

# Supplemental Material: Valley Crossing 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	Largest valley reached over time . . . . .	14
2.5	Largest valley reached throughout . . . . .	15
<b>3</b>	<b>Ordered exploitation results</b>	<b>19</b>
3.1	Data setup . . . . .	19
3.2	Performance over time . . . . .	19
3.3	Best performance throughout . . . . .	21
3.4	Largest valley reached over time . . . . .	23
3.5	Largest valley reached throughout . . . . .	25
<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
4.6	Largest valley reached throughout . . . . .	37
<b>5</b>	<b>Multi-path exploration results</b>	<b>41</b>
5.1	Data setup . . . . .	41
5.2	Activation gene coverage over time . . . . .	41
5.3	Final activation gene coverage . . . . .	43
5.4	Performance over time . . . . .	45
5.5	Best performance throughout . . . . .	47

5.6 Largest valley reached throughout . . . . .	49
---	----

# 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      _
## arch          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', '#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 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)
```

### 2.2 Performance over time

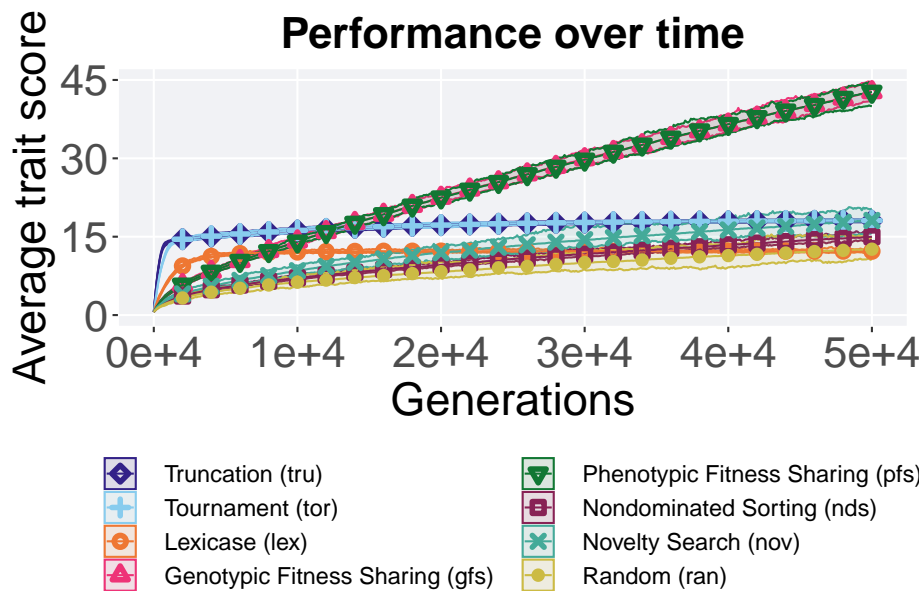
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
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, 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=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

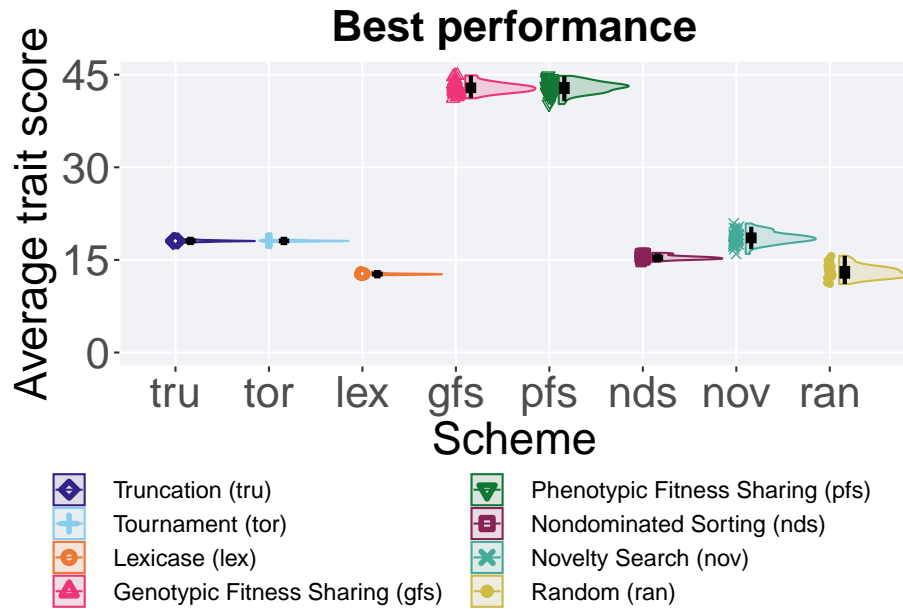
Best performance reached throughout 50,000 generations in a population.

```
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, 45),
    breaks=seq(0,45, 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') +
  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.
```



### 2.3.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max')
performance$acro = factor(performance$acro, levels = c('gfs', 'pfs', 'nov', 'tru', 'tor', 'lex'))
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  41.2  42.9  42.9  44.9  1.05
## 2 pfs     50     0  40.2  43.1  42.9  44.8  1.28
## 3 nov     50     0  16.0  18.6  18.6  20.9  1.07
## 4 tru     50     0  17.8  18.1  18.1  18.3  0.123
## 5 tor     50     0  17.9  18.1  18.1  18.3  0.0974
## 6 nds     50     0  14.7  15.3  15.4  16.1  0.378
## 7 ran     50     0  11.1  13.0  13.1  15.7  1.39
## 8 lex     50     0  12.5  12.7  12.7  13.1  0.128

```

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 = 366.61, 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      nov      tru      tor      nds      ran
## pfs 1.0000 -          -          -          -          -          -
## nov < 2e-16 < 2e-16 -          -          -          -          -
## tru < 2e-16 < 2e-16 0.0014 -          -          -          -
## tor < 2e-16 < 2e-16 0.0026 1.0000 -          -          -
## nds < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -          -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.2e-14 -
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.0000
##
## P value adjustment method: bonferroni

```

## 2.4 Largest valley reached over time

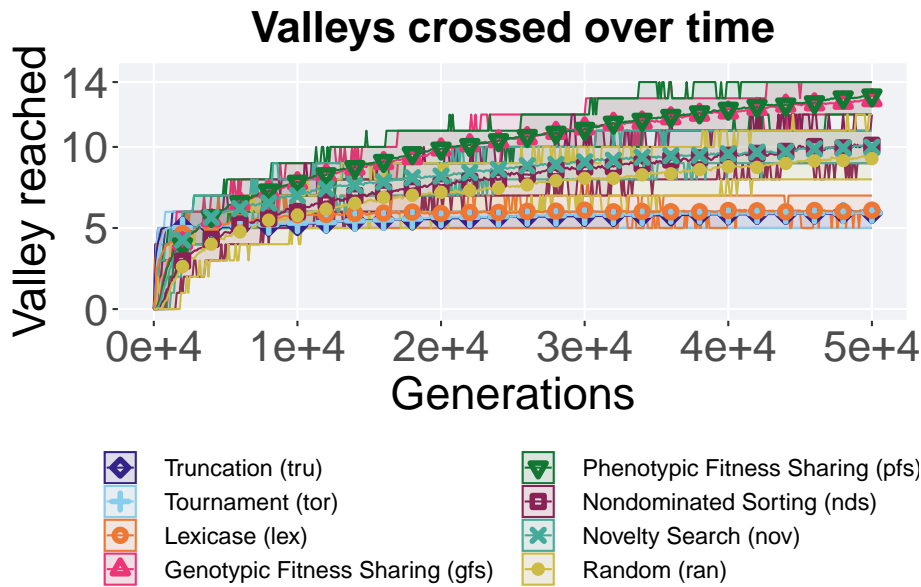
The largest valley reached in a single trait by the best performing solution in the population. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse data across 50 replicates.

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

## `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="Valley reached",
    limits=c(0, 14.5),
    breaks=c(0,5,10,14)
  ) +
  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('Valleys crossed 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
```



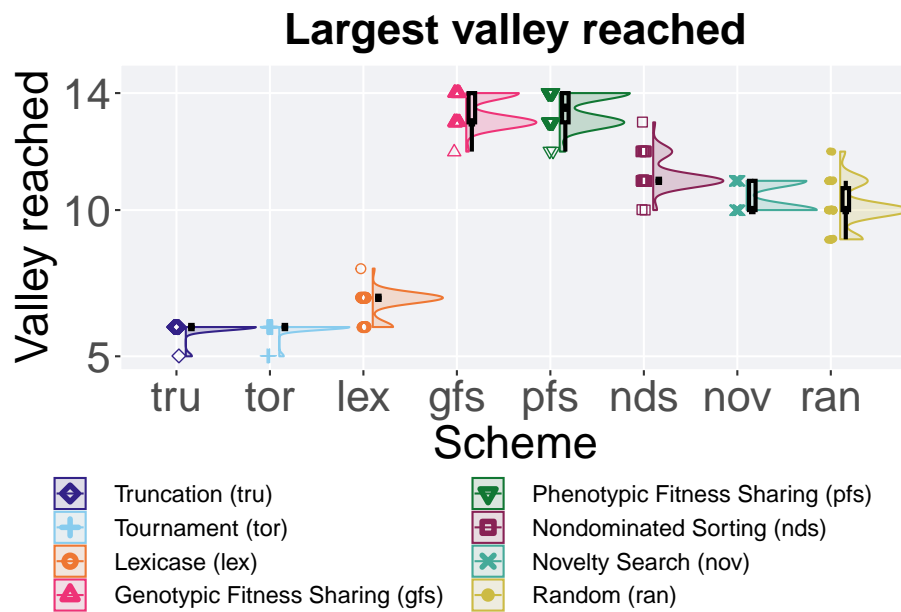
## 2.5 Largest valley reached throughout

Largest valley reached in a single trait by the best performing solution throughout an entire evolutionary run.

```
plot = filter(best_df, var == 'ele_big_peak') %>%
  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 = 0.2, 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="Valley reached",
    limits=c(5,14.5),
    breaks=c(5,10,14)
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Largest valley reached') +
  p_theme + theme(legend.title=element_blank())
```

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

```
## Warning: Removed 3 rows containing missing values (`geom_point()`).
```



### 2.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, var == 'ele_big_peak')
valleys$acro = factor(valleys$acro, levels = c('gfs', 'pfs', 'nds', 'nov', 'ran', 'lex', 'tru'))
valleys %>%
  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 gfs     50     0    12    13  13.4    14    1
## 2 pfs     50     0    12   13.5 13.5    14    1
## 3 nds     50     0    10    11  11.2    13    0
## 4 nov     50     0    10    10  10.5    11    1
## 5 ran     50     0     9    10  10.1    12  0.75
## 6 lex     50     0     6     7   6.8     8    0
## 7 tru     50     0     5     6   5.92    6    0
## 8 tor     50     0     5     6   5.94    6    0
```

Kruskal-Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 377.23, df = 7, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$acro
##
##      gfs      pfs      nds      nov      ran      lex      tru
## pfs 1.000    -      -      -      -      -      -
## nds < 2e-16 < 2e-16 -      -      -      -      -
## nov < 2e-16 < 2e-16 3.9e-08 -      -      -      -
## ran < 2e-16 < 2e-16 2.1e-10 0.099 -      -      -
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -      -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 4.6e-14 -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 4.3e-14 1.000
##
## 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 with valley crossing integrated. 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)
```

### 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, 14.5),
    breaks=c(0,5,10,14)
  ) +
  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.

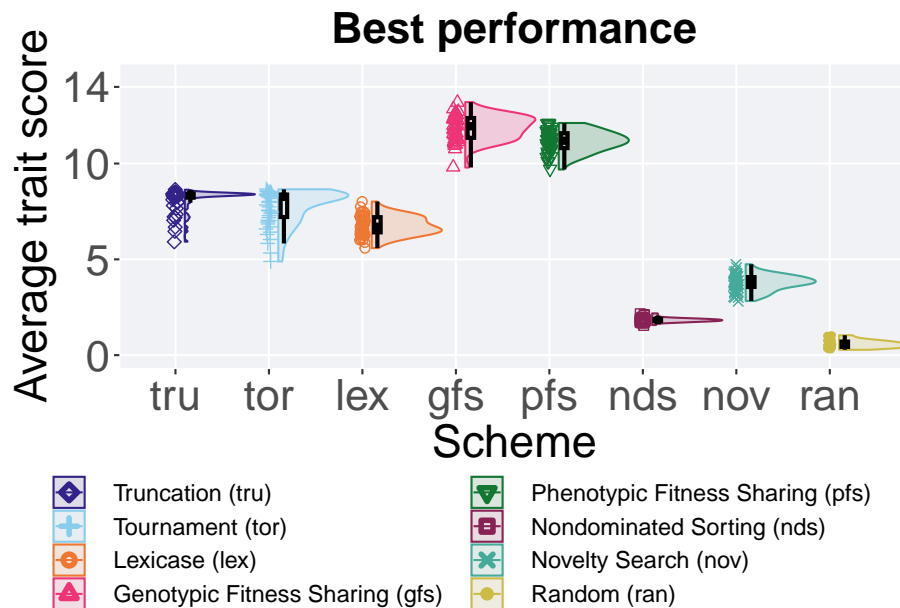
```
plot = filter(best_df, var == 'pop_fit_max') %>%
  ggplot(., aes(x = acro, y = val / DIMENSIONALITY, color = acro, fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = 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, 14.5),
    breaks=c(0,5,10,14) ) +
  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') +
  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('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  9.79  11.9  11.9  13.2  1.03
## 2 pfs      50      0  9.69  11.2  11.2  12.1  0.779
## 3 tru      50      0  5.92  8.36  8.13  8.63  0.203
## 4 tor      50      0  4.89  8.26  7.73  8.65  1.17
## 5 lex      50      0  5.59  6.70  6.76  8.02  0.792
## 6 nov      50      0  2.82  3.82  3.79  4.74  0.515
## 7 nds      50      0  1.57  1.83  1.84  2.18  0.116
## 8 ran      50      0  0.279 0.568 0.587  1.04  0.280
```

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 = 379.83, 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 5.0e-06 -          -          -          -          -
## tru < 2e-16 < 2e-16 -          -          -          -
## tor < 2e-16 < 2e-16 0.33  -          -          -
## lex < 2e-16 < 2e-16 7.8e-13 2.8e-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
```

### 3.4 Largest valley reached over time

The largest valley reached in a single trait by the best performing solution in the population. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse data

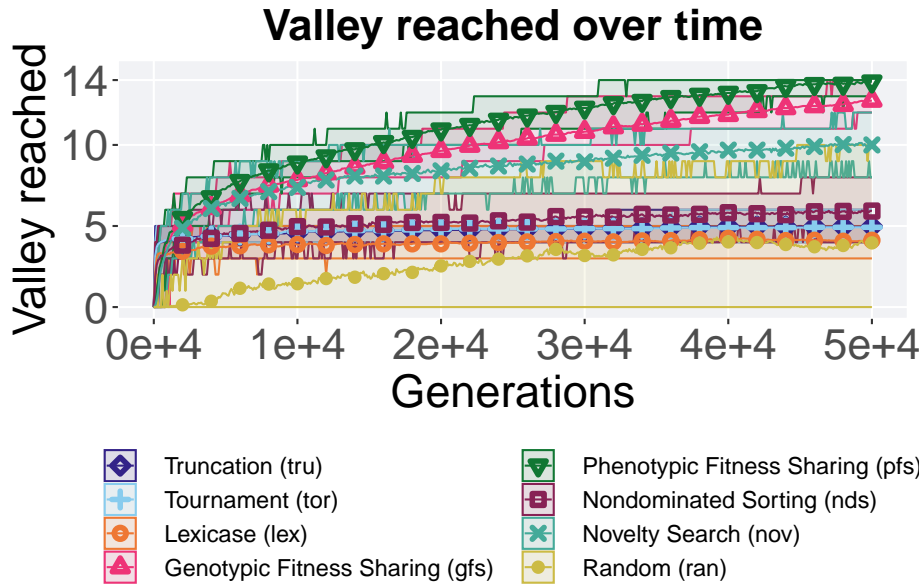
across 50 replicates.

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

## `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="Valley reached",
    limits=c(0, 14.5),
    breaks=c(0,5,10,14)
  ) +
  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('Valley reached 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
```



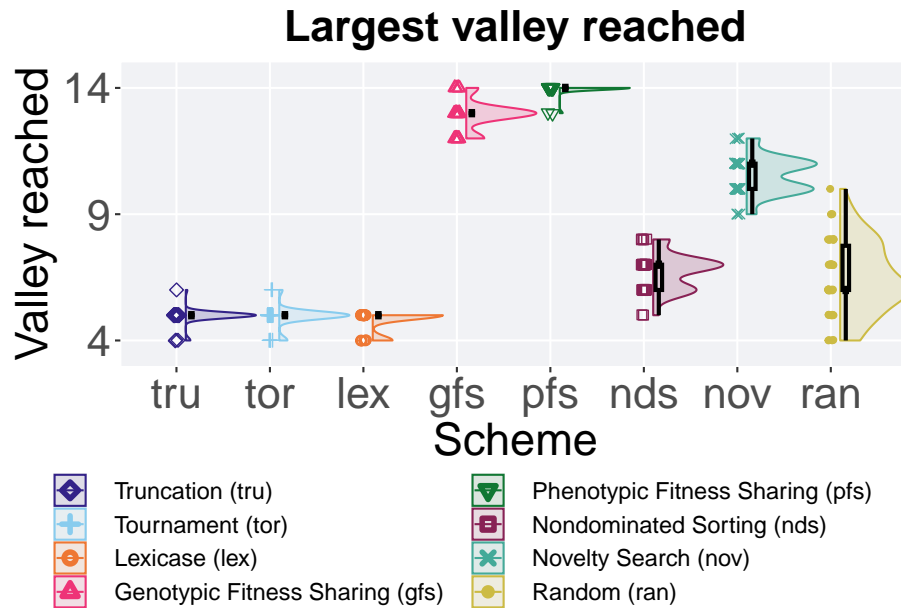


### 3.5 Largest valley reached throughout

Largest valley reached in a single trait by the best performing solution throughout an entire evolutionary run.

```
plot = filter(best_df, var == 'ele_big_peak') %>%
  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 = 0.2, 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="Valley reached",
    limits=c(3.5,14.5),
    breaks=c(4,9,14)
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Largest valley reached') +
  p_theme + theme(legend.title=element_blank())
```

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



### 3.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, var == 'ele_big_peak')
valleys$acro = factor(valleys$acro, levels = c('pfs', 'gfs', 'nov', 'nds', 'ran', 'tru', 'tor'))
valleys %>%
  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    13    14 14.0    14  0
## 2 gfs     50     0    12    13 12.9    14  0
## 3 nov     50     0     9    11 10.5    12  1
## 4 nds     50     0     5     7  6.72     8  1
## 5 ran     50     0     4     6  6.48    10 1.75
## 6 tru     50     0     4     5  4.96     6  0
## 7 tor     50     0     4     5  4.94     6  0
## 8 lex     50     0     4     5  4.78     5  0
```

Kruskal-Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 364.41, df = 7, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$acro
##
##      pfs      gfs      nov      nds      ran      tru      tor
## gfs 3.3e-15 -          -          -          -          -          -
## nov < 2e-16 < 2e-16 -          -          -          -          -
## nds < 2e-16 < 2e-16 < 2e-16 -          -          -          -
## ran < 2e-16 < 2e-16 < 2e-16 1.00      -          -          -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.6e-09 -          -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.8e-09 1.00      -
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.9e-10 0.21 0.72
##
## 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 with valley crossing integrated. 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, 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)
```

### 4.2 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

```



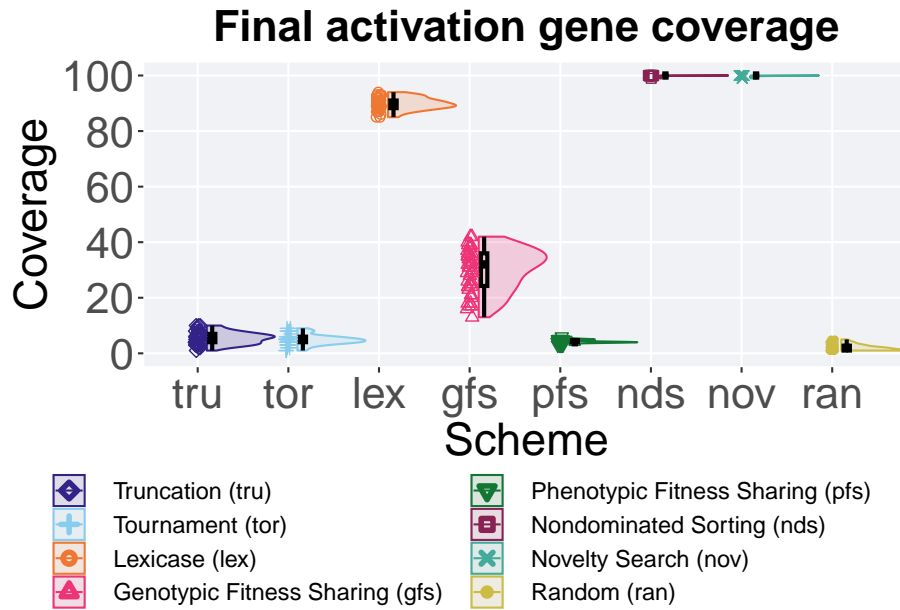
### 4.3 Final activation gene coverage

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 45 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', '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   85     90  89.8    94   2.75
## 4 gfs     50      0   13     32  30.1    42  11.8
## 5 tor     50      0    1      5   4.86     9    2
## 6 tru     50      0    1      6   5.5     10    3
## 7 pfs     50      0    3      4   4.18     6   0.75
## 8 ran     50      0    1      2   1.96     5   1.75
```

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 = 369.27, 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 1.0000 -          -          -          -          -          -
## lex < 2e-16 < 2e-16 -          -          -          -          -
## gfs < 2e-16 < 2e-16 < 2e-16 -          -          -          -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 -          -          -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.0000 -          -
## pfs < 2e-16 < 2e-16 < 2e-16 < 2e-16 0.1827 0.0095 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.3e-11 1.7e-12 5.0e-13
##
## P value adjustment method: bonferroni
```

## 4.4 Satisfactory trait coverage over time

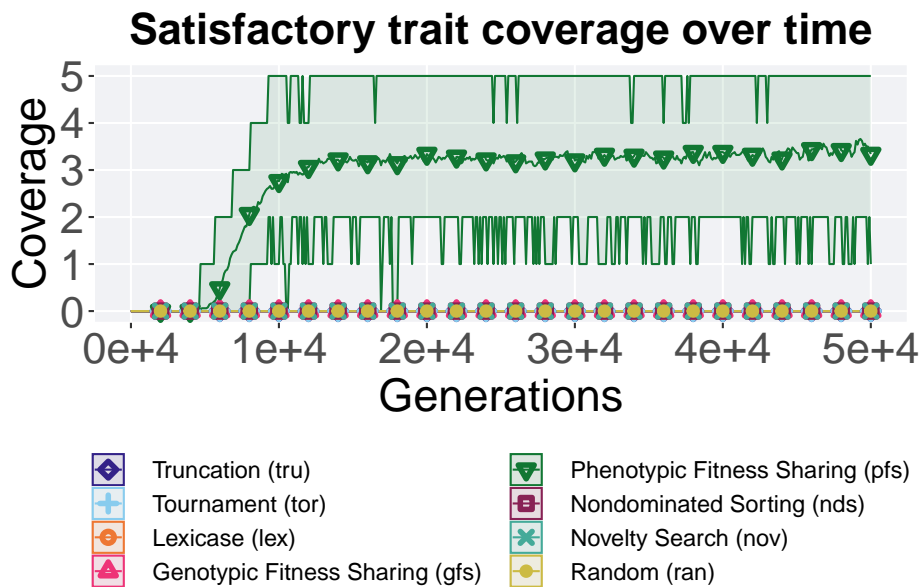
Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
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 =
  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) +
  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') +
  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, 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())

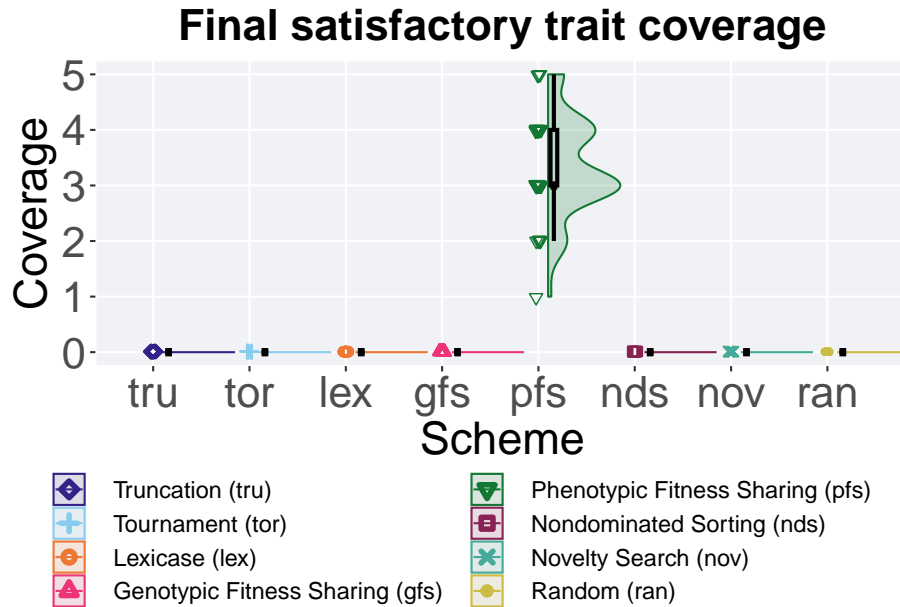
plot_grid(
  plot +
```

```

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

```

```
## Warning: Removed 167 rows containing missing values (`geom_point()`).
```



#### 4.5.1 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'
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      1      3  3.34      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(pop_uni_obj ~ acro, data = act_coverage)
```

```
##
##   Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by acro
## Kruskal-Wallis chi-squared = 396.94, 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 = "bonf",
                     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      -
## nov <2e-16 1      1      1      1
## ran <2e-16 1      1      1      1
##
## P value adjustment method: bonferroni
```

## 4.6 Largest valley reached throughout

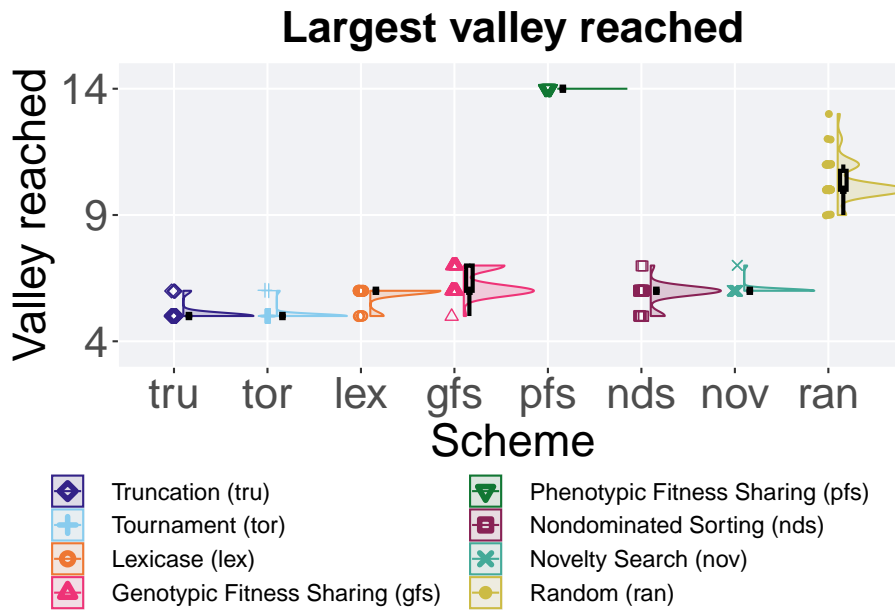
Largest valley reached in a single trait by the best performing solution throughout an entire evolutionary run.

```

plot = filter(best_df, var == 'ele_big_peak') %>%
  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="Valley reached",
    limits=c(3.5,14.5),
    breaks=c(4,9,14)
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Largest valley reached') +
  p_theme + theme(legend.title=element_blank())

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

```



#### 4.6.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, var == 'ele_big_peak')
valleys$acro = factor(valleys$acro, levels = c('pfs', 'ran', 'gfs', 'nov', 'nds', 'lex', 'tru', 'tor'))
valleys %>%
  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    14    14  14    14    0
## 2 ran     50     0     9    10 10.3    13  0.75
## 3 gfs     50     0     5     6  6.34     7  1
## 4 nov     50     0     6     6  6.04     7  0
```

```
## 5 nds      50      0      5      6 5.88      7 0
## 6 lex      50      0      5      6 5.84      6 0
## 7 tru      50      0      5      5 5.1       6 0
## 8 tor      50      0      5      5 5.04      6 0
```

Kruskal-Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 352.03, df = 7, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$acro
##
##      pfs      ran      gfs      nov      nds      lex      tru
## ran < 2e-16 -          -          -          -          -          -
## gfs < 2e-16 < 2e-16 -          -          -          -          -
## nov < 2e-16 < 2e-16 0.00347 -          -          -          -
## nds < 2e-16 < 2e-16 0.00018 0.26915 -          -          -
## lex < 2e-16 < 2e-16 1.3e-05 0.01917 1.00000 -          -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.8e-12 2.3e-12 -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.2e-14 1.6e-14 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 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

```

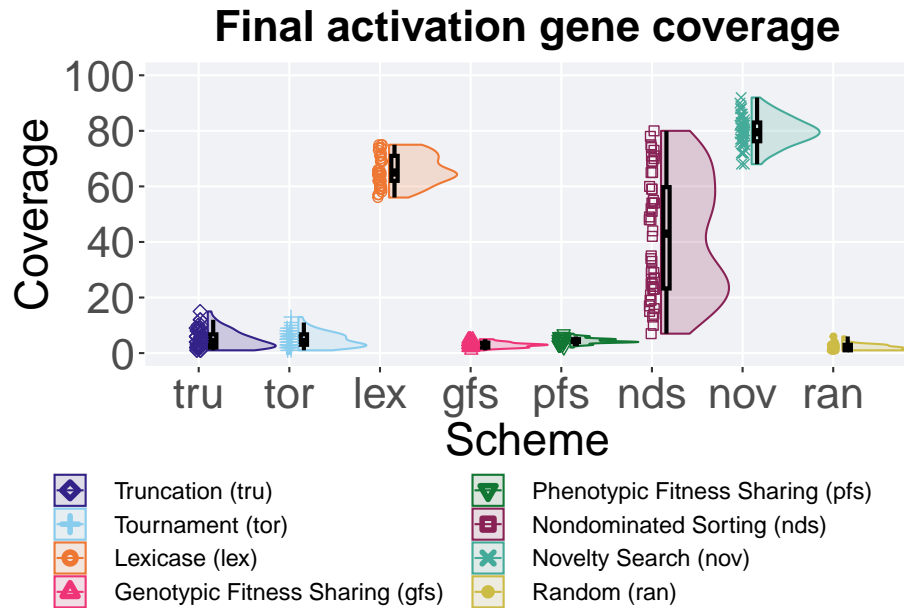


### 5.3 Final activation gene coverage

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', '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    68   79.5  79.9   92  6.75
## 2 lex     50     0    56   65   66.1   75   9
## 3 nds     50     0     7   43   42.5   80  36.5
## 4 tor     50     0     1    4    4.84   13  3.75
## 5 tru     50     0     1    4    4.9    15  4.75
## 6 pfs     50     0     2    4    4.4     7   1
## 7 gfs     50     0     1    3     3     5  1.75
## 8 ran     50     0     1    2    2.02    6   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 = 324.89, 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      pfs      gfs
## lex 1.4e-14 -          -          -          -          -
## nds 8.1e-15 1.5e-06 -          -          -          -
## tor < 2e-16 < 2e-16 < 2e-16 -          -          -
## tru < 2e-16 < 2e-16 2.7e-16 1.000    -          -
## pfs < 2e-16 < 2e-16 < 2e-16 1.000    1.000    -
## gfs < 2e-16 < 2e-16 < 2e-16 0.011    0.157    3.8e-08 -
## ran < 2e-16 < 2e-16 < 2e-16 3.7e-08 2.7e-06 1.3e-13 4.0e-05
##
## 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, 14.5),
    breaks=c(0,5,10,14)
  ) +
  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
```



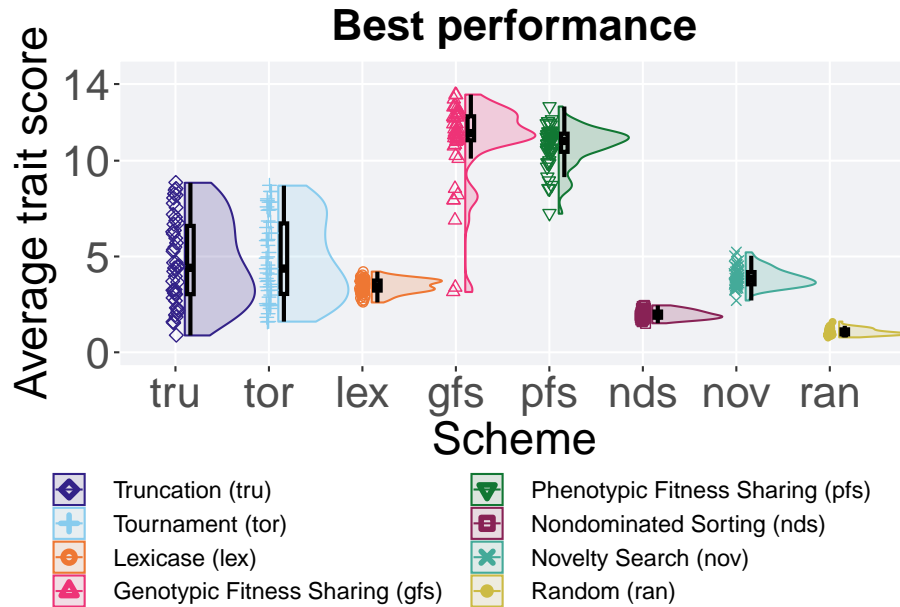
## 5.5 Best performance throughout

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, 14.5),
    breaks=c(0,5,10,14)
  ) +
  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') +
  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('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  3.15  11.4   11.1  13.4  1.25
## 2 pfs      50      0  7.23  11.0   10.8  12.8  0.949
## 3 tru      50      0  0.880  4.41   4.71   8.85  3.56
## 4 tor      50      0  1.60   4.37   4.84   8.70  3.68
## 5 nov      50      0  2.70   3.84   3.89   5.22  0.639
## 6 lex      50      0  2.60   3.44   3.45   4.21  0.523
## 7 nds      50      0  1.52   1.93   1.97   2.45  0.322
## 8 ran      50      0  0.780  0.998  1.06   1.60  0.261
```

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 = 327.4, 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 0.03925 -          -          -          -          -          -
## tru 2.2e-14 < 2e-16 -          -          -          -          -
## tor 2.4e-14 < 2e-16 1.00000 -          -          -          -
## nov 2.1e-14 < 2e-16 1.00000 1.00000 -          -          -
## lex 5.3e-15 < 2e-16 0.23671 0.04294 0.00042 -          -
## nds < 2e-16 < 2e-16 5.5e-10 8.2e-13 < 2e-16 < 2e-16 -
## ran < 2e-16 < 2e-16 1.4e-15 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

## 5.6 Largest valley reached throughout

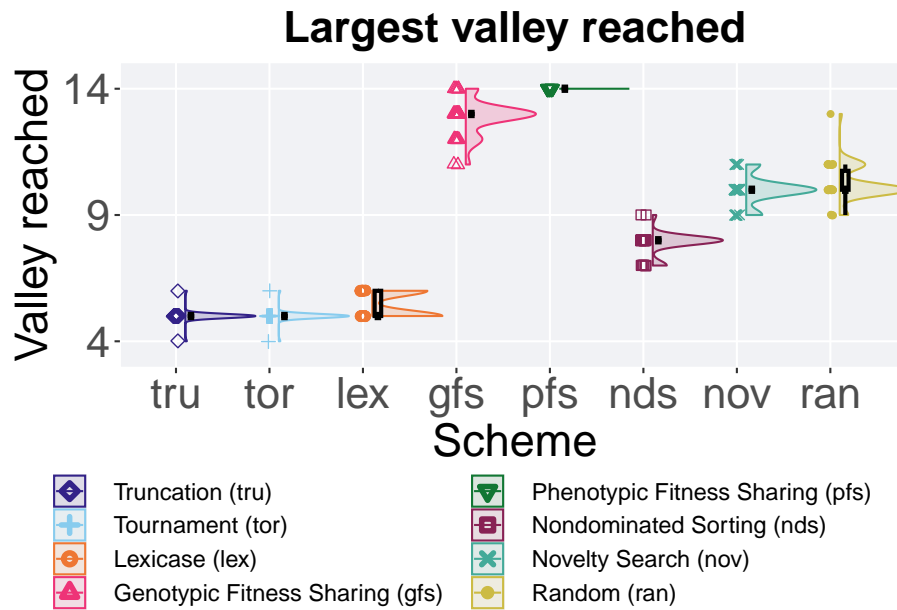
Largest valley reached in a single trait by the best performing solution throughout an entire evolutionary run.

```

plot = filter(best_df, var == 'ele_big_peak') %>%
  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="Valley reached",
    limits=c(3.5,14.5),
    breaks=c(4,9,14)
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Largest valley reached') +
  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 largest valley crossed.

```
valleys = filter(best_df, var == 'ele_big_peak')
valleys$acro = factor(valleys$acro, levels = c('pfs', 'gfs', 'ran', 'nov', 'nds', 'lex', 'tru', 'tor'))
valleys %>%
  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    14    14  14    14    0
## 2 gfs     50     0    11    13 12.9    14    0
## 3 ran     50     0     9    10 10.2    13  0.75
## 4 nov     50     0     9    10  9.98    11    0
```

```
## 5 nds      50      0      7      8 7.88      9 0
## 6 lex      50      0      5      5 5.44      6 1
## 7 tru      50      0      4      5 5        6 0
## 8 tor      50      0      4      5 5        6 0
```

Kruskal-Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acro
## Kruskal-Wallis chi-squared = 385.68, df = 7, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$acro
##
##      pfs      gfs      ran      nov      nds      lex      tru
## gfs < 2e-16 -          -          -          -          -          -
## ran < 2e-16 < 2e-16 -          -          -          -          -
## nov < 2e-16 < 2e-16 1          -          -          -          -
## nds < 2e-16 < 2e-16 < 2e-16 < 2e-16 -          -          -
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -          -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 7.6e-06 -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 7.6e-06 1
##
## P value adjustment method: bonferroni
```