

# Supplemental Material: Base Diagnostics

Jose Guadalupe Hernandez

2023-08-21



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	About our supplemental material . . . . .	5
1.2	Contributing authors . . . . .	5
1.3	Computer Setup . . . . .	5
1.4	Experimental setup . . . . .	6
<b>2</b>	<b>Exploitation rate results</b>	<b>9</b>
2.1	Data setup . . . . .	9
2.2	Performance over time . . . . .	9
2.3	Best performance throughout . . . . .	11
2.4	Generation satisfactory solution found . . . . .	14
<b>3</b>	<b>Ordered exploitation results</b>	<b>17</b>
3.1	Data setup . . . . .	17
3.2	Performance over time . . . . .	17
3.3	Best performance throughout . . . . .	19
3.4	Generation satisfactory solution found . . . . .	21
3.5	Streaks over time . . . . .	24
3.6	Longest streak throughout . . . . .	26
<b>4</b>	<b>Contradictory objectives results</b>	<b>29</b>
4.1	Data setup . . . . .	29
4.2	Activation gene coverage over time . . . . .	29
4.3	Final activation gene coverage . . . . .	31
4.4	Satisfactory trait coverage over time . . . . .	34
4.5	Final satisfactory trait coverage . . . . .	35
<b>5</b>	<b>Multi-path exploration results</b>	<b>39</b>
5.1	Data setup . . . . .	39
5.2	Activation gene coverage over time . . . . .	39
5.3	Final activation gene coverage . . . . .	41
5.4	Performance over time . . . . .	43
5.5	Best performance throughout . . . . .	45



# Chapter 1

## Introduction

This is the supplemental material for experiments with basic diagnostics.

### 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-Sche
```

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

```
# DATA_DIR = '~/Desktop/Repositories/ECJ-2023-Suite-Of-Diagnostic-Metrics-For-Characte
```

```
# graph variables
```

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

```
cb_palette <- c('#332288', '#8CCEE', '#EE7733', '#EE3377', '#117733', '#882255', '#44AA99',
```

```

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.

```

```

# colors for streak plots
STK_SHAPE = c(2,6,0,4,20,1)
stk_cb_palette <- c('#EE3377', '#117733', '#882255', '#44AA99', '#CCBB44', '#000000')

# 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)'

```





## Chapter 2

# Exploitation rate results

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

### 2.1 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)

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

### 2.2 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
  )

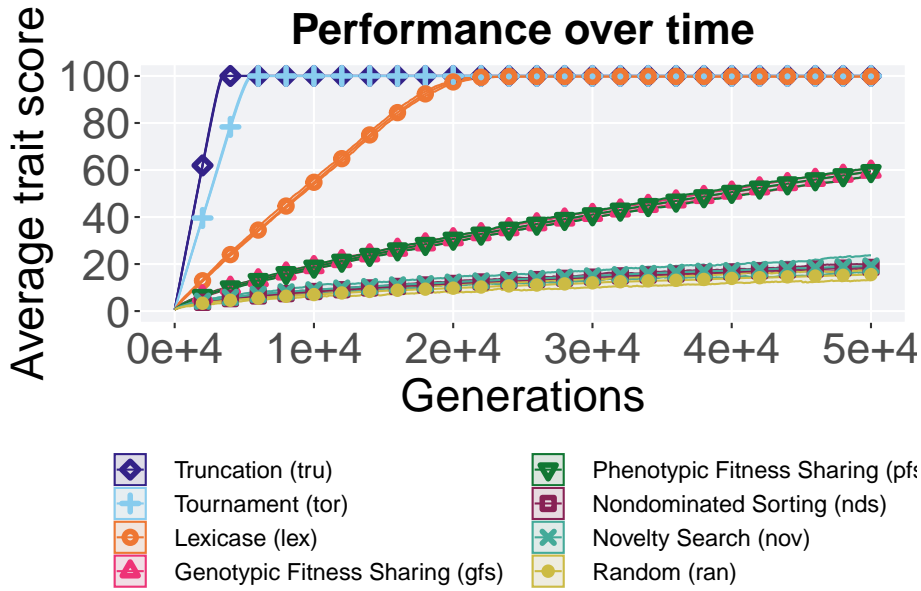
```

## `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, 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('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

```



## 2.3 Best performance throughout

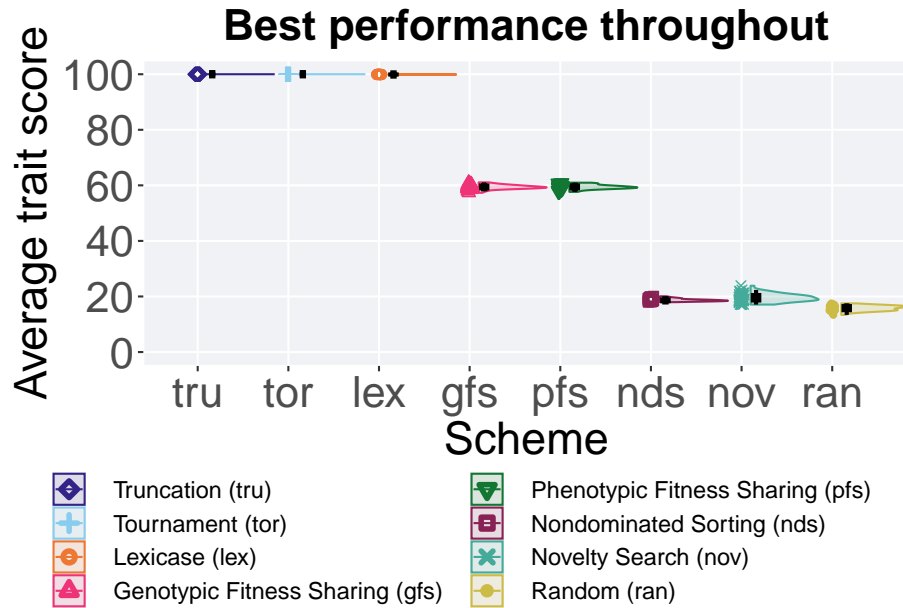
Best performance reached throughout 50,000 generations in a population.

```
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 = .1, 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 = 0.03, height = 0.02), 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 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.
```

```
## Warning: Removed 42 rows containing missing values (`geom_point()`).
```



### 2.3.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max')
performance$acro = factor(performance$acro, levels = c('tru','tor','lex','gfs','pfs','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 tru     50      0 100    100   100   100    0
## 2 tor     50      0 100    100   100   100    0
## 3 lex     50      0 99.9   99.9  99.9  99.9  0.0154
## 4 gfs     50      0 57.3   59.3  59.4  61.1  0.984
## 5 pfs     50      0 57.6   59.4  59.5  61.0  1.02
## 6 nov     50      0 17.1   19.5  19.5  23.9  1.95
## 7 nds     50      0 18.0   18.6  18.7  20.1  0.603
## 8 ran     50      0 13.4   15.9  15.8  17.5  1.46

```

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 = 385.26, 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
##
##      tru   tor   lex   gfs   pfs   nov   nds
## tor 1.000  -     -     -     -     -     -
## lex <2e-16 <2e-16 -     -     -     -     -
## gfs <2e-16 <2e-16 <2e-16 -     -     -     -
## pfs <2e-16 <2e-16 <2e-16 1.000 -     -     -
## nov <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -     -
## nds <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 0.018 -
## ran <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16

```

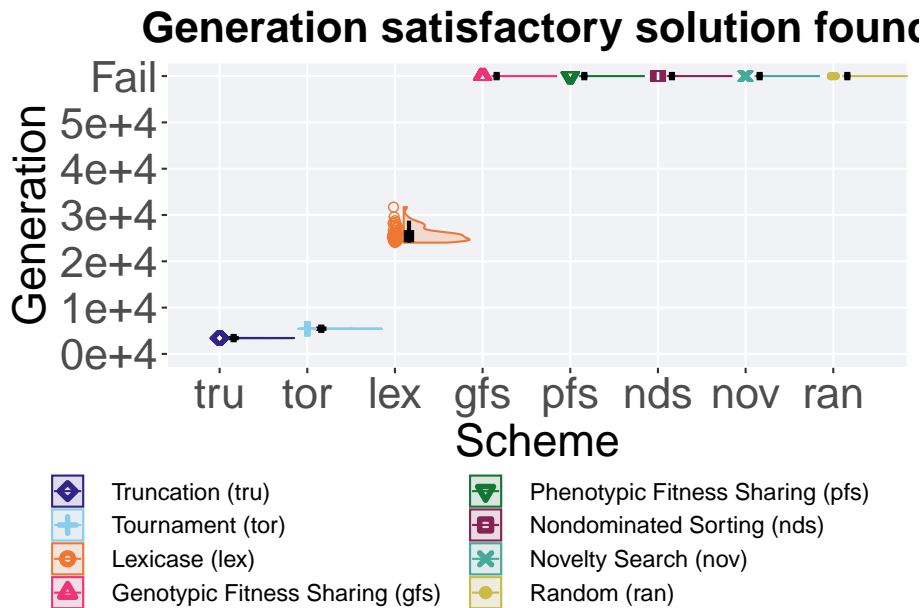
```
##
## P value adjustment method: bonferroni
```

## 2.4 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
plot = sati_df %>%
  ggplot(., aes(x = acro, y = gen , color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha = 0.5) +
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60001),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4", "Fail")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found') +
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )

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



### 2.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
ssf = filter(sati_df, gen <= GENERATIONS)
ssf$acro = factor(ssf$acro, levels = c('tru', 'tor', 'lex'))
ssf %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(gen)),
    min = min(gen, na.rm = TRUE),
    median = median(gen, na.rm = TRUE),
    mean = mean(gen, na.rm = TRUE),
    max = max(gen, na.rm = TRUE),
    IQR = IQR(gen, na.rm = TRUE)
  )
```

```
## # A tibble: 3 x 8
##   acro  count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 tru     50     0  3392  3422  3423.  3475   26
## 2 tor     50     0  5390  5444.  5447.  5509  43.2
## 3 lex     50     0 24036 25626. 25883. 31709 1739.
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```

kruskal.test(gen ~ acro, data = ssf)

##
##  Kruskal-Wallis rank sum test
##
## data:  gen by acro
## Kruskal-Wallis chi-squared = 132.46, df = 2, p-value < 2.2e-16
Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.
pairwise.wilcox.test(x = ssf$gen, g = ssf$acro, p.adjust.method = "bonferroni",
                    paired = FALSE, conf.int = FALSE, alternative = 'g')

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  ssf$gen and ssf$acro
##
##      tru      tor
## tor <2e-16 -
## lex <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```



## Chapter 3

# Ordered exploitation results

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

### 3.1 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)

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

### 3.2 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
  )

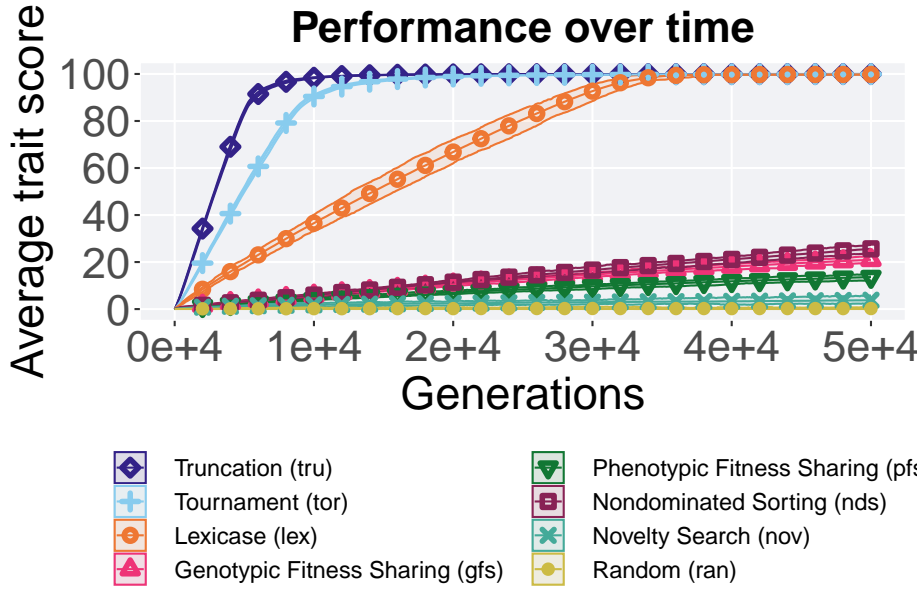
```

## `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, 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('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

```

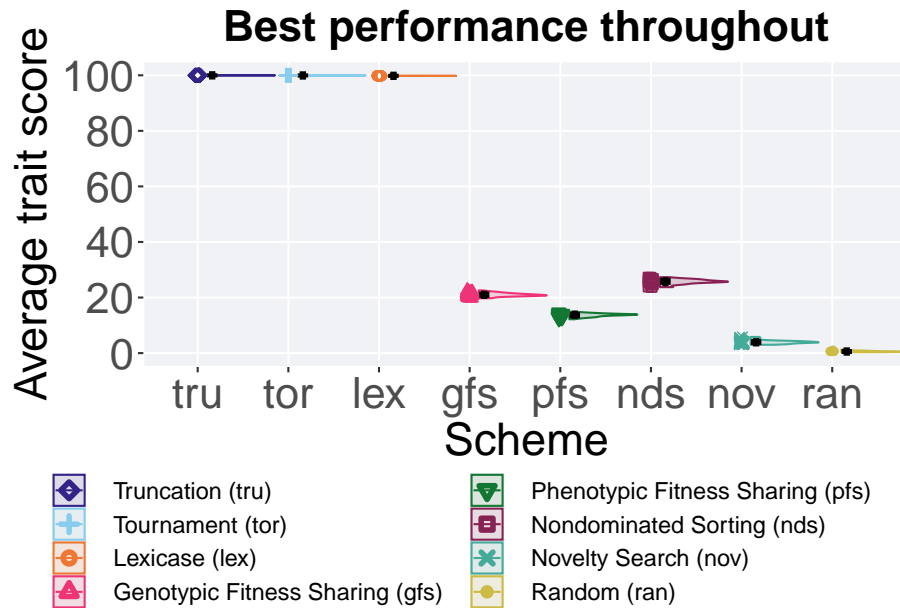


### 3.3 Best performance throughout

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 = .1, 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 = 0.03, height = 0.02), 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 performance throughout') +
  p_theme + theme(legend.title=element_blank())
```

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



### 3.3.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max')
performance$acro = factor(performance$acro, levels = c('tru','tor','lex','nds','gfs','pfs','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 tru     50      0  100.    100.    100.    100.    0.00168
## 2 tor     50      0  99.9    99.9    99.9    99.9    0.00650
## 3 lex     50      0  99.7    99.8    99.8    99.9    0.0247
## 4 nds     50      0  23.7    25.7    25.7    27.3    0.972
## 5 gfs     50      0  19.7    21.0    21.0    22.6    0.754
## 6 pfs     50      0  12.2    13.8    13.7    14.9    0.712
## 7 nov     50      0   3.00    3.90    4.00    5.83    0.666
## 8 ran     50      0   0.318   0.569   0.605    1.31    0.279
```

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 = 392.77, 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
##
##      tru    tor    lex    nds    gfs    pfs    nov
## tor <2e-16 -      -      -      -      -      -
## lex <2e-16 <2e-16 -      -      -      -      -
## nds <2e-16 <2e-16 <2e-16 -      -      -
## gfs <2e-16 <2e-16 <2e-16 <2e-16 -      -
## pfs <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -      -
## nov <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
```

### 3.4 Generation satisfactory solution found

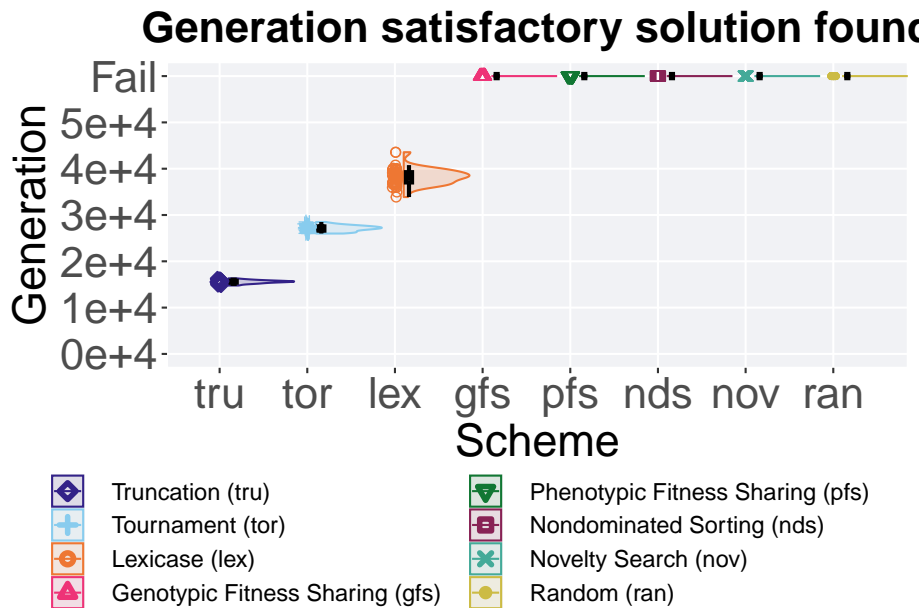
First generation a satisfactory solution is found throughout the 50,000 generations.

```

plot = sati_df %>%
  ggplot(., aes(x = acro, y = gen , color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha = 0.5) +
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60001),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4", "Fail")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found') +
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )

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

```



### 3.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
ssf = filter(sati_df, gen <= GENERATIONS)
ssf$acro = factor(ssf$acro, levels = c('tru', 'tor', 'lex'))
ssf %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(gen)),
    min = min(gen, na.rm = TRUE),
    median = median(gen, na.rm = TRUE),
    mean = mean(gen, na.rm = TRUE),
    max = max(gen, na.rm = TRUE),
    IQR = IQR(gen, na.rm = TRUE)
  )
```

```
## # A tibble: 3 x 8
##   acro  count na_cnt  min median  mean  max  IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 tru     50     0 14776 15585 15570. 16317 420.
## 2 tor     50     0 25996 27138 27105. 28495 913.
## 3 lex     50     0 33877 38288. 38265. 43565 2215.
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(gen ~ acro, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data:  gen by acro
## Kruskal-Wallis chi-squared = 132.45, df = 2, p-value < 2.2e-16

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

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  ssf$gen and ssf$acro
##
##      tru      tor
## tor <2e-16 -
## lex <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

### 3.5 Streaks over time

Longest streak of the best solution found in a population over time. A maximum streak value of 100 and a minimum streak value of 1 is possible. 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 = filter(over_time_df, acro != 'tor' & acro != 'tru' & acro != 'lex') %>%
  group_by(scheme, gen) %>%
  dplyr::summarise(
    min = min(ele_stk_cnt),
    mean = mean(ele_stk_cnt),
    max = max(ele_stk_cnt)
  )

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

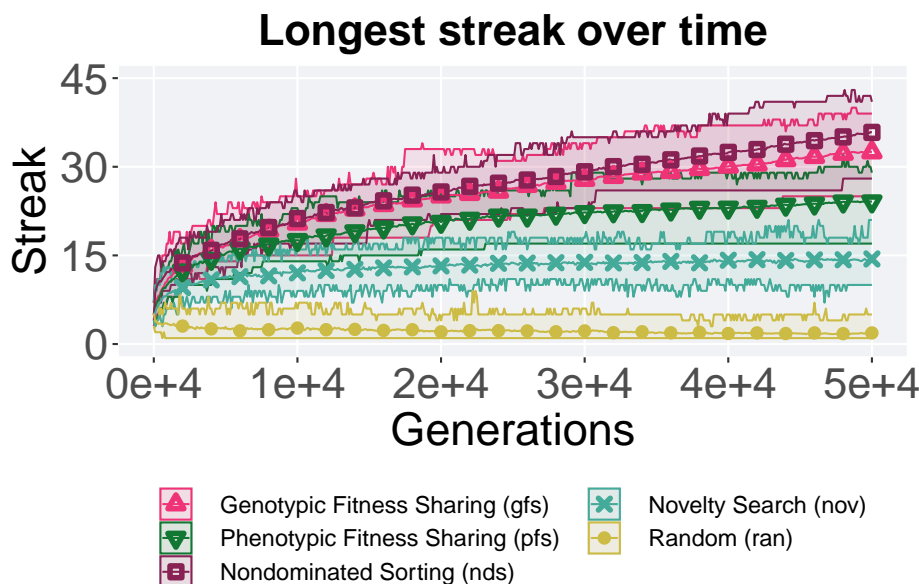
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = scheme, fill = scheme, color
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
```



```

    name="Streak",
    limits=c(0, 45),
    breaks=seq(0,45, 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=STK_SHAPE)+
scale_colour_manual(values = stk_cb_palette) +
scale_fill_manual(values = stk_cb_palette) +
ggtitle('Longest streak 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

```

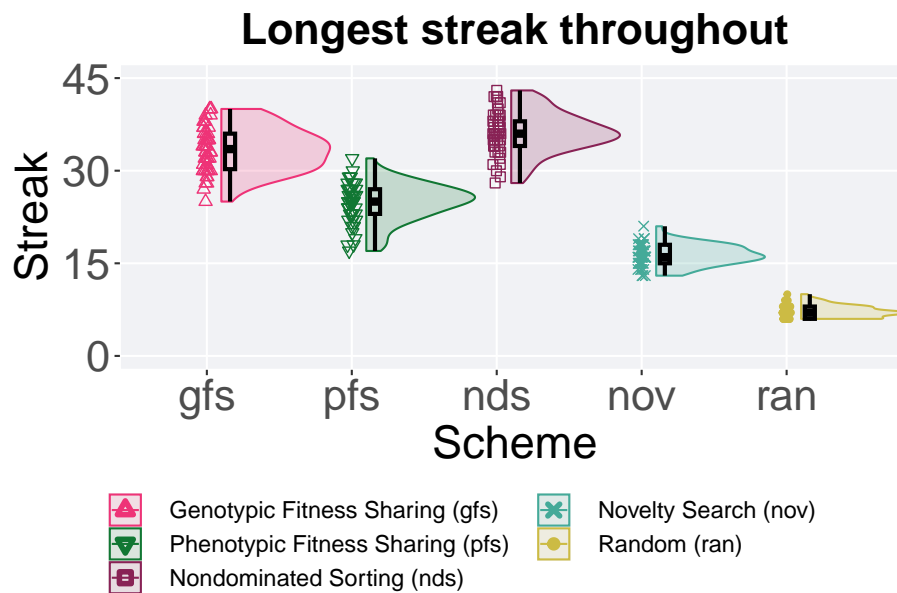


### 3.6 Longest streak throughout

Longest streak of the best solution found in the population throughout 50,000 generations.

```
plot = filter(best_df, var == 'ele_stk_cnt' & acro != 'tor' & acro != 'tru' & acro != 'fals')
ggplot(., aes(x = acro, y = val, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha = 0.5) +
  scale_y_continuous(
    name="Streak",
    limits=c(0, 45),
    breaks=seq(0,45, 15)
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=STK_SHAPE) +
  scale_colour_manual(values = stk_cb_palette) +
  scale_fill_manual(values = stk_cb_palette) +
  ggtitle('Longest streak throughout') +
  p_theme + theme(legend.title=element_blank())

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



### 3.6.1 Stats

Summary statistics for the longest streak

```
streak = filter(best_df, var == 'ele_stk_cnt' & acro != 'tor' & acro != 'tru' & acro != 'lex')
streak$acro = factor(streak$acro, levels = c('nds', 'gfs', 'pfs', 'nov', 'ran'))
streak %>%
  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: 5 x 8
##   acro  count na_cnt  min median  mean  max  IQR
##   <fct> <int>  <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 nds     50     0    28   36   36.2   43    4
## 2 gfs     50     0    25  33.5  33.4   40  5.75
## 3 pfs     50     0    17   25   24.8   32    4
## 4 nov     50     0    13   16   16.3   21    3
```

```
## 5 ran      50      0      6      7      7.18     10     2
```

Kruskal-Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 226.43, df = 4, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  streak$val and streak$acro
##
##      nds      gfs      pfs      nov
## gfs 0.0017 -          -          -
## pfs < 2e-16 2.7e-15 -          -
## nov < 2e-16 < 2e-16 3.1e-16 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

## 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. 50 replicates are conducted for each scheme explored.

### 4.1 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, st
over_time_df$uni_str_pos = over_time_df$uni_str_pos + over_time_df$arc_acti_gene - over_time_df$
over_time_df$scheme <- factor(over_time_df$scheme, levels = NAMES)
over_time_df$acro <- factor(over_time_df$acro, levels = ACRO)
```

### 4.2 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
```



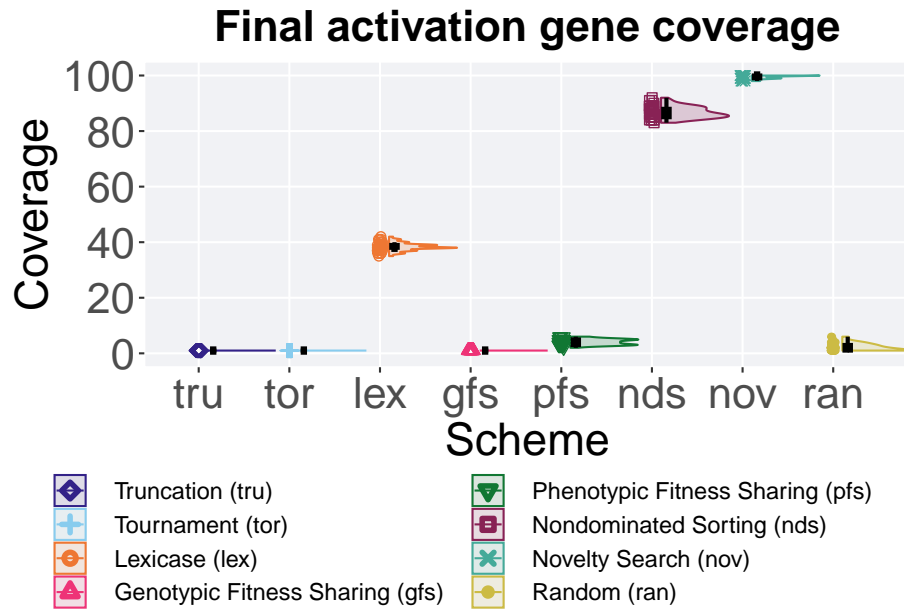
### 4.3 Final activation gene coverage

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 = .1, 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 = 0.03, height = 0.02), 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 18 rows containing missing values (`geom_point()`).
```



#### 4.3.1 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', 'pfs', 'ran'))
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   98    100 99.6   100     1
## 2 nds     50     0   83     86 86.7    92     3
## 3 lex     50     0   35     38 38.3    42     1
## 4 pfs     50     0    2      4  4.12     6     2
## 5 ran     50     0    1      2  2.22     6     2
## 6 gfs     50     0    1      1  1         1     0
## 7 tor     50     0    1      1  1         1     0
## 8 tru     50     0    1      1  1         1     0
```

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 = 381.66, 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      pfs      ran      gfs tor
## nds < 2e-16 -          -          -          -          - -
## lex < 2e-16 < 2e-16 -          -          -          - -
## pfs < 2e-16 < 2e-16 < 2e-16 -          -          - -
## ran < 2e-16 < 2e-16 < 2e-16 7.1e-09 -          - -
## gfs < 2e-16 < 2e-16 < 2e-16 < 2e-16 6.0e-09 -          -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 6.0e-09 1          -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 6.0e-09 1          1
##
## P value adjustment method: bonferroni
```

## 4.4 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",
    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('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
```



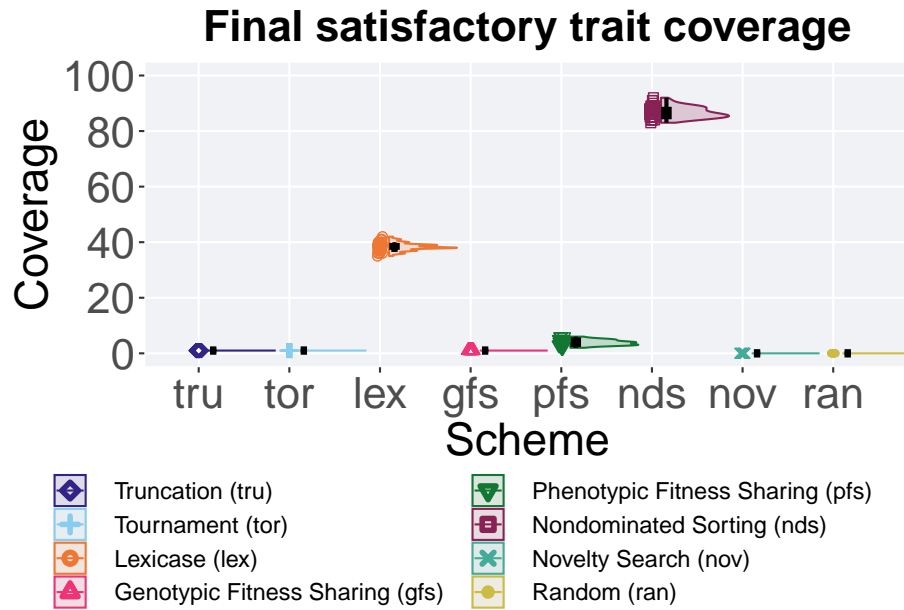
## 4.5 Final satisfactory trait coverage

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 = .1, 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 = 0.03, height = 0.02), 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 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 45 rows containing missing values (`geom_point()`).
```



#### 4.5.1 Stats

Summary statistics for the coverage found in the final population.

```
sat_coverage = filter(over_time_df, gen == 50000)
sat_coverage$acro = factor(sat_coverage$acro, levels = c('nds','lex','pfs','gfs','tor'))
sat_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 nds     50      0    83     86 86.7    92      3
## 2 lex     50      0    35     38 38.3    42      1
## 3 pfs     50      0      2      4  3.88     6      2
## 4 gfs     50      0      1      1  1         1      0
## 5 tor     50      0      1      1  1         1      0
## 6 tru     50      0      1      1  1         1      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 = sat_coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_uni_obj by acro
## Kruskal-Wallis chi-squared = 396.63, 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$pop_uni_obj, g = sat_coverage$acro, p.adjust.method = "bonf",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

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



## Chapter 5

# Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each selection scheme on the multi-path exploration diagnostic. 50 replicates are conducted for each scheme explored.

### 5.1 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)
```

### 5.2 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)
  )

```

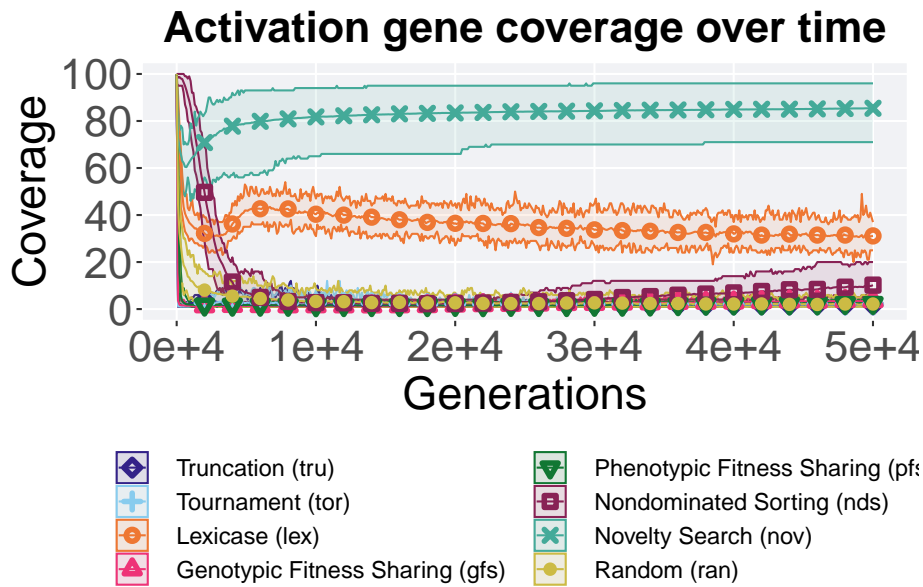
## `summarise()` has grouped output by 'scheme'. You can override using the  
## `.groups` argument.

```

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

```



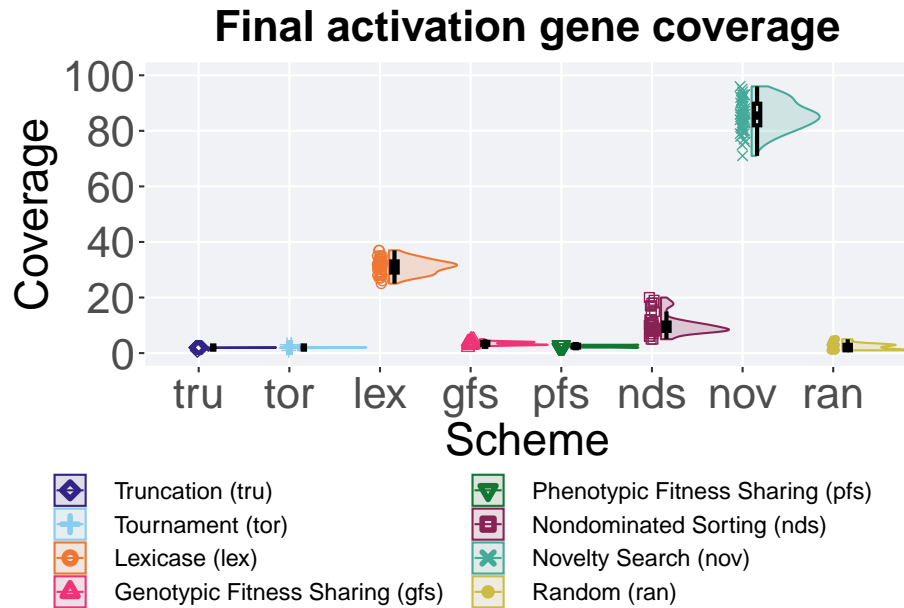


### 5.3 Final activation gene coverage

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 = .1, 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 = 0.03, height = 0.02), 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)
)
```



### 5.3.1 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', 'gfs', 'pfs'))
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    71   85.5 85.4    96  7.75
## 2 lex     50     0    25    31  31.1    37   4
## 3 nds     50     0     5     9  10.2    20   3
## 4 gfs     50     0     2     3   3.48     5   1
## 5 pfs     50     0     2     2   2.5     3   1
## 6 ran     50     0     1     2   2.16     5   2
## 7 tor     50     0     1     2   2.04     3   0
## 8 tru     50     0     1     2   1.98     2   0
```

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 = 350.25, 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      gfs      pfs      ran tor
## lex < 2e-16 -          -          -          -          -
## nds < 2e-16 < 2e-16 -          -          -          -
## gfs < 2e-16 < 2e-16 < 2e-16 -          -          -
## pfs < 2e-16 < 2e-16 < 2e-16 3.6e-10 -          -
## ran < 2e-16 < 2e-16 < 2e-16 9.4e-08 0.4          -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.2e-05 1.0 -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.1e-07 1.0 1.0
##
## P value adjustment method: bonferroni
```

## 5.4 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, 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('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
```

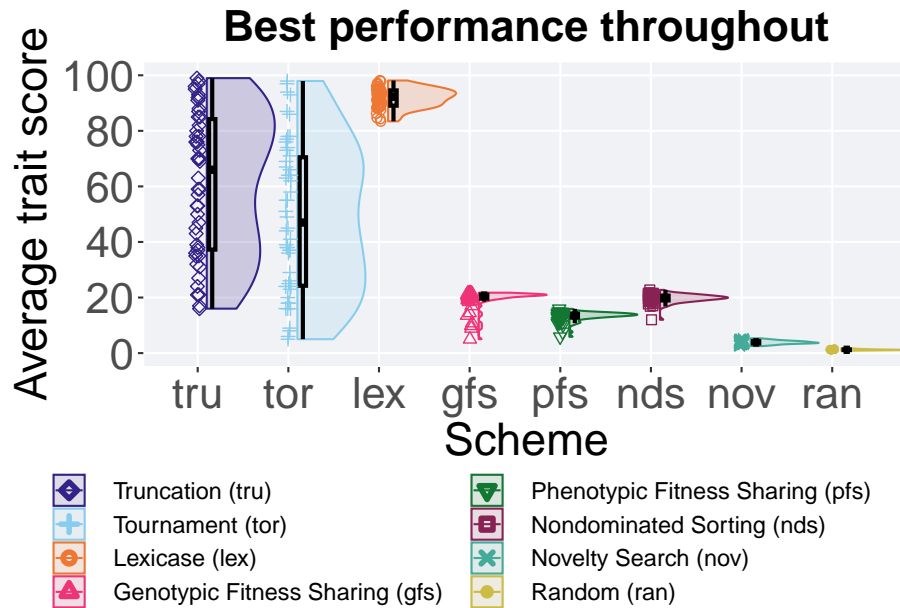


## 5.5 Best performance throughout

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 = .1, 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 = 0.03, height = 0.02), 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 performance throughout') +
  p_theme + theme(legend.title=element_blank())
```

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



### 5.5.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max')
performance$acro = factor(performance$acro, levels = c('lex', 'tru', 'tor', 'gfs', 'nds', 'pfs', '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 lex      50      0 83.4   92.3  91.9  98.1  5.51
## 2 tru      50      0 16     66.0  61.1  99.0  47.0
## 3 tor      50      0 5      47.0  48.9  97.9  46.2
## 4 gfs      50      0 4.99   20.7  19.3  21.7  1.45
## 5 nds      50      0 12.0   19.9  19.7  22.8  1.72
## 6 pfs      50      0 5.87   13.6  13.2  15.9  1.39
## 7 nov      50      0 2.52   3.83  3.87  5.33  0.793
## 8 ran      50      0 0.865  1.19  1.23  1.72  0.247
```

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 = 356.22, 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
##
##    lex    tru    tor    gfs    nds    pfs    nov
## tru 4.4e-09 -      -      -      -      -      -
## tor 1.1e-12 0.33   -      -      -      -      -
## gfs < 2e-16 2.3e-13 4.3e-07 -      -      -      -
## nds < 2e-16 1.2e-13 6.8e-07 0.47  -      -      -
## pfs < 2e-16 < 2e-16 5.0e-12 2.3e-11 1.3e-15 -      -
## nov < 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
```