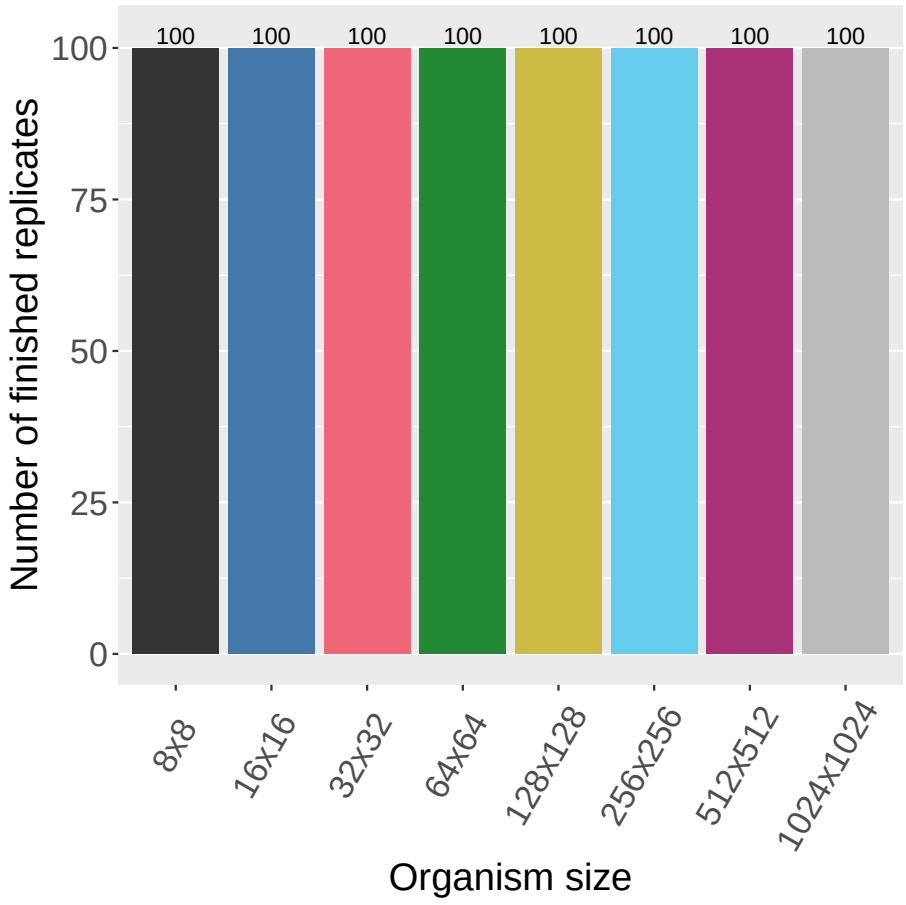
```
# Group the data by size and summarize
data_grouped = dplyr::group_by(df2, MCSIZE)
data_summary = dplyr::summarize(data_grouped, mean_ones = mean(ave_ones), n = dplyr::n
```

We clean the data and create a few helper variables to make plotting easier.

```
# Calculate restraint value (x - 60 because genome length is 100 here)
df2$restraint_value = df2$ave_ones - 60
# Make a nice, clean factor for size
df2$size_str = paste0(df2$MCSIZE, 'x', df2$MCSIZE)
df2$size_factor = factor(df2$size_str, levels = c('8x8', '16x16', '32x32', '64x64', '1
df2$size_factor_reversed = factor(df2$size_str, levels = rev(c('8x8', '16x16', '32x32'
data_summary$size_str = paste0(data_summary$MCSIZE, 'x', data_summary$MCSIZE)
data_summary$size_factor = factor(data_summary$size_str, levels = c('8x8', '16x16', '3
# Create a map of colors we'll use to plot the different organism sizes
color_vec = as.character(khroma::color('bright')(7))
color_map = c(
  '8x8' =       '#333333',
  '16x16' =     color_vec[1],
  '32x32' =     color_vec[2],
  '64x64' =     color_vec[3],
  '128x128' =   color_vec[4],
  '256x256' =   color_vec[5],
  '512x512' =   color_vec[6],
  '1024x1024' = color_vec[7]
)
# Set the sizes for text in plots
text_major_size = 18
text_minor_size = 16
```

2.2 Data integrity check

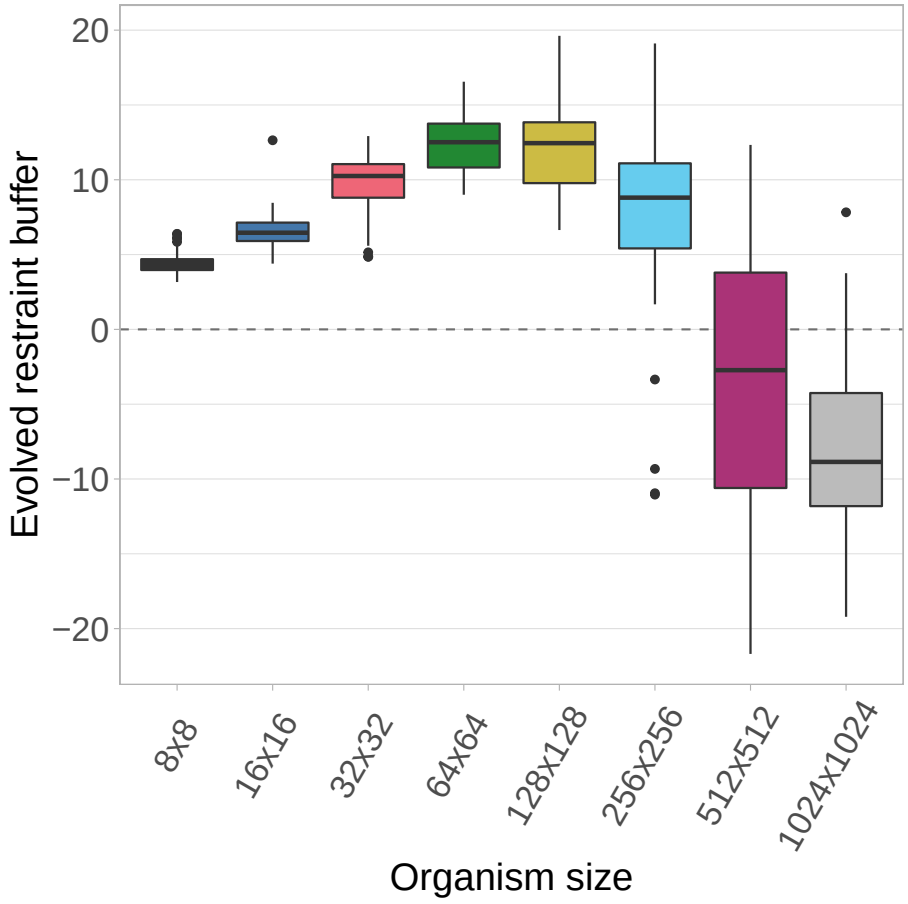
Now we plot the number of finished replicates for each treatment to make sure all data are present. Each bar/color shows a different organism size.



2.3 Aggregate plots

Here we plot all the data at once.

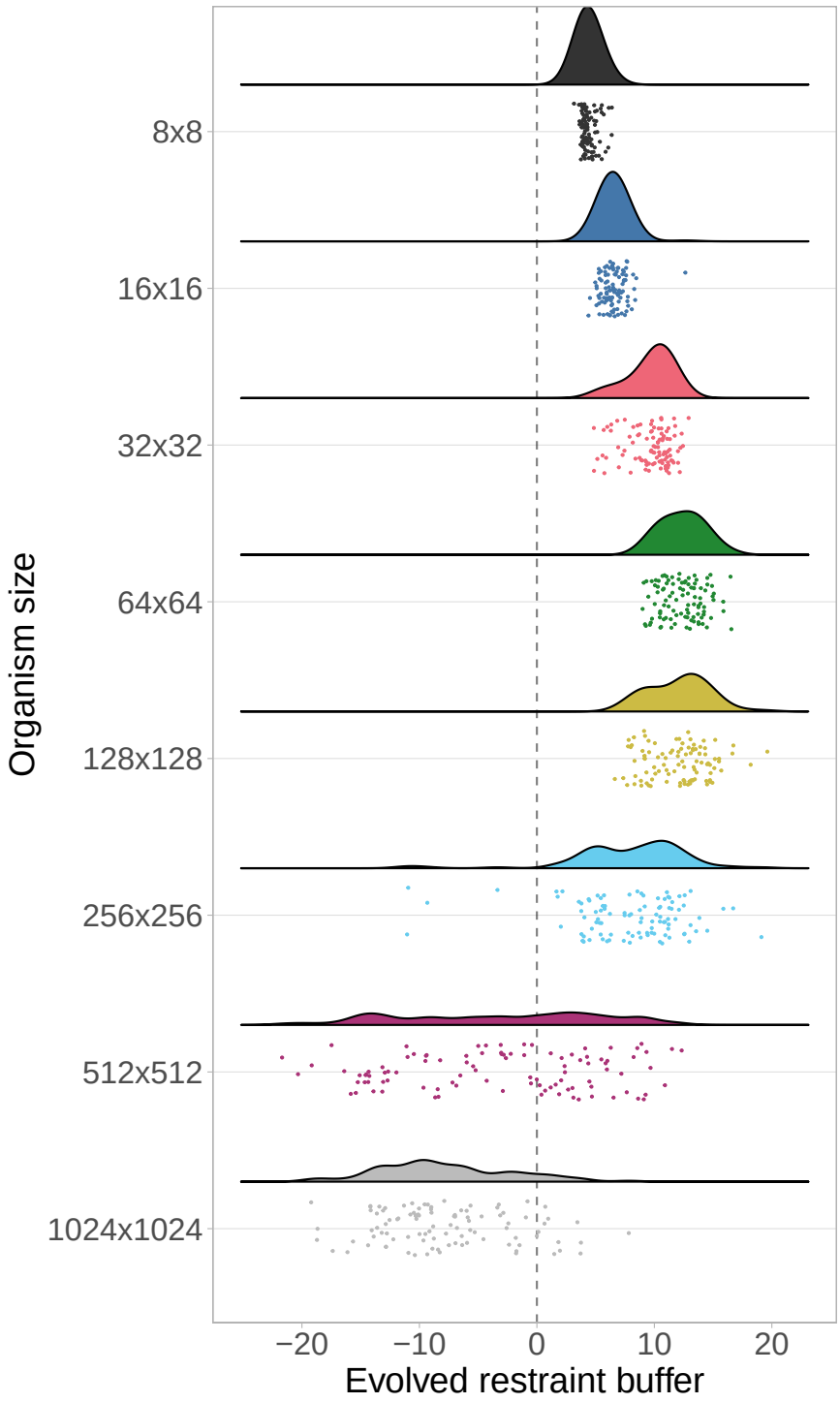
2.3.1 Boxplots



2.3.2 Raincloud plots

We can plot the same data via raincloud plots.

```
## Picking joint bandwidth of 1.16
```


Organism size
8x8
16x16
32x32
64x64
128x128
256x256
512x512
1024x1024
−20
−10
0
10
20
Evolved restraint buffer

2.4 Statistics

First, we perform a Kruskal-Wallis test across all organism sizes to indicate if variance exists. If variance exists, we then perform a pairwise Wilcoxon Rank-Sum test to show which pairs of organism sizes significantly differ. Finally, we perform Bonferroni-Holm corrections for multiple comparisons.

```
  res = kruskal.test(df2$restraint_value ~ df2$MCSIZE, df2)
  df_kruskal = data.frame(data = matrix(nrow = 0, ncol = 3))
  colnames(df_kruskal) = c('p_value', 'chi_squared', 'df')
  df_kruskal[nrow(df_kruskal) + 1,] = c(res$p.value, as.numeric(res$statistic)[1], as.n
  df_kruskal$less_0.01 = df_kruskal$p_value < 0.01
  print(df_kruskal)
```

```
##         p_value chi_squared df less_0.01
## 1 1.506351e-127    610.2553  7      TRUE
```

We see that significant variation exists, so we perform pariwise Wilcoxon tests on each to see which pais of sizes are significantly different.

```
size_vec = c(16, 32, 64, 128, 256, 512)
df_test = df2
df_wilcox = data.frame(data = matrix(nrow = 0, ncol = 5))
colnames(df_wilcox) = c('size_a', 'size_b', 'p_value_corrected', 'p_value_raw', 'W')
for(size_idx_a in 1:(length(size_vec) - 1)){
  size_a = size_vec[size_idx_a]
  for(size_idx_b in (size_idx_a + 1):length(size_vec)){
    size_b = size_vec[size_idx_b]
    res = wilcox.test(df_test[df_test$MCSIZE == size_a,]$restraint_value, df_test[df_te
    df_wilcox[nrow(df_wilcox) + 1,] = c(size_a, size_b, 0, res$p.value, as.numeric(res$
  }
}
df_wilcox$p_value_corrected = p.adjust(df_wilcox$p_value_raw, method = 'holm')
df_wilcox$less_0.01 = df_wilcox$p_value_corrected < 0.01
print(df_wilcox)
```

```
##    size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1      16     32      4.406735e-21 4.406735e-22 1045.5      TRUE
## 2      16     64      1.790650e-32 1.193767e-33   51.5      TRUE
## 3      16    128      2.585339e-31 1.988723e-32  147.0      TRUE
## 4      16    256      1.864978e-03 6.216595e-04 3599.0      TRUE
## 5      16    512      3.596138e-17 4.495172e-18 8547.0      TRUE
## 6      32     64      2.103060e-15 3.004372e-16 1654.5      TRUE
## 7      32    128      1.857809e-09 4.644523e-10 2449.5      TRUE
## 8      32    256      8.472946e-03 4.236473e-03 6171.0      TRUE
## 9      32    512      1.338207e-26 1.216552e-27 9459.5      TRUE
## 10     64    128      4.429461e-01 4.429461e-01 5314.5     FALSE
## 11     64    256      2.515682e-15 4.192803e-16 8329.0      TRUE
```

```
## 12     64    512      1.552625e-31 1.109018e-32 9873.0      TRUE
## 13    128    256      4.763656e-12 9.527311e-13 7921.5      TRUE
## 14    128    512      3.610598e-30 3.008832e-31 9759.0      TRUE
## 15    256    512      7.155324e-19 7.950361e-20 8730.5      TRUE
```

Chapter 3

Somatic Mutation Rate Sweep

This experiment was one of the prelimary experiments we conducted to find the default parameters for Primordium. Here, we vary the somatic mutation rate, the probability that a cell replication will result in the offspring cell having a different restraint value from its parent.

We settled on a somatic mutation rate of 0.5 (*i.e.*, each cell replication has a 50% chance of mutation).

The configuration script and data for the experiment can be found under `2021_02_27__soma_mut_fin/` in the experiments directory of the git repository.

3.1 Data cleaning

Load necessary libraries

```
library(dplyr)
library(ggplot2)
library(ggridges)
library(scales)
library(khroma)
```

Load the data and trim all the unecessay bits (*e.g.*, we initially ran sizes 8x8, 1024x1024 but cut them from the paper to make plots easier to read).

```
# Load the data
df = read.csv('../experiments/2021_02_27__soma_mut_fin/evolution/data/scraped_evolution_data.csv'
# Trim off NAs (artifiacts of how we scraped the data) and trim to only have gen 10,000
```

```
df2 = df[!is.na(df$MCSIZE) & df$generation == 10000,]
# Ignore data for size 8x8 and 1024x1024
df2 = df2[df2$MCSIZE != 8 & df2$MCSIZE != 1024,]
```

We group and summarize the data to make to ensure all replicates are present.

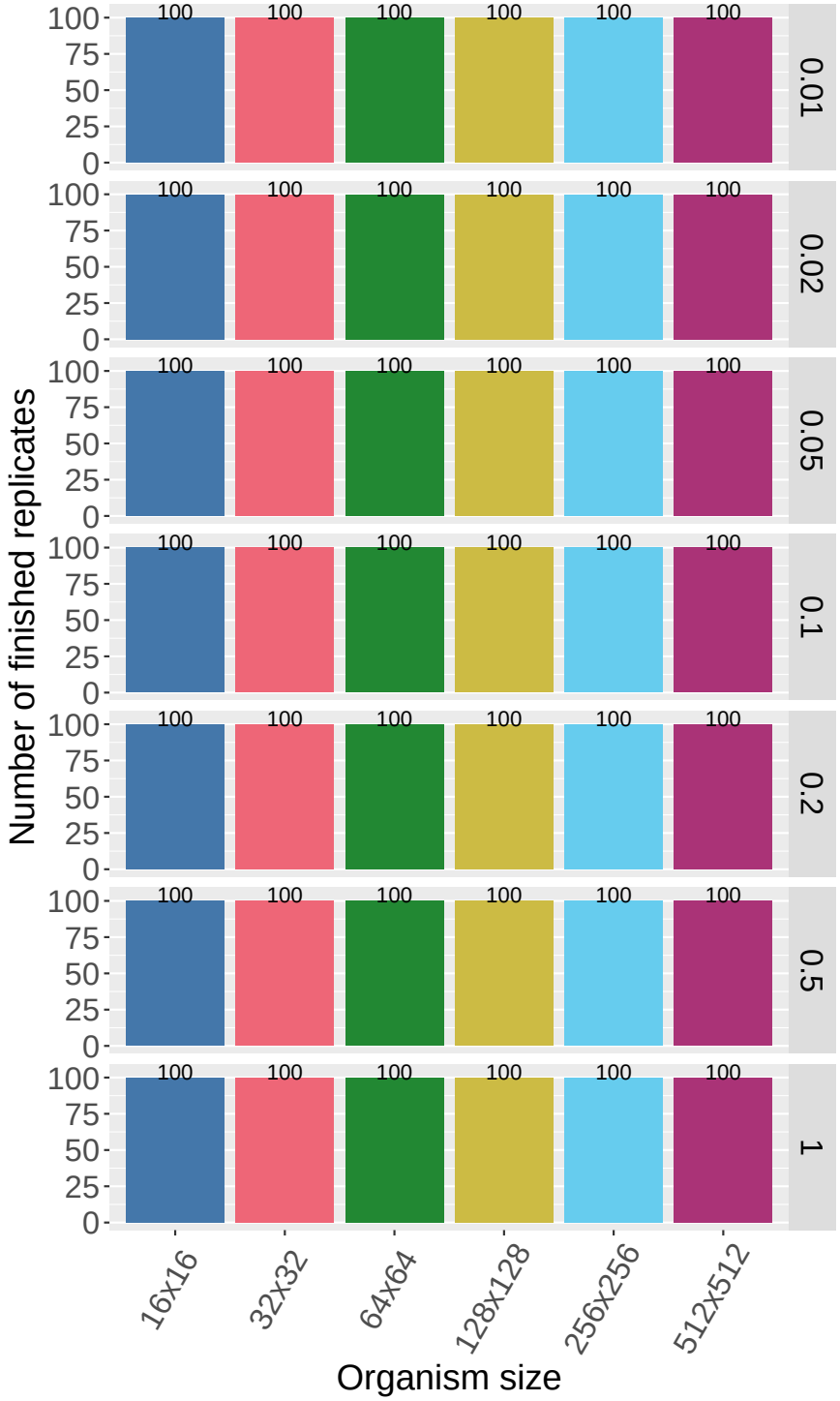
```
# Group the data by size and summarize
data_grouped = dplyr::group_by(df2, MCSIZE, CELLMUT)
data_summary = dplyr::summarize(data_grouped, mean_ones = mean(ave_ones), n = dplyr::n
```

We clean the data and create a few helper variables to make plotting easier.

```
# Calculate restraint value (x - 60 because genome length is 100 here)
df2$restraint_value = df2$ave_ones - 60
# Make a nice, clean factor for size
df2$size_str = paste0(df2$MCSIZE, 'x', df2$MCSIZE)
df2$size_factor = factor(df2$size_str, levels = c('16x16', '32x32', '64x64', '128x128'
df2$size_factor_reversed = factor(df2$size_str, levels = rev(c('16x16', '32x32', '64x6
df2$soma_mut_str = paste('soma CELLMUT', df2$CELLMUT)
df2$mut_factor = factor(df2$CELLMUT, levels = c(0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.0
data_summary$size_str = paste0(data_summary$MCSIZE, 'x', data_summary$MCSIZE)
data_summary$size_factor = factor(data_summary$size_str, levels = c('16x16', '32x32',
data_summary$soma_mut_str = paste('soma CELLMUT', data_summary$CELLMUT)
data_summary$mut_factor = factor(data_summary$CELLMUT, levels = c(0.01, 0.02, 0.05, 0.1
# Create a map of colors we'll use to plot the different organism sizes
color_vec = as.character(khroma::color('bright')(7))
color_map = c(
  '16x16' =     color_vec[1],
  '32x32' =     color_vec[2],
  '64x64' =     color_vec[3],
  '128x128' =   color_vec[4],
  '256x256' =   color_vec[5],
  '512x512' =   color_vec[6],
  '1024x1024' = color_vec[7]
)
# Set the sizes for text in plots
text_major_size = 18
text_minor_size = 16
```

3.2 Data integrity check

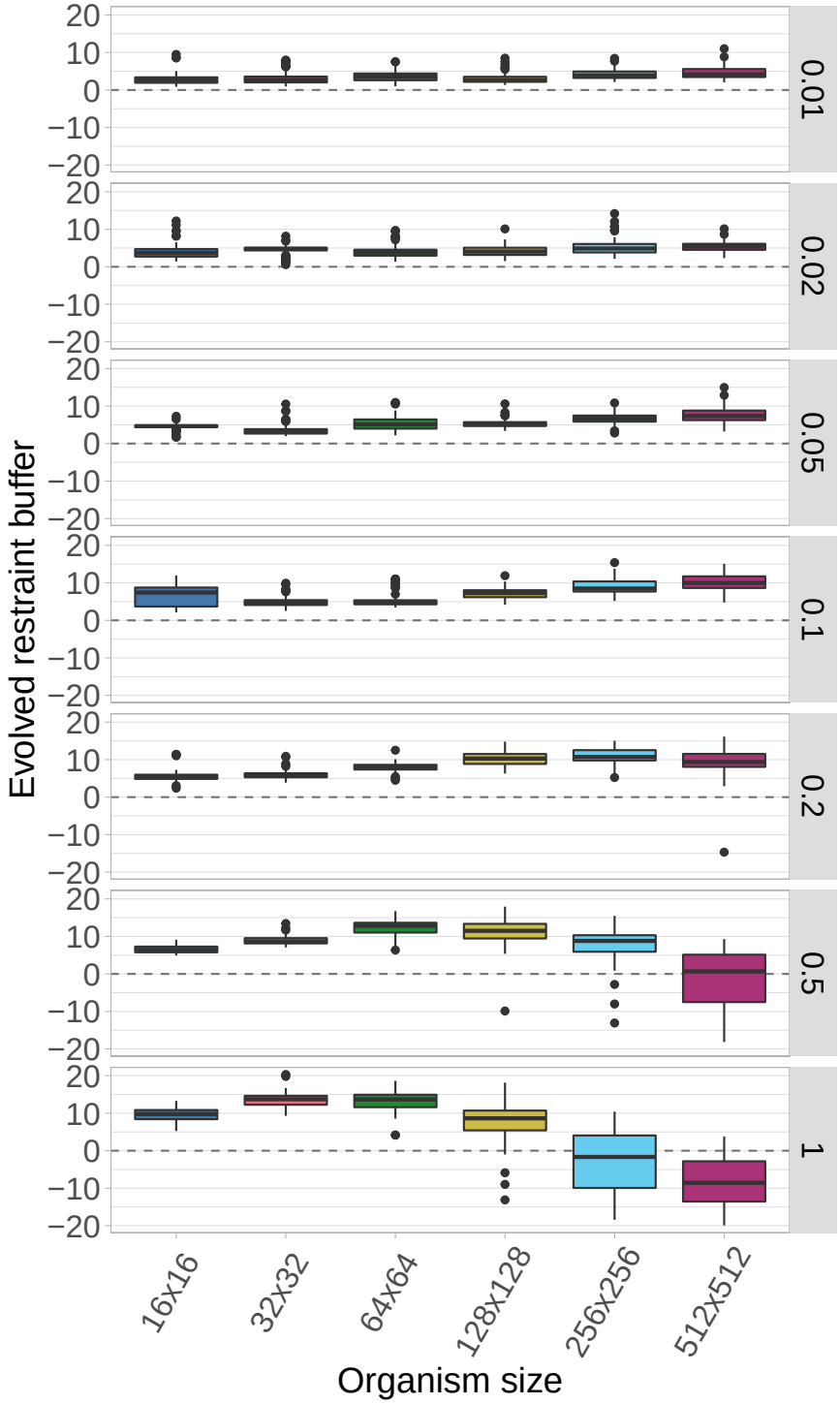
Now we plot the number of finished replicates for each treatment to make sure all data are present. Each row shows a different somatic mutation rate. Each bar/color shows a different organism size.


Number of finished replicates
Organism size
16x16
32x32
64x64
128x128
256x256
512x512
0.01
0.02
0.05
0.1
0.2
0.5
1
100
75
50
25
0

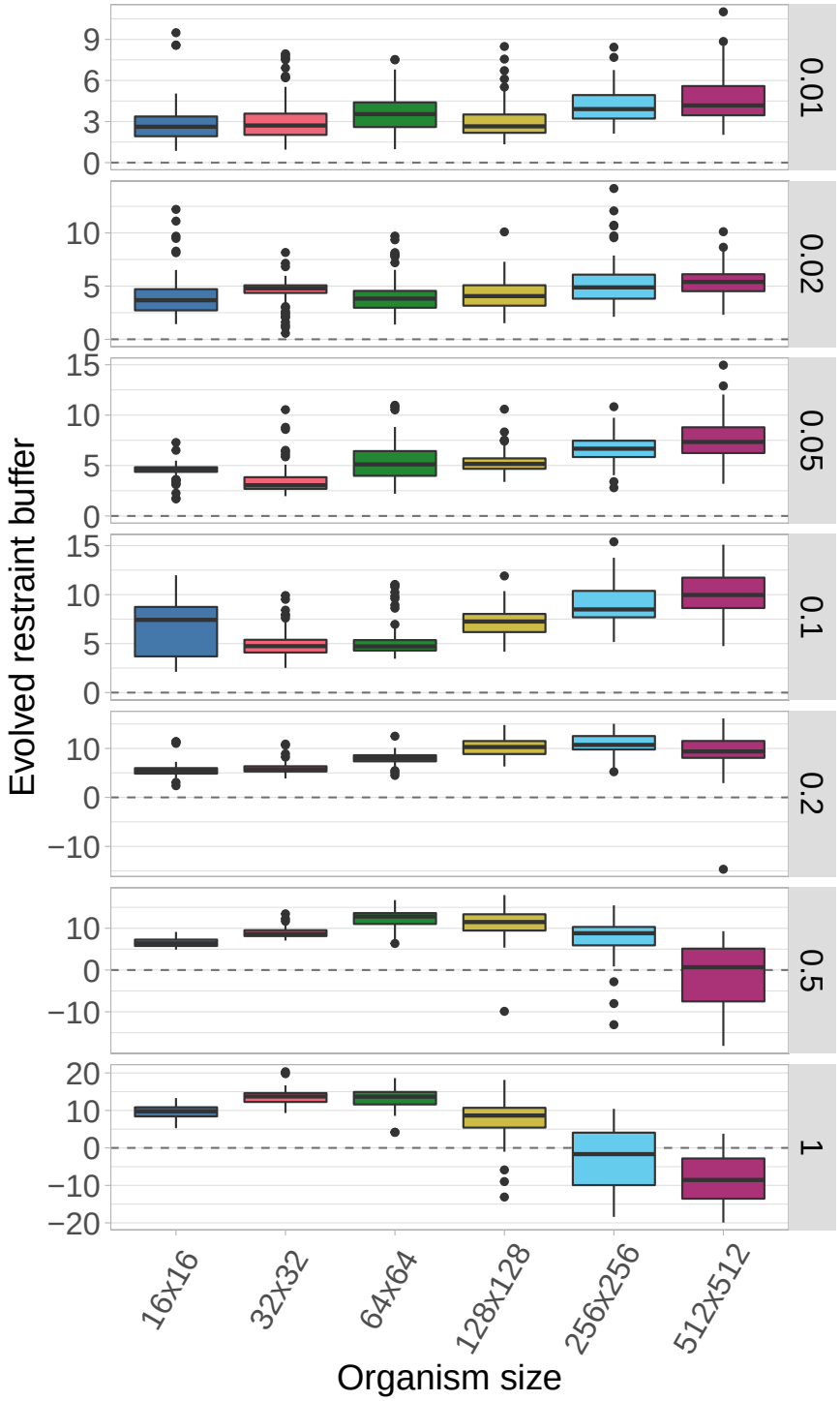
3.3 Aggregate plots

3.3.1 Facet by somatic mutation rate

Here we plot all the data at once. Each row showing a different somatic mutation rate and each boxplot shows a given organism size.

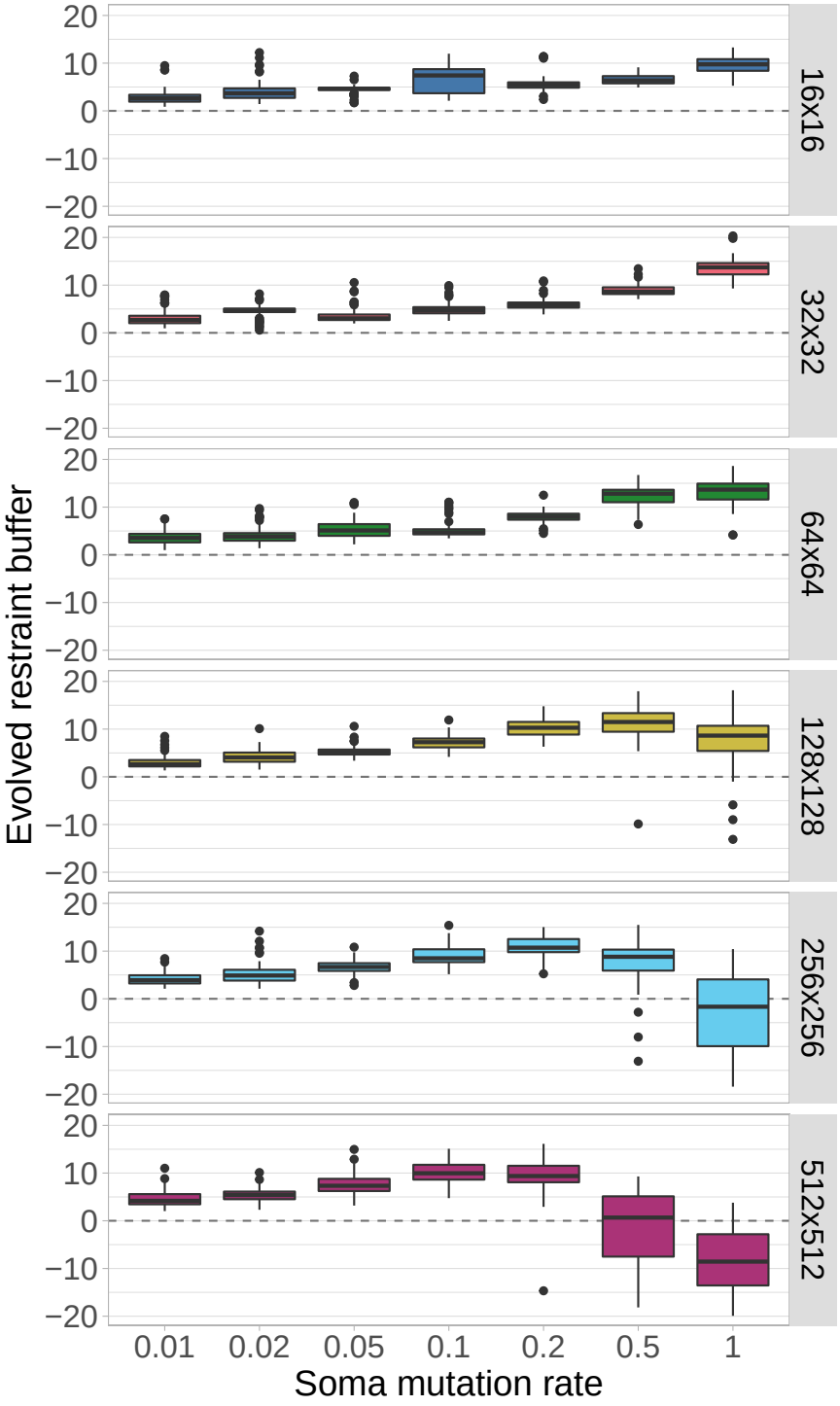

Evolved restraint buffer
20
10
0
−10
−20
0.01
0.02
0.05
0.1
0.2
0.5
1
16x16
32x32
64x64
128x128
256x256
512x512
Organism size

Here we plot the same data, only we allow the y-axis to vary between rows.

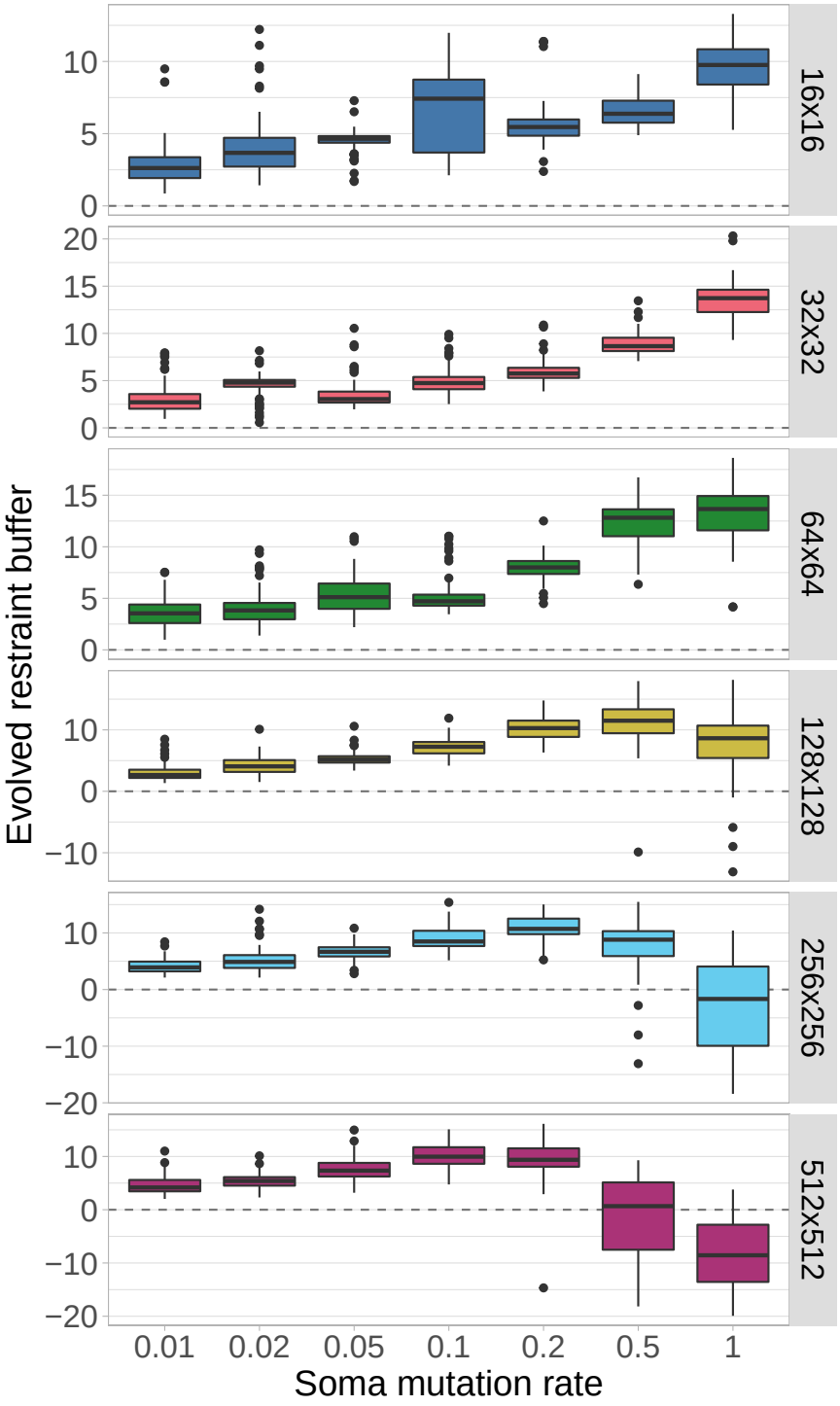

Evolved restraint buffer
0.01
0.02
0.05
0.1
0.2
0.5
1
16x16
32x32
64x64
128x128
256x256
512x512
Organism size

3.3.2 Facet by organism size

Next, we plot the same data, but this time each row corresponds to a certain organism size while somatic mutation rate changes along the x-axis.


Evolved restraint buffer
20
10
0
−10
−20
16x16
32x32
64x64
128x128
256x256
512x512
0.01
0.02
0.05
0.1
0.2
0.5
1
Soma mutation rate

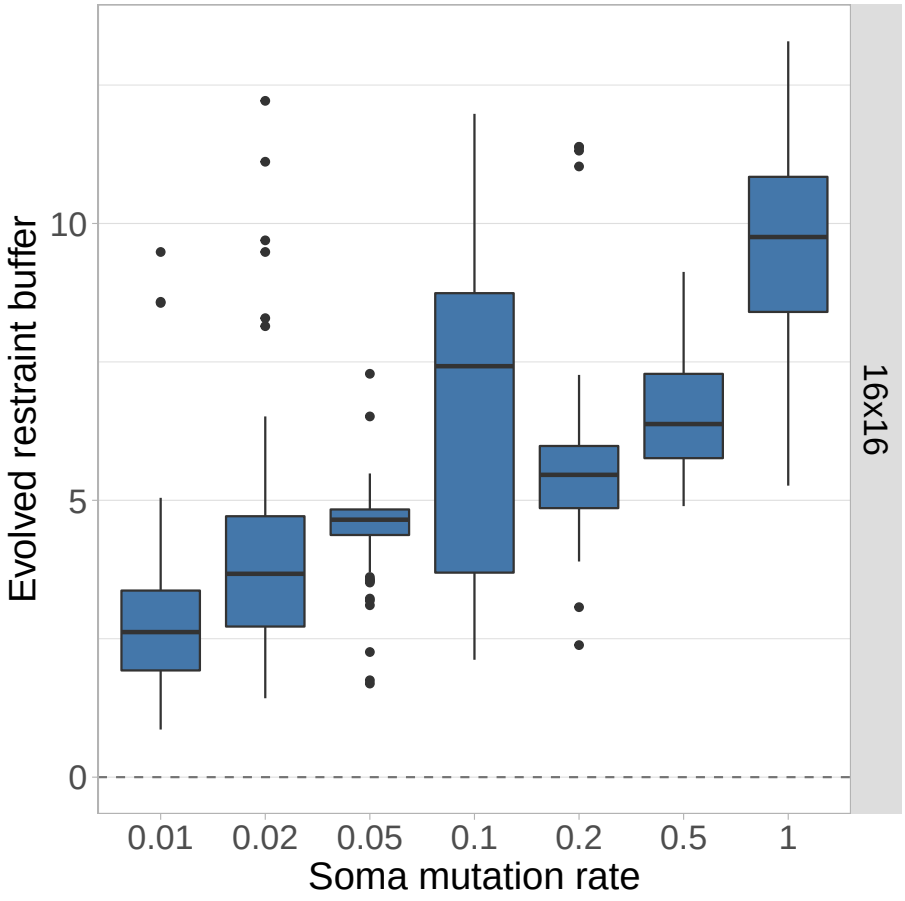
Again, we replot the same data but allow the y-axis to vary between rows.


Evolved restraint buffer
16x16
32x32
64x64
128x128
256x256
512x512
0.01
0.02
0.05
0.1
0.2
0.5
1
Soma mutation rate

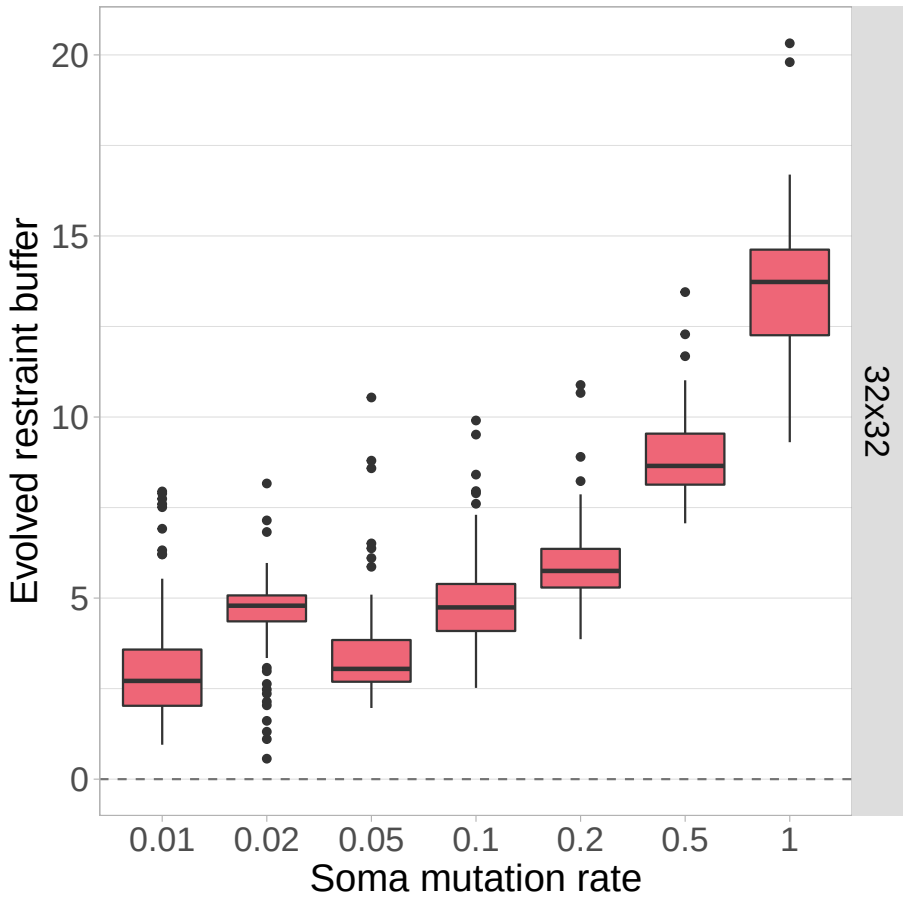
3.4 Single organism size plots

Here we plot each organism size independently, with the somatic mutation rate on the x-axis.

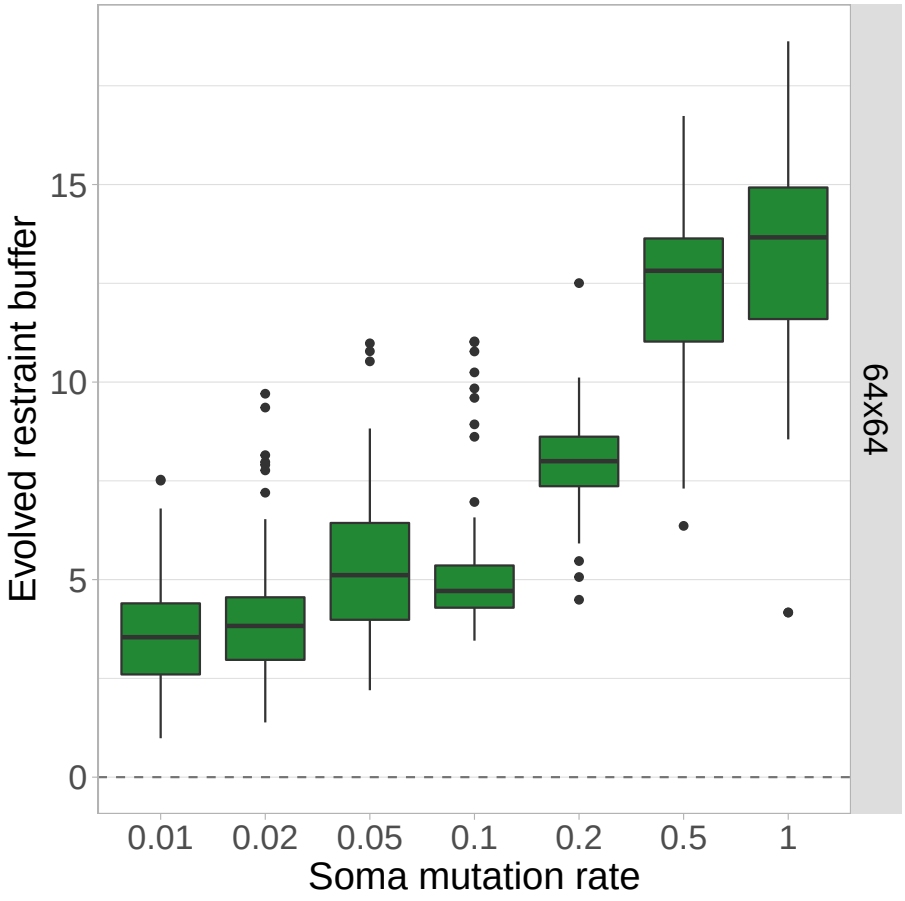
3.4.1 Organism size 16x16



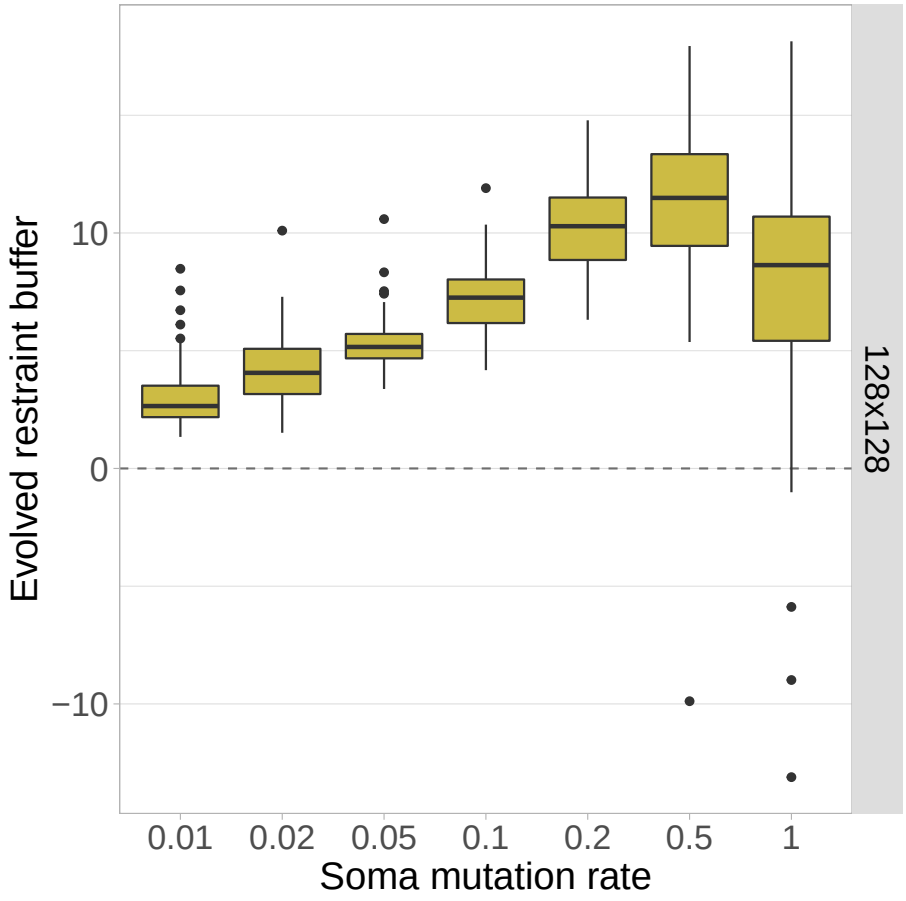
3.4.2 Organism size 32x32



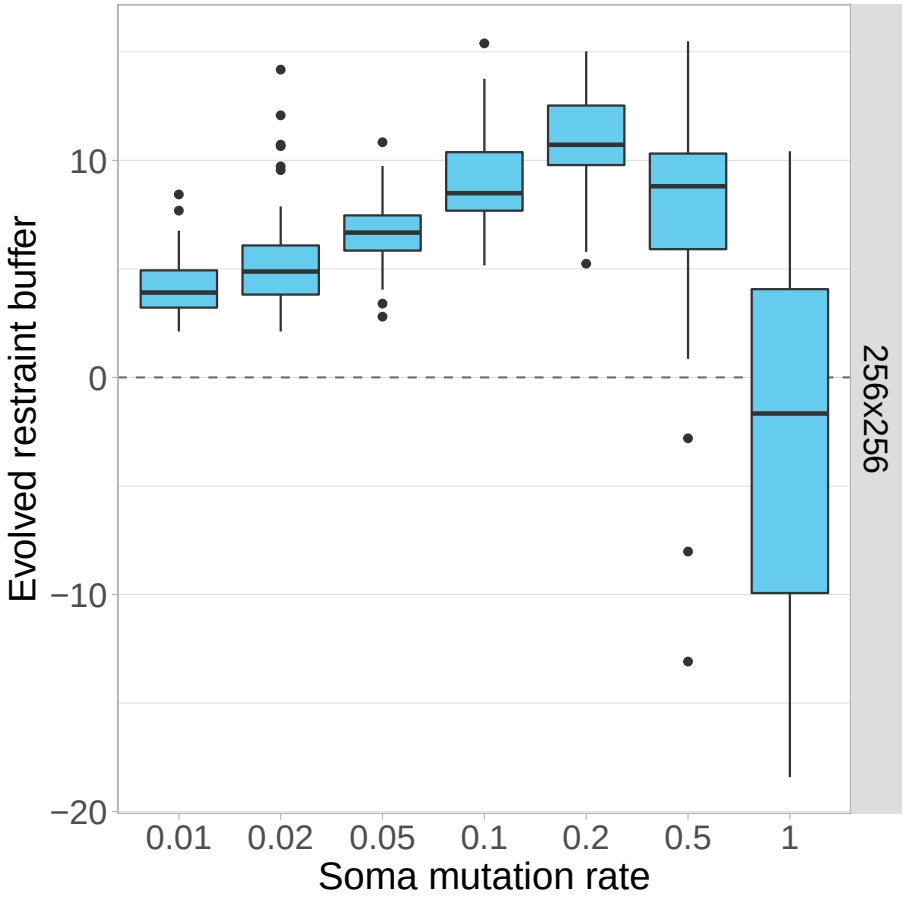
3.4.3 Organism size 64x64



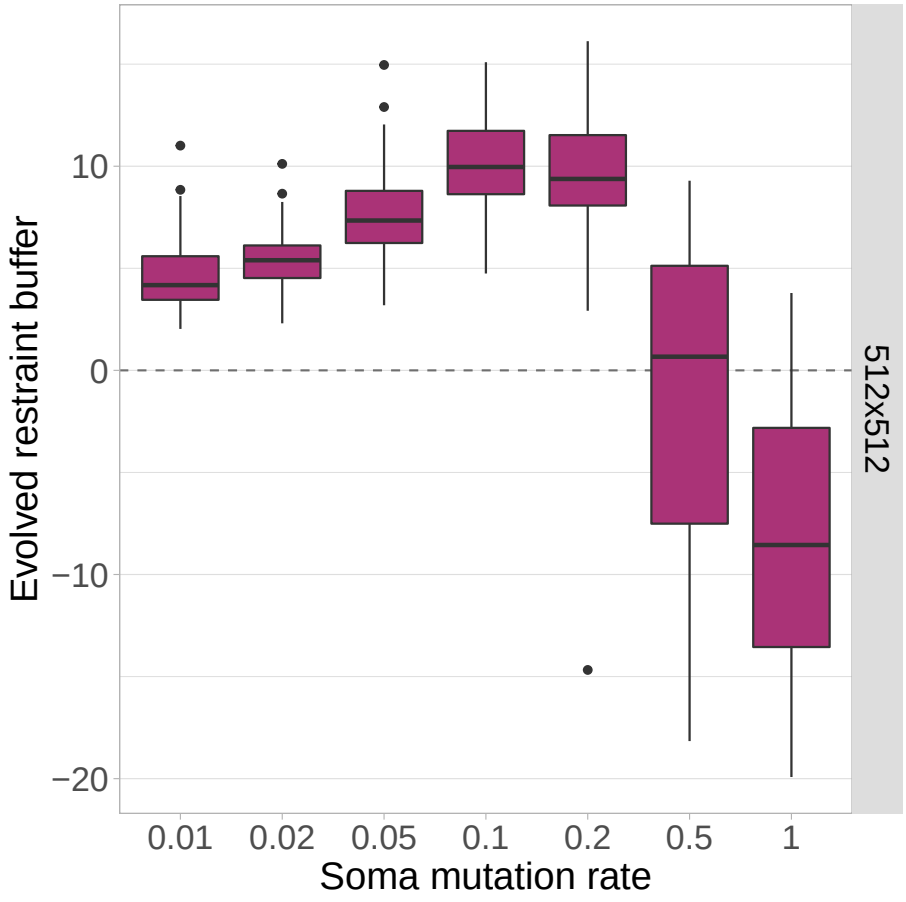
3.4.4 Organism size 128x128



3.4.5 Organism size 256x256



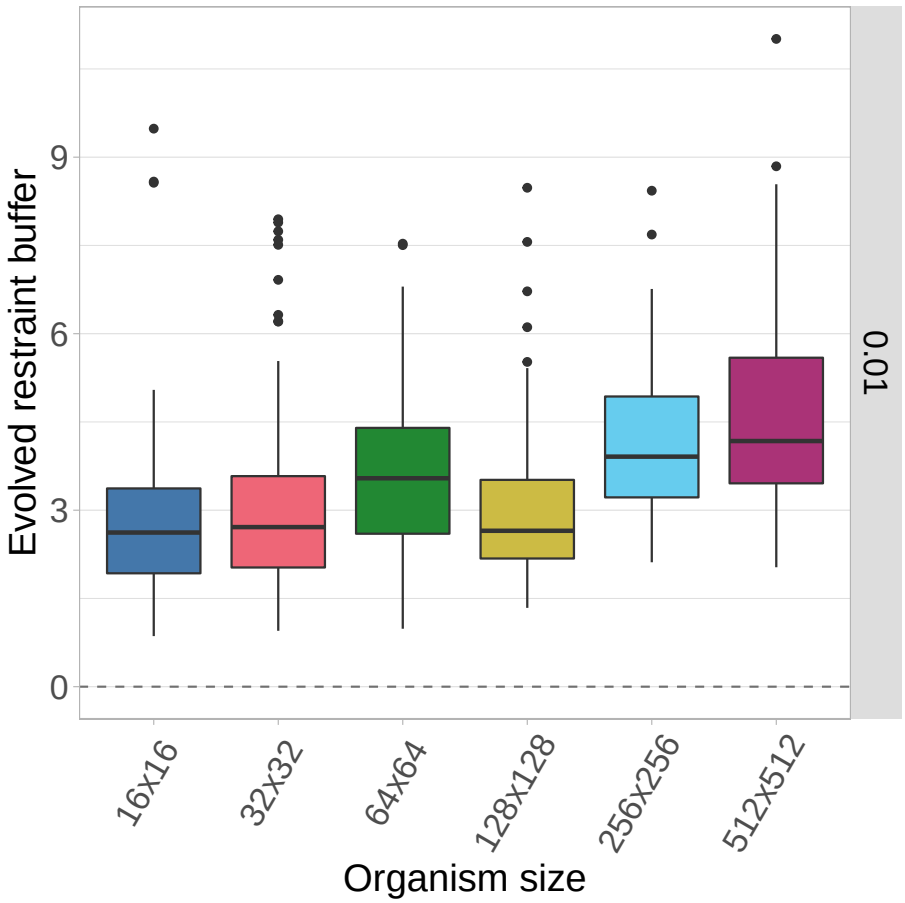
3.4.6 Organism size 512x512



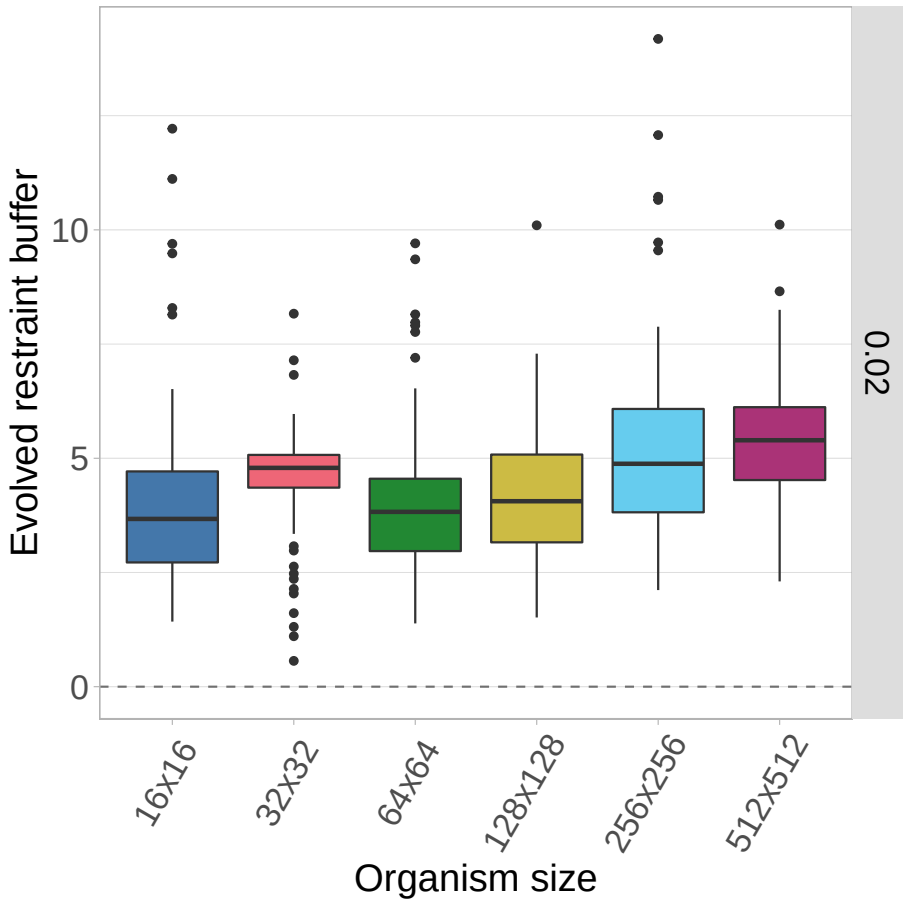
3.5 Single somatic mutation rate plots

Here we plot each somatic mutation rate independently, with organism size varying on the x-axis.

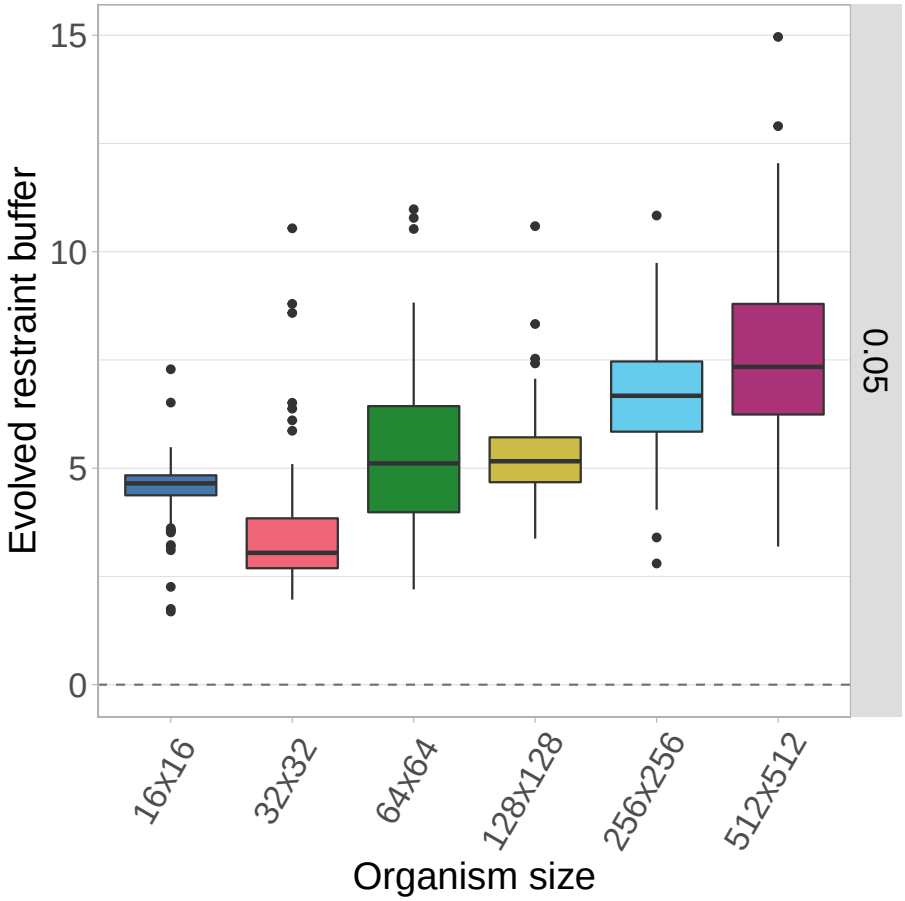
3.5.1 Somatic mut. rate 0.01



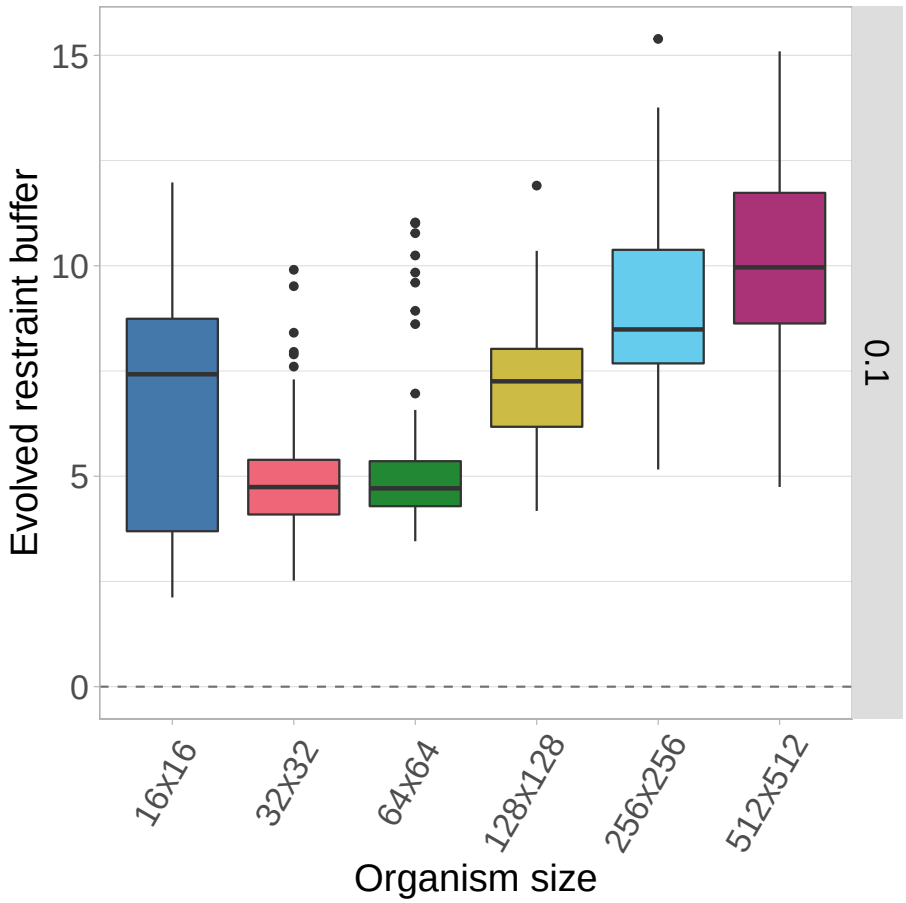
3.5.2 Somatic mut. rate 0.02



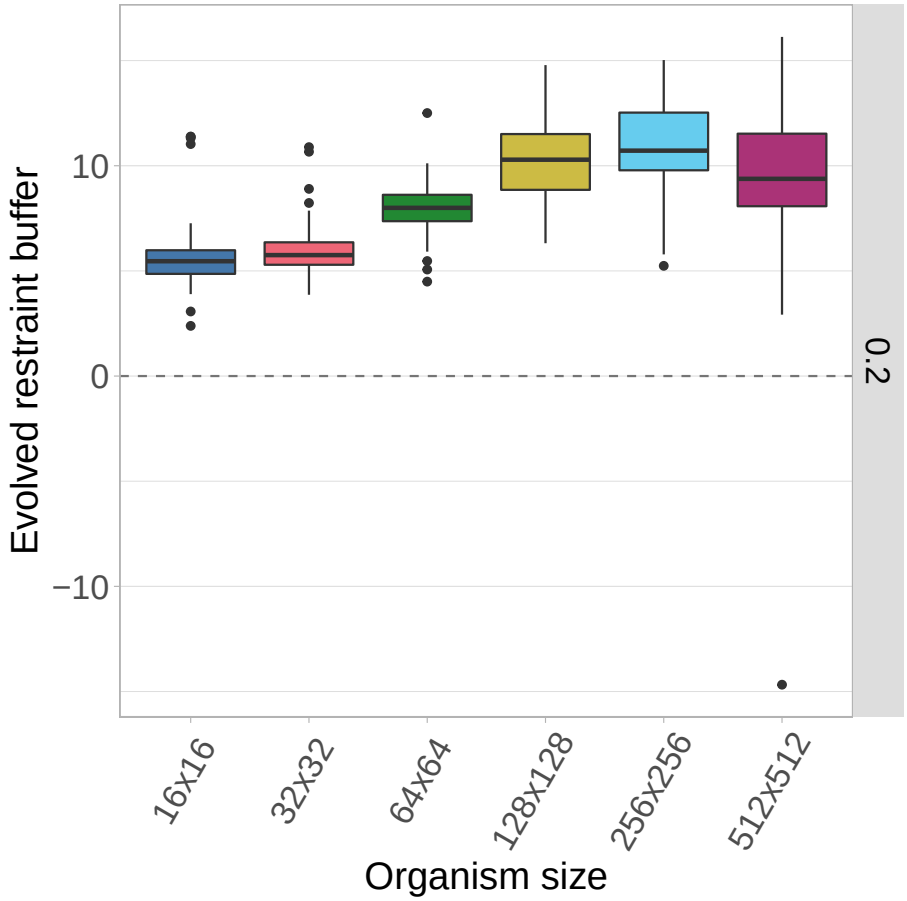
3.5.3 Somatic mut. rate 0.05



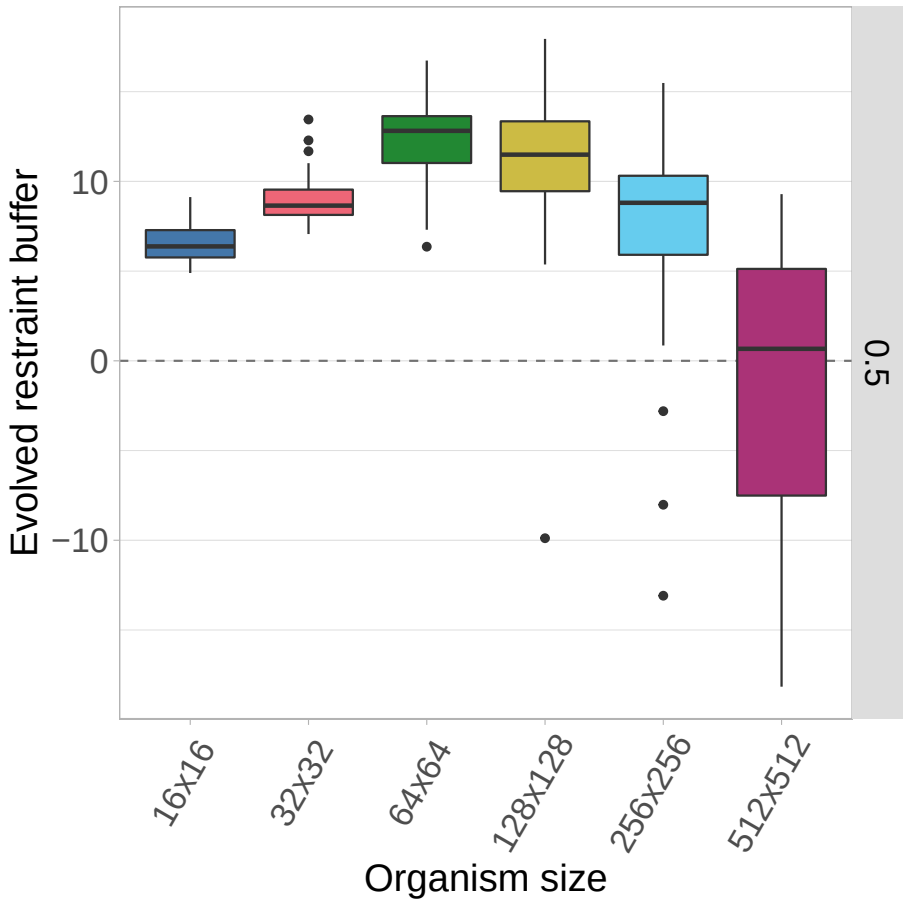
3.5.4 Somatic mut. rate 0.1



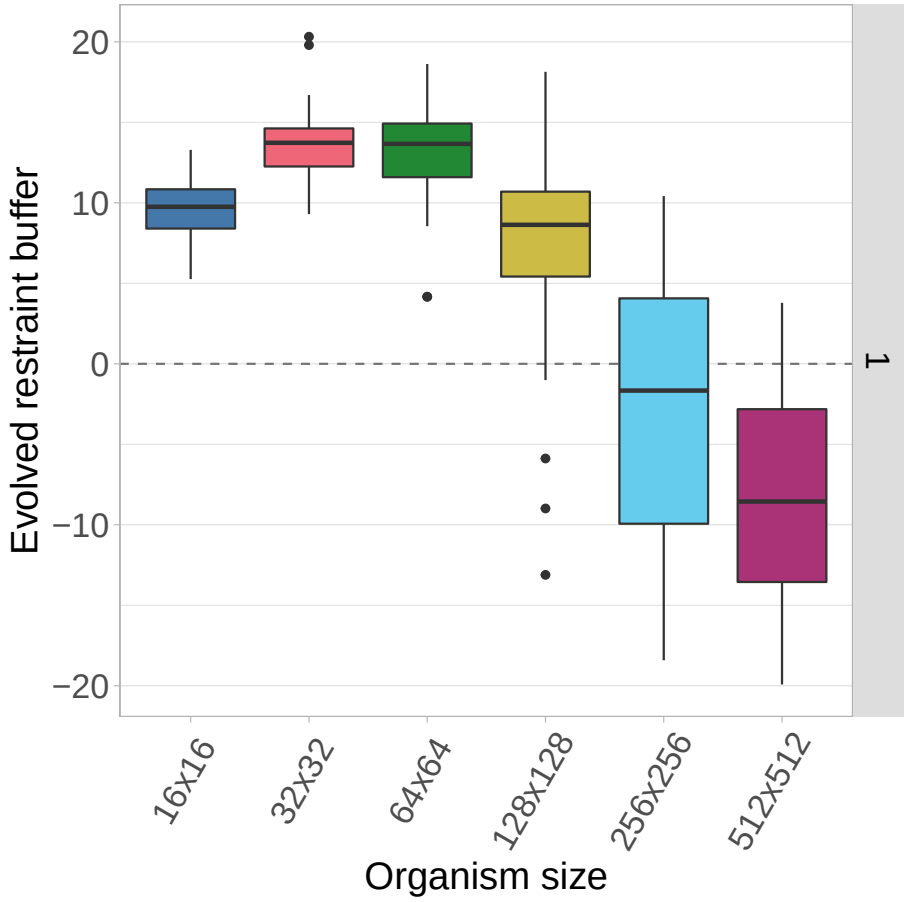
3.5.5 Somatic mut. rate 0.2



3.5.6 Somatic mut. rate 0.5



3.5.7 Somatic mut. rate 1.0



3.6 Statistics

Since organism size is our main point of comparison, we calculate stats for each somatic mutation rate.

First, we perform a Kruskal-Wallis test across all organism sizes to indicate if variance exists at that mutation rate. If variance exists, we then perfrm a pairwise Wilcoxon Rank-Sum test to show which pairs of organism sizes significantly differ. Finally, we perform Bonferroni-Holm corrections for multiple comparisons.

```
mut_vec = c(0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1)
df_kruskal = data.frame(data = matrix(nrow = 0, ncol = 4))
colnames(df_kruskal) = c('soma_mut_rate', 'p_value', 'chi_squared', 'df')
for(mut_rate in mut_vec){
  df_test = df2[df2$CELLMUT == mut_rate,]
```

```
    res = kruskal.test(df_test$restraint_value ~ df_test$MCSIZE, df_test)
    df_kruskal[nrow(df_kruskal) + 1,] = c(mut_rate, res$p.value, as.numeric(res$statistic)[1], as
  }
  df_kruskal$less_0.01 = df_kruskal$p_value < 0.01
  print(df_kruskal)
```

```
##   soma_mut_rate      p_value chi_squared df less_0.01
## 1          0.01 2.661659e-25    125.0566  5      TRUE
## 2          0.02 4.808020e-19     95.4471  5      TRUE
## 3          0.05 1.142677e-63    304.3847  5      TRUE
## 4          0.10 3.945761e-64    306.5323  5      TRUE
## 5          0.20 4.924029e-79    375.7743  5      TRUE
## 6          0.50 5.011460e-85    403.5832  5      TRUE
## 7          1.00 5.474947e-99    468.3229  5      TRUE
```

We see that significant variation exists within each mutation rate, so we perform pariwise Wilcoxon tests on each to see which pais of sizes are significantly different.

```
size_vec = c(16, 32, 64, 128, 256, 512)
mut_vec = c(0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1)
for(mut_rate in mut_vec){
  df_test = df2[df2$CELLMUT == mut_rate,]
  df_wilcox = data.frame(data = matrix(nrow = 0, ncol = 6))
  colnames(df_wilcox) = c('mut_rate', 'size_a', 'size_b', 'p_value_corrected', 'p_value_raw', 'W'
  for(size_idx_a in 1:(length(size_vec) - 1)){
    size_a = size_vec[size_idx_a]
    for(size_idx_b in (size_idx_a + 1):length(size_vec)){
      size_b = size_vec[size_idx_b]
      res = wilcox.test(df_test[df_test$MCSIZE == size_a,]$restraint_value, df_test[df_test$MCSIZ
      df_wilcox[nrow(df_wilcox) + 1,] = c(mut_rate, size_a, size_b, 0, res$p.value, as.numeric(re
    }
  }
  df_wilcox$p_value_corrected = p.adjust(df_wilcox$p_value_raw, method = 'holm')
  df_wilcox$less_0.01 = df_wilcox$p_value_corrected < 0.01
  print(paste0('Somatic mutation rate: ', mut_rate))
  print(df_wilcox)
}
```

```
## [1] "Somatic mutation rate: 0.01"
##    mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1      0.01     16     32      9.390497e-01 4.695249e-01 4703.5     FALSE
## 2      0.01     16     64      2.988154e-04 3.735192e-05 3312.0      TRUE
## 3      0.01     16    128      7.079843e-01 2.359948e-01 4514.5     FALSE
## 4      0.01     16    256      2.034819e-12 1.453442e-13 1974.5      TRUE
## 5      0.01     16    512      4.368517e-15 2.912344e-16 1653.0      TRUE
## 6      0.01     32     64      1.074876e-02 1.535537e-03 3703.0     FALSE
```

```
## 7      0.01     32    128      9.390497e-01 7.176323e-01 4851.5     FALSE
## 8      0.01     32    256      8.111610e-09 8.111610e-10 2485.5      TRUE
## 9      0.01     32    512      1.748038e-11 1.456698e-12 2102.5      TRUE
## 10     0.01     64    128      1.074876e-02 1.601365e-03 6292.0     FALSE
## 11     0.01     64    256      1.397091e-02 2.794183e-03 3776.0     FALSE
## 12     0.01     64    512      7.748038e-05 8.608931e-06 3178.5      TRUE
## 13     0.01    128    256      3.676583e-09 3.342348e-10 2428.5      TRUE
## 14     0.01    128    512      2.110112e-12 1.623163e-13 1980.5      TRUE
## 15     0.01    256    512      2.266729e-01 5.666822e-02 4219.5     FALSE
## [1] "Somatic mutation rate: 0.02"
##    mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1      0.02     16     32      3.611494e-05 4.012771e-06 3112.5      TRUE
## 2      0.02     16     64      4.740405e-01 4.740405e-01 4706.5     FALSE
## 3      0.02     16    128      2.648393e-01 5.296786e-02 4207.5     FALSE
## 4      0.02     16    256      6.698428e-07 5.582024e-08 2776.5      TRUE
## 5      0.02     16    512      4.142268e-11 2.761512e-12 2139.0      TRUE
## 6      0.02     32     64      1.240992e-05 1.240992e-06 6985.0      TRUE
## 7      0.02     32    128      2.150816e-02 3.584693e-03 6192.5     FALSE
## 8      0.02     32    256      3.993493e-01 9.983733e-02 4326.0     FALSE
## 9      0.02     32    512      1.117168e-04 1.396459e-05 3221.5      TRUE
## 10     0.02     64    128      4.025666e-01 2.012833e-01 4476.5     FALSE
## 11     0.02     64    256      5.648464e-06 5.134967e-07 2944.5      TRUE
## 12     0.02     64    512      6.120346e-11 4.371676e-12 2165.5      TRUE
## 13     0.02    128    256      3.129242e-04 4.470345e-05 3329.0      TRUE
## 14     0.02    128    512      1.760116e-08 1.353935e-09 2519.0      TRUE
## 15     0.02    256    512      3.993493e-01 1.013587e-01 4329.0     FALSE
## [1] "Somatic mutation rate: 0.05"
##    mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1      0.05     16     32      8.163575e-15 9.070638e-16 8290.5      TRUE
## 2      0.05     16     64      1.254683e-03 4.182276e-04 3555.5      TRUE
## 3      0.05     16    128      2.819711e-09 5.639421e-10 2462.0      TRUE
## 4      0.05     16    256      1.007639e-23 8.396990e-25  791.0      TRUE
## 5      0.05     16    512      3.169326e-24 2.437943e-25  742.5      TRUE
## 6      0.05     32     64      9.865308e-14 1.409330e-14 1850.0      TRUE
## 7      0.05     32    128      9.672216e-22 8.792924e-23  978.5      TRUE
## 8      0.05     32    256      4.456762e-26 3.183402e-27  576.5      TRUE
## 9      0.05     32    512      1.225797e-27 8.171978e-29  441.0      TRUE
## 10     0.05     64    128      9.619980e-01 9.619980e-01 4980.0     FALSE
## 11     0.05     64    256      4.409184e-09 1.102296e-09 2505.5      TRUE
## 12     0.05     64    512      1.967988e-13 3.279979e-14 1894.5      TRUE
## 13     0.05    128    256      3.061979e-14 3.827473e-15 1782.5      TRUE
## 14     0.05    128    512      4.080298e-17 4.080298e-18 1448.5      TRUE
## 15     0.05    256    512      2.648877e-03 1.324439e-03 3685.5      TRUE
## [1] "Somatic mutation rate: 0.1"
##    mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1       0.1     16     32      3.903716e-03 9.759291e-04 6350.0      TRUE
```

```
## 2       0.1     16     64      9.815188e-02 3.271729e-02 5874.5     FALSE
## 3       0.1     16    128      6.061146e-01 3.140880e-01 4587.5     FALSE
## 4       0.1     16    256      3.278276e-08 5.463793e-09 2612.5      TRUE
## 5       0.1     16    512      9.506115e-18 1.188264e-18 1391.5      TRUE
## 6       0.1     32     64      6.061146e-01 3.030573e-01 4578.0     FALSE
## 7       0.1     32    128      8.673971e-21 8.673971e-22 1074.0      TRUE
## 8       0.1     32    256      6.950798e-29 4.964856e-30  340.0      TRUE
## 9       0.1     32    512      1.934395e-30 1.289597e-31  211.5      TRUE
## 10      0.1     64    128      2.239733e-18 2.488592e-19 1320.5      TRUE
## 11      0.1     64    256      1.194130e-25 9.951080e-27  619.5      TRUE
## 12      0.1     64    512      1.966283e-27 1.512525e-28  463.5      TRUE
## 13      0.1    128    256      8.038941e-11 1.148420e-11 2222.0      TRUE
## 14      0.1    128    512      1.880691e-21 1.709719e-22 1006.0      TRUE
## 15      0.1    256    512      3.931365e-04 7.862729e-05 3383.5      TRUE
## [1] "Somatic mutation rate: 0.2"
##    mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1       0.2     16     32      1.077048e-02 5.385238e-03 3860.5     FALSE
## 2       0.2     16     64      6.720281e-24 8.400351e-25  791.0      TRUE
## 3       0.2     16    128      1.215721e-28 1.013101e-29  365.5      TRUE
## 4       0.2     16    256      4.359012e-29 3.353086e-30  326.0      TRUE
## 5       0.2     16    512      3.611807e-25 3.283461e-26  665.0      TRUE
## 6       0.2     32     64      5.255254e-22 7.507505e-23  972.0      TRUE
## 7       0.2     32    128      3.542154e-29 2.530110e-30  316.0      TRUE
## 8       0.2     32    256      3.153758e-30 2.102505e-31  228.5      TRUE
## 9       0.2     32    512      1.346976e-24 1.496640e-25  723.5      TRUE
## 10      0.2     64    128      1.237545e-13 2.062574e-14 1870.0      TRUE
## 11      0.2     64    256      6.129521e-25 6.129521e-26  689.0      TRUE
## 12      0.2     64    512      1.436552e-07 2.873105e-08 2728.5      TRUE
## 13      0.2    128    256      6.935985e-03 2.311995e-03 3752.5      TRUE
## 14      0.2    128    512      1.987108e-01 1.987108e-01 5526.5     FALSE
## 15      0.2    256    512      3.309684e-04 8.274210e-05 6611.5      TRUE
## [1] "Somatic mutation rate: 0.5"
##    mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1       0.5     16     32      1.212403e-25 1.212403e-26  627.0      TRUE
## 2       0.5     16     64      1.029212e-31 7.351512e-33  113.0      TRUE
## 3       0.5     16    128      1.432034e-27 1.301849e-28  458.0      TRUE
## 4       0.5     16    256      3.887685e-06 1.295895e-06 3018.5      TRUE
## 5       0.5     16    512      1.499786e-19 2.499644e-20 8781.5      TRUE
## 6       0.5     32     64      3.854284e-24 4.282538e-25  764.5      TRUE
## 7       0.5     32    128      6.344735e-14 1.268947e-14 1844.5      TRUE
## 8       0.5     32    256      6.346151e-01 6.346151e-01 5195.0     FALSE
## 9       0.5     32    512      3.036159e-31 2.335507e-32 9847.5      TRUE
## 10      0.5     64    128      9.397051e-03 4.698526e-03 6157.5      TRUE
## 11      0.5     64    256      6.907801e-20 9.868288e-21 8822.0      TRUE
## 12      0.5     64    512      9.160009e-33 6.106673e-34 9971.0      TRUE
## 13      0.5    128    256      4.999760e-11 1.249940e-11 7773.0      TRUE
```

```
## 14      0.5    128    512      6.054856e-31 5.045714e-32 9821.0      TRUE
## 15      0.5    256    512      4.216225e-21 5.270281e-22 8947.0      TRUE
## [1] "Somatic mutation rate: 1"
##    mut_rate size_a size_b p_value_corrected  p_value_raw       W less_0.01
## 1         1     16     32      2.812620e-27 3.515774e-28   494.5      TRUE
## 2         1     16     64      5.606003e-22 9.343338e-23   981.0      TRUE
## 3         1     16    128      2.202125e-02 1.101063e-02  6041.0     FALSE
## 4         1     16    256      4.073858e-28 4.526509e-29  9580.5      TRUE
## 5         1     16    512      3.841268e-33 2.561566e-34 10000.0      TRUE
## 6         1     32     64      7.619035e-01 7.619035e-01  5124.5     FALSE
## 7         1     32    128      2.931097e-22 4.187282e-23  9052.0      TRUE
## 8         1     32    256      3.841268e-33 2.976903e-34  9995.0      TRUE
## 9         1     32    512      3.841268e-33 2.561711e-34 10000.0      TRUE
## 10        1     64    128      1.456083e-19 3.640207e-20  8765.0      TRUE
## 11        1     64    256      2.413338e-32 2.193944e-33  9928.0      TRUE
## 12        1     64    512      3.841268e-33 2.560845e-34 10000.0      TRUE
## 13        1    128    256      1.180975e-20 2.361951e-21  8883.5      TRUE
## 14        1    128    512      1.253447e-30 1.253447e-31  9789.5      TRUE
## 15        1    256    512      6.072904e-07 2.024301e-07  7127.5      TRUE
```

Chapter 4

Germ Mutation Rate Sweep

This experiment was one of the prelimary experiments we conducted to find the default parameters for Primordium. We varied the mutation rate, the probability that an offspring experineces a mutation to its restraint buffer during organism reproduction.

The final default germ mutation rate was 0.02 (*i.e.*, each organism reproduction has a 2% chance of mutation).

The configuration script and data for the experiment can be found under `2021_02_16__germ_mut_fin/` in the experiments directory of the git repository.

Load necessary libraries

```
library(dplyr)
library(ggplot2)
library(ggridges)
library(scales)
library(khroma)
```

Load the data and trim all the unecessay bits (*e.g.*, we initially ran sizes 8x8, 1024x1024 but cut them from the paper to make plots easier to read).

```
# Load the data
df = read.csv('../experiments/2021_02_16__germ_mut_fin/evolution/data/scraped_evolution_data.csv'
#df = read.csv('/research/rogue_cell/Primordium/experiments/2021_02_16__germ_mut_fin/evolution/da
# Trim off NAs (artifiacts of how we scraped the data) and trim to only have gen 10,000
cat(colnames(df), '\n')
```

```
## X generation ave_ones ave_repro_time min_ones max_ones var_ones rep_id MCSIZE COST GENS MUT PO
```

```
df2 = df[!is.na(df$MCSIZE) & df$generation == 10000,]
# Ignore data for size 8x8 and 1024x1024
```

```
df2 = df2[df2$MCSIZE != 8 & df2$MCSIZE != 1024,]
```

We group and summarize the data to make to ensure all replicates are present.

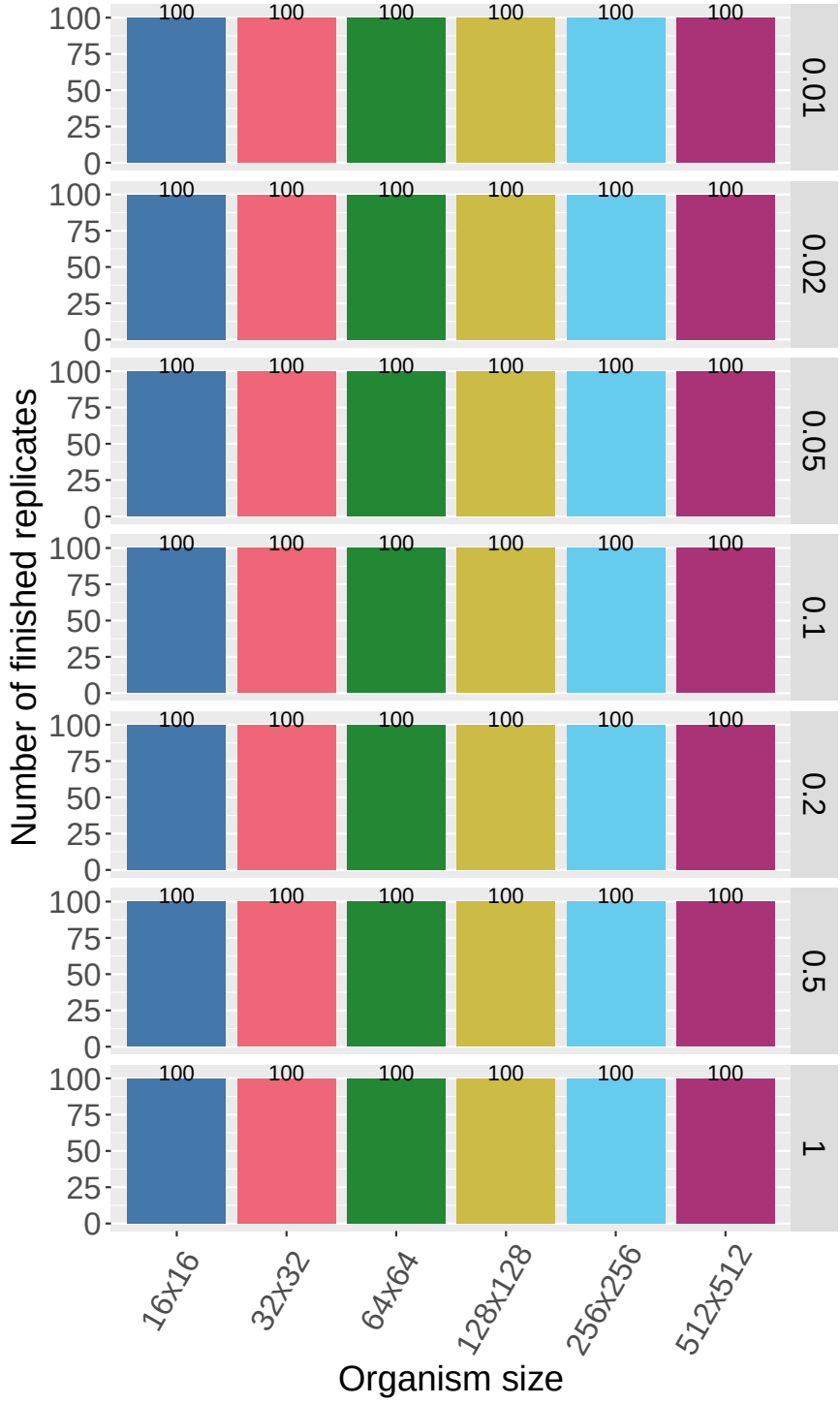
```
# Group the data by size and summarize
data_grouped = dplyr::group_by(df2, MCSIZE, MUT)
data_summary = dplyr::summarize(data_grouped, mean_ones = mean(ave_ones), n = dplyr::n
```

Further cleaning of the data plus adding some variables to make plotting easier.

```
# Calculate restraint value (x - 60 because genome length is 100 here)
df2$restraint_value = df2$ave_ones - 60
# Make a nice, clean factor for size
df2$size_str = paste0(df2$MCSIZE, 'x', df2$MCSIZE)
df2$size_factor = factor(df2$size_str, levels = c('16x16', '32x32', '64x64', '128x128'
df2$size_factor_reversed = factor(df2$size_str, levels = rev(c('16x16', '32x32', '64x6
df2$germ_mut_str = paste('GERM MUT', df2$MUT)
df2$mut_factor = factor(df2$MUT, levels = c(0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.00))
data_summary$size_str = paste0(data_summary$MCSIZE, 'x', data_summary$MCSIZE)
data_summary$size_factor = factor(data_summary$size_str, levels = c('16x16', '32x32',
data_summary$germ_mut_str = paste('GERM MUT', data_summary$MUT)
data_summary$mut_factor = factor(data_summary$MUT, levels = c(0.01, 0.02, 0.05, 0.10, 
# Create a map of colors we'll use to plot the different organism sizes
color_vec = as.character(khroma::color('bright')(7))
color_map = c(
  '16x16' =     color_vec[1],
  '32x32' =     color_vec[2],
  '64x64' =     color_vec[3],
  '128x128' =   color_vec[4],
  '256x256' =   color_vec[5],
  '512x512' =   color_vec[6],
  '1024x1024' = color_vec[7]
)
# Set the sizes for text in plots
text_major_size = 18
text_minor_size = 16
boxplot_color = '#9ecae1'
```

4.1 Data integrity check

Now we plot the number of finished replicates for each treatment to make sure all data are present. Each row shows a different germ mutation rate. Each bar/color

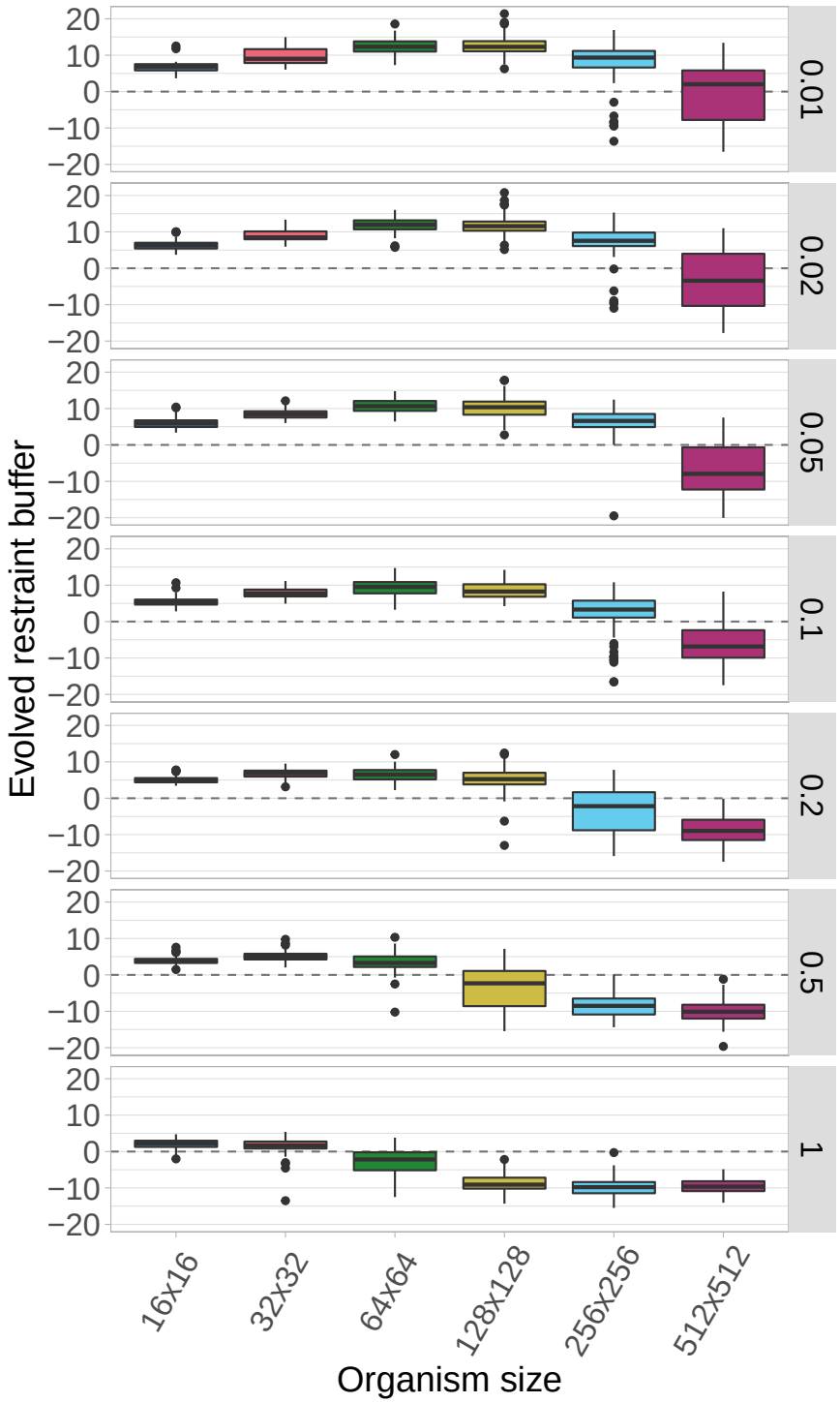


shows a different organism size.

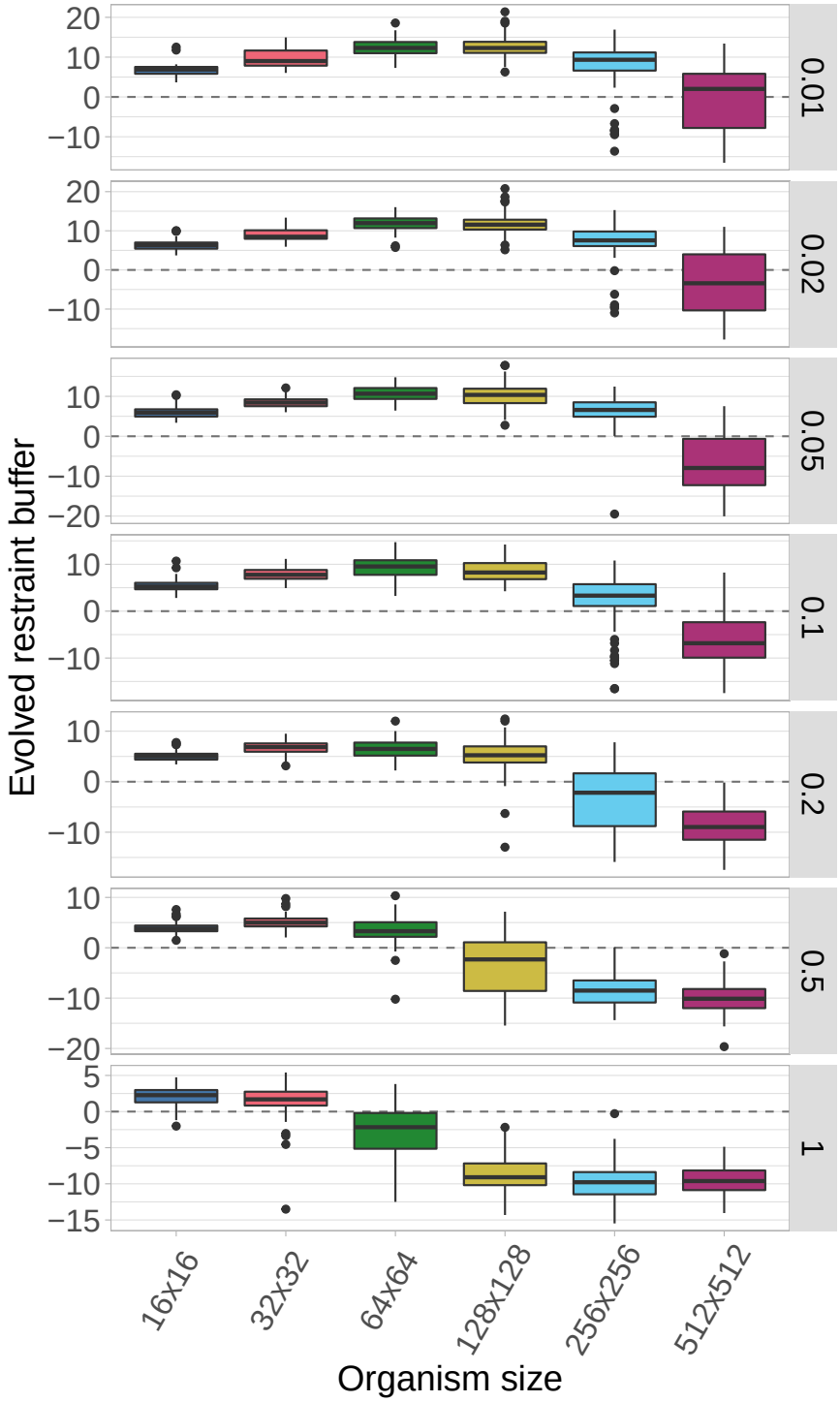
4.2 Aggregate plots

4.2.1 Facet by germ mutation rate

Here we plot all the data at once. Each row shows a different germ mutation rate and each boxplot shows a given organism size.

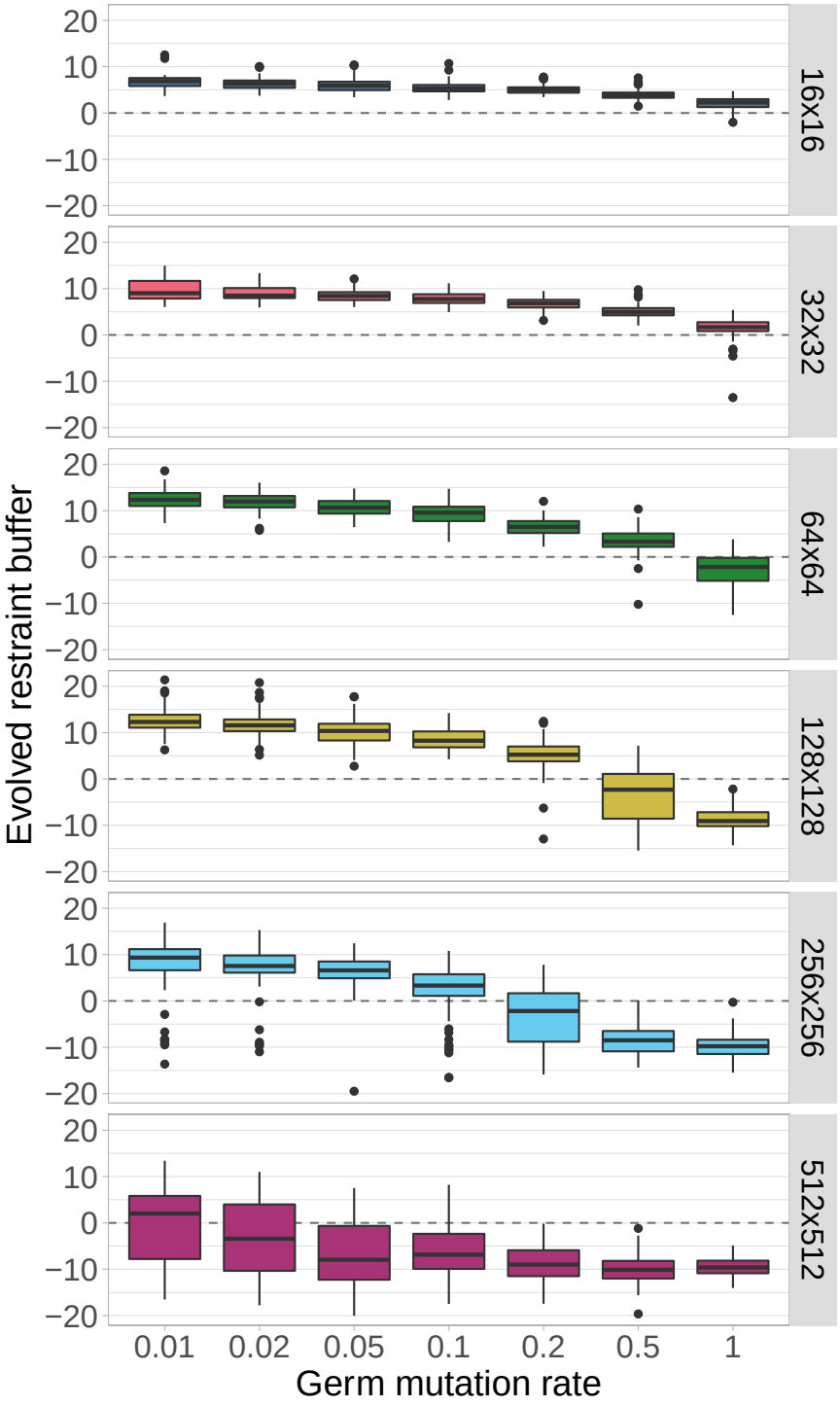

Evolved restraint buffer
20
10
0
−10
−20
0.01
0.02
0.05
0.1
0.2
0.5
1
16x16
32x32
64x64
128x128
256x256
512x512
Organism size

Here is the same data, plotted identically other than now each row can have a different y-axis.

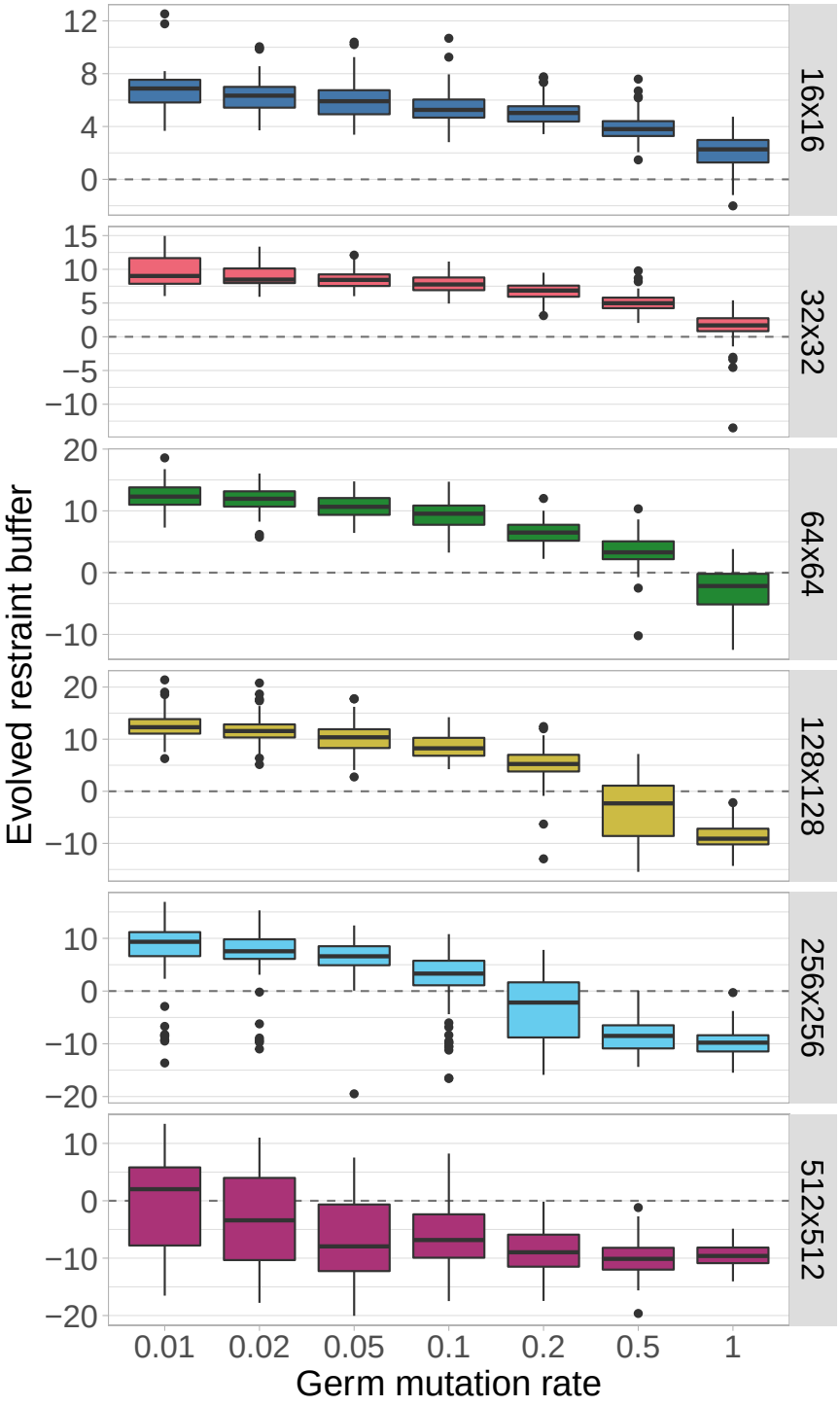

Evolved restraint buffer
0.01
0.02
0.05
0.1
0.2
0.5
1
16x16
32x32
64x64
128x128
256x256
512x512
Organism size

4.2.2 Facet by organism size

Next, we plot the same data, but this time each row corresponds to a certain organism size while germ mutation rate varies along the x-axis.


Evolved restraint buffer
20
10
0
−10
−20
16x16
32x32
64x64
128x128
256x256
512x512
0.01
0.02
0.05
0.1
0.2
0.5
1
Germ mutation rate

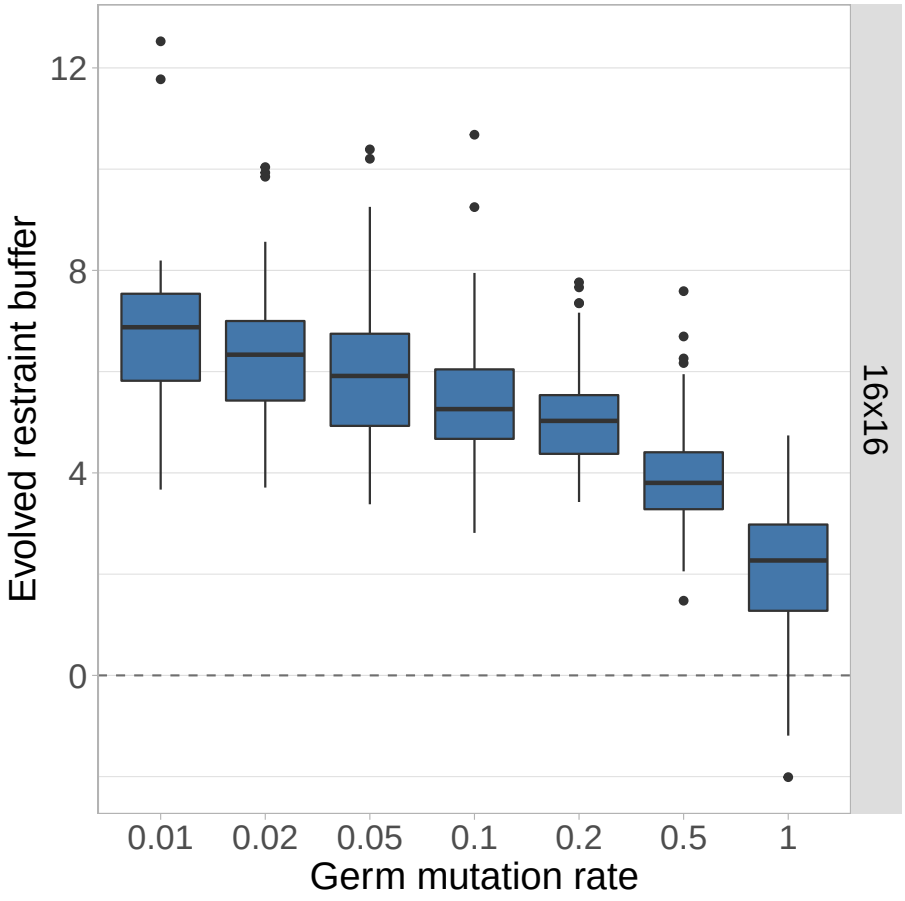
Again, we plot the same data again, but now the y-axis can change between rows.


Evolved restraint buffer
Germ mutation rate
0.01
0.02
0.05
0.1
0.2
0.5
1
16x16
32x32
64x64
128x128
256x256
512x512

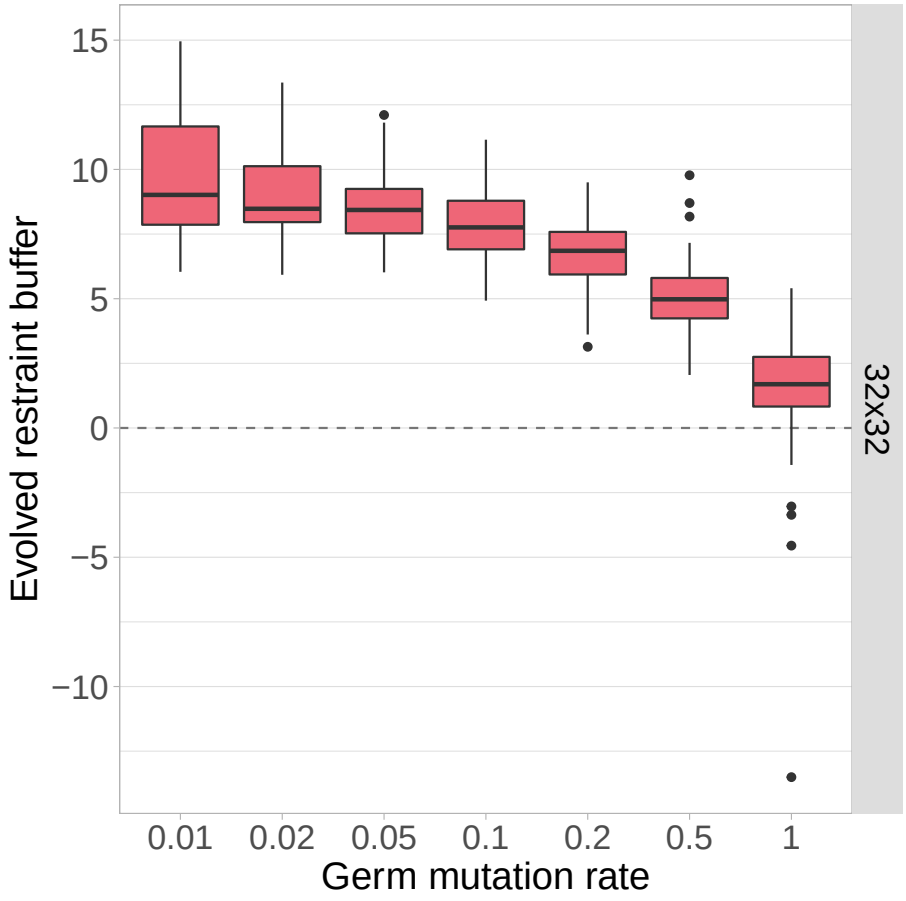
4.3 Single organism size plots

Here we plot each organism size independently, with the germ mutation rate on the x-axis.

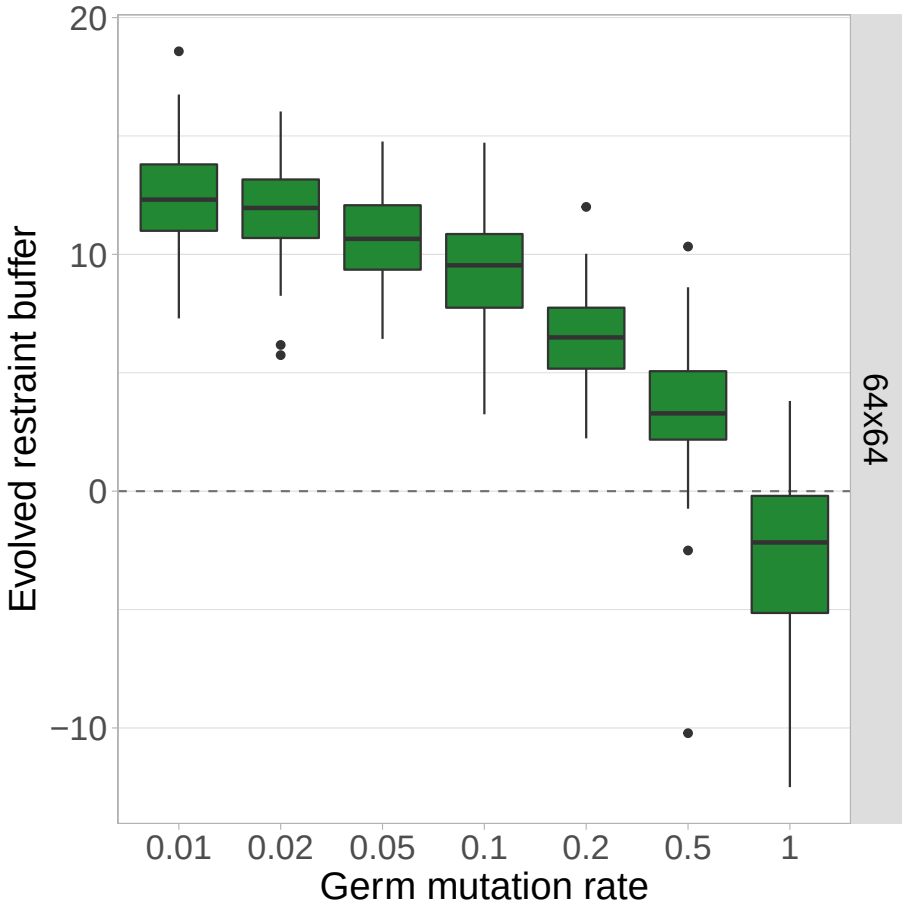
4.3.1 Organism size 16x16



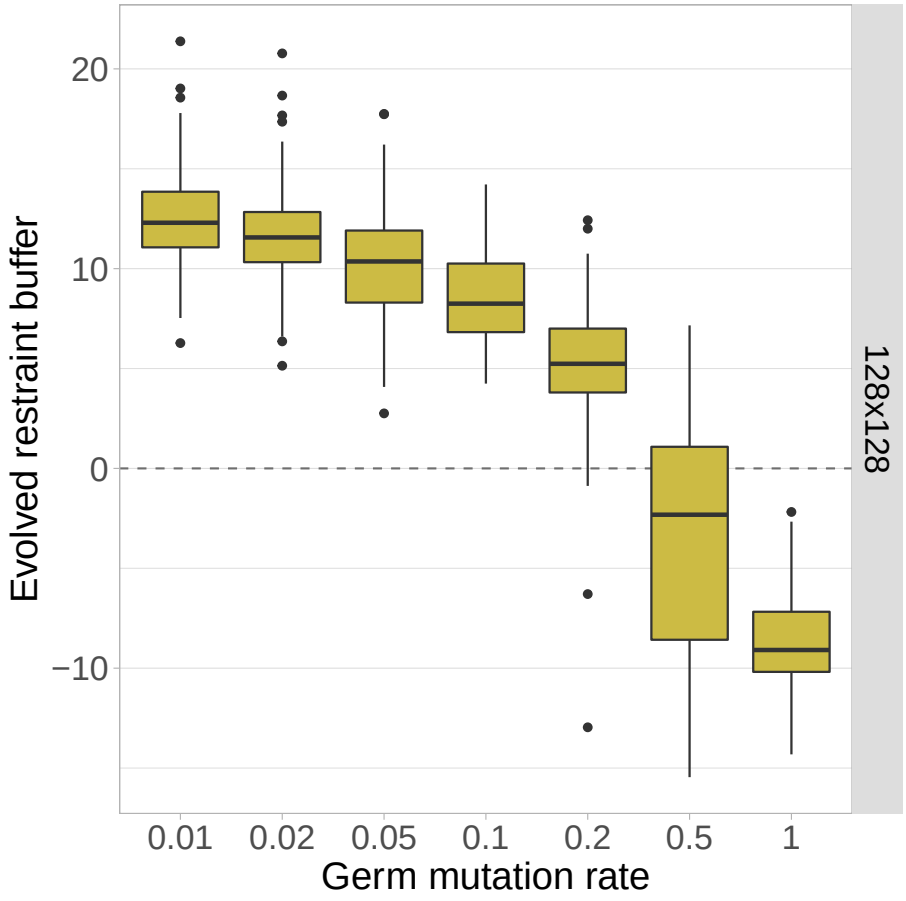
4.3.2 Organism size 32x32



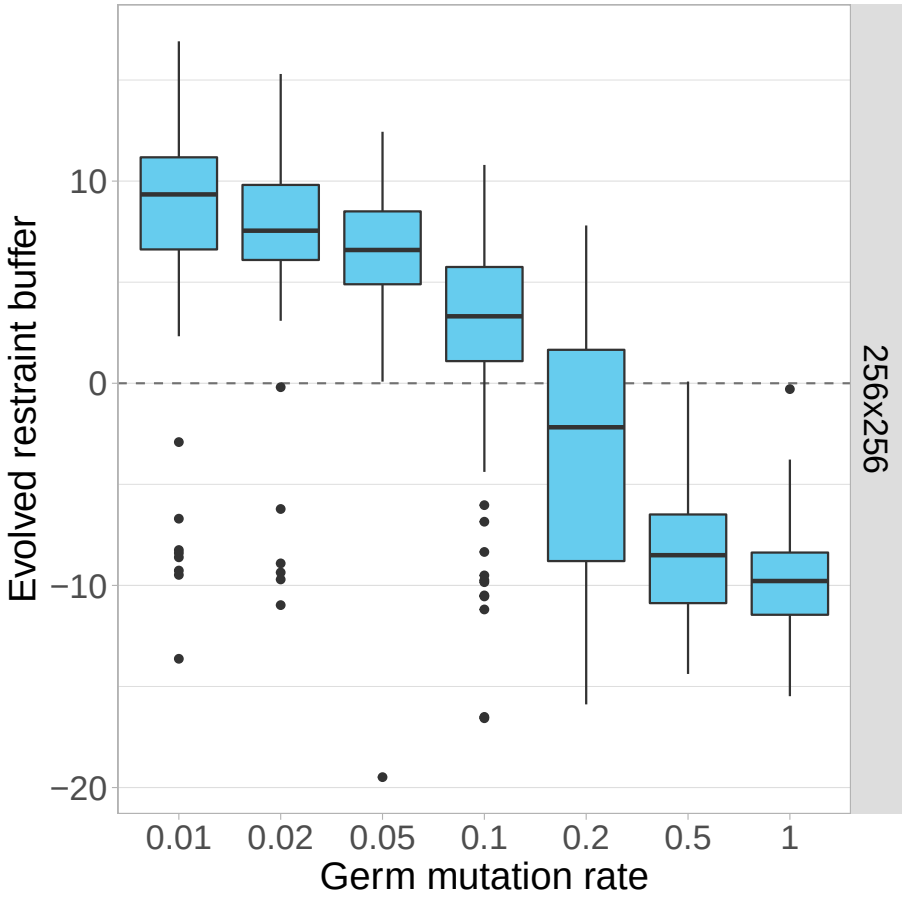
4.3.3 Organism size 64x64



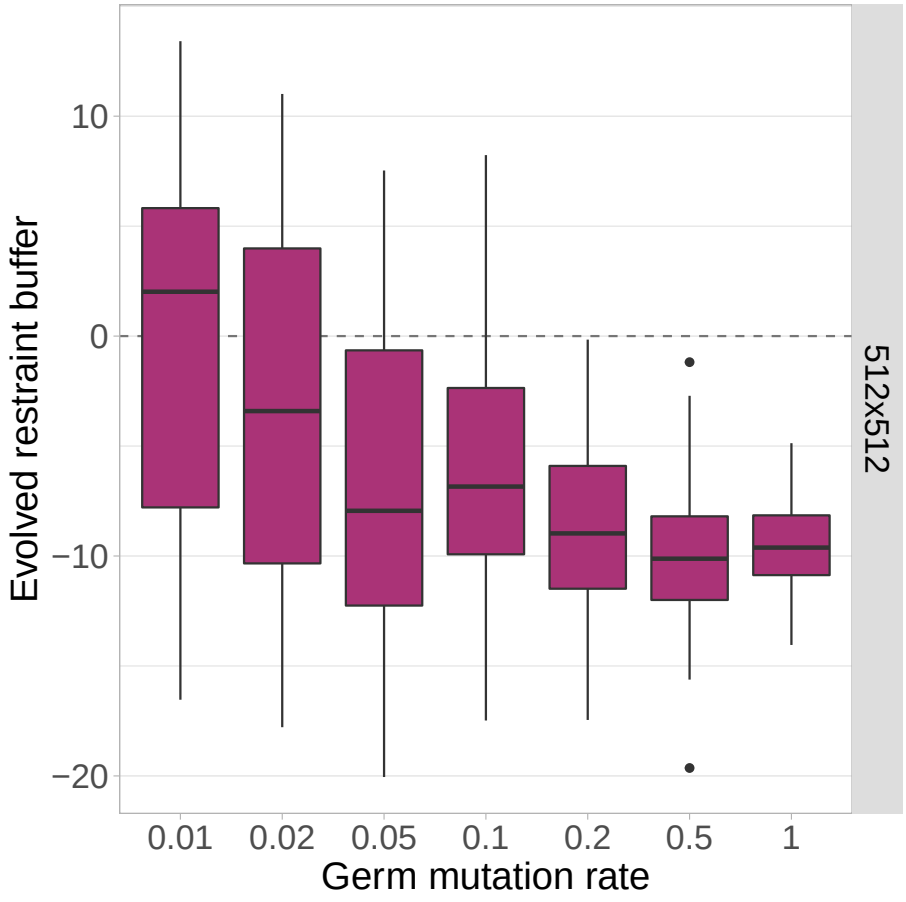
4.3.4 Organism size 128x128



4.3.5 Organism size 256x256



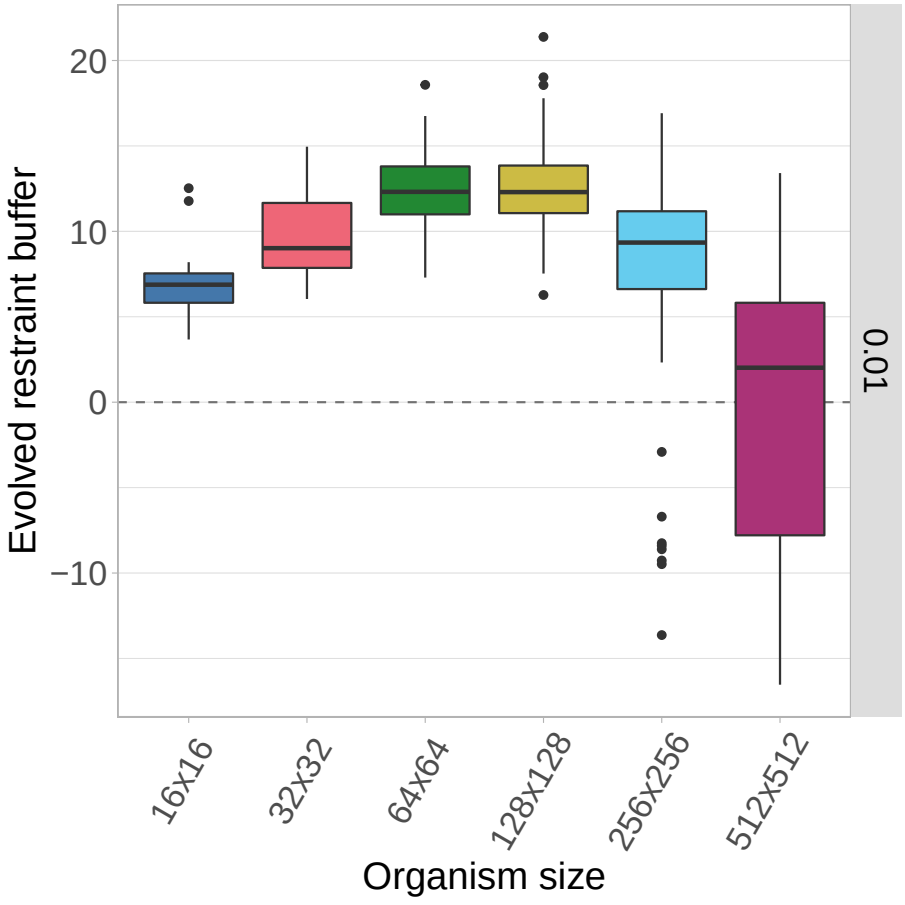
4.3.6 Organism size 512x512



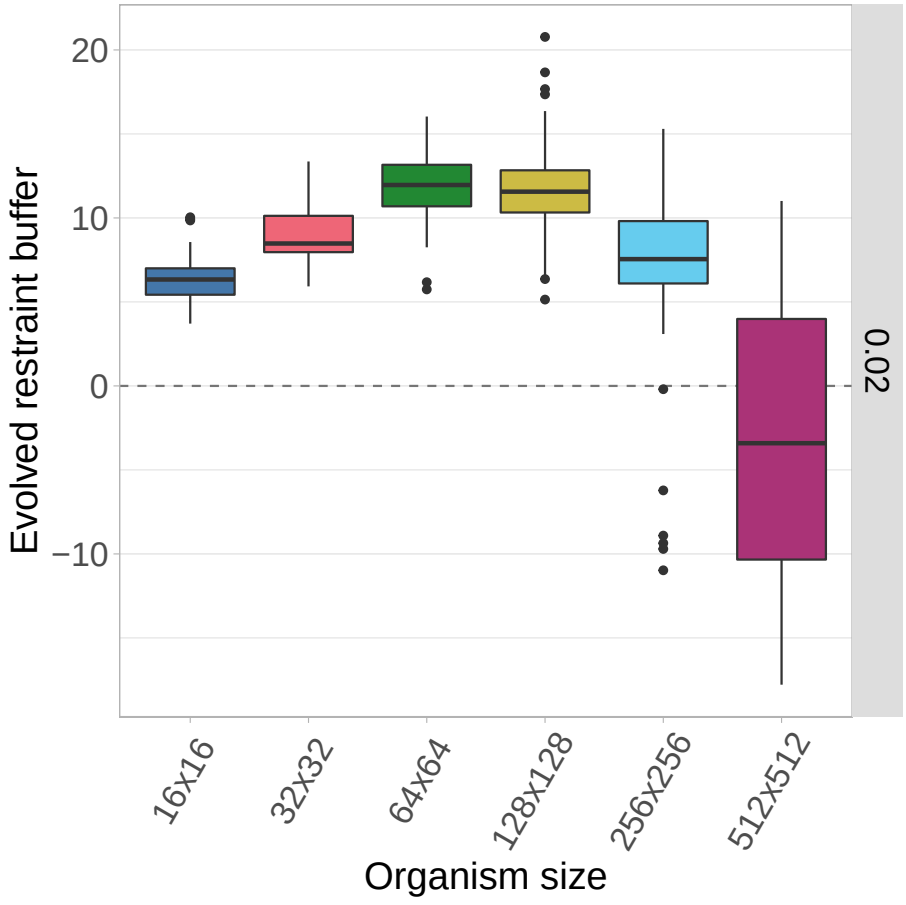
4.4 Single organism size plots

Similarly, here we plot each germ mutation rate independently, with the organism size on the x-axis.

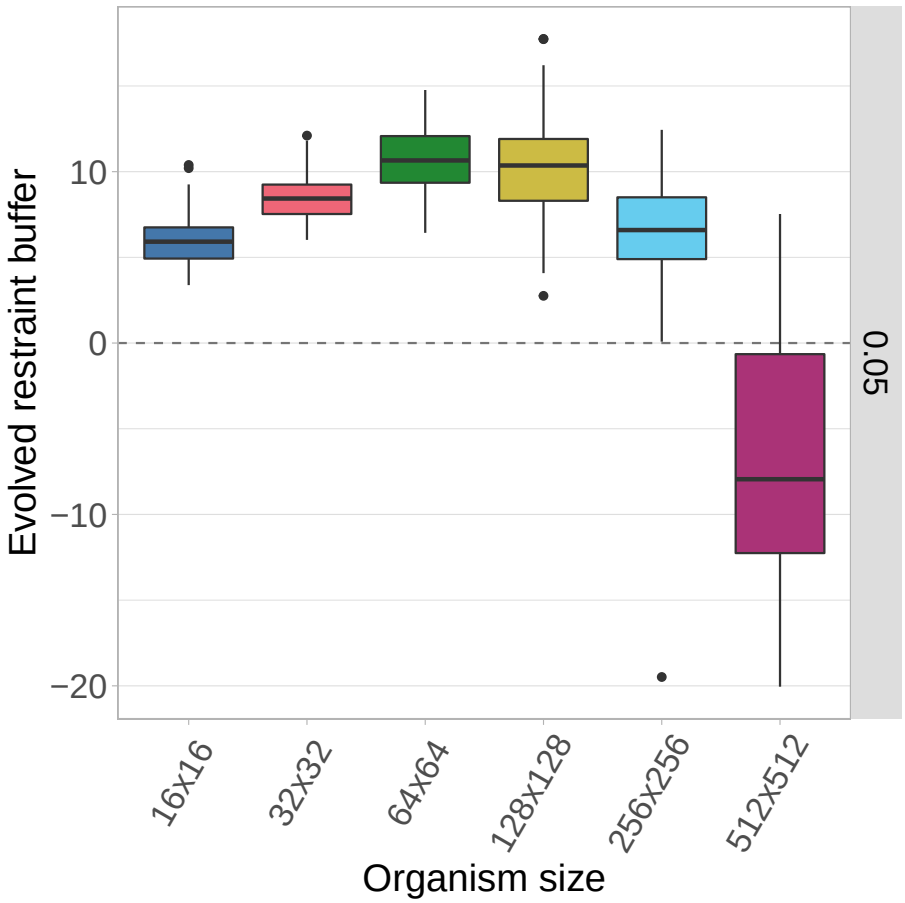
4.4.1 Germ mut. rate 0.01



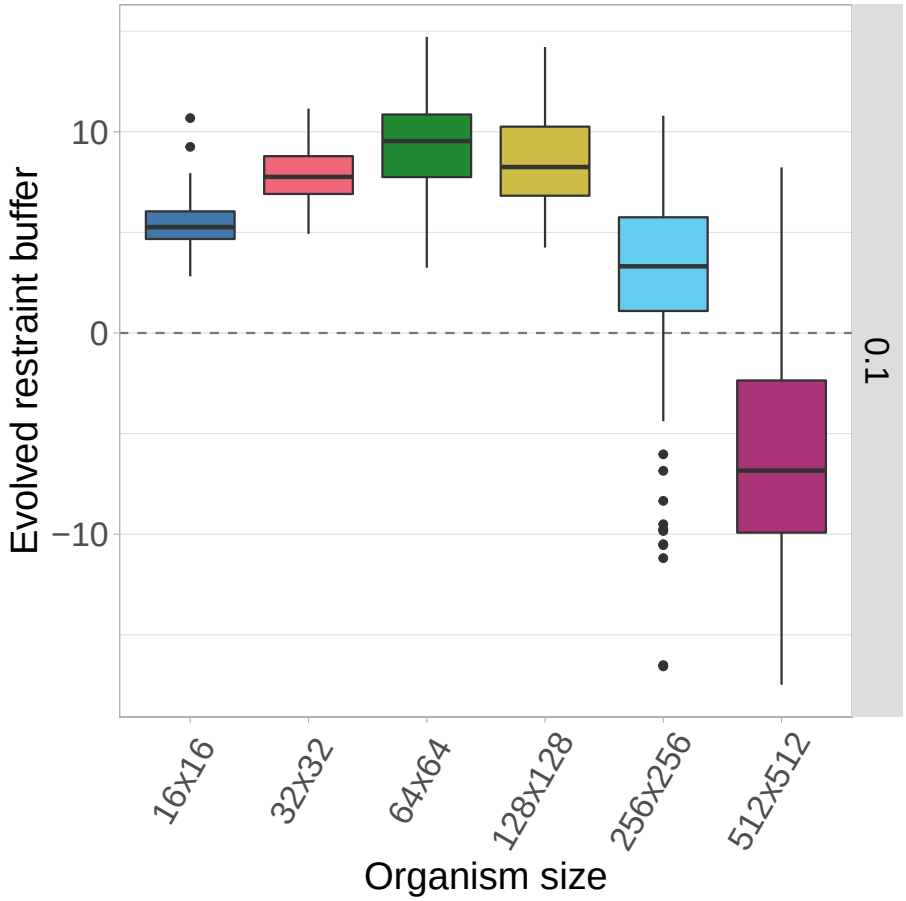
4.4.2 Germ mut. rate 0.02



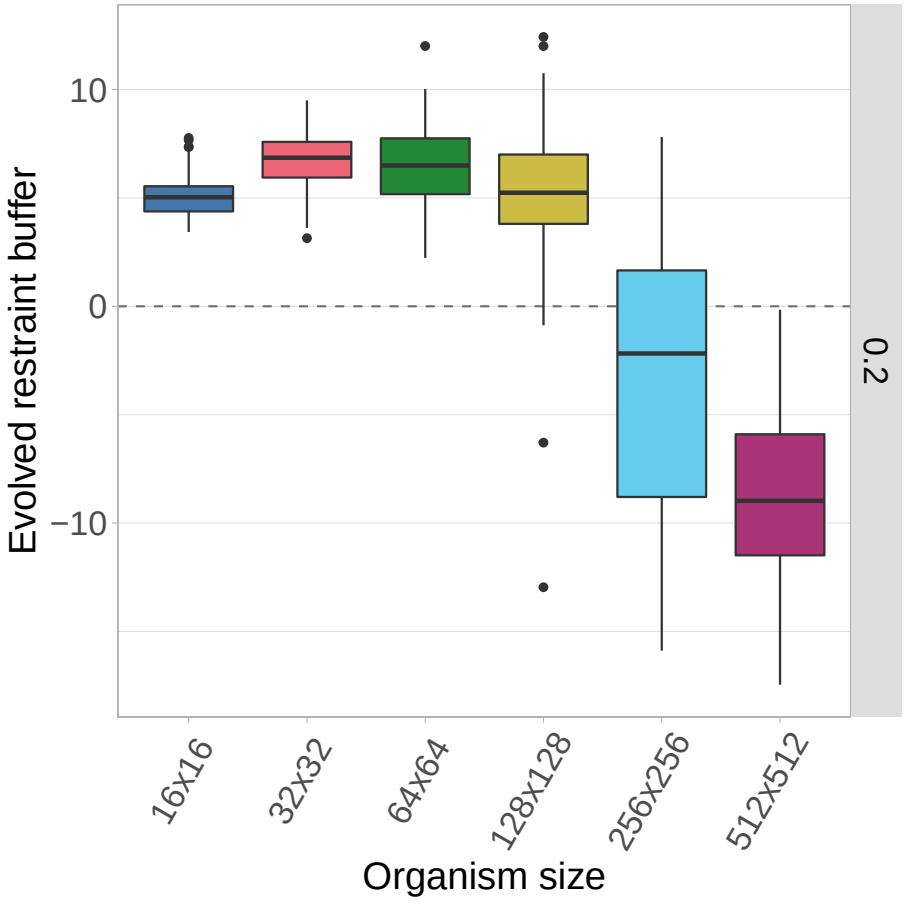
4.4.3 Germ mut. rate 0.05



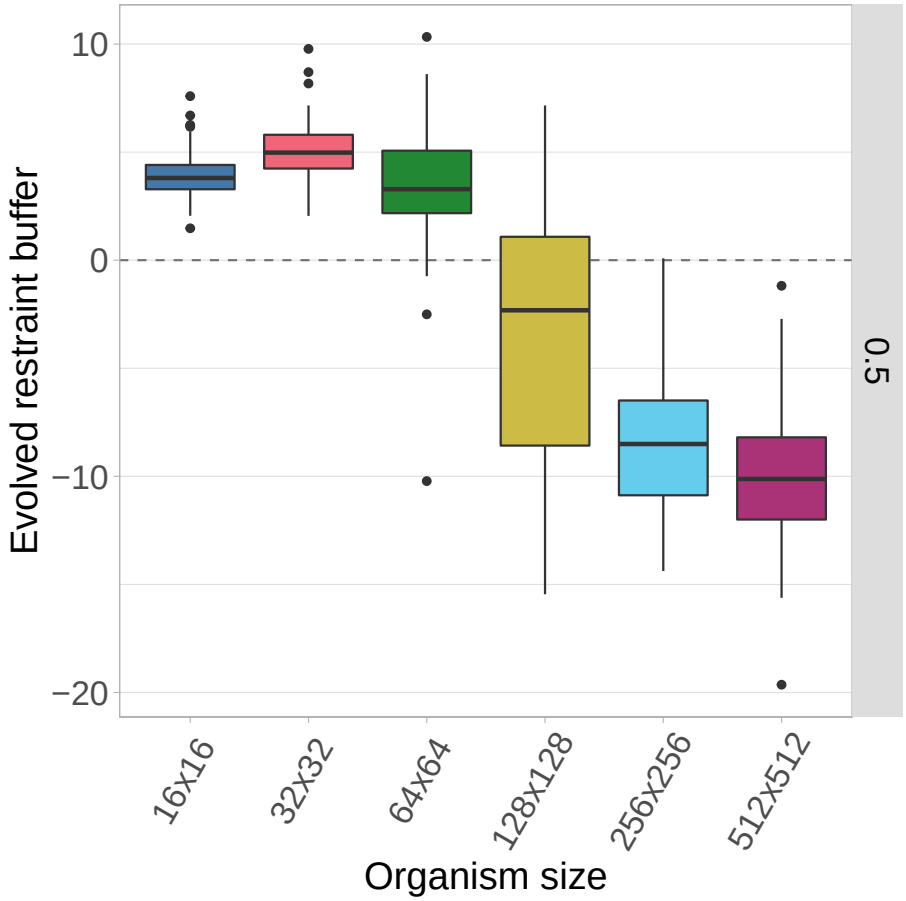
4.4.4 Germ mut. rate 0.1



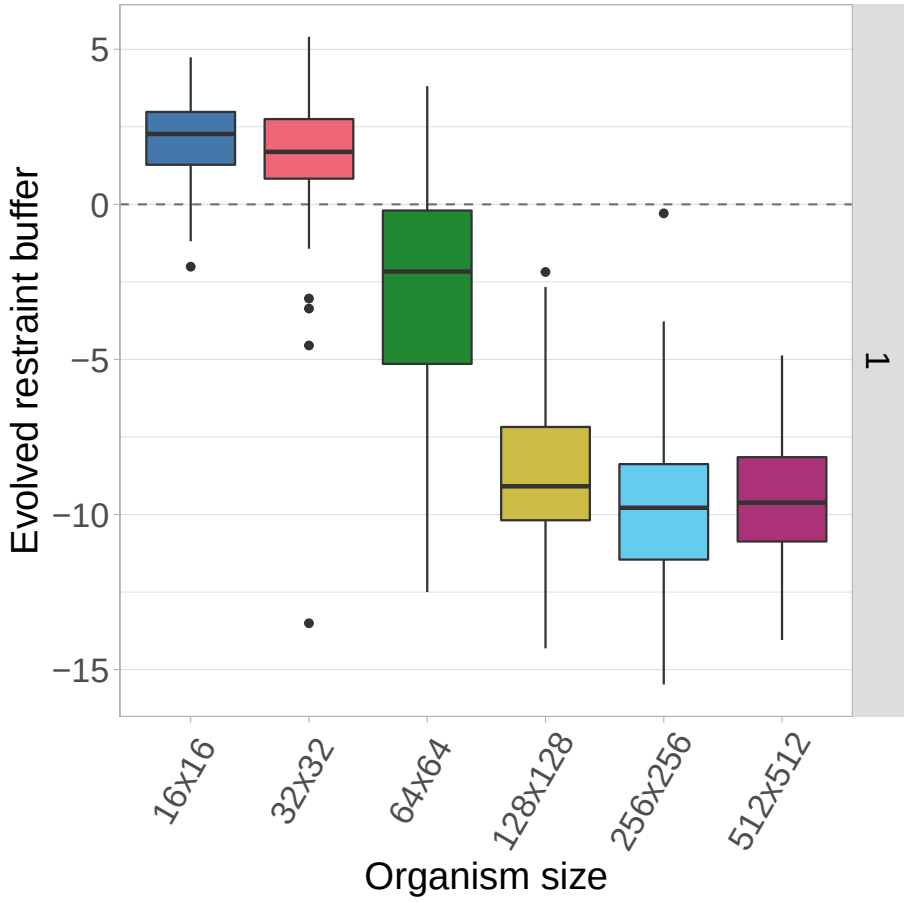
4.4.5 Germ mut. rate 0.2



4.4.6 Germ mut. rate 0.5



4.4.7 Germ mut. rate 1.0



4.5 Statistics

Since organism size is our main point of comparison, we calculate stats for each germ mutation rate.

First, we perform a Kruskal-Wallis test across all organism sizes to indicate if variance exists at that mutation rate. If variance exists, we then perfrm a pairwise Wilcoxon Rank-Sum test to show which pairs of organism sizes significantly differ. Finally, we perform Bonferroni-Holm corrections for multiple comparisons.

```
mut_vec = c(0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1)
df_kruskal = data.frame(data = matrix(nrow = 0, ncol = 4))
colnames(df_kruskal) = c('germ_mut_rate', 'p_value', 'chi_squared', 'df')
for(mut_rate in mut_vec){
  df_test = df2[df2$MUT == mut_rate,]
```

```
    res = kruskal.test(df_test$restraint_value ~ df_test$MCSIZE, df_test)
    df_kruskal[nrow(df_kruskal) + 1,] = c(mut_rate, res$p.value, as.numeric(res$statistic)[1], as
  }
  df_kruskal$less_0.01 = df_kruskal$p_value < 0.01
  print(df_kruskal)
```

```
##   germ_mut_rate      p_value chi_squared df less_0.01
## 1          0.01 9.191452e-79    374.5160  5      TRUE
## 2          0.02 6.227269e-82    389.2251  5      TRUE
## 3          0.05 1.934895e-82    391.5809  5      TRUE
## 4          0.10 1.983976e-83    396.1708  5      TRUE
## 5          0.20 3.180895e-85    404.4991  5      TRUE
## 6          0.50 4.313881e-91    431.7152  5      TRUE
## 7          1.00 2.144229e-92    437.7600  5      TRUE
```

We see that significant variation exists within each mutation rate, so we perform pariwise Wilcoxon tests on each to see which pais of sizes are significantly different.

```
size_vec = c(16, 32, 64, 128, 256, 512)
mut_vec = c(0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1)
for(mut_rate in mut_vec){
  df_test = df2[df2$MUT == mut_rate,]
  df_wilcox = data.frame(data = matrix(nrow = 0, ncol = 6))
  colnames(df_wilcox) = c('germ_mut_rate', 'size_a', 'size_b', 'p_value_corrected', 'p_value_raw'
  for(size_idx_a in 1:(length(size_vec) - 1)){
    size_a = size_vec[size_idx_a]
    for(size_idx_b in (size_idx_a + 1):length(size_vec)){
      size_b = size_vec[size_idx_b]
      res = wilcox.test(df_test[df_test$MCSIZE == size_a,]$restraint_value, df_test[df_test$MCSIZ
      df_wilcox[nrow(df_wilcox) + 1,] = c(mut_rate, size_a, size_b, 0, res$p.value, as.numeric(re
    }
  }
  df_wilcox$p_value_corrected = p.adjust(df_wilcox$p_value_raw, method = 'holm')
  df_wilcox$less_0.01 = df_wilcox$p_value_corrected < 0.01
  print(paste0('Germ mutation rate: ', mut_rate))
  print(df_wilcox)
}
```

```
## [1] "Germ mutation rate: 0.01"
##    germ_mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1           0.01     16     32      1.161192e-21 1.161192e-22  990.0      TRUE
## 2           0.01     16     64      1.990837e-31 1.484433e-32  137.0      TRUE
## 3           0.01     16    128      2.032847e-30 1.694039e-31  221.0      TRUE
## 4           0.01     16    256      1.721090e-07 5.736966e-08 2778.5      TRUE
## 5           0.01     16    512      1.237738e-13 2.062896e-14 8130.0      TRUE
## 6           0.01     32     64      4.401194e-15 5.501492e-16 1684.5      TRUE
```

```
## 7           0.01     32    128      1.423615e-13 2.847230e-14 1887.0      TRUE
## 8           0.01     32    256      2.438849e-01 1.219425e-01 5633.5     FALSE
## 9           0.01     32    512      5.140604e-27 4.673276e-28 9495.0      TRUE
## 10          0.01     64    128      9.221418e-01 9.221418e-01 5040.5     FALSE
## 11          0.01     64    256      4.110744e-14 5.872491e-15 8195.5      TRUE
## 12          0.01     64    512      6.122051e-32 4.081368e-33 9907.0      TRUE
## 13          0.01    128    256      5.020912e-13 1.255228e-13 8033.5      TRUE
## 14          0.01    128    512      1.990837e-31 1.422026e-32 9864.5      TRUE
## 15          0.01    256    512      1.882508e-17 2.091675e-18 8582.5      TRUE
## [1] "Germ mutation rate: 0.02"
##    germ_mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1           0.02     16     32      5.385908e-24 5.385908e-25  773.5      TRUE
## 2           0.02     16     64      3.620092e-31 2.585780e-32  156.0      TRUE
## 3           0.02     16    128      5.876058e-29 4.896715e-30  339.5      TRUE
## 4           0.02     16    256      7.355430e-06 2.451810e-06 3071.0      TRUE
## 5           0.02     16    512      2.935849e-18 3.669812e-19 8662.0      TRUE
## 6           0.02     32     64      5.800574e-18 8.286535e-19 1375.0      TRUE
## 7           0.02     32    128      4.715120e-12 1.178780e-12 2090.5      TRUE
## 8           0.02     32    256      4.080762e-04 2.040381e-04 6520.5      TRUE
## 9           0.02     32    512      6.645814e-27 6.041649e-28 9485.5      TRUE
## 10          0.02     64    128      2.889472e-01 2.889472e-01 5434.5     FALSE
## 11          0.02     64    256      2.039804e-17 3.399674e-18 8560.0      TRUE
## 12          0.02     64    512      4.109271e-32 2.739514e-33 9920.5      TRUE
## 13          0.02    128    256      2.514342e-14 5.028683e-15 8203.5      TRUE
## 14          0.02    128    512      4.123142e-31 3.171647e-32 9837.0      TRUE
## 15          0.02    256    512      4.866893e-20 5.407659e-21 8848.0      TRUE
## [1] "Germ mutation rate: 0.05"
##    germ_mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1           0.05     16     32      1.591362e-24 1.768180e-25  730.0      TRUE
## 2           0.05     16     64      1.063762e-30 8.864684e-32  198.5      TRUE
## 3           0.05     16    128      3.321119e-21 4.151399e-22 1043.0      TRUE
## 4           0.05     16    256      2.538532e-02 1.269266e-02 3979.5     FALSE
## 5           0.05     16    512      1.050337e-26 9.548517e-28 9468.5      TRUE
## 6           0.05     32     64      3.387540e-14 5.645899e-15 1802.5      TRUE
## 7           0.05     32    128      1.306936e-05 4.356453e-06 3119.5      TRUE
## 8           0.05     32    256      1.528740e-07 3.821850e-08 7251.0      TRUE
## 9           0.05     32    512      3.162116e-32 2.258654e-33 9927.0      TRUE
## 10          0.05     64    128      1.546546e-01 1.546546e-01 5583.0     FALSE
## 11          0.05     64    256      4.390965e-19 6.272808e-20 8741.0      TRUE
## 12          0.05     64    512      6.208612e-33 4.139074e-34 9984.0      TRUE
## 13          0.05    128    256      2.838701e-12 5.677403e-13 7950.5      TRUE
## 14          0.05    128    512      9.845016e-32 7.573090e-33 9886.0      TRUE
## 15          0.05    256    512      3.142822e-26 3.142822e-27 9424.0      TRUE
## [1] "Germ mutation rate: 0.1"
##    germ_mut_rate size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1            0.1     16     32      2.006447e-24 2.229385e-25  739.0      TRUE
```

```
## 2            0.1     16     64      2.197505e-25 1.997732e-26  646.0      TRUE
## 3            0.1     16    128      3.982057e-19 6.636762e-20 1261.5      TRUE
## 4            0.1     16    256      2.853915e-06 9.513050e-07 7006.5      TRUE
## 5            0.1     16    512      1.146029e-26 9.550238e-28 9468.5      TRUE
## 6            0.1     32     64      6.866683e-07 1.716671e-07 2860.0      TRUE
## 7            0.1     32    128      5.714627e-02 5.714627e-02 4221.0     FALSE
## 8            0.1     32    256      7.451552e-21 9.314440e-22 8923.0      TRUE
## 9            0.1     32    512      1.091653e-31 7.797522e-33 9885.0      TRUE
## 10           0.1     64    128      2.618271e-02 1.309135e-02 6016.0     FALSE
## 11           0.1     64    256      6.893655e-25 6.893655e-26 9306.5      TRUE
## 12           0.1     64    512      4.295636e-32 2.863757e-33 9919.0      TRUE
## 13           0.1    128    256      9.294756e-21 1.327822e-21 8908.0      TRUE
## 14           0.1    128    512      2.475810e-31 1.904469e-32 9854.5      TRUE
## 15           0.1    256    512      7.440793e-16 1.488159e-16 8380.0      TRUE
## [1] "Germ mutation rate: 0.2"
##    germ_mut_rate size_a size_b p_value_corrected  p_value_raw       W less_0.01
## 1            0.2     16     32      6.711164e-17 9.587377e-18  1488.5      TRUE
## 2            0.2     16     64      2.652853e-08 5.305706e-09  2610.5      TRUE
## 3            0.2     16    128      5.723537e-01 4.033561e-01  4657.5     FALSE
## 4            0.2     16    256      9.414689e-28 1.046077e-28  9550.0      TRUE
## 5            0.2     16    512      3.841700e-33 2.561134e-34 10000.0      TRUE
## 6            0.2     32     64      5.723537e-01 2.861769e-01  5437.0     FALSE
## 7            0.2     32    128      2.713788e-06 6.784470e-07  7033.5      TRUE
## 8            0.2     32    256      2.557355e-30 2.324869e-31  9768.0      TRUE
## 9            0.2     32    512      3.841700e-33 2.561422e-34 10000.0      TRUE
## 10           0.2     64    128      8.967634e-04 2.989211e-04  6480.5      TRUE
## 11           0.2     64    256      3.597156e-29 3.597156e-30  9671.5      TRUE
## 12           0.2     64    512      3.841700e-33 2.561422e-34 10000.0      TRUE
## 13           0.2    128    256      3.256203e-24 4.070254e-25  9237.5      TRUE
## 14           0.2    128    512      1.052551e-31 8.771259e-33  9881.0      TRUE
## 15           0.2    256    512      1.558734e-09 2.597889e-10  7587.5      TRUE
## [1] "Germ mutation rate: 0.5"
##    germ_mut_rate size_a size_b p_value_corrected  p_value_raw       W less_0.01
## 1            0.5     16     32      3.627488e-11 7.254975e-12  2195.0      TRUE
## 2            0.5     16     64      1.774145e-01 1.774145e-01  5552.5     FALSE
## 3            0.5     16    128      9.003159e-21 1.125395e-21  8915.0      TRUE
## 4            0.5     16    256      3.840402e-33 2.560268e-34 10000.0      TRUE
## 5            0.5     16    512      3.840402e-33 2.560412e-34 10000.0      TRUE
## 6            0.5     32     64      1.574642e-07 5.248808e-08  7228.0      TRUE
## 7            0.5     32    128      3.547680e-25 3.941867e-26  9328.0      TRUE
## 8            0.5     32    256      3.840402e-33 2.560701e-34 10000.0      TRUE
## 9            0.5     32    512      3.840402e-33 2.560845e-34 10000.0      TRUE
## 10           0.5     64    128      4.292292e-17 6.131846e-18  8532.5      TRUE
## 11           0.5     64    256      3.128938e-32 3.128938e-33  9916.0      TRUE
## 12           0.5     64    512      1.333109e-32 1.211917e-33  9948.0      TRUE
## 13           0.5    128    256      2.868826e-09 7.172065e-10  7522.5      TRUE
```

```
## 14           0.5    128    512      9.393819e-14 1.565636e-14  8144.5      TRUE
## 15           0.5    256    512      3.381624e-03 1.690812e-03  6285.5      TRUE
## [1] "Germ mutation rate: 1"
##    germ_mut_rate size_a size_b p_value_corrected  p_value_raw       W less_0.01
## 1              1     16     32      1.080497e-01 5.402483e-02  5789.0     FALSE
## 2              1     16     64      2.560330e-24 2.844811e-25  9251.5      TRUE
## 3              1     16    128      3.840402e-33 2.560268e-34 10000.0      TRUE
## 4              1     16    256      3.840402e-33 2.887894e-34  9996.0      TRUE
## 5              1     16    512      3.840402e-33 2.560412e-34 10000.0      TRUE
## 6              1     32     64      1.004804e-19 1.674674e-20  8799.0      TRUE
## 7              1     32    128      8.265949e-32 8.265949e-33  9883.0      TRUE
## 8              1     32    256      7.190219e-32 6.536563e-33  9891.0      TRUE
## 9              1     32    512      5.842919e-32 4.869099e-33  9901.0      TRUE
## 10             1     64    128      1.007238e-18 2.014476e-19  8689.0      TRUE
## 11             1     64    256      7.963405e-23 1.137629e-23  9105.0      TRUE
## 12             1     64    512      5.680932e-23 7.101164e-24  9124.0      TRUE
## 13             1    128    256      1.357430e-02 3.393576e-03  6199.5     FALSE
## 14             1    128    512      3.704384e-02 1.234795e-02  6024.5     FALSE
## 15             1    256    512      4.892624e-01 4.892624e-01  4716.5     FALSE
```

Chapter 5

Genome Length Sweep

By default, all genomes are bitstrings with 100 bits. Here, we look into the effects of varying this genome length (we use values 25, 50, 100, 200, and 400 bits).

The configuration script and data for the experiment can be found under `2021_02_27__genome_length/` in the experiments directory of the git repository.

5.1 Data cleaning

Load necessary libraries

```
library(dplyr)
library(ggplot2)
library(ggridges)
library(scales)
library(khroma)
```

Load the data and trim all the unecessay bits (*e.g.*, we initially ran sizes 8x8, 1024x1024 but cut them from the paper to make plots easier to read).

```
# Load the data
df = read.csv('../experiments/2021_02_27__genome_length/evolution/data/scraped_evolution_data_len
df = rbind(df, read.csv('../experiments/2021_02_27__genome_length/evolution/data/scraped_evolutio
df = rbind(df, read.csv('../experiments/2021_02_27__genome_length/evolution/data/scraped_evolutio
df = rbind(df, read.csv('../experiments/2021_02_27__genome_length/evolution/data/scraped_evolutio
df = rbind(df, read.csv('../experiments/2021_02_27__genome_length/evolution/data/scraped_evolutio
# Trim off NAs (artifiacts of how we scraped the data) and trim to only have gen 10,000
df2 = df[!is.na(df$MCSIZE) & df$generation == 10000,]
# Ignore data for size 8x8 and 1024x1024
df2 = df2[df2$MCSIZE != 8 & df2$MCSIZE != 1024,]
```

We group and summarize the data to make to ensure all replicates are present.

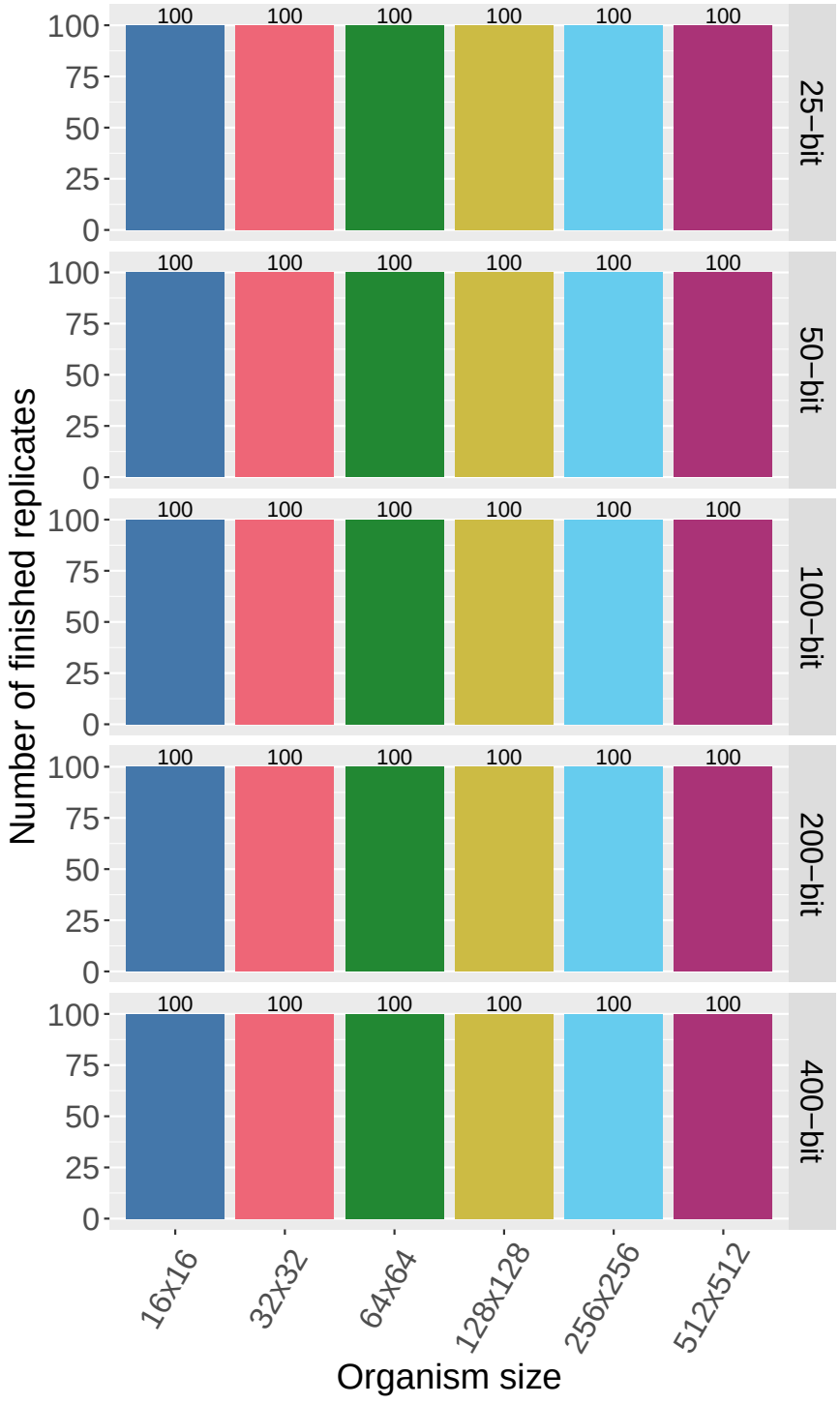
```
# Group the data by size and summarize
data_grouped = dplyr::group_by(df2, MCSIZE, LENGTH)
data_summary = dplyr::summarize(data_grouped, mean_ones = mean(ave_ones), n = dplyr::n
```

We clean the data and create a few helper variables to make plotting easier.

```
## Set variables to make plotting easier
# Calculate restraint value (x - 60% of the genome length)
df2$restraint_value = df2$ave_ones - (df2$LENGTH * 0.6)
# Make a nice, clean factor for size
df2$size_str = paste0(df2$MCSIZE, 'x', df2$MCSIZE)
df2$size_factor = factor(df2$size_str, levels = c('16x16', '32x32', '64x64', '128x128'
df2$size_factor_reversed = factor(df2$size_str, levels = rev(c('16x16', '32x32', '64x6
df2$length_str = paste0(df2$LENGTH, '-bit')
df2$length_factor = factor(df2$length_str, levels = c('25-bit', '50-bit', '100-bit', '
data_summary$size_str = paste0(data_summary$MCSIZE, 'x', data_summary$MCSIZE)
data_summary$size_factor = factor(data_summary$size_str, levels = c('16x16', '32x32',
data_summary$length_str = paste0(data_summary$LENGTH, '-bit')
data_summary$length_factor = factor(data_summary$length_str, levels = c('25-bit', '50-
# Create a map of colors we'll use to plot the different organism sizes
color_vec = as.character(khroma::color('bright')(7))
color_map = c(
  '16x16' =     color_vec[1],
  '32x32' =     color_vec[2],
  '64x64' =     color_vec[3],
  '128x128' =   color_vec[4],
  '256x256' =   color_vec[5],
  '512x512' =   color_vec[6],
  '1024x1024' = color_vec[7]
)
# Set the sizes for text in plots
text_major_size = 18
text_minor_size = 16
```

5.2 Data integrity check

Now we plot the number of finished replicates for each treatment to make sure all data are present. Each row shows a different genome length (in bits). Each

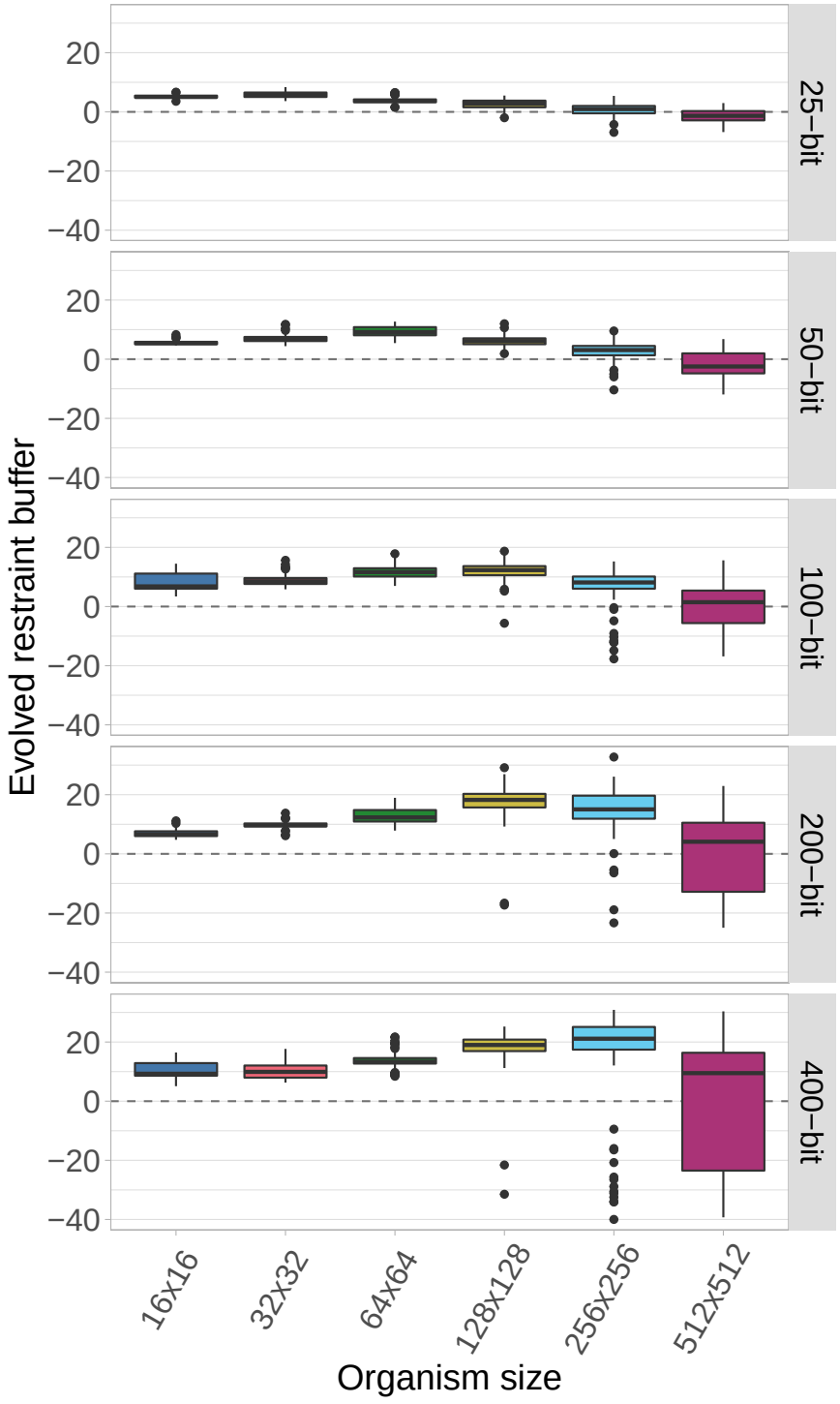


bar/color shows a different organism size.

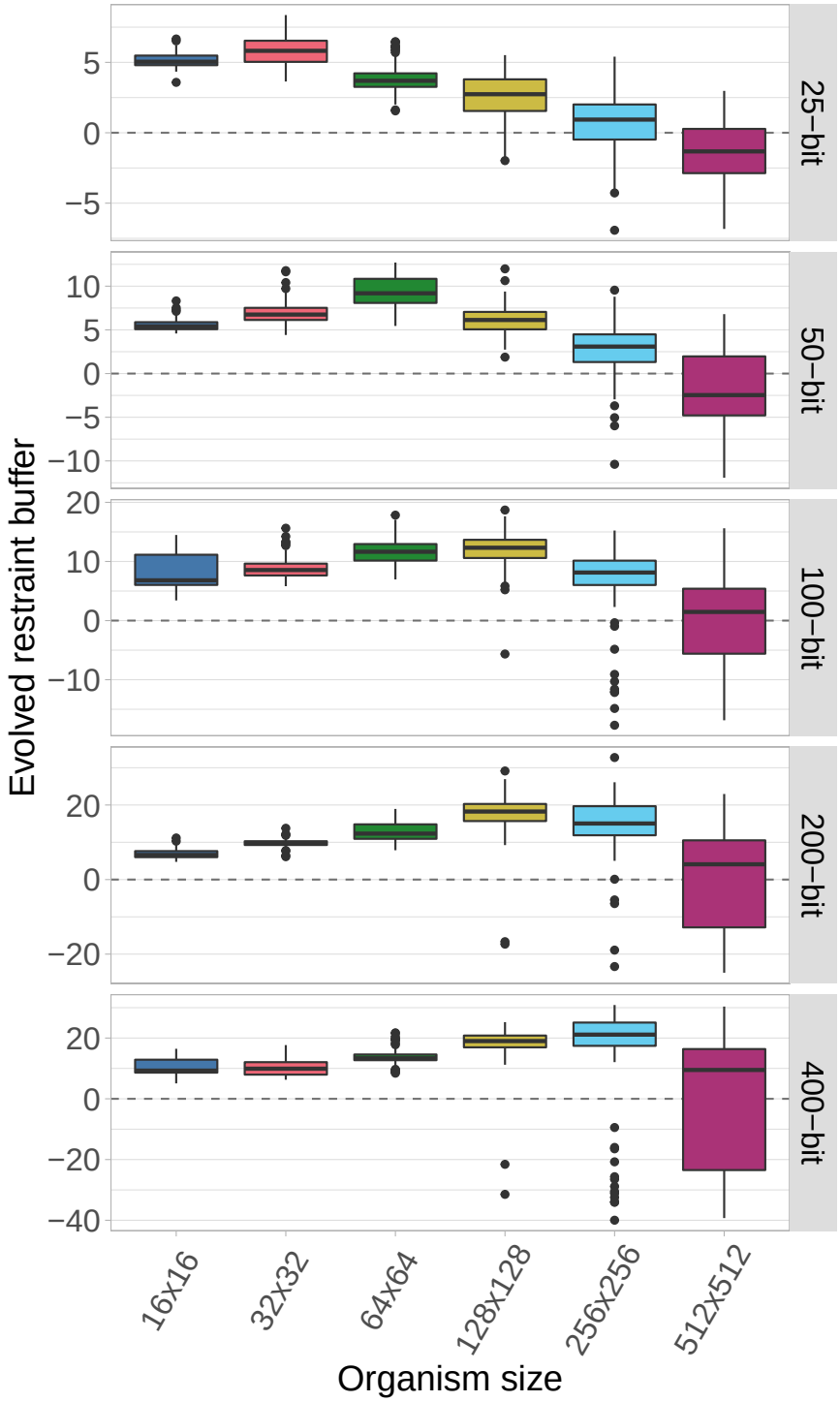
5.3 Aggregate plots

5.3.1 Facet by genome length

Here we plot all the data at once. Each row showing a different somatic mutation rate and each boxplot shows a given organism size.

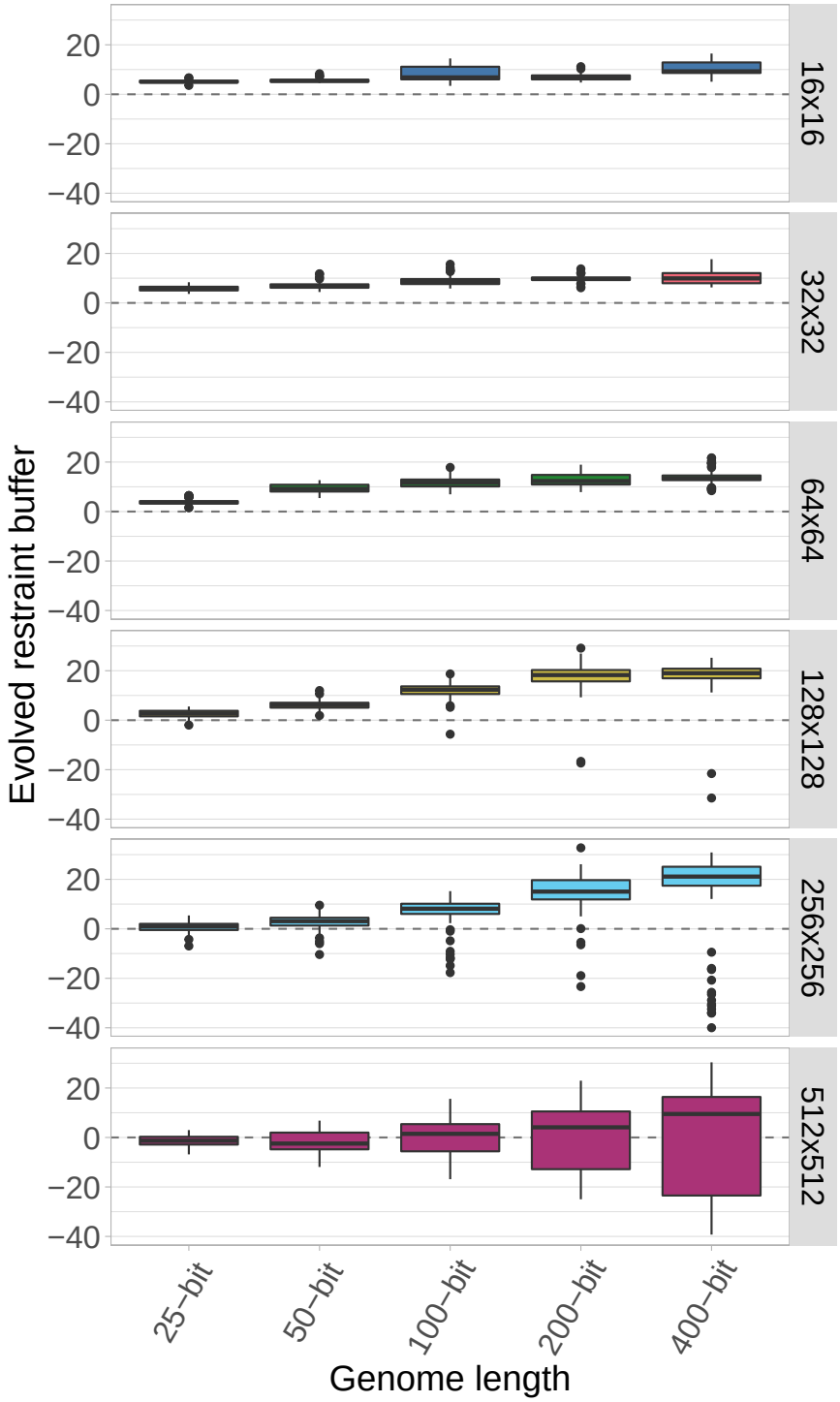

Evolved restraint buffer
20
0
−20
−40
25−bit
50−bit
100−bit
200−bit
400−bit
16x16
32x32
64x64
128x128
256x256
512x512
Organism size

Here we plot the same data, only we allow the y-axis to vary between rows.

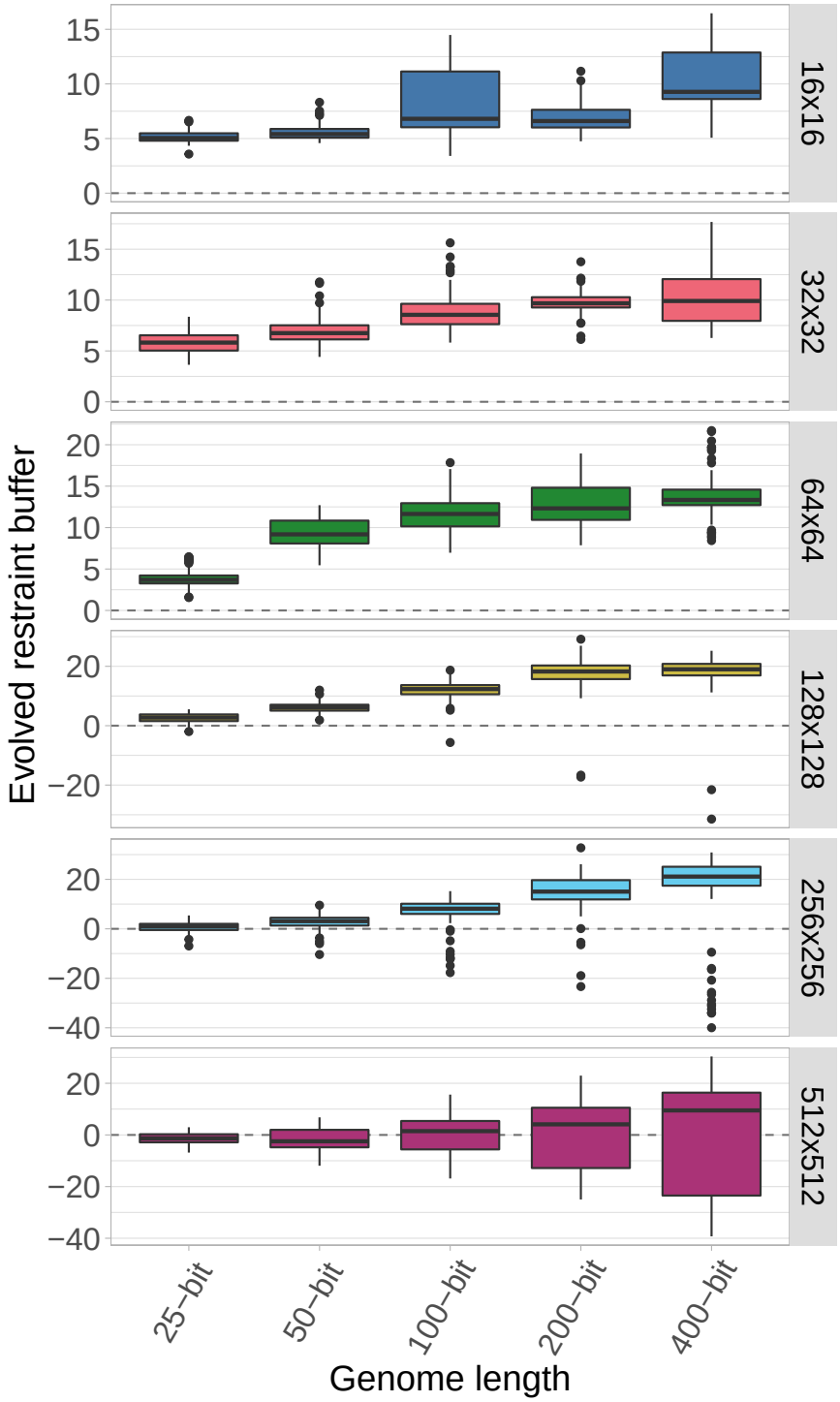

Evolved restraint buffer
25−bit
50−bit
100−bit
200−bit
400−bit
16x16
32x32
64x64
128x128
256x256
512x512
Organism size

5.3.2 Facet by organism size

Here we plot the same data again, only now each row shows an organims size while genome length varies on the x-axis.


Evolved restraint buffer
20
0
−20
−40
16x16
32x32
64x64
128x128
256x256
512x512
25−bit
50−bit
100−bit
200−bit
400−bit
Genome length

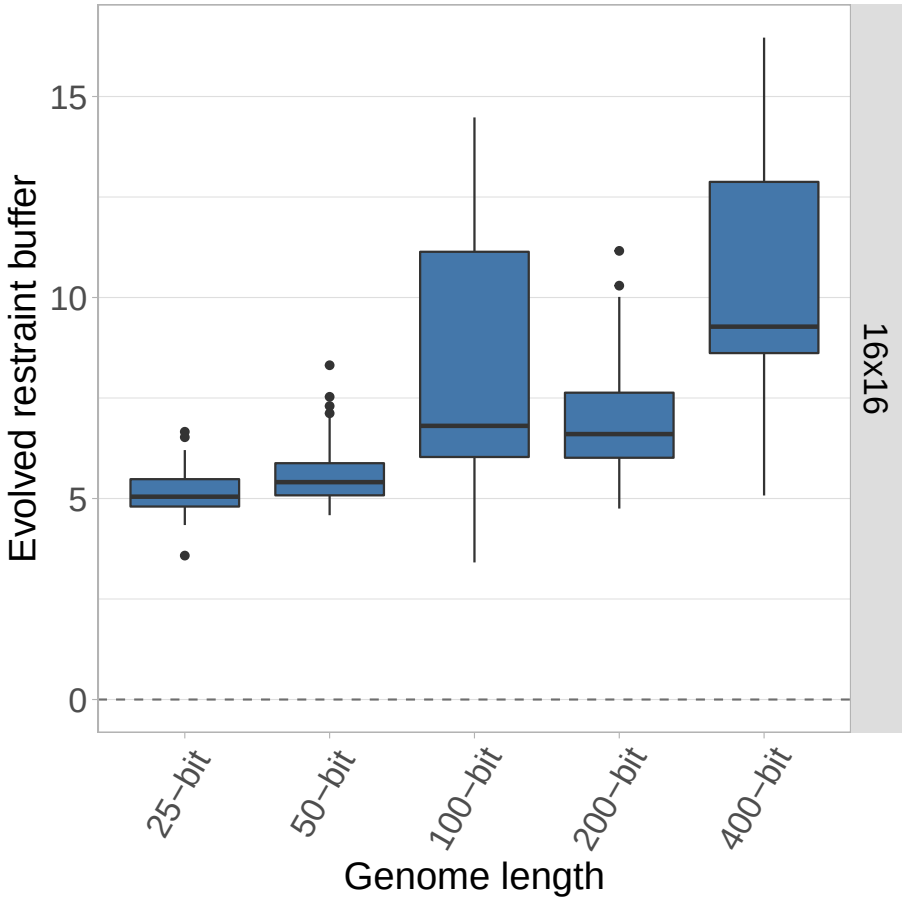
Here is the identical plot but now we allow the y-axis to vary between the rows.


Evolved restraint buffer
16x16
32x32
64x64
128x128
256x256
512x512
25−bit
50−bit
100−bit
200−bit
400−bit
Genome length

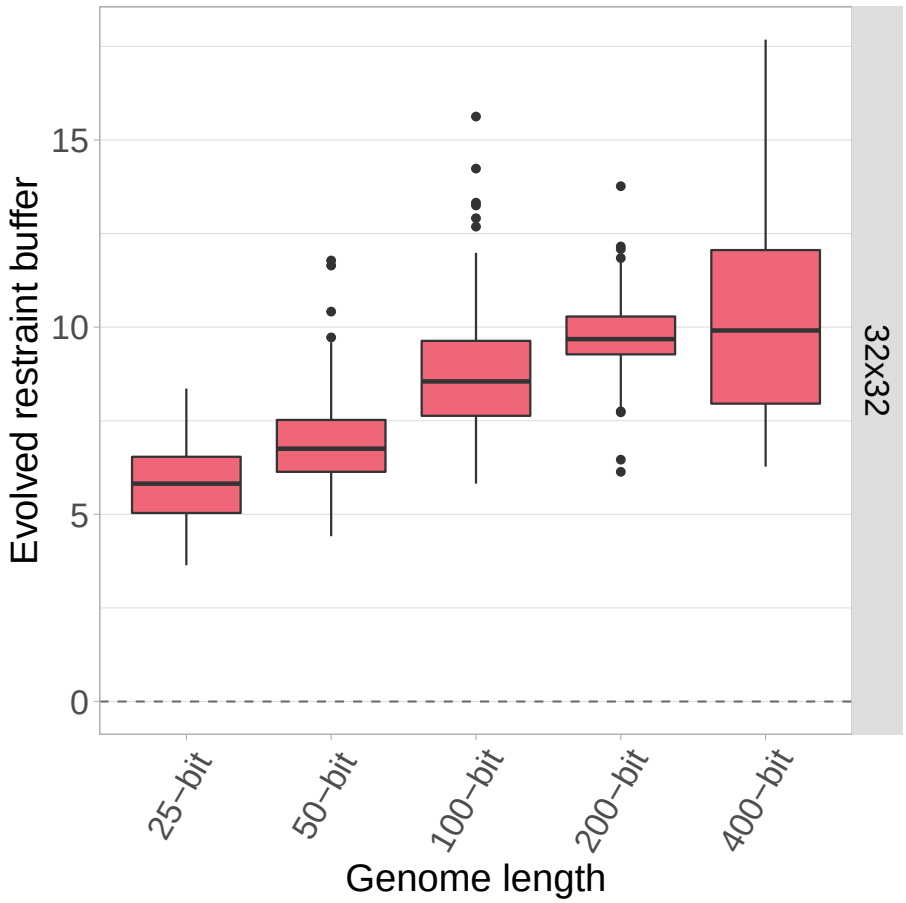
5.4 Single organism size plots

Here we plot each organism size independently, with the genome length on the x-axis.

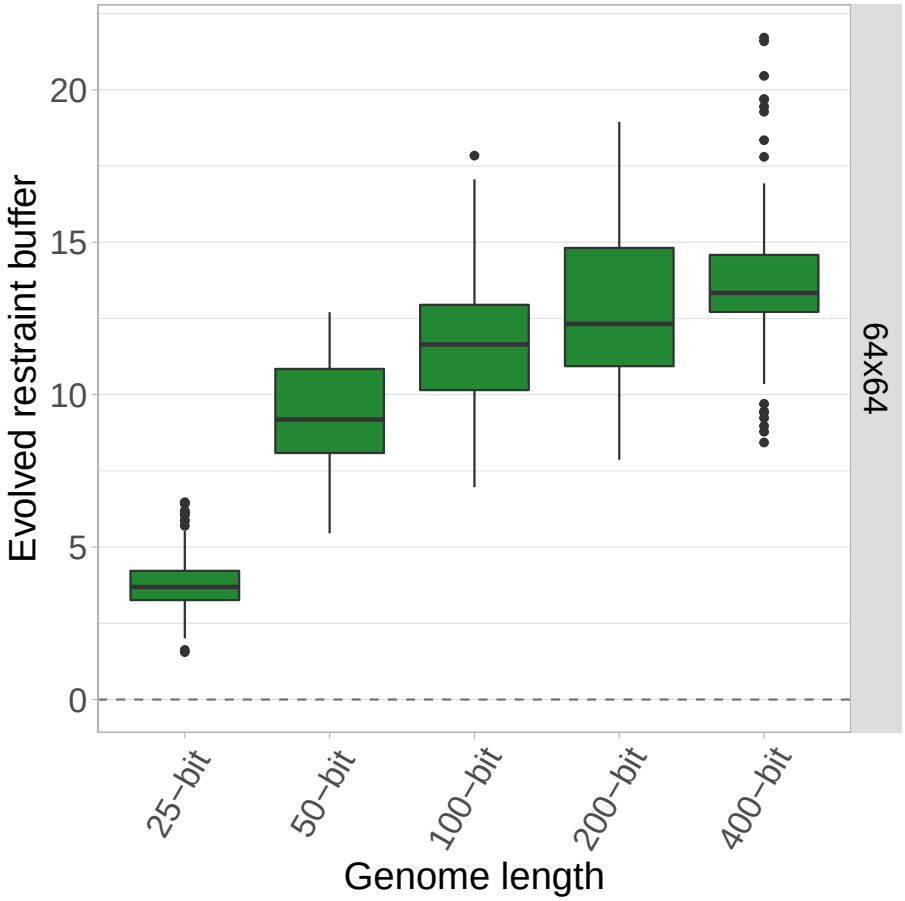
5.4.1 Organism size 16x16



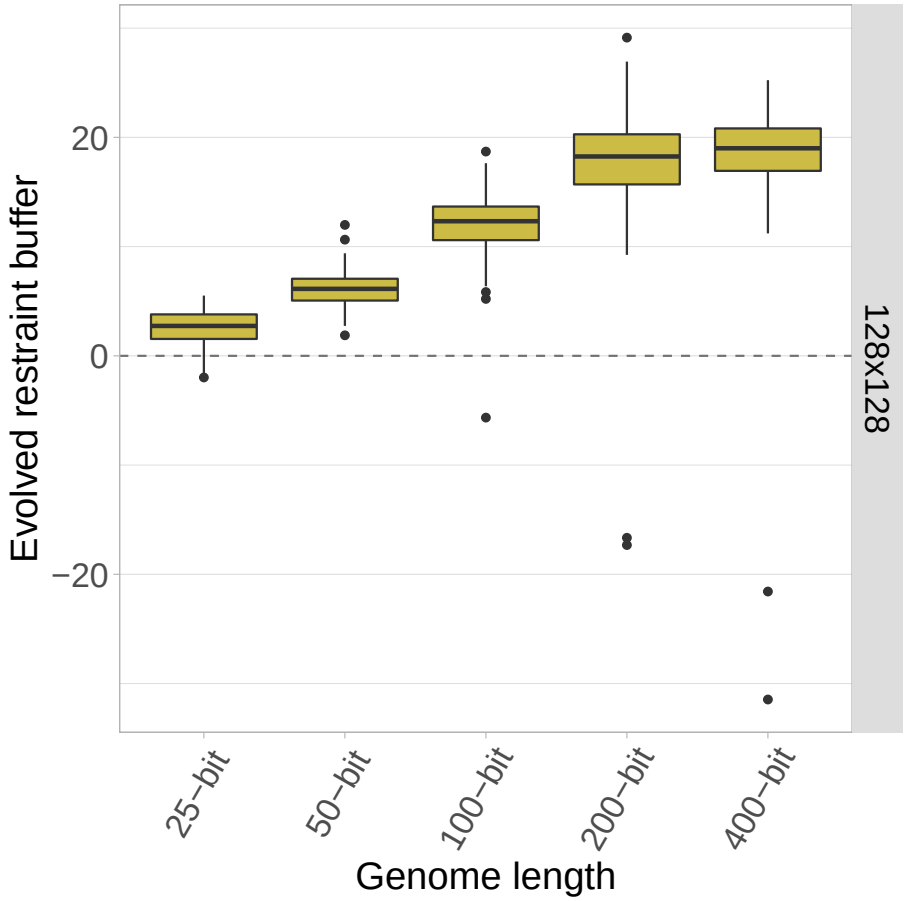
5.4.2 Organism size 32x32



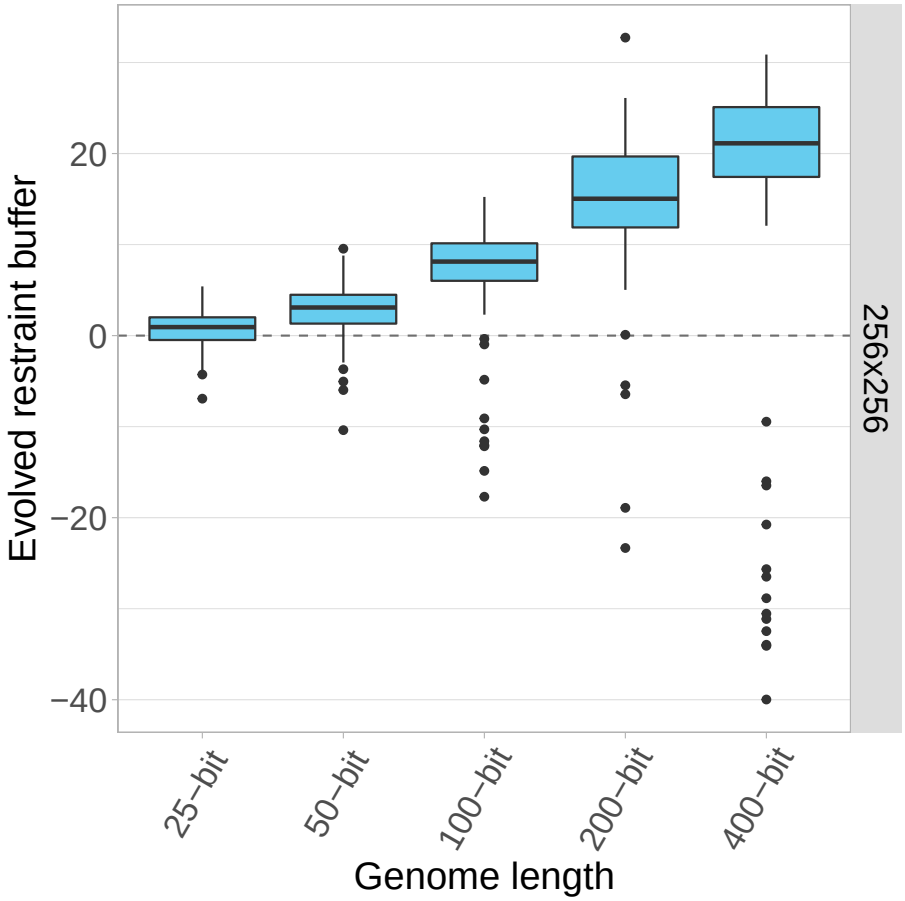
5.4.3 Organism size 64x64



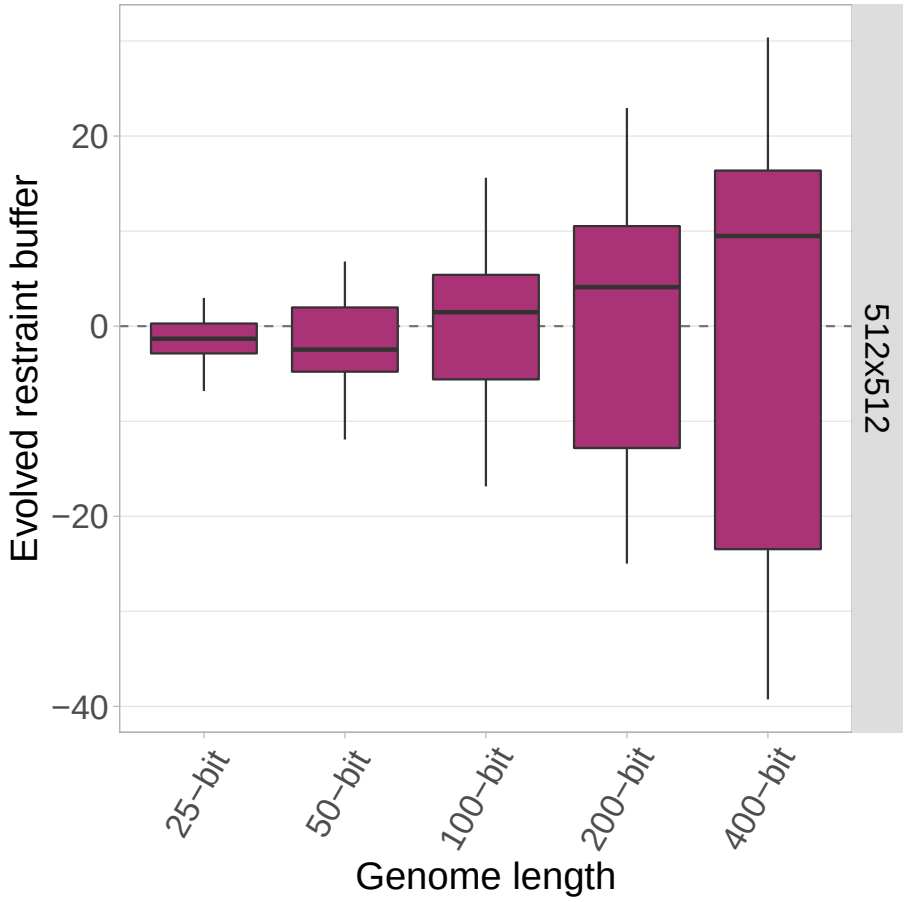
5.4.4 Organism size 128x128



5.4.5 Organism size 256x256



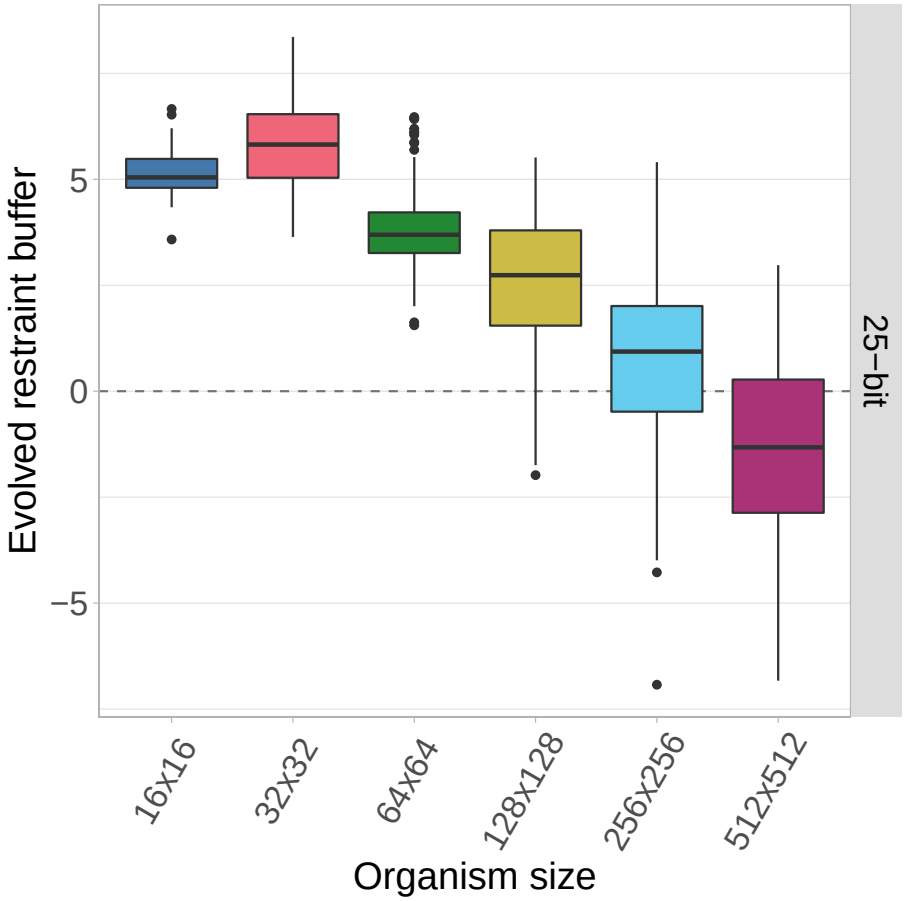
5.4.6 Organism size 512x512



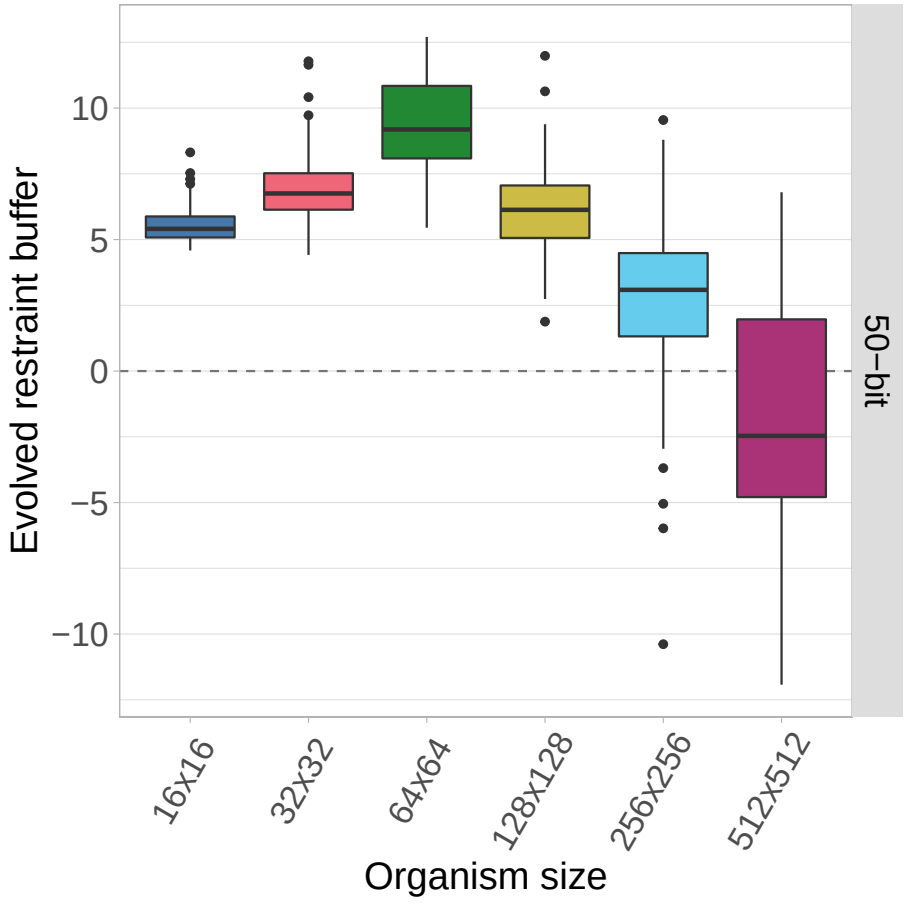
5.5 Single genome length plots

Here we plot each genome length independently, with the organism size on the x-axis.

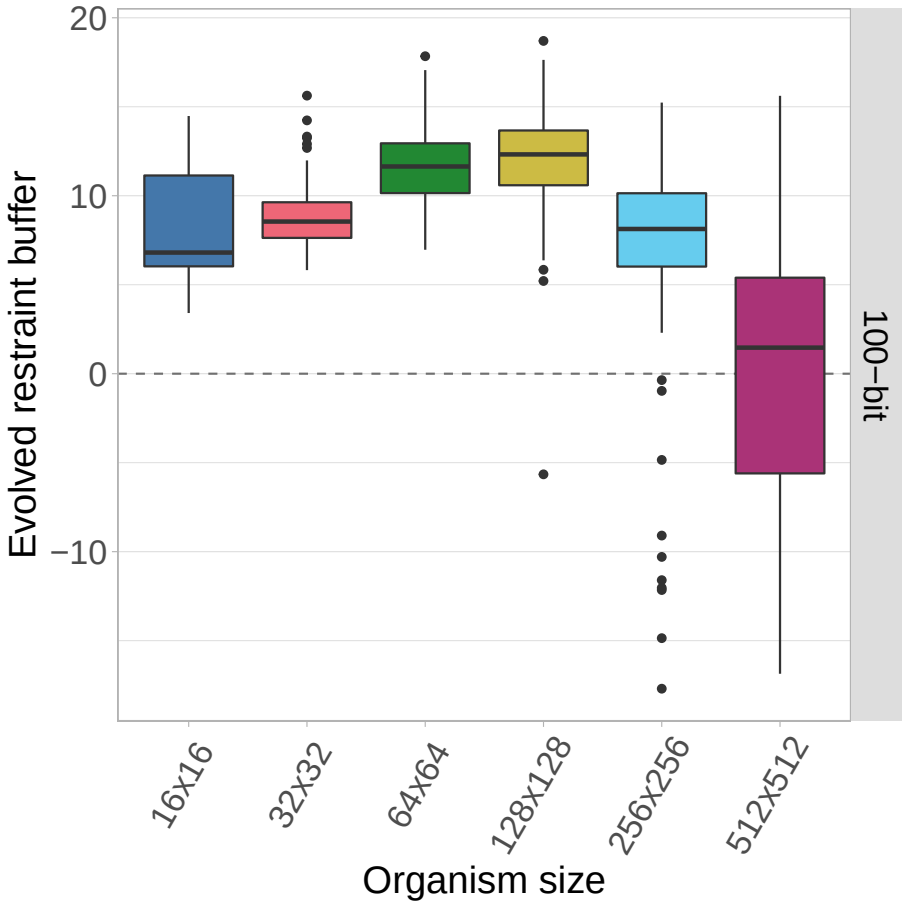
5.5.1 25-bit genomes



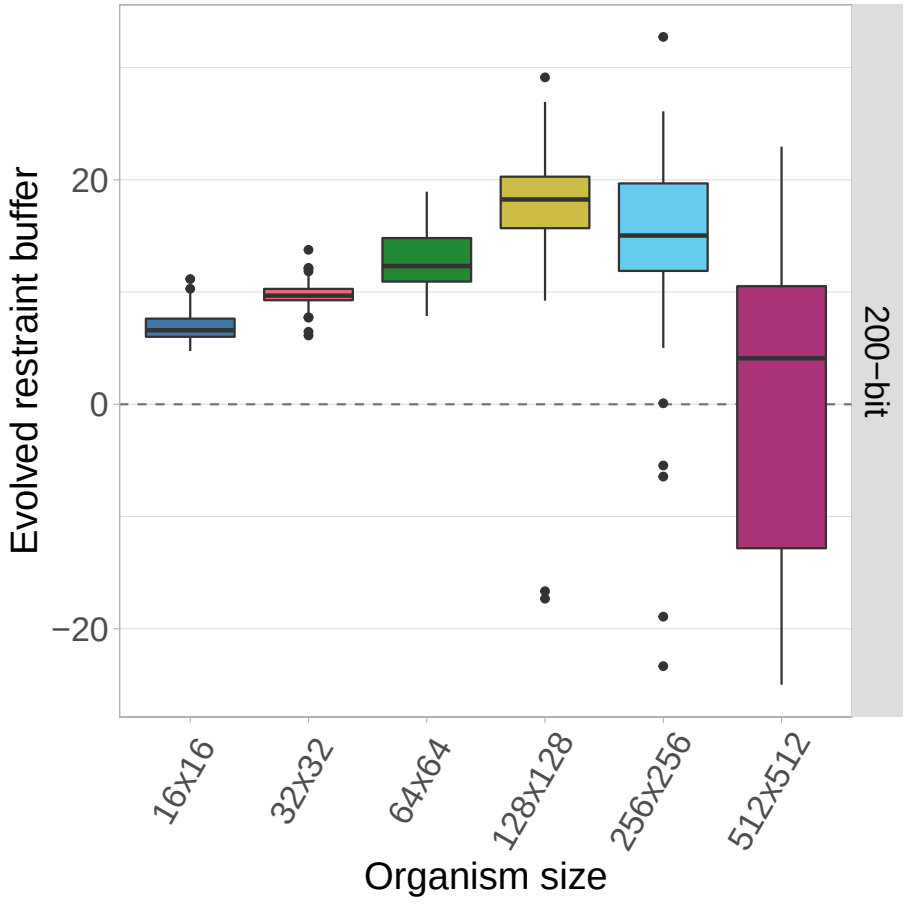
5.5.2 50-bit genomes



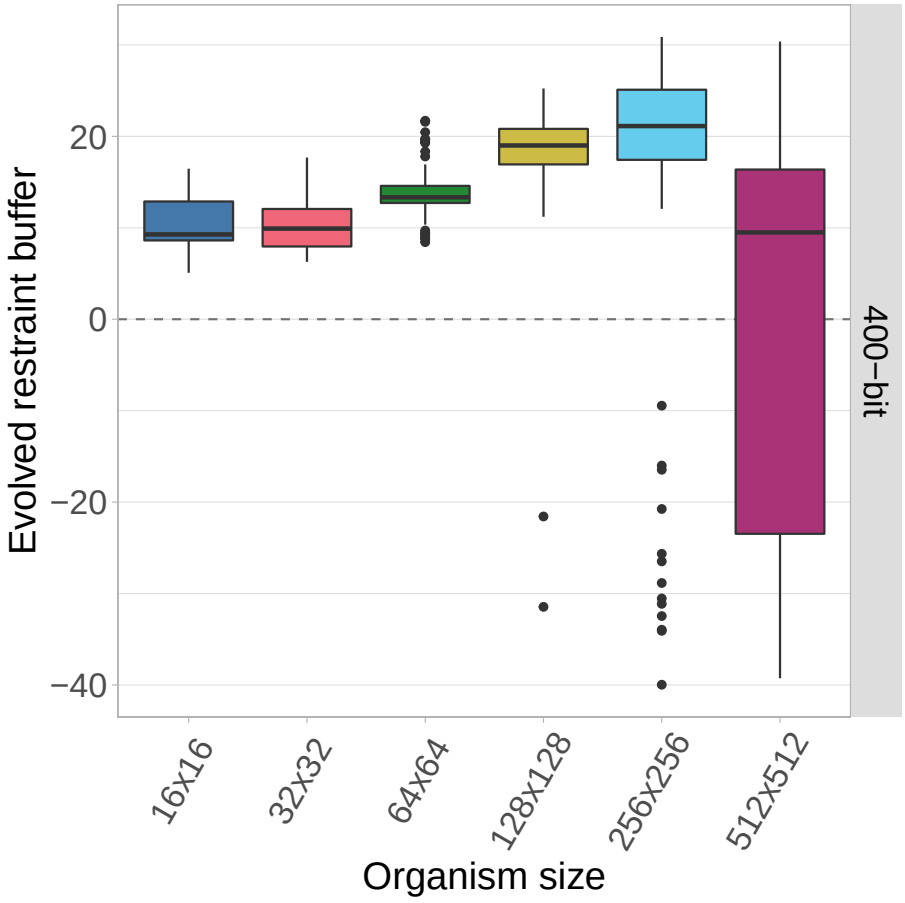
5.5.3 100-bit genomes



5.5.4 200-bit genomes



5.5.5 400-bit genomes



5.6 Statistics

Since organism size is our main point of comparison, we calculate stats for each genome length.

First, we perform a Kruskal-Wallis test across all organism sizes to indicate if variance exists at that mutation rate. If variance exists, we then perfrm a pairwise Wilcoxon Rank-Sum test to show which pairs of organism sizes significantly differ. Finally, we perform Bonferroni-Holm corrections for multiple comparisons.

```
length_vec = c(25, 50, 100, 200, 400)
df_kruskal = data.frame(data = matrix(nrow = 0, ncol = 4))
colnames(df_kruskal) = c('genome_length', 'p_value', 'chi_squared', 'df')
for(genome_length in length_vec){
```

```
    df_test = df2[df2$LENGTH == genome_length,]
    res = kruskal.test(df_test$restraint_value ~ df_test$MCSIZE, df_test)
    df_kruskal[nrow(df_kruskal) + 1,] = c(genome_length, res$p.value, as.numeric(res$statistic)[1
  }
  df_kruskal$less_0.01 = df_kruskal$p_value < 0.01
  print(df_kruskal)
```

```
##   genome_length      p_value chi_squared df less_0.01
## 1            25 1.508889e-97    461.6473  5      TRUE
## 2            50 9.772852e-87    411.5159  5      TRUE
## 3           100 7.491319e-60    286.6294  5      TRUE
## 4           200 1.626963e-75    359.4358  5      TRUE
## 5           400 2.857912e-49    237.3380  5      TRUE
```

We see that significant variation exists within each genome length, so we perform pariwise Wilcoxon tests on each to see which pais of sizes are significantly different.

```
size_vec = c(16, 32, 64, 128, 256, 512)
length_vec = c(25, 50, 100, 200, 400)
for(genome_length in length_vec){
  df_test = df2[df2$LENGTH == genome_length,]
  df_wilcox = data.frame(data = matrix(nrow = 0, ncol = 6))
  colnames(df_wilcox) = c('genome_length', 'size_a', 'size_b', 'p_value_corrected', 'p_value_raw'
  for(size_idx_a in 1:(length(size_vec) - 1)){
    size_a = size_vec[size_idx_a]
    for(size_idx_b in (size_idx_a + 1):length(size_vec)){
      size_b = size_vec[size_idx_b]
      res = wilcox.test(df_test[df_test$MCSIZE == size_a,]$restraint_value, df_test[df_test$MCSIZ
      df_wilcox[nrow(df_wilcox) + 1,] = c(genome_length, size_a, size_b, 0, res$p.value, as.numer
    }
  }
  df_wilcox$p_value_corrected = p.adjust(df_wilcox$p_value_raw, method = 'holm')
  df_wilcox$less_0.01 = df_wilcox$p_value_corrected < 0.01
  print(paste0('Genome length: ', genome_length))
  print(df_wilcox)
}
```

```
## [1] "Genome length: 25"
##    genome_length size_a size_b p_value_corrected  p_value_raw       W less_0.01
## 1             25     16     32      2.337475e-07 2.337475e-07  2883.5      TRUE
## 2             25     16     64      6.069986e-18 1.213997e-18  8607.5      TRUE
## 3             25     16    128      1.663209e-24 2.376012e-25  9258.5      TRUE
## 4             25     16    256      6.203828e-32 5.639844e-33  9896.0      TRUE
## 5             25     16    512      3.838888e-33 2.559259e-34 10000.0      TRUE
## 6             25     32     64      1.447210e-22 2.412016e-23  9074.5      TRUE
## 7             25     32    128      1.283820e-27 1.283820e-28  9542.5      TRUE
```

```
## 8             25     32    256      2.711275e-32 2.259396e-33  9927.0      TRUE
## 9             25     32    512      3.838888e-33 2.560557e-34 10000.0      TRUE
## 10            25     64    128      1.187343e-07 5.936715e-08  7219.0      TRUE
## 11            25     64    256      2.378014e-26 2.642238e-27  9430.5      TRUE
## 12            25     64    512      1.298354e-32 9.987336e-34  9954.5      TRUE
## 13            25    128    256      1.433015e-10 3.582536e-11  7710.0      TRUE
## 14            25    128    512      7.524923e-25 9.406154e-26  9294.5      TRUE
## 15            25    256    512      4.120336e-10 1.373445e-10  7627.5      TRUE
## [1] "Genome length: 50"
##    genome_length size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1             50     16     32      2.588989e-15 5.177978e-16 1681.5      TRUE
## 2             50     16     64      2.694326e-30 2.072558e-31  228.0      TRUE
## 3             50     16    128      3.224011e-03 3.224011e-03 3794.0      TRUE
## 4             50     16    256      4.089878e-15 1.022470e-15 8284.5      TRUE
## 5             50     16    512      1.182991e-27 1.182991e-28 9545.5      TRUE
## 6             50     32     64      1.797027e-17 2.567181e-18 1427.0      TRUE
## 7             50     32    128      7.415731e-04 3.707866e-04 6457.5      TRUE
## 8             50     32    256      1.165570e-21 1.295078e-22 9005.5      TRUE
## 9             50     32    512      4.567933e-31 3.262810e-32 9836.0      TRUE
## 10            50     64    128      2.265727e-21 2.832159e-22 8973.0      TRUE
## 11            50     64    256      8.866606e-30 7.388839e-31 9727.5      TRUE
## 12            50     64    512      7.213069e-33 4.808713e-34 9979.0      TRUE
## 13            50    128    256      4.869262e-16 8.115436e-17 8409.5      TRUE
## 14            50    128    512      4.777498e-28 4.343180e-29 9582.0      TRUE
## 15            50    256    512      3.236734e-12 1.078911e-12 7914.5      TRUE
## [1] "Genome length: 100"
##    genome_length size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1            100     16     32      7.697389e-02 1.924347e-02 4041.5     FALSE
## 2            100     16     64      3.952168e-13 7.904337e-14 1941.5      TRUE
## 3            100     16    128      2.398968e-14 2.998710e-15 1770.0      TRUE
## 4            100     16    256      4.158441e-01 4.158441e-01 5333.5     FALSE
## 5            100     16    512      5.614119e-18 4.678432e-19 8651.0      TRUE
## 6            100     32     64      1.034976e-13 1.478537e-14 1852.5      TRUE
## 7            100     32    128      3.085548e-17 3.085548e-18 1435.5      TRUE
## 8            100     32    256      1.117541e-01 3.725137e-02 5853.0     FALSE
## 9            100     32    512      2.483010e-22 1.910008e-23 9084.0      TRUE
## 10           100     64    128      1.117541e-01 4.890986e-02 4193.5     FALSE
## 11           100     64    256      5.002561e-15 5.558401e-16 8315.0      TRUE
## 12           100     64    512      5.082595e-28 3.388396e-29 9591.0      TRUE
## 13           100    128    256      1.590814e-17 1.446195e-18 8599.5      TRUE
## 14           100    128    512      9.444159e-28 6.745828e-29 9566.0      TRUE
## 15           100    256    512      2.634367e-13 4.390611e-14 8090.0      TRUE
## [1] "Genome length: 200"
##    genome_length size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1            200     16     32      4.663546e-26 3.886289e-27  584.0      TRUE
## 2            200     16     64      7.523609e-32 5.015739e-33  100.0      TRUE
```

```
## 3             200     16    128      2.146093e-30 1.532923e-31  217.5      TRUE
## 4             200     16    256      1.997886e-24 1.911300e-25  733.0      TRUE
## 5             200     16    512      9.462181e-03 9.462181e-03 6062.5      TRUE
## 6             200     32     64      1.344008e-20 1.493343e-21 1097.0      TRUE
## 7             200     32    128      5.645064e-28 4.342357e-29  418.0      TRUE
## 8             200     32    256      1.309572e-19 1.636965e-20 1200.0      TRUE
## 9             200     32    512      2.440723e-07 6.101808e-08 7217.0      TRUE
## 10            200     64    128      1.166719e-18 1.666742e-19 1302.5      TRUE
## 11            200     64    256      9.151807e-05 3.050602e-05 3293.0      TRUE
## 12            200     64    512      6.237644e-15 1.247529e-15 8274.5      TRUE
## 13            200    128    256      9.982635e-05 4.991318e-05 6660.5      TRUE
## 14            200    128    512      1.997886e-24 1.816260e-25 9269.0      TRUE
## 15            200    256    512      1.717006e-17 2.861676e-18 8568.0      TRUE
## [1] "Genome length: 400"
##    genome_length size_a size_b p_value_corrected  p_value_raw      W less_0.01
## 1             400     16     32      5.405382e-01 5.348472e-01 5254.5     FALSE
## 2             400     16     64      3.472338e-14 3.472338e-15 1777.5      TRUE
## 3             400     16    128      4.072814e-28 2.715209e-29  401.0      TRUE
## 4             400     16    256      1.163125e-16 9.692706e-18 1489.0      TRUE
## 5             400     16    512      5.405382e-01 1.801794e-01 5549.0     FALSE
## 6             400     32     64      5.784416e-12 8.263451e-13 2070.5      TRUE
## 7             400     32    128      1.489024e-26 1.063588e-27  535.5      TRUE
## 8             400     32    256      2.187697e-16 1.988815e-17 1523.0      TRUE
## 9             400     32    512      5.405382e-01 1.825748e-01 5546.0     FALSE
## 10            400     64    128      2.993077e-18 2.302367e-19 1317.0      TRUE
## 11            400     64    256      3.197608e-12 3.997010e-13 2030.0      TRUE
## 12            400     64    512      8.293488e-05 1.658698e-05 6763.0      TRUE
## 13            400    128    256      1.019174e-02 2.547935e-03 3764.5     FALSE
## 14            400    128    512      9.137039e-13 1.015227e-13 8045.0      TRUE
## 15            400    256    512      6.084030e-12 1.014005e-12 7918.0      TRUE
```

Chapter 6

Timing sample count experiment

By default, we calculated 100 timing samples for each combination of organism size and restraint buffer value to use for fitness in Primordium (a fresh batch for each experiment). With this experiment we showed that increasing this number to 10,000 has no qualitative difference on results. This was done by replicating the baseline experiment using 10,000 samples and comparing the results to a fresh run with 100 samples.

6.1 Data cleaning

Load necessary libraries

```
library(dplyr)
library(ggplot2)
library(ggridges)
library(scales)
library(khroma)
```

Load the data and trim all the unecessay bits (*e.g.*, we initially ran sizes 8x8, 1024x1024 but cut them from the paper to make plots easier to read).

```
# Load the data
df = read.csv('../experiments/2021_02_24__finite_10k_samples/evolution/data/scraped_evolution_dat
df = rbind(df, read.csv('../experiments/2021_02_24__finite_10k_samples/evolution/data/scraped_evo
df = rbind(df, read.csv('../experiments/2021_02_24__finite_10k_samples/evolution/data/scraped_evo
df$LENGTH = 100
df = rbind(df, read.csv('../experiments/2021_02_24__finite_10k_samples/evolution/data/scraped_evo
df = rbind(df, read.csv('../experiments/2021_02_24__finite_10k_samples/evolution/data/scraped_evo
```

```
# Trim off NAs (artifiacts of how we scraped the data) and trim to only have gen 10,00
df2 = df[!is.na(df$MCSIZE) & df$generation == 10000,]
# Ignore data for size 8x8 and 1024x1024
df2 = df2[df2$MCSIZE != 8 & df2$MCSIZE != 1024,]
```

We group and summarize the data to make to ensure all replicates are present.

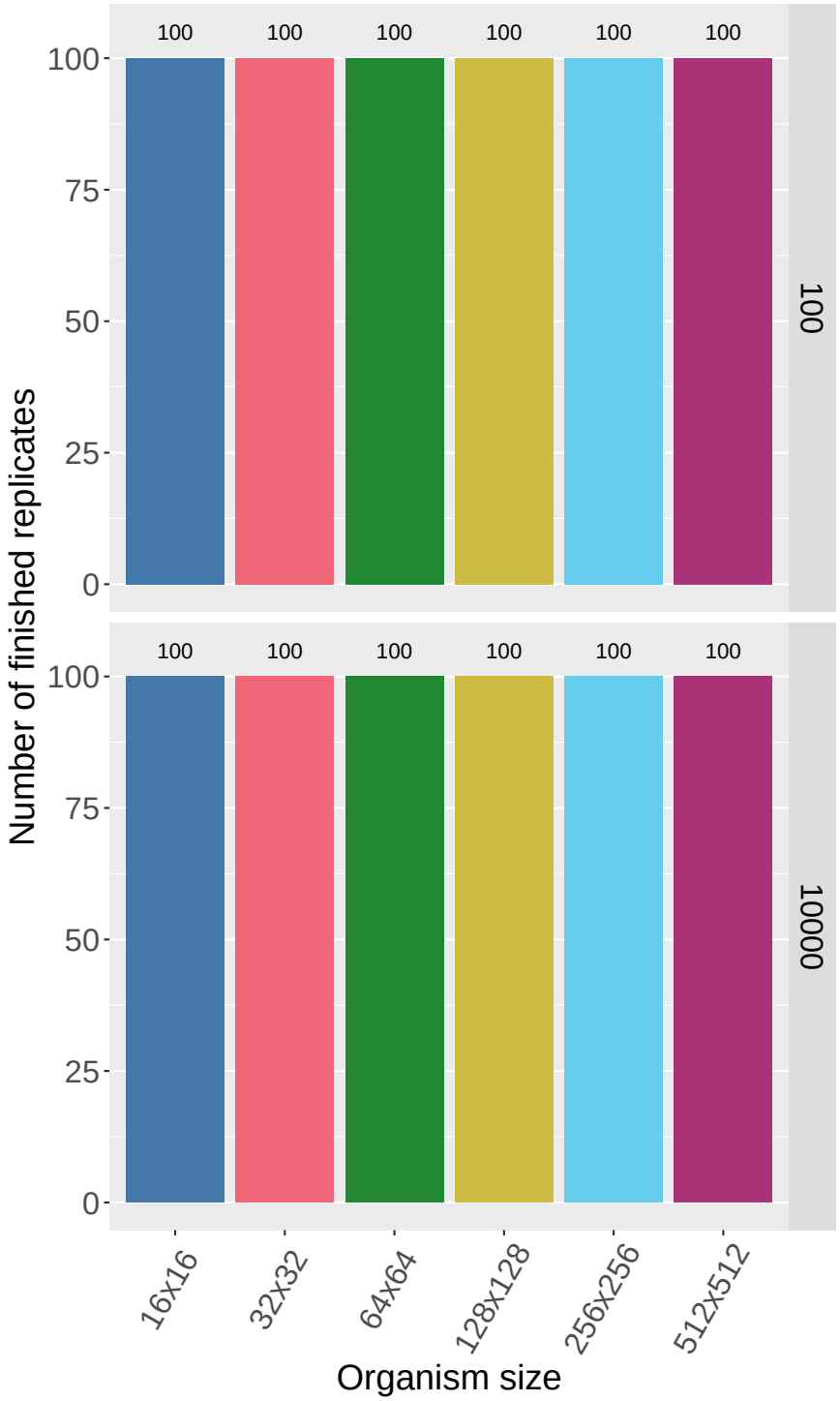
```
# Group the data by size and summarize
data_grouped = dplyr::group_by(df2, MCSIZE, SAMPLES)
data_summary = dplyr::summarize(data_grouped, mean_ones = mean(ave_ones), n = dplyr::n
```

We clean the data and create a few helper variables to make plotting easier.

```
# Calculate restraint value (x - 60 because genome length is 100 here)
df2$restraint_value = df2$ave_ones - 60
# Make a nice, clean factor for size
df2$size_str = paste0(df2$MCSIZE, 'x', df2$MCSIZE)
df2$size_factor = factor(df2$size_str, levels = c('16x16', '32x32', '64x64', '128x128'
df2$size_factor_reversed = factor(df2$size_str, levels = rev(c('16x16', '32x32', '64x6
data_summary$size_str = paste0(data_summary$MCSIZE, 'x', data_summary$MCSIZE)
data_summary$size_factor = factor(data_summary$size_str, levels = c('16x16', '32x32',
# Create a map of colors we'll use to plot the different organism sizes
color_vec = as.character(khroma::color('bright')(7))
color_map = c(
  '16x16' =     color_vec[1],
  '32x32' =     color_vec[2],
  '64x64' =     color_vec[3],
  '128x128' =   color_vec[4],
  '256x256' =   color_vec[5],
  '512x512' =   color_vec[6],
  '1024x1024' = color_vec[7]
)
# Set the sizes for text in plots
text_major_size = 18
text_minor_size = 16
```

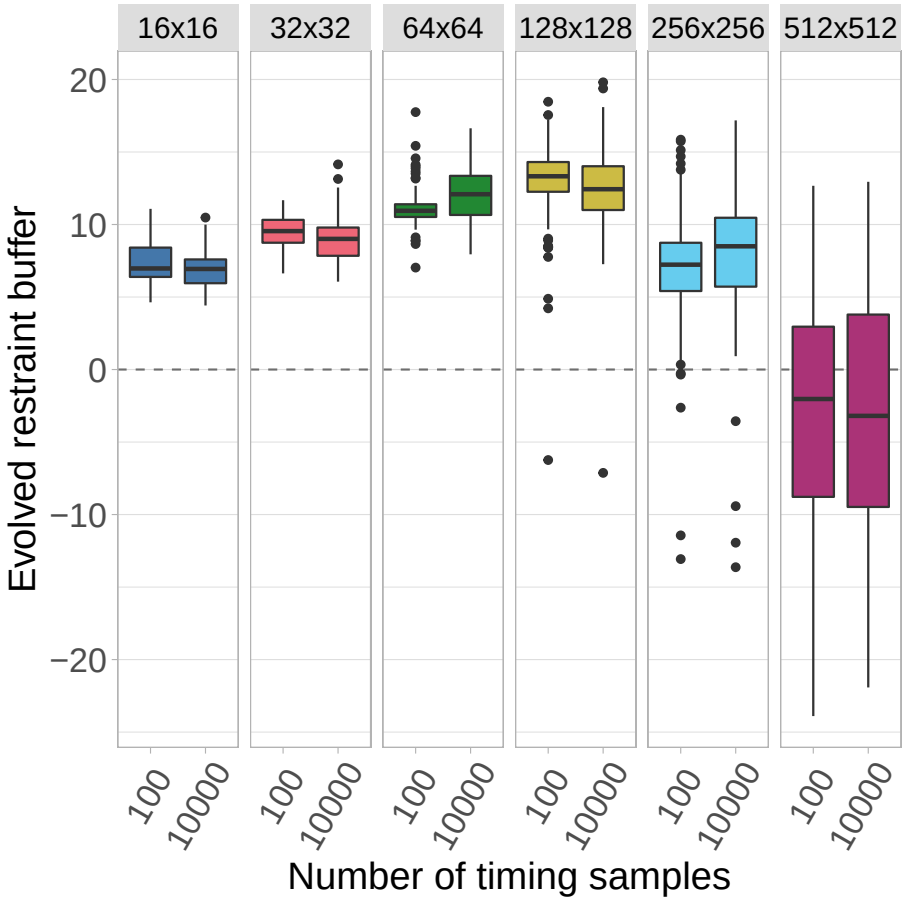
6.2 Data integrity check

Now we plot the number of finished replicates for each treatment to make sure all data are present. Rows show the number of samples used for fitness. Each bar/color shows a different organism size.


100
100
100
100
100
100
100
75
50
25
0
100
Number of finished replicates
10000
16x16
32x32
64x64
128x128
256x256
512x512
Organism size

6.3 Plot

Here we plot all the data. The figure is split into 6 subplots, each showing a different organism size. Inside each subplot, the number of timing samples is shown on



the x-axis.

6.4 Statistics

The plot shows that the general trend, that the evolved restraint buffer initially increases with organism but then decreases, holds true at both sample counts. Further, we see that the evolved buffer values are fairly consistent between the two sample counts.

While we concluded this was sufficient evidence to use only 100 samples (10,000 is intractable to run for multiple experiments), we include the statistics here. Since we treat each organism size as a group, we simply take a Wilcoxon Rank-Sum test between 100 samples and 10,000 samples

```
##   org_size      p_value      W less_0.01
## 1       16 4.243294e-02 5831.0     FALSE
## 2       32 3.489808e-04 6464.0      TRUE
## 3       64 4.913265e-05 3338.0      TRUE
## 4      128 3.021256e-02 5887.5     FALSE
## 5      256 2.561216e-02 4086.0     FALSE
## 6      512 9.066359e-01 5048.5     FALSE
```

Chapter 7

Interactive web app

TBD