

Supplemental Material: Valley Crossing Diagnostics

Jose Guadalupe Hernandez

2023-07-28

Contents

1	Introduction	5
1.1	About our supplemental material	5
1.2	Contributing authors	5
1.3	Computer Setup	5
1.4	Experimental setup	6
2	Exploitation rate results	9
2.1	Analysis dependencies	9
2.2	Data setup	9
2.3	Performance over time	9
2.4	Best performance throughout	11
2.5	Performance interval comparison	14
3	Ordered exploitation results	29
3.1	Analysis dependencies	29
3.2	Data setup	29
3.3	Performance over time	29
3.4	Best performance throughout	31
3.5	Performance interval comparison	33
4	Contradictory objectives results	49
4.1	Analysis dependencies	49
4.2	Data setup	49
4.3	Activation gene coverage over time	50
4.4	Final activation gene coverage	51
4.5	Satisfactory trait coverage over time	54
4.6	Best satisfactory trait coverage throughout	55
4.7	Final satisfactory trait coverage	58
4.8	Best trait value over time	60
4.9	Best trait value throughout	62
5	Multi-path exploration results	65
5.1	Analysis dependencies	65
5.2	Data setup	65

5.3	Activation gene coverage over time	66
5.4	Final activation gene coverage	67
5.5	Performance over time	69
5.6	Best performance throughout	71

Chapter 1

Introduction

This is the supplemental material for experiments with diagnostics and integrated valleys.

1.1 About our supplemental material

This supplemental material is hosted on GitHub using GitHub pages. The source code and configuration files used to generate this supplemental material can be found in this GitHub repository. We compiled our data analyses and supplemental documentation into this nifty web-accessible book using bookdown.

Our supplemental material includes the following paper figures and statistics:

- Exploitation rate results (Section 2)
- Ordered exploitation results (Section 3)
- Contradictory objectives results (Section 4)
- Multi-path exploration results (Section 5)

1.2 Contributing authors

- Jose Guadalupe Hernandez
- Alexander Lalejini
- Charles Ofria

1.3 Computer Setup

These analyses were conducted in the following computing environment:

```
print(version)
```

```
##
## platform      x86_64-pc-linux-gnu
## arch          x86_64
## os            linux-gnu
## system        x86_64, linux-gnu
## status
## major         4
## minor         3.1
## year          2023
## month         06
## day           16
## svn rev       84548
## language      R
## version.string R version 4.3.1 (2023-06-16)
## nickname      Beagle Scouts
```

1.4 Experimental setup

Setting up required variables.

```
# libraries we are using
```

```
library(ggplot2)
```

```
library(cowplot)
```

```
library(dplyr)
```

```
##
```

```
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
##      filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##      intersect, setdiff, setequal, union
```

```
library(PupillometryR)
```

```
## Loading required package: rlang
```

```
# data diractory for gh-pages
```

```
DATA_DIR = '/opt/ECJ-2023-Suite-Of-Diagnostic-Metrics-For-Characterizing-Selection-Sch
```

```
# data diractory for local testing
```

```
# DATA_DIR = 'C:/Users/jgh9094/Desktop/Research/Projects/SelectionDiagnostics/ECJ-2023
```

```
# graph variables
```

```
SHAPE = c(5,3,1,2,6,0,4,20,1)
```

```

cb_palette <- c('#332288', '#88CCEE', '#EE7733', '#EE3377', '#117733', '#882255', '#44AA99', '#CCBB44',
TSIZE = 26
p_theme <- theme(
  text = element_text(size = 28),
  plot.title = element_text( face = "bold", size = 22, hjust=0.5),
  panel.border = element_blank(),
  panel.grid.minor = element_blank(),
  legend.title=element_text(size=22),
  legend.text=element_text(size=23),
  axis.title = element_text(size=23),
  axis.text = element_text(size=22),
  legend.position="bottom",
  panel.background = element_rect(fill = "#f1f2f5",
                                   colour = "white",
                                   size = 0.5, linetype = "solid")
)

```

```

## Warning: The `size` argument of `element_rect()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

```

```

# default variables
REPLICATES = 50
DIMENSIONALITY = 100
GENERATIONS = 50000

# selection scheme related stuff
ACRO = c('tru', 'tor', 'lex', 'gfs', 'pfs', 'nds', 'nov', 'ran')
NAMES = c('Truncation (tru)', 'Tournament (tor)', 'Lexicase (lex)', 'Genotypic Fitness Sharing (gfs)')

# valley crossing comparisons
mvc_col = c('#1A85FF', '#D41159')

```


Chapter 2

Exploitation rate results

Here we present the results for **best performances** found by each selection scheme on the exploitation rate diagnostic with valley crossing integrated. 50 replicates are conducted for each scheme explored.

2.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

2.2 Data setup

```
DIR = paste(DATA_DIR, 'EXPLOITATION_RATE/', sep = "", collapse = NULL)
over_time_df <- read.csv(paste(DIR, 'over-time.csv', sep = "", collapse = NULL), header = TRUE, stringsAsFactors = FALSE)
over_time_df$scheme <- factor(over_time_df$scheme, levels = NAMES)

best_df <- read.csv(paste(DIR, 'best.csv', sep = "", collapse = NULL), header = TRUE, stringsAsFactors = FALSE)
best_df$acro <- factor(best_df$acro, levels = ACRO)
```

2.3 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```

lines = over_time_df %>%
  group_by(scheme, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )

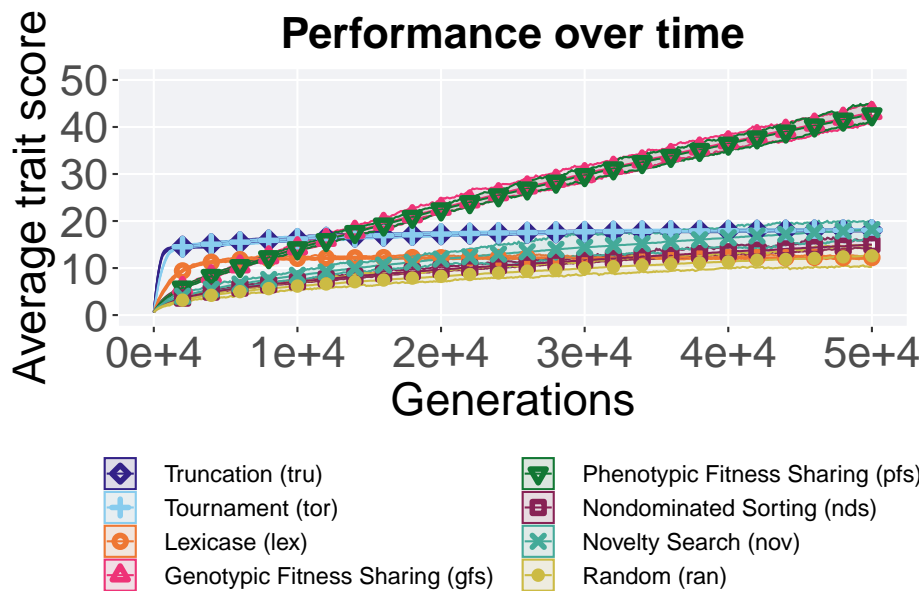
```

`summarise()` has grouped output by 'scheme'. You can override using the
`.groups` argument.

```

over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = scheme, fill = scheme, color
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %>= 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 50),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank(),legend.text=element_text(size=12)) +
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  )
over_time_plot

```



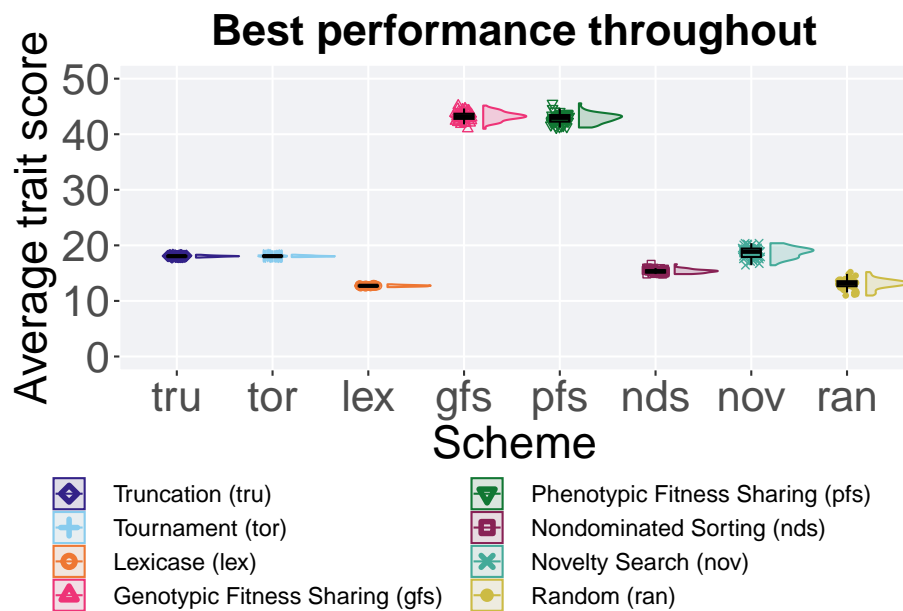
2.4 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max') %>%
  ggplot(., aes(x = acro, y = val / DIMENSIONALITY, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 50),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout') +
  p_theme + theme(legend.title=element_blank())
```

```
plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

```
## Warning: Using the `size` aesthetic with geom_polygon was deprecated in ggplot2 3.4
## i Please use the `linewidth` aesthetic instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



2.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max')
performance$acro = factor(performance$acro, levels = c('gfs','pfs','tru','tor','nov','ran'))
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
```

```

min = min(val / DIMENSIONALITY, na.rm = TRUE),
median = median(val / DIMENSIONALITY, na.rm = TRUE),
mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
max = max(val / DIMENSIONALITY, na.rm = TRUE),
IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
)

```

```

## # A tibble: 8 x 8
##   acro  count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 gfs     50     0  41.0  43.2  43.2  45.2  0.923
## 2 pfs     50     0  41.2  43.0  42.9  45.6  1.17
## 3 tru     50     0  17.8  18.1  18.0  18.3  0.140
## 4 tor     50     0  17.9  18.1  18.1  18.3  0.150
## 5 nov     50     0  16.4  18.8  18.7  20.4  1.51
## 6 nds     50     0  14.8  15.3  15.4  16.6  0.418
## 7 ran     50     0  11.0  13.2  13.1  15.2  0.900
## 8 lex     50     0  12.5  12.7  12.7  13.0  0.154

```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ acro, data = performance)
```

```

##
##  Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 367.97, df = 7, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```

pairwise.wilcox.test(x = performance$val, g = performance$acro, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')

```

```

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$acro
##
##      gfs      pfs      tru      tor      nov      nds      ran
## pfs 0.4845 -          -          -          -          -          -
## tru < 2e-16 < 2e-16 -          -          -          -          -
## tor < 2e-16 < 2e-16 1.0000 -          -          -          -
## nov < 2e-16 < 2e-16 1.0000 1.0000 -          -          -
## nds < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -          -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.9e-16 -
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 0.0035
##

```

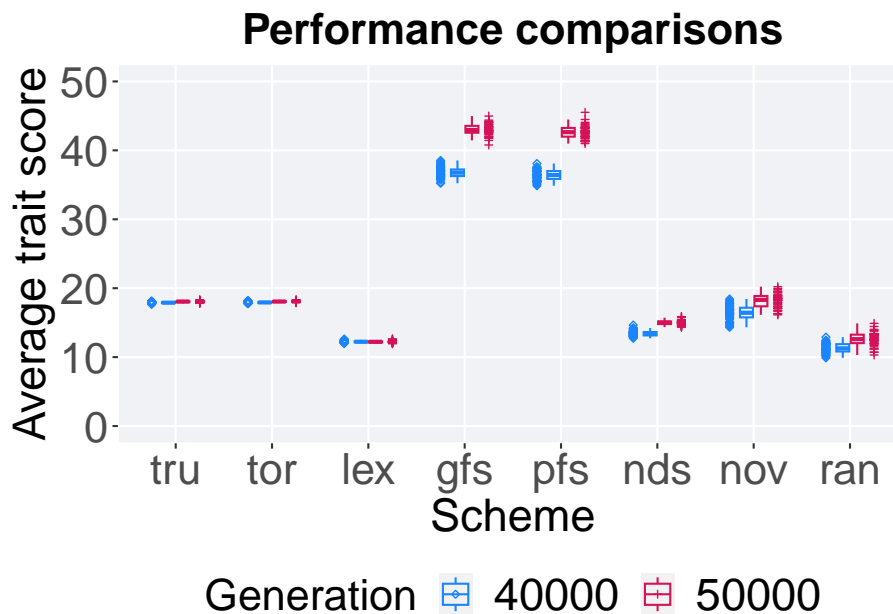
```
## P value adjustment method: bonferroni
```

2.5 Performance interval comparison

Here we compare the best performance found in the population at generation 40,000 & 50,000 for all selection schemes.

```
interval_df = filter(filter(over_time_df, (gen == 50000 | gen == 40000)))
interval_df$Generation <- factor(interval_df$gen, levels = c(40000,50000))
interval_df$acro <- factor(interval_df$acro, levels = ACRO)

ggplot(interval_df, aes(x=acro, y=pop_fit_max/DIMENSIONALITY,fill=Generation,color=Generation)) +
  geom_boxplot(width = .3, outlier.shape = NA, alpha = 0.0) +
  geom_point(size = 1.0, alpha = 1.0, position=position_dodge(width=1.0)) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 50),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons') +
  p_theme
```



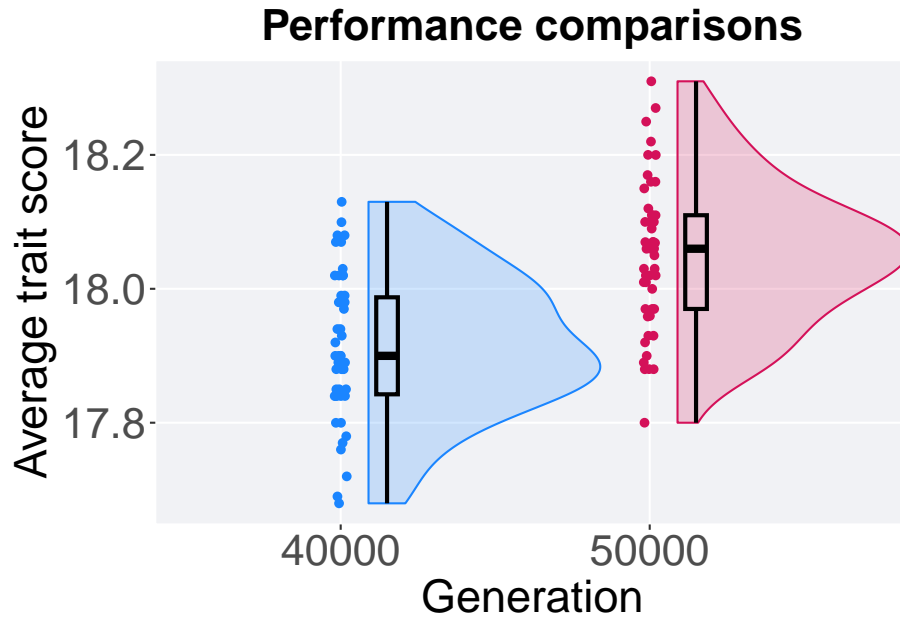
2.5.1 Stats by interval comparisons

Here we present statistical analysis between the performances found at different generation intervals.

2.5.1.1 Truncation

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'tru') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation, shape=Generation)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = 'dodge') +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons') +
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'tru') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  17.7  17.9  17.9  18.1  0.145
## 2 50000         50     0  17.8  18.1  18.0  18.3  0.140
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'tru' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'tru' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

```
##
```

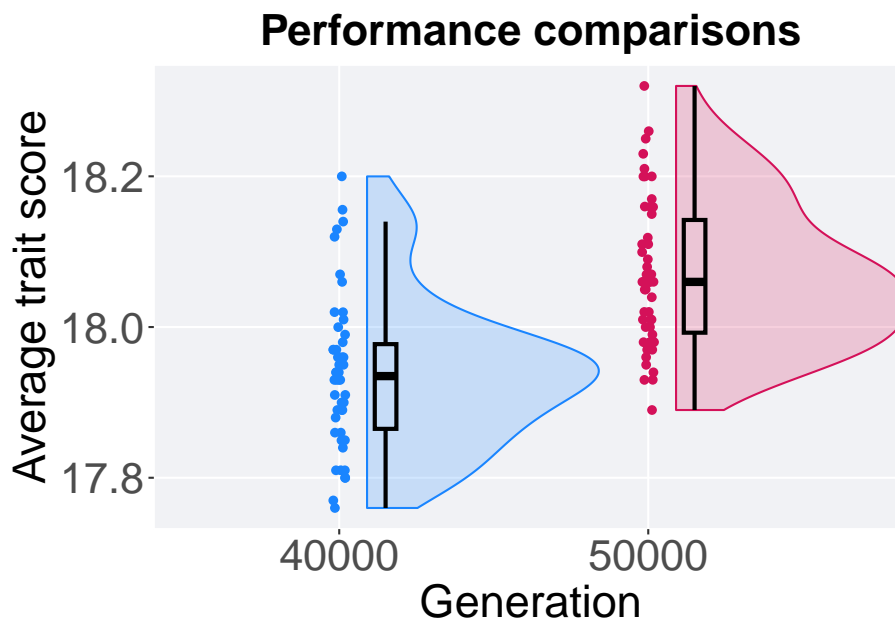


```
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "tru" & Generation == 50000)$pop_fit_max and filter(interval_df, acro == "tru" & Generation == 40000)$pop_fit_max
## W = 2013.5, p-value = 7.103e-08
## alternative hypothesis: true location shift is greater than 0
```

2.5.1.2 Tournament

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'tor') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation, shape=Generation)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16)) +
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons') +
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'tor') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  17.8  17.9  17.9  18.2 0.112
## 2 50000         50     0  17.9  18.1  18.1  18.3 0.150
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'tor' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'tor' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

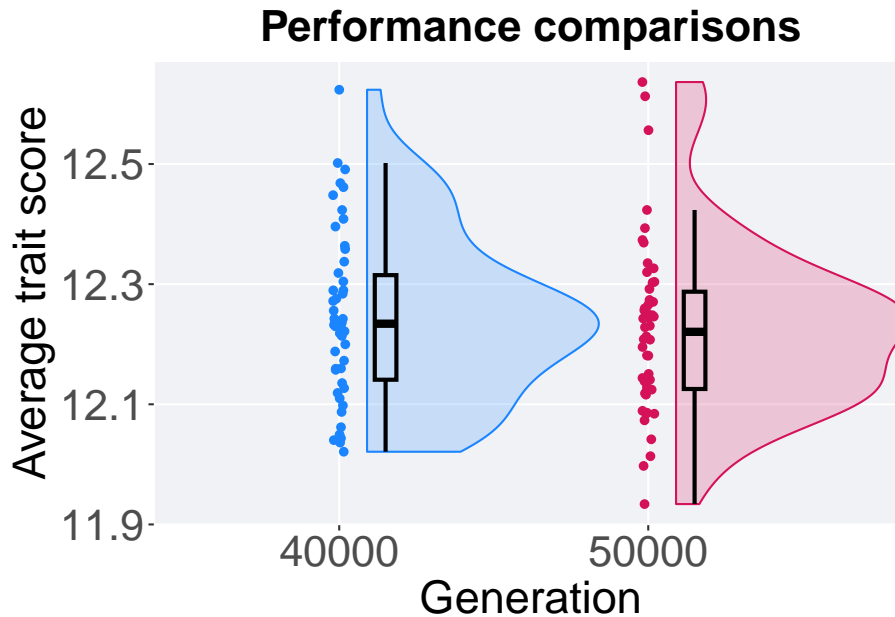
```
##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "tor" & Generation == 50000)$pop_fit_max and fil
## W = 2095.5, p-value = 2.818e-09
## alternative hypothesis: true location shift is greater than 0
```

2.5.1.3 Lexicase

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'lex') %>%
  ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation)) +
    geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.5) +
    geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1) +
    geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
    scale_y_continuous(
      name="Average trait score",
    ) +
    scale_x_discrete(
      name="Generation"
    ) +
```

```
scale_shape_manual(values=c(16,16))+
scale_colour_manual(values = mvc_col, ) +
scale_fill_manual(values = mvc_col) +
ggtitle('Performance comparisons')+
p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'lex') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  12.0  12.2  12.2  12.6  0.174
## 2 50000         50     0  11.9  12.2  12.2  12.6  0.162
```

Results for post-hoc Wilcoxon rank-sum test.

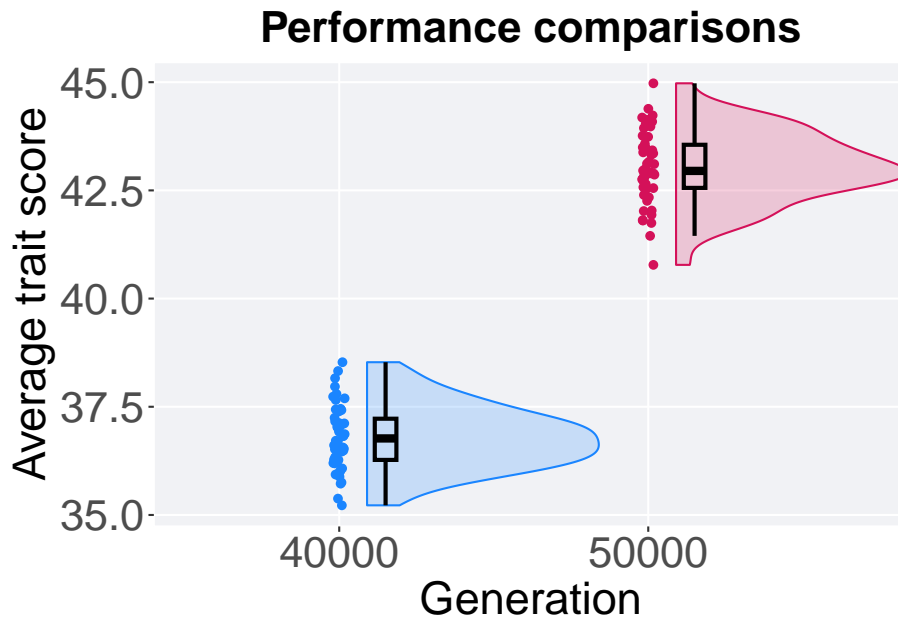
```
wilcox.test(x = filter(interval_df, acro == 'lex' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'lex' & Generation == 40000)$pop_fit_max,
            alternative = 't')
```

```
##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "lex" & Generation == 50000)$pop_fit_max and fil
## W = 1155, p-value = 0.5147
## alternative hypothesis: true location shift is not equal to 0
```

2.5.1.4 Genotypic fitness sharing

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'gfs') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY,fill=Generation,color=Generat.
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons')+
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'gfs') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  35.2  36.8  36.8  38.5  0.952
## 2 50000         50     0  40.8  43.0  43.0  45.0  0.996
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'gfs' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'gfs' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

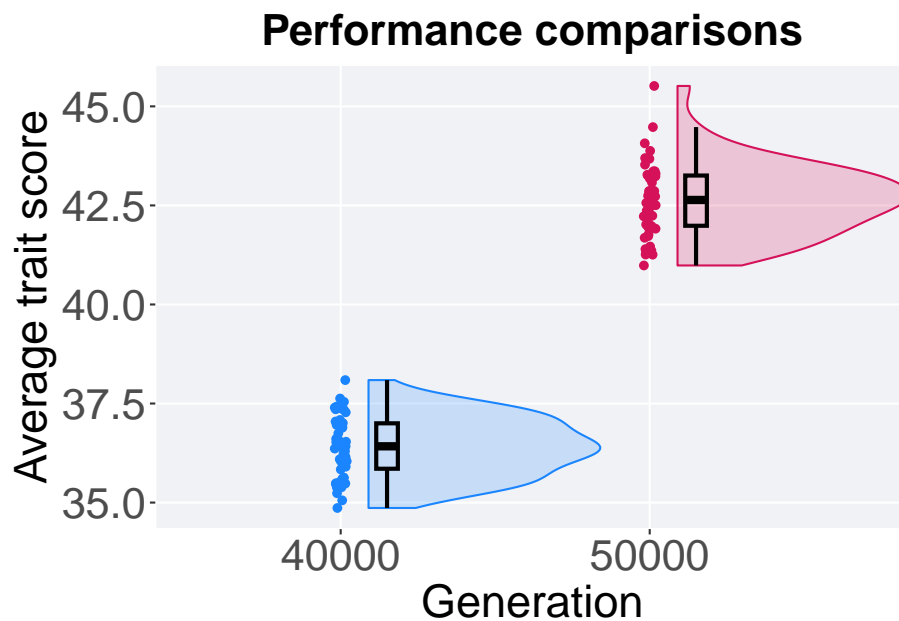
```
##
```

```
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "gfs" & Generation == 50000)$pop_fit_max and fil
## W = 2500, p-value < 2.2e-16
## alternative hypothesis: true location shift is greater than 0
```

2.5.1.5 Phenotypic fitness sharing

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'pfs') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generat
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons')+
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'pfs') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  34.9  36.4  36.4  38.1  1.15
## 2 50000         50     0  41.0  42.6  42.6  45.5  1.27
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'pfs' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'pfs' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

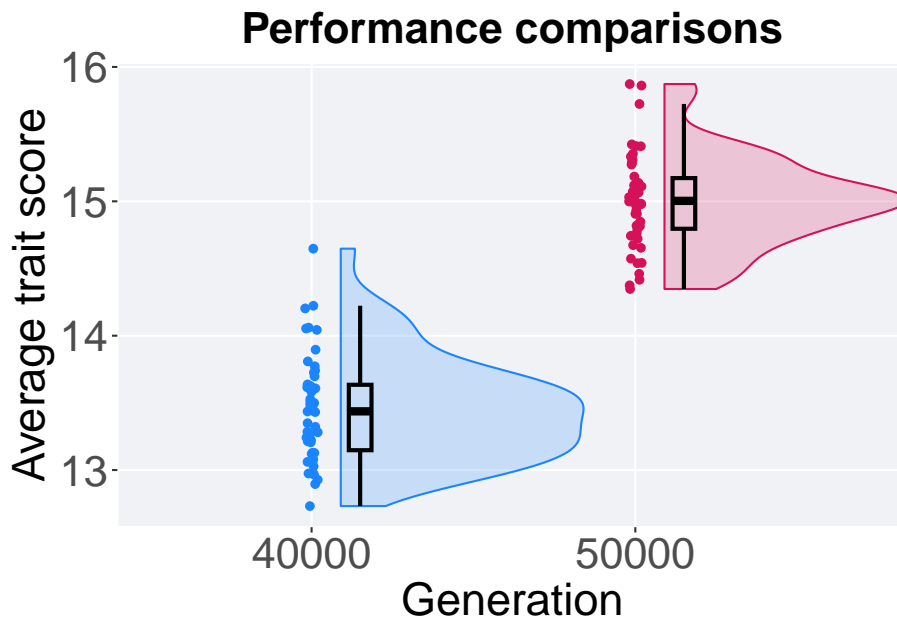
```
##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "pfs" & Generation == 50000)$pop_fit_max and filter(interval_df, acro == "pfs" & Generation == 40000)$pop_fit_max
## W = 2500, p-value < 2.2e-16
## alternative hypothesis: true location shift is greater than 0
```

2.5.1.6 Nondominated sorting

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'nds') %>%
  ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation, shape=Generation)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
```

```
scale_shape_manual(values=c(16,16))+
scale_colour_manual(values = mvc_col, ) +
scale_fill_manual(values = mvc_col) +
ggtitle('Performance comparisons')+
p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'nds') %>%
group_by(Generation) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_fit_max)),
  min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
  median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
  mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
  max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
  IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
)
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  12.7  13.4  13.5  14.6  0.487
## 2 50000         50     0  14.3  15.0  15.0  15.9  0.377
```


Results for post-hoc Wilcoxon rank-sum test.

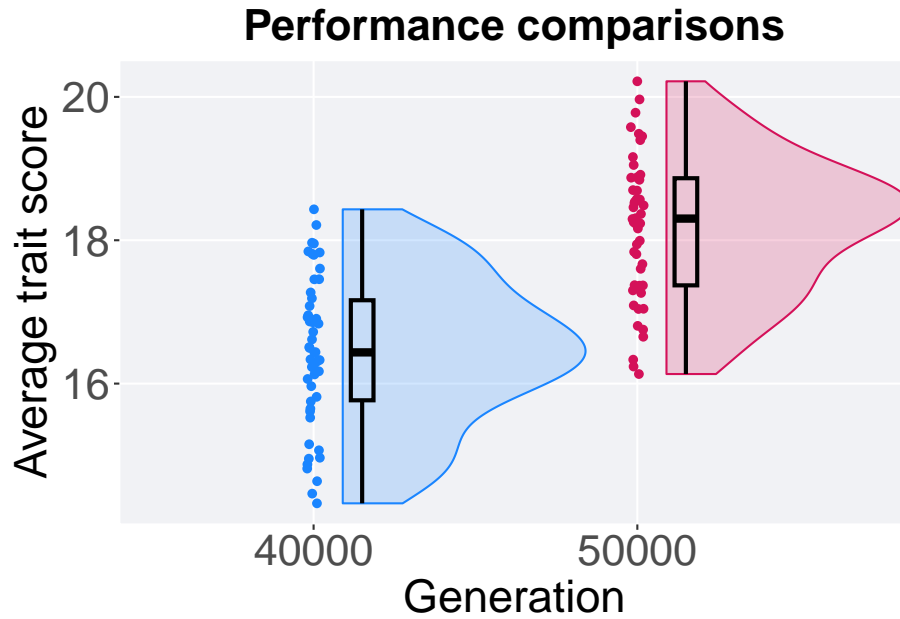
```
wilcox.test(x = filter(interval_df, acro == 'nds' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'nds' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

```
##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "nds" & Generation == 50000)$pop_fit_max and filter(interval_df, acro == "nds" & Generation == 40000)$pop_fit_max
## W = 2493, p-value < 2.2e-16
## alternative hypothesis: true location shift is greater than 0
```

2.5.1.7 Novelty search

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'nov') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation, shape=Generation)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons') +
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'nov') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  14.3  16.4  16.4  18.4  1.40
## 2 50000         50     0  16.1  18.3  18.2  20.2  1.49
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'nov' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'nov' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

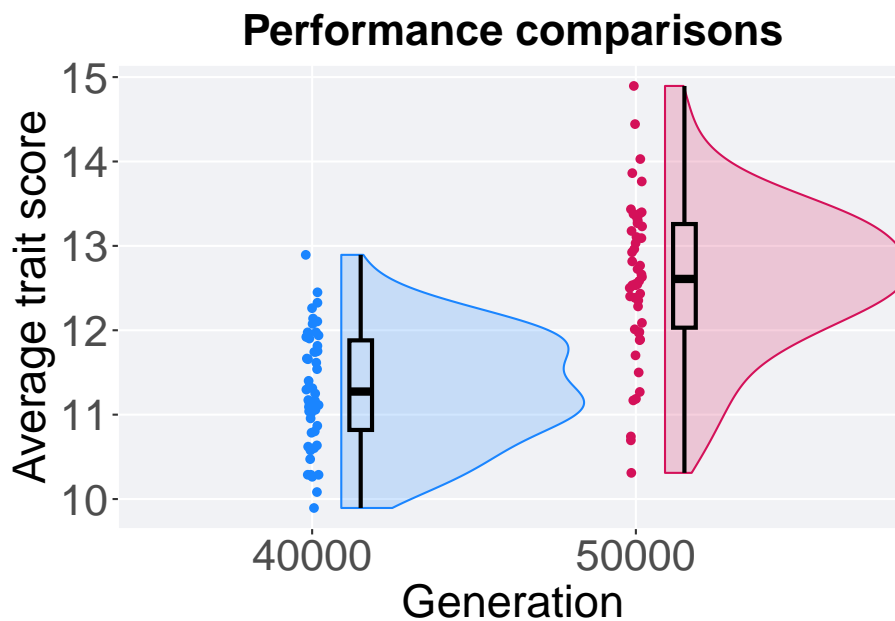
```
##
```

```
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "nov" & Generation == 50000)$pop_fit_max and filter(interval_df, acro == "nov" & Generation == 40000)$pop_fit_max
## W = 2209, p-value = 1.951e-11
## alternative hypothesis: true location shift is greater than 0
```

2.5.1.8 Random

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'ran') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation, shape=Generation)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = 'dodge') +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons')+
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'ran') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  9.89  11.3  11.3  12.9  1.06
## 2 50000         50     0 10.3   12.6  12.6  14.9  1.23
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'ran' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'ran' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

```
##
## Wilcoxon rank sum test with continuity correction
##
## data:  filter(interval_df, acro == "ran" & Generation == 50000)$pop_fit_max and fil
## W = 2185, p-value = 5.885e-11
## alternative hypothesis: true location shift is greater than 0
```

Chapter 3

Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme on the ordered exploitation diagnostic with valley crossing integrated. 50 replicates are conducted for each scheme explored.

3.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

3.2 Data setup

```
DIR = paste(DATA_DIR, 'ORDERED_EXPLOITATION/', sep = "", collapse = NULL)
over_time_df <- read.csv(paste(DIR, 'over-time.csv', sep = "", collapse = NULL), header = TRUE, stringsAsFactors = FALSE)
over_time_df$scheme <- factor(over_time_df$scheme, levels = NAMES)

best_df <- read.csv(paste(DIR, 'best.csv', sep = "", collapse = NULL), header = TRUE, stringsAsFactors = FALSE)
best_df$acro <- factor(best_df$acro, levels = ACRO)
```

3.3 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```

lines = over_time_df %>%
  group_by(scheme, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )

```

`summarise()` has grouped output by 'scheme'. You can override using the
`.groups` argument.

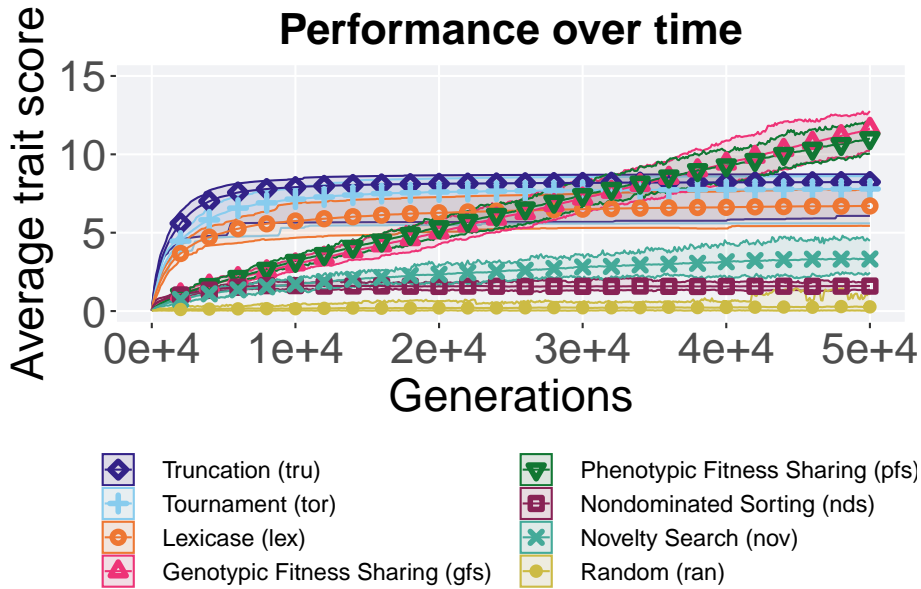
```

over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = scheme, fill = scheme, color
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 15),
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank(),legend.text=element_text(size=12)) +
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  )

over_time_plot

```



3.4 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

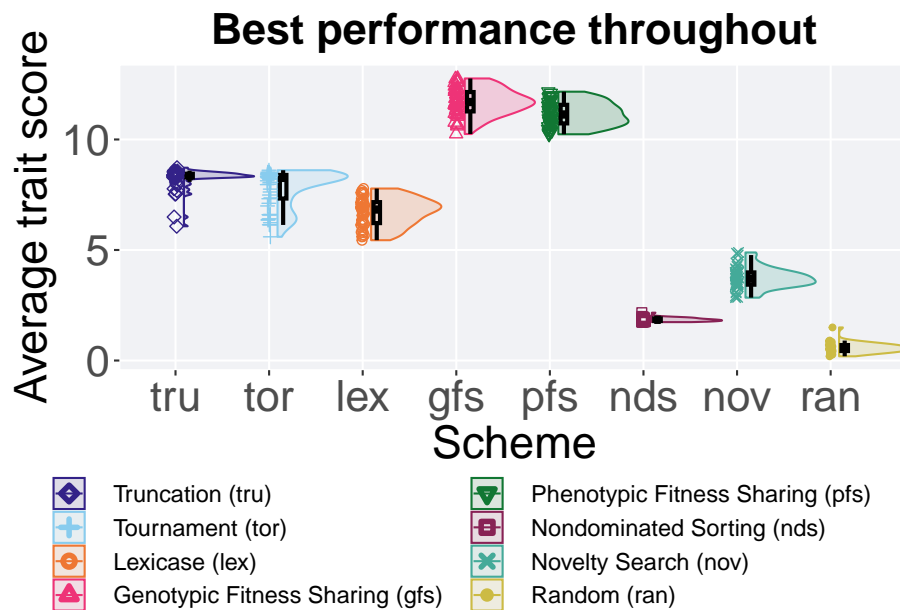
```
plot = filter(best_df, var == 'pop_fit_max') %>%
  ggplot(., aes(x = acro, y = val / DIMENSIONALITY, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout') +
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```

legend,
nrow=2,
rel_heights = c(3,1)
)

```



3.4.1 Stats

Summary statistics for the best performance.

```

performance = filter(best_df, var == 'pop_fit_max')
performance$acro = factor(performance$acro, levels = c('gfs','pfs','tru','tor','lex','nds','nov','ran'))
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )

```

```
## # A tibble: 8 x 8
```

```
##   acro   count na_cnt   min median   mean   max   IQR
```



```
##      <fct> <int>  <int>  <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 gfs      50      0 10.2   11.7   11.7  12.8  0.876
## 2 pfs      50      0 10.2   11.1   11.2  12.2  0.840
## 3 tru      50      0  6.07   8.35   8.23   8.72  0.166
## 4 tor      50      0  5.60   8.26   7.79   8.61  1.07
## 5 lex      50      0  5.44   6.83   6.72   7.78  0.949
## 6 nov      50      0  2.84   3.70   3.74   4.89  0.555
## 7 nds      50      0  1.74   1.84   1.85   2.16  0.114
## 8 ran      50      0  0.192  0.552  0.570  1.50  0.269
```

Kruskal-Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ acro, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 380.24, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$val, g = performance$acro, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$acro
##
##      gfs      pfs      tru      tor      lex      nov      nds
## pfs 0.00041 -          -          -          -          -          -
## tru < 2e-16 < 2e-16 -          -          -          -          -
## tor < 2e-16 < 2e-16 0.09539 -          -          -          -
## lex < 2e-16 < 2e-16 2.1e-14 2.0e-07 -          -          -
## nov < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -          -
## nds < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

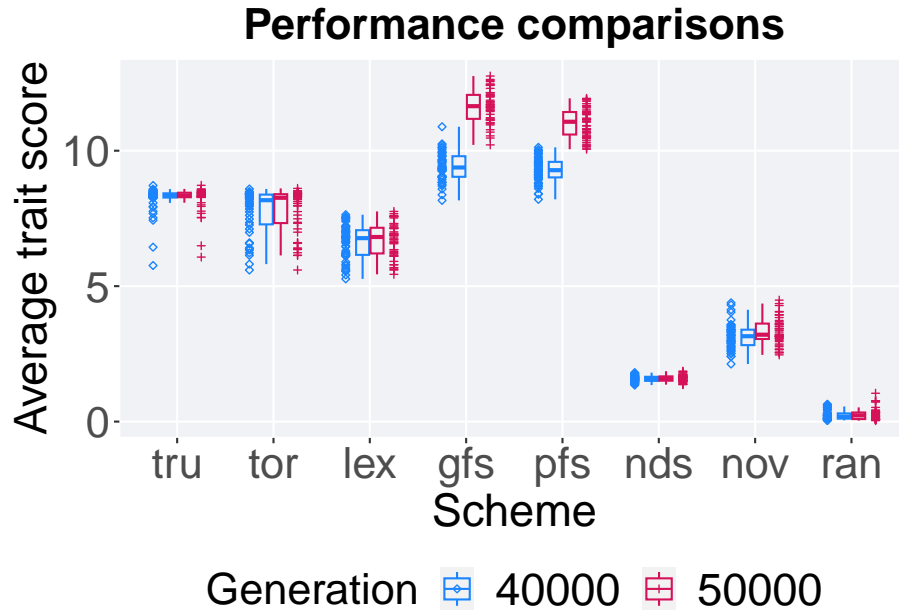
3.5 Performance interval comparison

Here we compare the best performance found in the population at generation 40,000 & 50,000 for all selection schemes.

```
interval_df = filter(filter(over_time_df, (gen == 50000 | gen == 40000)))
interval_df$Generation <- factor(interval_df$gen, levels = c(40000, 50000))
```

```
interval_df$acro <- factor(interval_df$acro, levels = ACRO)

ggplot(interval_df, aes(x=acro, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Gen
  geom_boxplot(width = .3, outlier.shape = NA, alpha = 0.0) +
  geom_point(size = 1.0, alpha = 1.0, position=position_dodge(width=1.0)) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons') +
  p_theme
```



3.5.1 Stats by interval comparisons

Here we present statistical analysis between the performances found at different generation intervals.

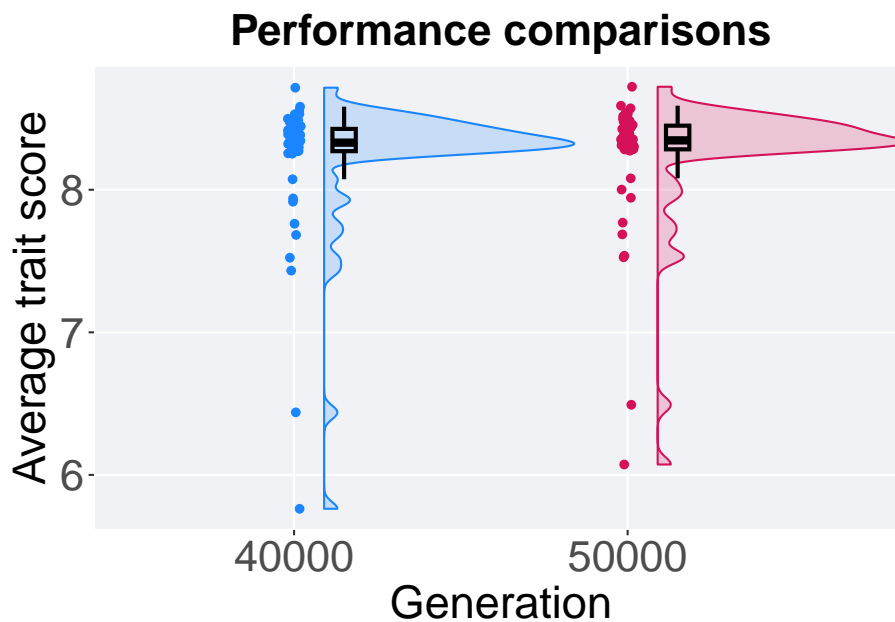
3.5.1.1 Truncation

Plot for comparing performances at interval 40,000 & 50,000.

```

filter(interval_df, acro == 'tru') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY,fill=Generation,color=Generation,shape=O)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = 'dodge') +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons')+
  p_theme + theme(legend.position="none")

```



Summary statistics for the best performance.

```

filter(interval_df, acro == 'tru') %>%
group_by(Generation) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_fit_max)),
  min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),

```

```

median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
)

```

```

## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>         <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  5.76  8.33  8.21  8.72 0.156
## 2 50000         50     0  6.07  8.35  8.23  8.72 0.166

```

Results for post-hoc Wilcoxon rank-sum test.

```

wilcox.test(x = filter(interval_df, acro == 'tru' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'tru' & Generation == 40000)$pop_fit_max,
            alternative = 't')

```

```

##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "tru" & Generation == 50000)$pop_fit_max and fil
## W = 1351, p-value = 0.4884
## alternative hypothesis: true location shift is not equal to 0

```

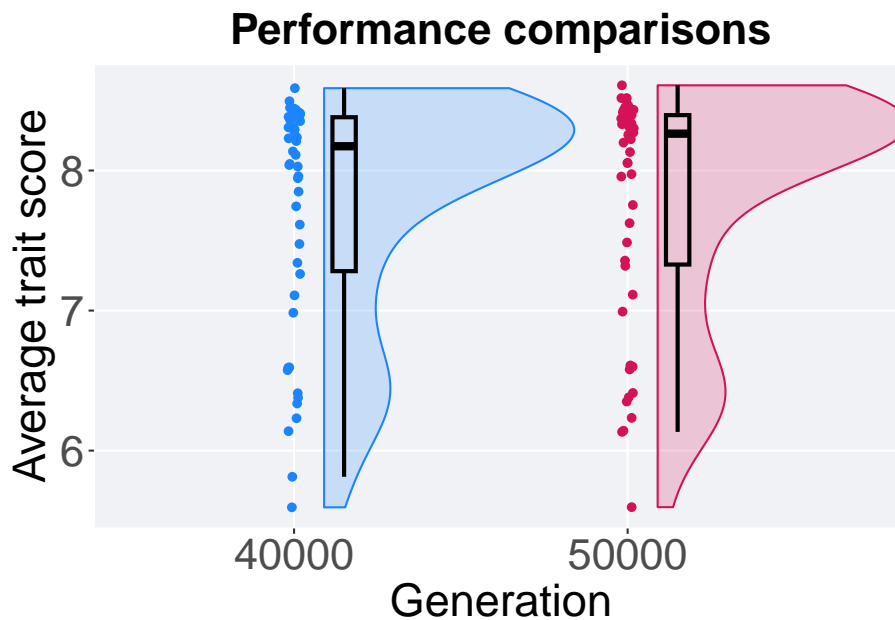
3.5.1.2 Tournament

Plot for comparing performances at interval 40,000 & 50,000.

```

filter(interval_df, acro == 'tor') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY,fill=Generation,color=Generat.
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons')+
  p_theme + theme(legend.position="none")

```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'tor') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int>  <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50      0  5.59  8.17  7.75  8.59  1.10
## 2 50000         50      0  5.60  8.26  7.79  8.61  1.07
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'tor' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'tor' & Generation == 40000)$pop_fit_max,
            alternative = 't')
```

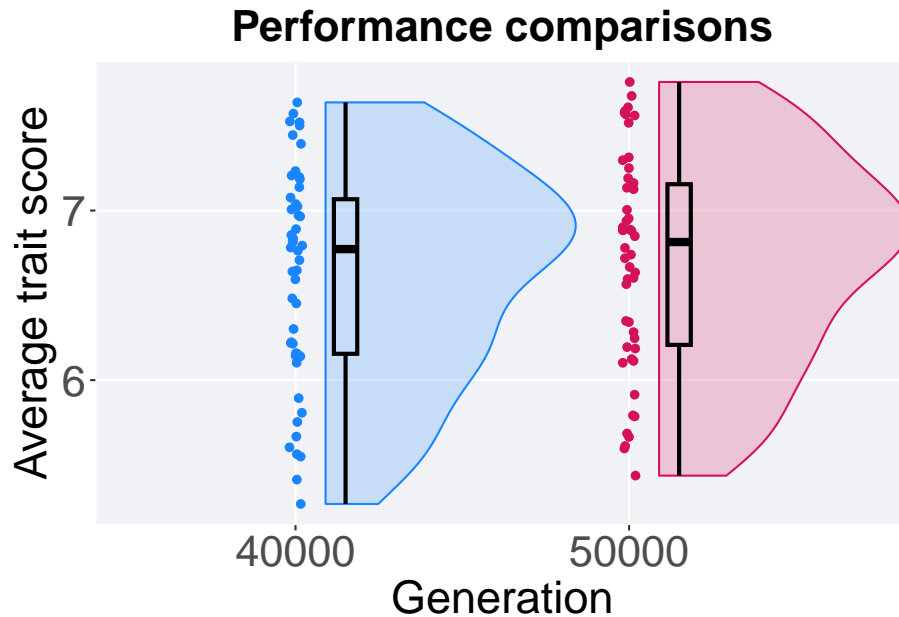
```
##
```

```
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "tor" & Generation == 50000)$pop_fit_max and fil
## W = 1343, p-value = 0.5237
## alternative hypothesis: true location shift is not equal to 0
```

3.5.1.3 Lexicase

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'lex') %>%
ggplot(. , aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generat
geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons')+
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'lex') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  5.27  6.77  6.62  7.64  0.912
## 2 50000         50     0  5.44  6.81  6.71  7.76  0.948
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'lex' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'lex' & Generation == 40000)$pop_fit_max,
            alternative = 't')
```

```
##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "lex" & Generation == 50000)$pop_fit_max and filter(interval_df, acro == "lex" & Generation == 40000)$pop_fit_max
## W = 1346, p-value = 0.5103
## alternative hypothesis: true location shift is not equal to 0
```

3.5.1.4 Genotypic fitness sharing

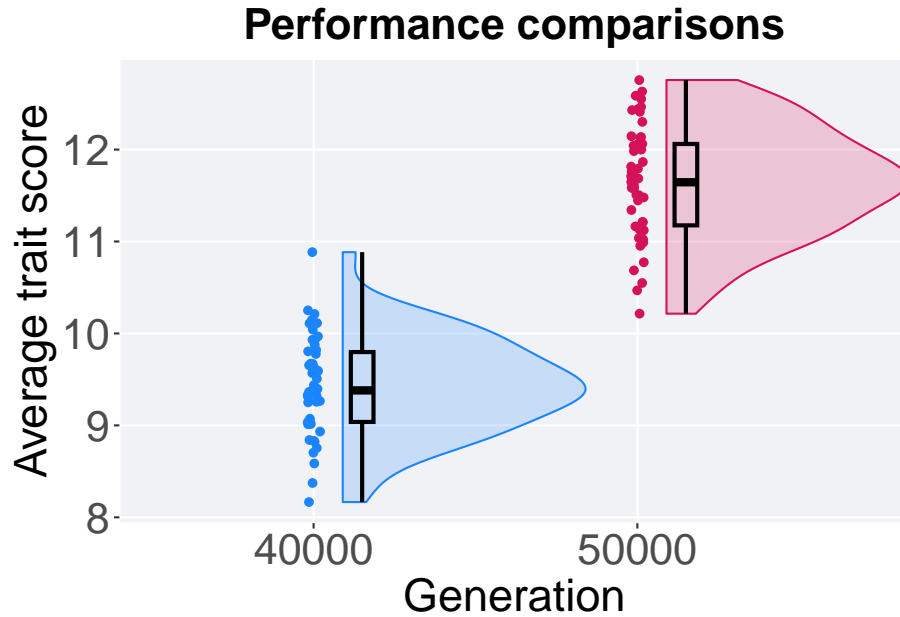
Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'gfs') %>%
  ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation, shape=Generation)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
```

```

scale_shape_manual(values=c(16,16))+
scale_colour_manual(values = mvc_col, ) +
scale_fill_manual(values = mvc_col) +
ggtitle('Performance comparisons')+
p_theme + theme(legend.position="none")

```



Summary statistics for the best performance.

```

filter(interval_df, acro == 'gfs') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

```

```

## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  8.17  9.38  9.43 10.9 0.760
## 2 50000         50     0 10.2 11.6 11.6 12.8 0.885

```


Results for post-hoc Wilcoxon rank-sum test.

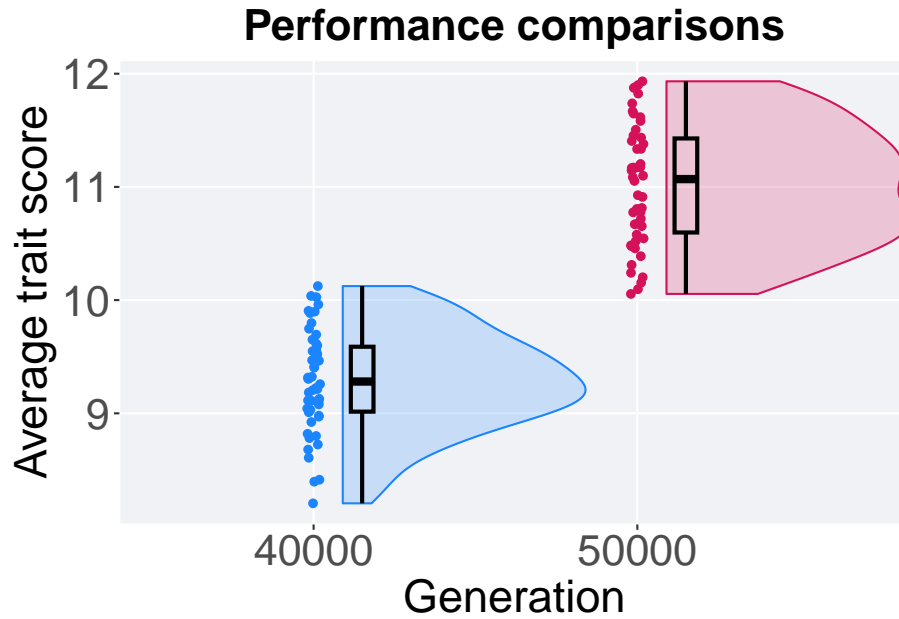
```
wilcox.test(x = filter(interval_df, acro == 'gfs' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'gfs' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

```
##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "gfs" & Generation == 50000)$pop_fit_max and filter(interval_df, acro == "gfs" & Generation == 40000)$pop_fit_max
## W = 2494, p-value < 2.2e-16
## alternative hypothesis: true location shift is greater than 0
```

3.5.1.5 Phenotypic fitness sharing

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'pfs') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation, shape=Generation)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons') +
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'pfs') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  8.20  9.28  9.28  10.1 0.574
## 2 50000         50     0 10.1  11.1 11.0  11.9 0.831
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'pfs' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'pfs' & Generation == 40000)$pop_fit_max,
            alternative = 'g')
```

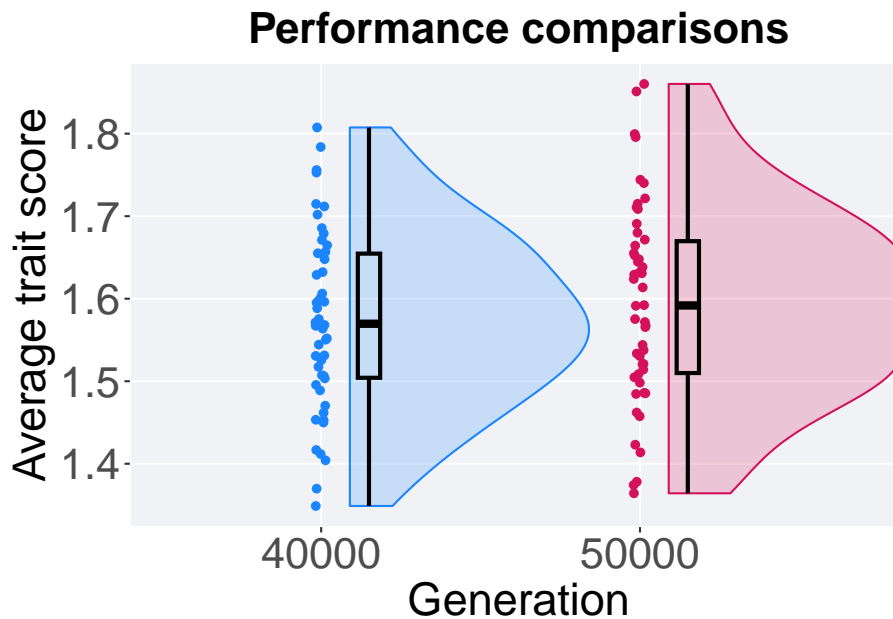
```
##
```

```
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "pfs" & Generation == 50000)$pop_fit_max and filter(interval_df, acro == "pfs" & Generation == 40000)$pop_fit_max
## W = 2498, p-value < 2.2e-16
## alternative hypothesis: true location shift is greater than 0
```

3.5.1.6 Nondominated sorting

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'nds') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generation, shape=Generation)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16)) +
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons') +
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'nds') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50     0  1.35  1.57  1.57  1.81 0.150
## 2 50000         50     0  1.36  1.59  1.59  1.86 0.160
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'nds' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'nds' & Generation == 40000)$pop_fit_max,
            alternative = 't')
```

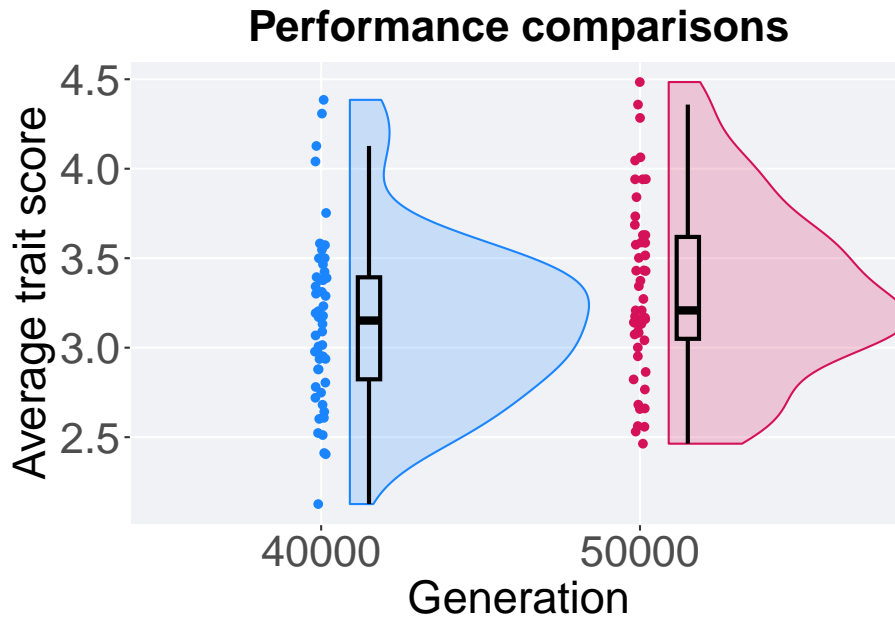
```
##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "nds" & Generation == 50000)$pop_fit_max and fil
## W = 1359, p-value = 0.4545
## alternative hypothesis: true location shift is not equal to 0
```

3.5.1.7 Novelty search

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'nov') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY, fill=Generation, color=Generat
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
```

```
scale_shape_manual(values=c(16,16))+
scale_colour_manual(values = mvc_col, ) +
scale_fill_manual(values = mvc_col) +
ggtitle('Performance comparisons')+
p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'nov') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt   min median   mean   max   IQR
##   <fct>      <int>  <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50      0  2.13  3.15  3.15  4.38 0.570
## 2 50000         50      0  2.46  3.21  3.32  4.48 0.569
```

Results for post-hoc Wilcoxon rank-sum test.

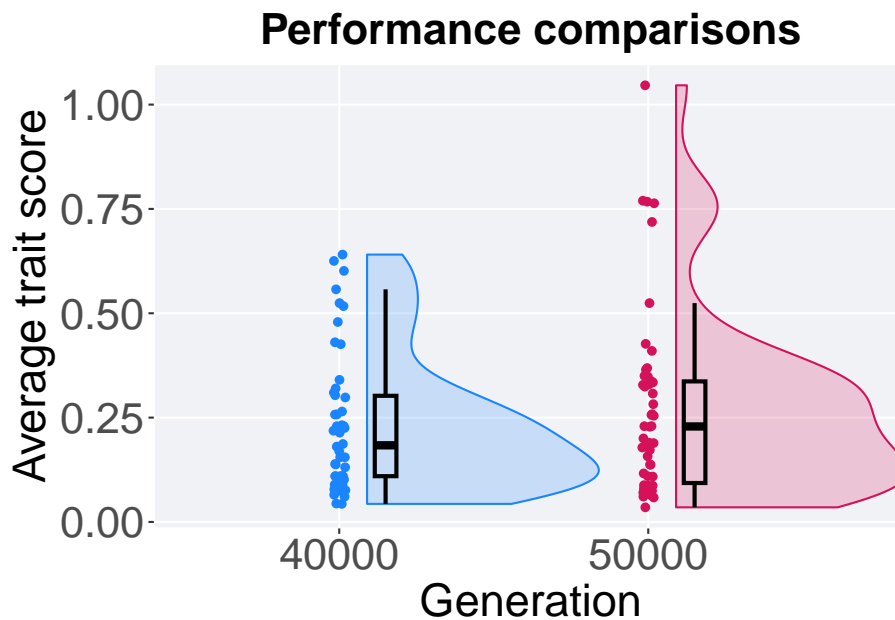
```
wilcox.test(x = filter(interval_df, acro == 'nov' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'nov' & Generation == 40000)$pop_fit_max,
            alternative = 't')
```

```
##
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "nov" & Generation == 50000)$pop_fit_max and fil
## W = 1506, p-value = 0.07818
## alternative hypothesis: true location shift is not equal to 0
```

3.5.1.8 Random

Plot for comparing performances at interval 40,000 & 50,000.

```
filter(interval_df, acro == 'ran') %>%
ggplot(., aes(x=Generation, y=pop_fit_max/DIMENSIONALITY,fill=Generation,color=Generat.
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
  ) +
  scale_x_discrete(
    name="Generation"
  ) +
  scale_shape_manual(values=c(16,16))+
  scale_colour_manual(values = mvc_col, ) +
  scale_fill_manual(values = mvc_col) +
  ggtitle('Performance comparisons')+
  p_theme + theme(legend.position="none")
```



Summary statistics for the best performance.

```
filter(interval_df, acro == 'ran') %>%
  group_by(Generation) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 8
##   Generation count na_cnt    min median  mean   max   IQR
##   <fct>      <int>  <int>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 40000         50      0 0.0433  0.184 0.233 0.641 0.193
## 2 50000         50      0 0.0350  0.229 0.270 1.05  0.243
```

Results for post-hoc Wilcoxon rank-sum test.

```
wilcox.test(x = filter(interval_df, acro == 'ran' & Generation == 50000)$pop_fit_max,
            y = filter(interval_df, acro == 'ran' & Generation == 40000)$pop_fit_max,
            alternative = 't')
```

```
##
```

```
## Wilcoxon rank sum test with continuity correction
##
## data: filter(interval_df, acro == "ran" & Generation == 50000)$pop_fit_max and fil
## W = 1335, p-value = 0.5602
## alternative hypothesis: true location shift is not equal to 0
```


Chapter 4

Contradictory objectives results

Here we present the results for **activation gene coverage** and **satisfactory trait coverage** found by each selection scheme on the contradictory objectives diagnostic with valley crossing integrated. 50 replicates are conducted for each scheme explored.

4.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

4.2 Data setup

```
DIR = paste(DATA_DIR, 'CONTRADICTIONARY_OBJECTIVES/', sep = "", collapse = NULL)
over_time_df <- read.csv(paste(DIR, 'over-time.csv', sep = "", collapse = NULL), header = TRUE, stringsAsFactors = FALSE)
over_time_df$uni_str_pos = over_time_df$uni_str_pos + over_time_df$arc_acti_gene - over_time_df$arc_acti_gene
over_time_df$scheme <- factor(over_time_df$scheme, levels = NAMES)
over_time_df$acro <- factor(over_time_df$acro, levels = ACRO)

best_df <- read.csv(paste(DIR, 'best.csv', sep = "", collapse = NULL), header = TRUE, stringsAsFactors = FALSE)
best_df$acro <- factor(best_df$acro, levels = ACRO)
```

4.3 Activation gene coverage over time

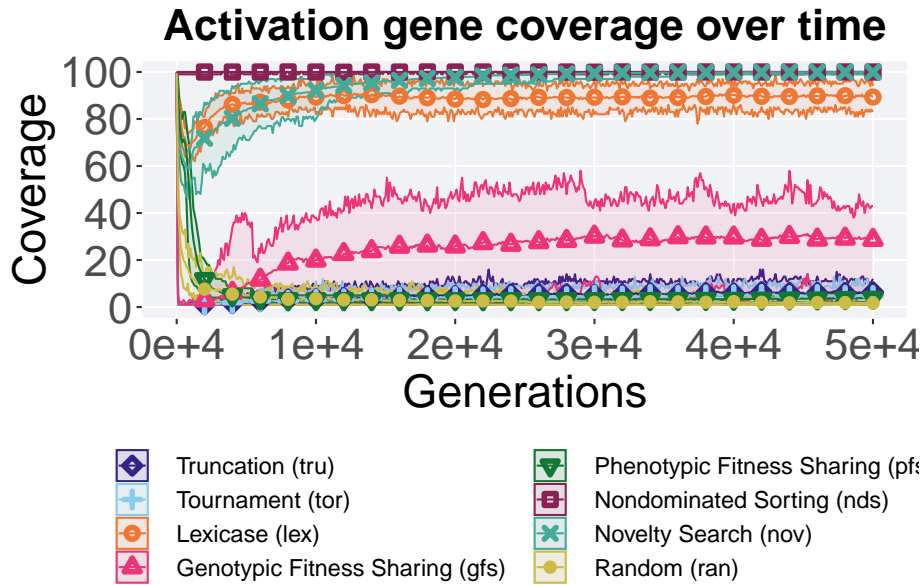
Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = over_time_df %>%
  group_by(scheme, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )

## `summarise()` has grouped output by 'scheme'. You can override using the
## `.groups` argument.

over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = scheme, fill = scheme, color
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 100),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank(),legend.text=element_text(size=12)) +
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  )

over_time_plot
```



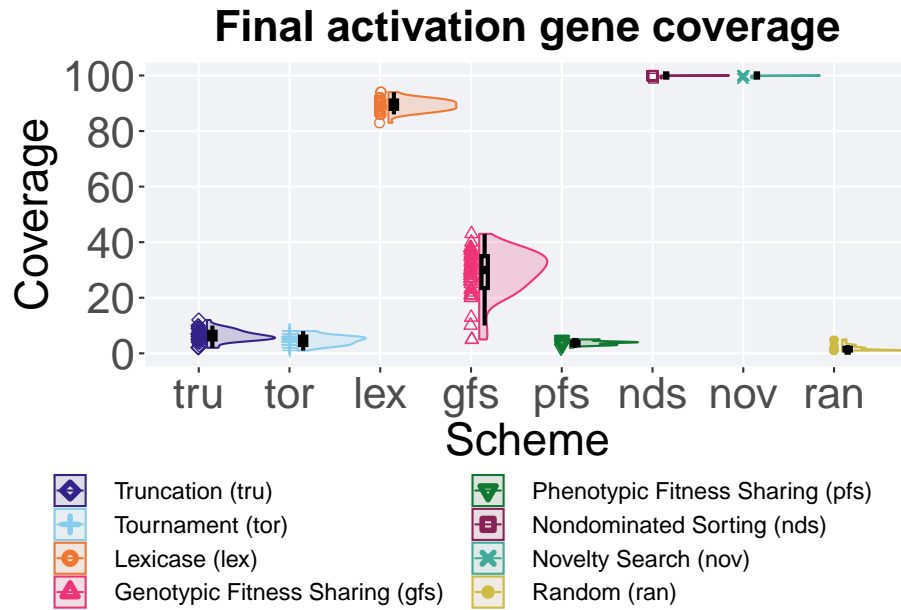
4.4 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000) %>%
  ggplot(., aes(x = acro, y = uni_str_pos, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position
  geom_point(position = position_jitter(width = 0.01, height = 0.1), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 100),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + theme(legend.title=element_blank())
```

```
plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

```
## Warning: Removed 51 rows containing missing values (`geom_point()`).
```



4.4.1 Stats

Summary statistics for the coverage found in the final population.

```
act_coverage = filter(over_time_df, gen == 50000)
act_coverage$acro = factor(act_coverage$acro, levels = c('nov', 'nds', 'lex', 'gfs', 'tor'))
act_coverage %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
```

```
IQR = IQR(uni_str_pos, na.rm = TRUE)
)
```

```
## # A tibble: 8 x 8
##   acro count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 nov     50     0   99    100  99.9   100    0
## 2 nds     50     0   99    100 100.    100    0
## 3 lex     50     0   83     89  89.3    94    3
## 4 gfs     50     0    5     30  28.7    43   11.5
## 5 tor     50     0    1      5   4.72     8    3
## 6 tru     50     0    2      6   6.28    12   2.75
## 7 pfs     50     0    2      4   3.84     5    1
## 8 ran     50     0    1      1   1.72     5    1
```

Kruskal-Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ acro, data = act_coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by acro
## Kruskal-Wallis chi-squared = 373.95, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$acro, p.adjust.method = "bonf",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: act_coverage$uni_str_pos and act_coverage$acro
##
##      nov      nds      lex      gfs      tor      tru      pfs
## nds 1.00      -      -      -      -      -      -
## lex < 2e-16 < 2e-16 -      -      -      -      -
## gfs < 2e-16 < 2e-16 < 2e-16 -      -      -      -
## tor < 2e-16 < 2e-16 < 2e-16 1.3e-15 -      -      -
## tru < 2e-16 < 2e-16 < 2e-16 7.3e-15 1.00      -      -
## pfs < 2e-16 < 2e-16 < 2e-16 < 2e-16 0.05    3.4e-10 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.4e-12 1.9e-15 1.8e-12
##
## P value adjustment method: bonferroni
```

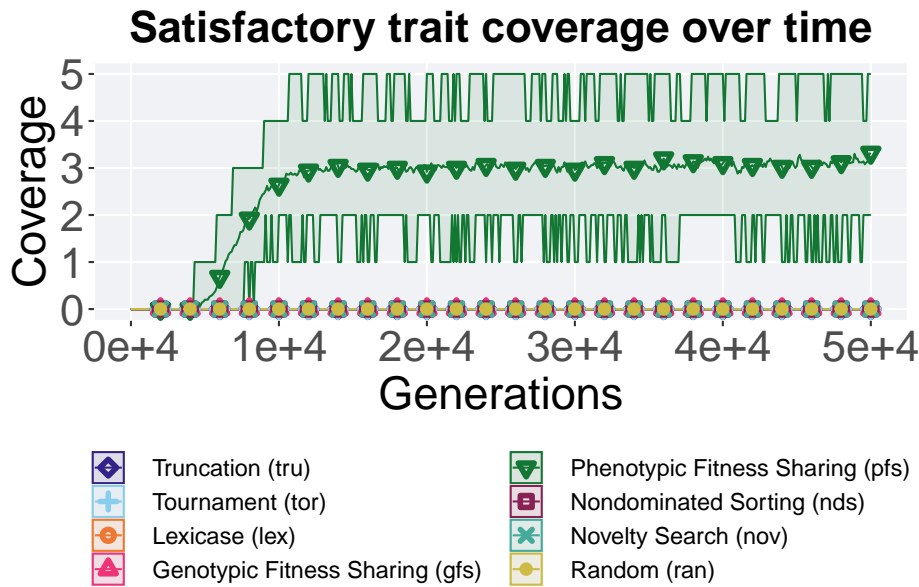
4.5 Satisfactory trait coverage over time

Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = over_time_df %>%
  group_by(scheme, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )
```

`summarise()` has grouped output by 'scheme'. You can override using the
`.groups` argument.

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = scheme, fill = scheme, color
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time')+
  p_theme + theme(legend.title=element_blank(),legend.text=element_text(size=12)) +
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  )
over_time_plot
```



4.6 Best satisfactory trait coverage throughout

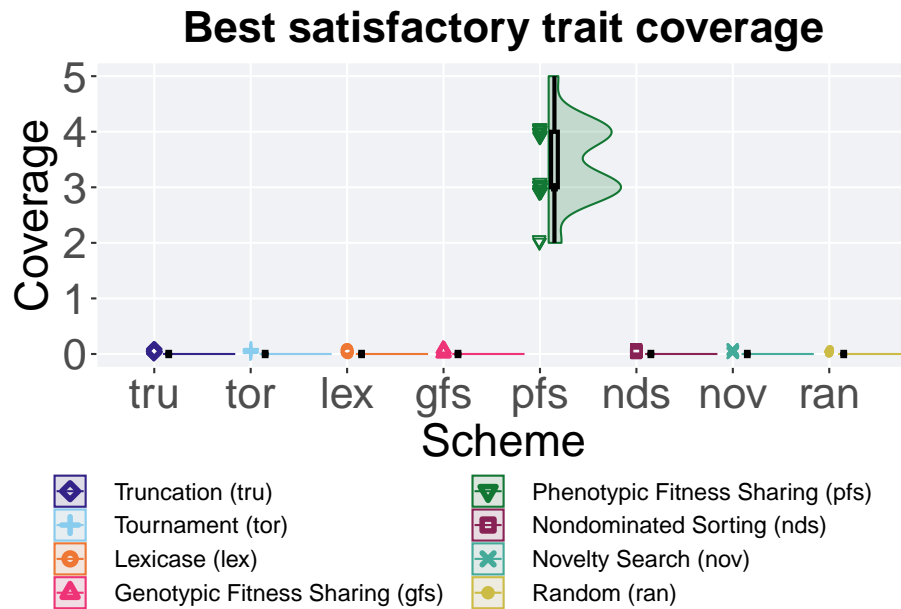
Best satisfactory trait coverage reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_uni_obj') %>%
  ggplot(., aes(x = acro, y = val, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = 1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = 'dodge') +
  geom_point(position = position_jitter(width = 0.01, height = 0.1), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name = "Coverage",
    limits = c(0, 5)
  ) +
  scale_x_discrete(
    name = "Scheme"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best satisfactory trait coverage') +
  p_theme + theme(legend.title = element_blank())

plot_grid(
```

```
plot +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

```
## Warning: Removed 174 rows containing missing values (`geom_point()`).
```



4.6.1 Stats

Summary statistics for the best coverage.

```
sat_coverage = filter(best_df, var == 'pop_uni_obj')
sat_coverage$acro = factor(sat_coverage$acro, levels = c('pfs','nds','lex','gfs','tor'))
sat_coverage %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
    IQR = IQR(val, na.rm = TRUE)
```



```
)

## # A tibble: 8 x 8
##   acro count na_cnt   min median  mean  max  IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 pfs     50      0     2     3  3.44     5     1
## 2 nds     50      0     0     0  0.00     0     0
## 3 lex     50      0     0     0  0.00     0     0
## 4 gfs     50      0     0     0  0.00     0     0
## 5 tor     50      0     0     0  0.00     0     0
## 6 tru     50      0     0     0  0.00     0     0
## 7 nov     50      0     0     0  0.00     0     0
## 8 ran     50      0     0     0  0.00     0     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ acro, data = sat_coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 397.02, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = sat_coverage$val, g = sat_coverage$acro, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  sat_coverage$val and sat_coverage$acro
##
##      pfs    nds lex gfs tor tru nov
## nds <2e-16 -   -   -   -   -
## lex <2e-16 1   -   -   -   -
## gfs <2e-16 1   1   -   -   -
## tor <2e-16 1   1   1   -   -
## tru <2e-16 1   1   1   1   -
## nov <2e-16 1   1   1   1   1
## ran <2e-16 1   1   1   1   1
##
## P value adjustment method: bonferroni
```

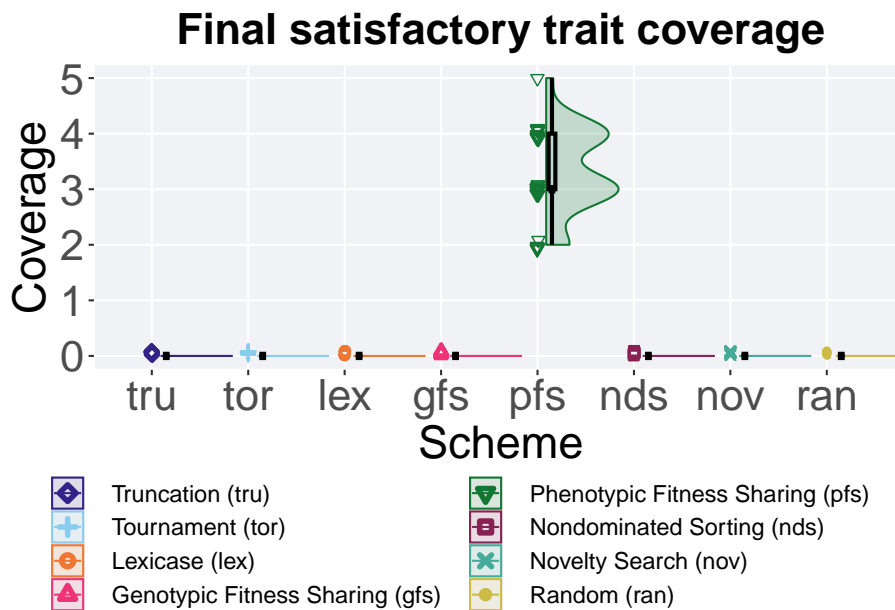
4.7 Final satisfactory trait coverage

Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000) %>%
  ggplot(., aes(x = acro, y = pop_uni_obj, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.01, height = 0.1), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage') +
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

```
## Warning: Removed 168 rows containing missing values (`geom_point()`).
```



4.7.1 Stats

Summary statistics for the coverage found in the final population.

```
act_coverage = filter(over_time_df, gen == 50000)
act_coverage$acro = factor(act_coverage$acro, levels = c('pfs', 'nds', 'lex', 'gfs', 'tor', 'tru', 'nov'))
act_coverage %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_uni_obj)),
    min = min(pop_uni_obj, na.rm = TRUE),
    median = median(pop_uni_obj, na.rm = TRUE),
    mean = mean(pop_uni_obj, na.rm = TRUE),
    max = max(pop_uni_obj, na.rm = TRUE),
    IQR = IQR(pop_uni_obj, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   acro  count na_cnt  min median  mean  max  IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 pfs     50     0     2     3  3.32     5     1
## 2 nds     50     0     0     0  0.00     0     0
## 3 lex     50     0     0     0  0.00     0     0
## 4 gfs     50     0     0     0  0.00     0     0
```

```
## 5 tor      50      0      0      0 0      0      0
## 6 tru      50      0      0      0 0      0      0
## 7 nov      50      0      0      0 0      0      0
## 8 ran      50      0      0      0 0      0      0
```

Kruskal-Wallis test illustrates evidence of statistical differences.

```
kruskal.test(pop_uni_obj ~ acro, data = act_coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_uni_obj by acro
## Kruskal-Wallis chi-squared = 396.97, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = act_coverage$pop_uni_obj, g = act_coverage$acro, p.adjust.method = 'p.adjust.method',
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: act_coverage$pop_uni_obj and act_coverage$acro
##
##      pfs      nds lex gfs tor tru nov
## nds <2e-16 -    -    -    -    -
## lex <2e-16 1    -    -    -    -
## gfs <2e-16 1    1    -    -    -
## tor <2e-16 1    1    1    -    -
## tru <2e-16 1    1    1    1    -
## nov <2e-16 1    1    1    1    1
## ran <2e-16 1    1    1    1    1
##
## P value adjustment method: bonferroni
```

4.8 Best trait value over time

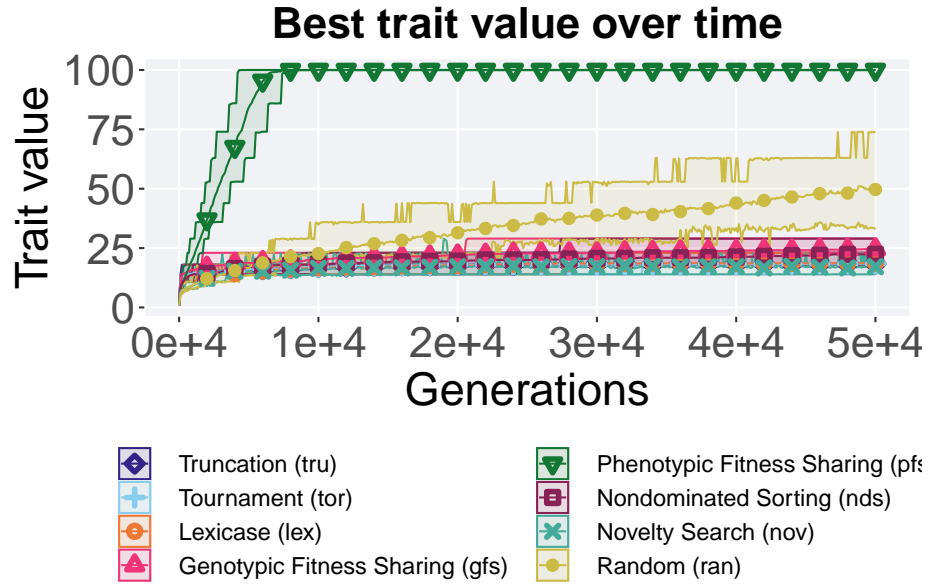
Best trait value in a population over time. Data points on the graph is the average trait value across 50 replicates every 2000 generations. Shading comes from the best and worse trait value across 50 replicates.

```
lines = over_time_df %>%
  group_by(scheme, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
```

```
)
```

```
## `summarise()` has grouped output by 'scheme'. You can override using the
## `.groups` argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = scheme, fill = scheme, color = scheme,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Trait value"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best trait value over time')+
  p_theme + theme(legend.title=element_blank(),legend.text=element_text(size=12)) +
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  )
over_time_plot
```



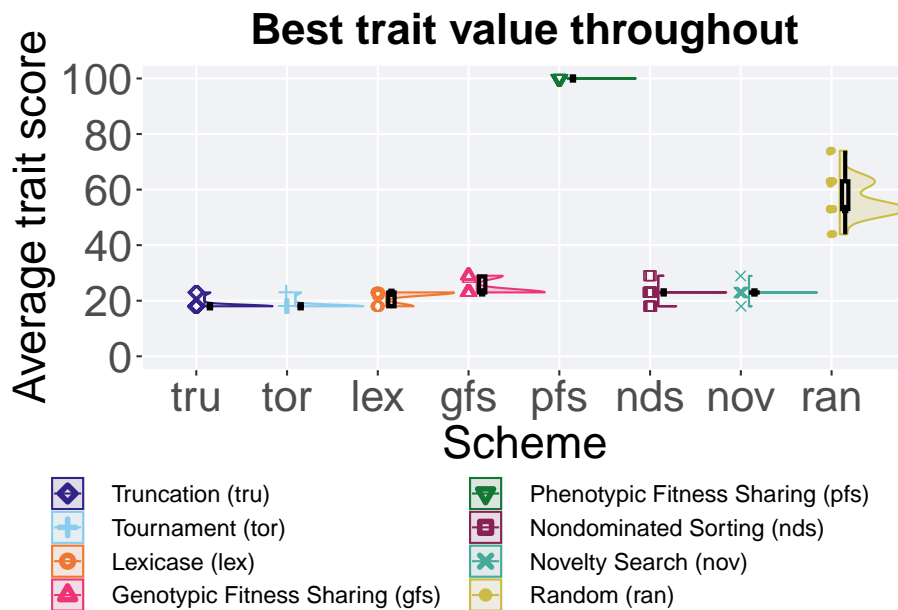
4.9 Best trait value throughout

Best trait value reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max') %>%
  ggplot(., aes(x = acro, y = val, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 100),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best trait value throughout') +
  p_theme + theme(legend.title=element_blank())
```

```
plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

```
## Warning: Removed 14 rows containing missing values (`geom_point()`).
```



4.9.1 Stats

Summary statistics for the best trait value.

```
trait_val = filter(best_df, var == 'pop_fit_max')
trait_val$acro = factor(trait_val$acro, levels = c('pfs', 'ran', 'gfs', 'nov', 'nds', 'lex', 'tru', 'tor'))
trait_val %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
```

```
IQR = IQR(val, na.rm = TRUE)
)
```

```
## # A tibble: 8 x 8
##   acro count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 pfs     50     0 100.   100   100.  100  0.000100
## 2 ran     50     0 43.9   53.0   56.7  74.0  9.99
## 3 gfs     50     0 23.0    23    24.7   29   5.76
## 4 nov     50     0 18.0   23.0   23.0  28.9  0.0528
## 5 nds     50     0 18      23    22.4   29   0.00328
## 6 lex     50     0 18      22.9  21.5   23   4.98
## 7 tru     50     0 18.0    18    18.4   23    0
## 8 tor     50     0 18.0    18    18.4   23    0
```

Kruskal-Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ acro, data = trait_val)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: val by acro
## Kruskal-Wallis chi-squared = 337.68, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = trait_val$val, g = trait_val$acro, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: trait_val$val and trait_val$acro
##
##      pfs      ran      gfs      nov      nds      lex      tru
## ran < 2e-16 -          -          -          -          -          -
## gfs < 2e-16 < 2e-16 -          -          -          -          -
## nov < 2e-16 < 2e-16 3.2e-16 -          -          -          -
## nds < 2e-16 < 2e-16 0.00015 1.00000 -          -          -
## lex < 2e-16 < 2e-16 3.7e-15 0.00722 1.4e-05 -          -
## tru < 2e-16 < 2e-16 < 2e-16 4.5e-12 3.0e-11 1.0e-07 -
## tor < 2e-16 < 2e-16 < 2e-16 6.3e-12 7.2e-11 9.4e-08 1.00000
##
## P value adjustment method: bonferroni
```


Chapter 5

Multi-path exploration results

Here we present the results for **best performances** found by each selection scheme on the multi-path exploration diagnostic with valley crossing integrated. 50 replicates are conducted for each scheme explored.

5.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

5.2 Data setup

```
DIR = paste(DATA_DIR, 'MULTIPATH_EXPLORATION/', sep = "", collapse = NULL)
over_time_df <- read.csv(paste(DIR, 'over-time.csv', sep = "", collapse = NULL), header = TRUE, stringsAsFactors = FALSE)
over_time_df$uni_str_pos = over_time_df$uni_str_pos + over_time_df$arc_acti_gene - over_time_df$arc_acti_gene
over_time_df$scheme <- factor(over_time_df$scheme, levels = NAMES)
over_time_df$acro <- factor(over_time_df$acro, levels = ACRO)

best_df <- read.csv(paste(DIR, 'best.csv', sep = "", collapse = NULL), header = TRUE, stringsAsFactors = FALSE)
best_df$acro <- factor(best_df$acro, levels = ACRO)
```

5.3 Activation gene coverage over time

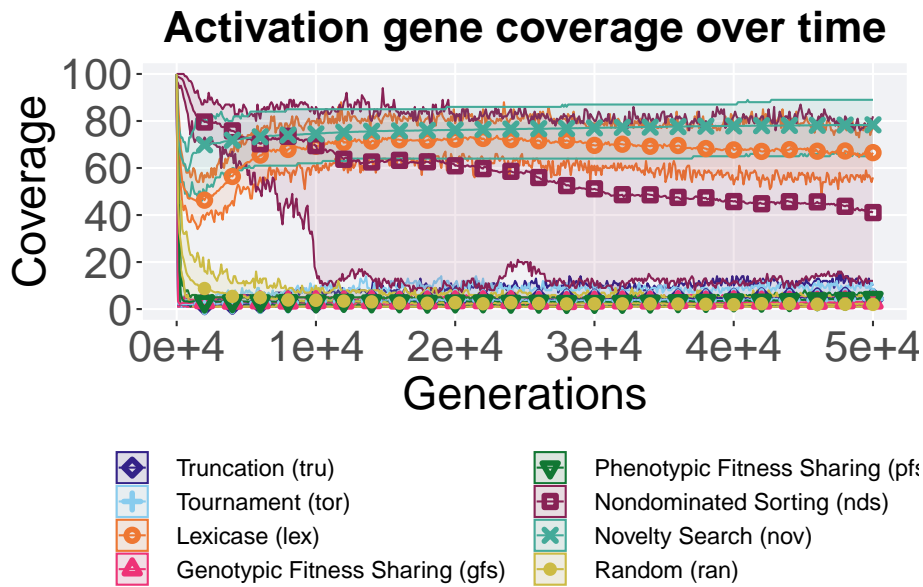
Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = over_time_df %>%
  group_by(scheme, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )

## `summarise()` has grouped output by 'scheme'. You can override using the
## `.groups` argument.

over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = scheme, fill = scheme, color
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 100),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank(),legend.text=element_text(size=12)) +
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  )

over_time_plot
```

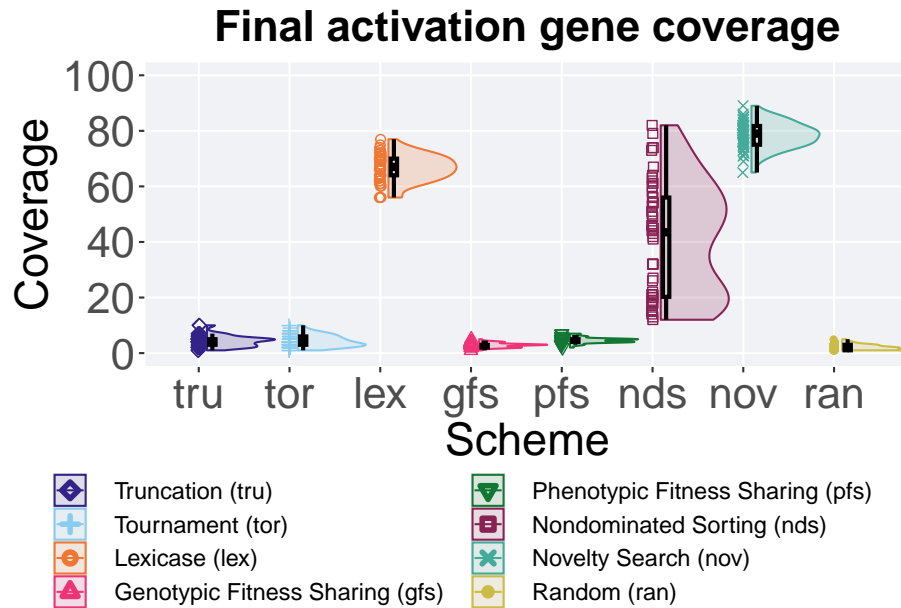


5.4 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000) %>%
  ggplot(., aes(x = acro, y = uni_str_pos, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = .1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = 0.01, height = 0.1), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 100),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + theme(legend.title=element_blank())
```

```
plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



5.4.1 Stats

Summary statistics for the coverage found in the final population.

```
act_coverage = filter(over_time_df, gen == 50000)
act_coverage$acro = factor(act_coverage$acro, levels = c('nov', 'lex', 'nds', 'tor', 'tru'))
act_coverage %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   acro count na_cnt   min median  mean   max   IQR
##   <fct> <int>   <int> <int>   <dbl> <dbl> <int> <dbl>
## 1 nov     50     0    65    79   78.4    89  6.75
## 2 lex     50     0    56    67   66.5    77   6
## 3 nds     50     0    12   43.5  41.0    82  35.8
## 4 tor     50     0     1     4    4.4    10   3
## 5 tru     50     0     1     4    4.28   10   2
## 6 gfs     50     0     1     3    2.88    5   1
## 7 pfs     50     0     2     5    4.64    7   1
## 8 ran     50     0     1     2    1.98    5   2
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ acro, data = act_coverage)
```

```
##
##   Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by acro
## Kruskal-Wallis chi-squared = 333.43, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$acro, p.adjust.method = "bonf",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##   Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$acro
##
##      nov      lex      nds      tor      tru      gfs      pfs
## lex 3.0e-14 -          -          -          -          -
## nds 1.5e-14 2.0e-09 -          -          -          -
## tor < 2e-16 < 2e-16 < 2e-16 -          -          -
## tru < 2e-16 < 2e-16 < 2e-16 1.0000 -          -
## gfs < 2e-16 < 2e-16 < 2e-16 0.0032 0.0012 -          -
## pfs < 2e-16 < 2e-16 < 2e-16 1.0000 1.0000 1.0000 -
## ran < 2e-16 < 2e-16 < 2e-16 1.6e-08 1.1e-08 5.8e-05 2.2e-14
##
## P value adjustment method: bonferroni
```

5.5 Performance over time

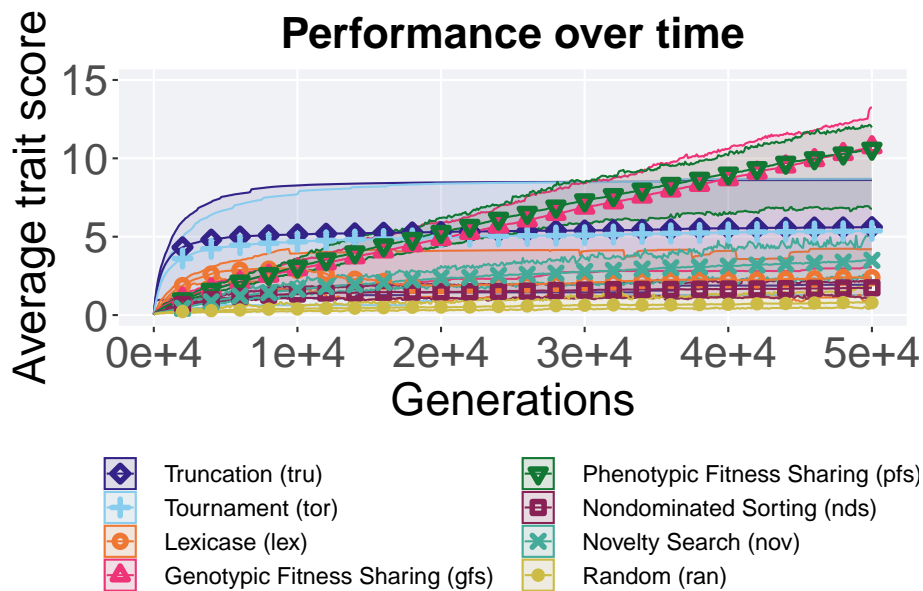
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes

from the best and worse performance across 50 replicates.

```
lines = over_time_df %>%
  group_by(scheme, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

`summarise()` has grouped output by 'scheme'. You can override using the
`.groups` argument.

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = scheme, fill = scheme, color
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %>= 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 15),
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank(),legend.text=element_text(size=12)) +
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  )
over_time_plot
```



5.6 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

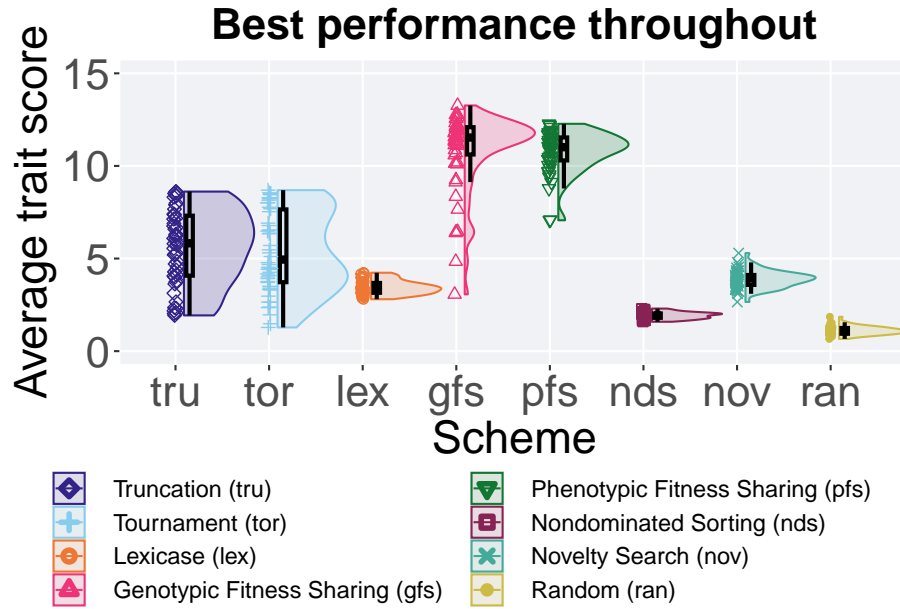
```
plot = filter(best_df, var == 'pop_fit_max') %>%
  ggplot(., aes(x = acro, y = val / DIMENSIONALITY, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = .1) +
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = .09, y = 0)) +
  geom_point(position = position_jitter(width = .02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(0, 15),
  ) +
  scale_x_discrete(
    name = "Scheme"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout') +
  p_theme + theme(legend.title = element_blank())

plot_grid(
  plot +
```

```

theme(legend.position="none"),
legend,
nrow=2,
rel_heights = c(3,1)
)

```



5.6.1 Stats

Summary statistics for the best performance.

```

performance = filter(best_df, var == 'pop_fit_max')
performance$acro = factor(performance$acro, levels = c('gfs','pfs','tru','tor','nov',''))
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )

```

A tibble: 8 x 8


```
##   acro count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 gfs      50      0 3.07   11.5  10.8  13.3  1.48
## 2 pfs      50      0 7.07   11.0  10.8  12.3  1.24
## 3 tru      50      0 1.93    5.82  5.61  8.62  3.24
## 4 tor      50      0 1.28    4.94  5.37  8.69  3.93
## 5 nov      50      0 2.66    3.92  3.88  5.29  0.536
## 6 lex      50      0 2.80    3.41  3.45  4.23  0.531
## 7 nds      50      0 1.59    1.95  1.92  2.30  0.235
## 8 ran      50      0 0.659   1.08  1.11  1.86  0.311
```

Kruskal-Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ acro, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 341.54, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$val, g = performance$acro, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$acro
##
##      gfs      pfs      tru      tor      nov      lex      nds
## pfs 0.3635 -          -          -          -          -          -
## tru 4.1e-13 5.9e-16 -          -          -          -          -
## tor 3.2e-13 5.2e-16 1.0000 -          -          -          -
## nov 1.9e-15 < 2e-16 3.2e-05 0.0055 -          -          -
## lex 1.1e-15 < 2e-16 3.1e-07 1.6e-05 6.3e-05 -          -
## nds < 2e-16 < 2e-16 2.5e-15 1.2e-13 < 2e-16 < 2e-16 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 3.5e-16
##
## P value adjustment method: bonferroni
```