Supplemental Material: Selection Scheme Parameter Sweep MVC Diagnostics

Jose Guadalupe Hernandez

2023-08-28

Contents

Chapter 1

Introduction

This is the supplemental material for selection scheme parameter sweep experiments with diagnostics combined with 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.

This supplemental material includes the following selection schemes:

- Truncation (Section 2)
- Tournament (Section 3)
- Genotypic fitness sharing (Section 4)
- Phenotypic fitness sharing (Section 5)
- Nondominated sorting (Section 6)
- Novelty search (Section 7)

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 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(5,3,1,2,6,0,4,20,8)
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",
                                  linewidth = 0.5, linetype = "solid")
)

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

# selection scheme params exploring
TR_LIST = c('1','2','4','8','16','32','64','128','256')
TS_LIST = c('2','4','8','16','32','64','128','256','512')
FS_LIST = c('0','0.1','0.3','0.6','1.2','2.5','5')
ND_LIST = c('0','0.1','0.3','0.6','1.2','2.5','5')
NS_LIST = c('1','2','4','8','15','30')
```

Chapter 2

Truncation selection

Results for the truncation selection parameter sweep on the diagnostics with valleys.

2.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME-MVC/tru.csv', sep = "", collapse = NULL), head
over_time_df$T <- factor(over_time_df$T, levels = TR_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST-MVC/tru.csv', sep = "", collapse = NULL), header = TRUE,
best_df$T <- factor(best_df$T, levels = TR_LIST)
```

2.2 Exploitation rate results

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

2.2.1 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 = filter(over_time_df, acro == 'exp') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

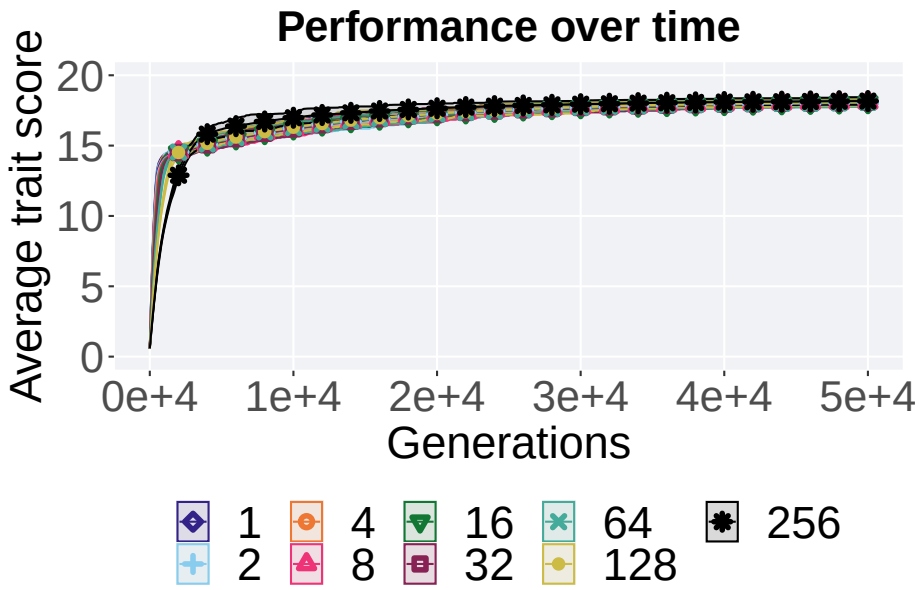
```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape
  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, 20)
  ) +
  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()) +
  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")
  )

over_time_plot
```



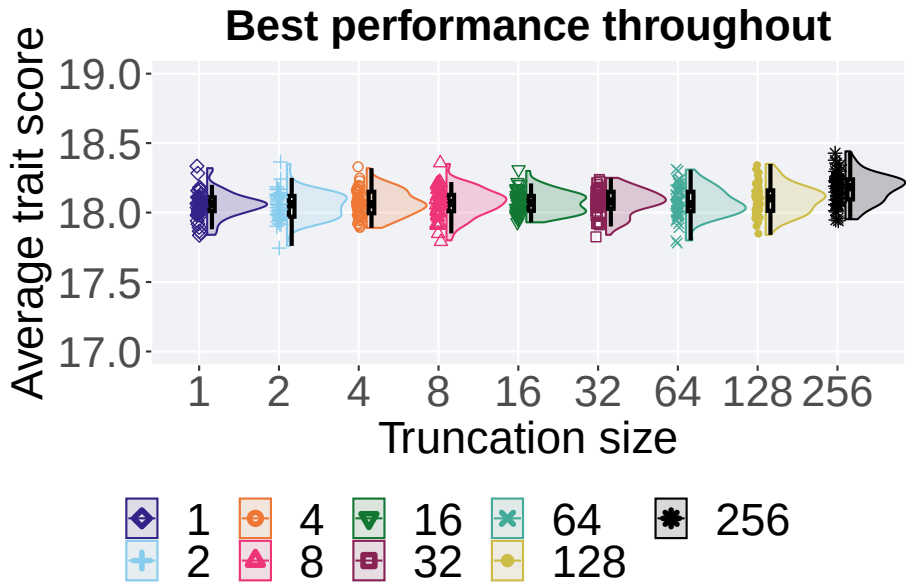
2.2.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'exp' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = T, y = val / DIMENSIONALITY, color = T, fill = T, shape = T)) +
  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, positio
  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(17, 19)
  ) +
  scale_x_discrete(
    name="Truncation size"
  )+
  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)
)
```



2.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'exp' &  var == 'pop_fit_max')
performance$T = factor(performance$T, levels = TR_LIST)
performance %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max    IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
```

```
## 1 1        50      0  17.8   18.1  18.1  18.3 0.0999
## 2 2        50      0  17.8   18.1  18.1  18.3 0.150
## 3 4        50      0  17.9   18.1  18.1  18.3 0.155
## 4 8        50      0  17.8   18.1  18.1  18.3 0.117
## 5 16       50      0  17.9   18.1  18.1  18.3 0.107
## 6 32       50      0  17.8   18.1  18.1  18.2 0.122
## 7 64       50      0  17.8   18.1  18.1  18.3 0.140
## 8 128      50      0  17.8   18.1  18.1  18.3 0.147
## 9 256      50      0  18.0   18.2  18.2  18.4 0.144
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 40.639, df = 8, p-value = 2.435e-06
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$T
##
##     1       2       4       8       16      32      64      128
## 2   1.00000 -       -       -       -       -       -       -
## 4   1.00000 1.00000 -       -       -       -       -       -
## 8   1.00000 1.00000 1.00000 -       -       -       -       -
## 16  1.00000 1.00000 1.00000 1.00000 -       -       -       -
## 32  1.00000 1.00000 1.00000 1.00000 1.00000 -       -       -
## 64  1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 -       -
## 128 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 -
## 256 1.7e-05 1.6e-05 0.00018 0.00043 0.00023 0.00104 0.00114 0.03366
##
## P value adjustment method: bonferroni
```

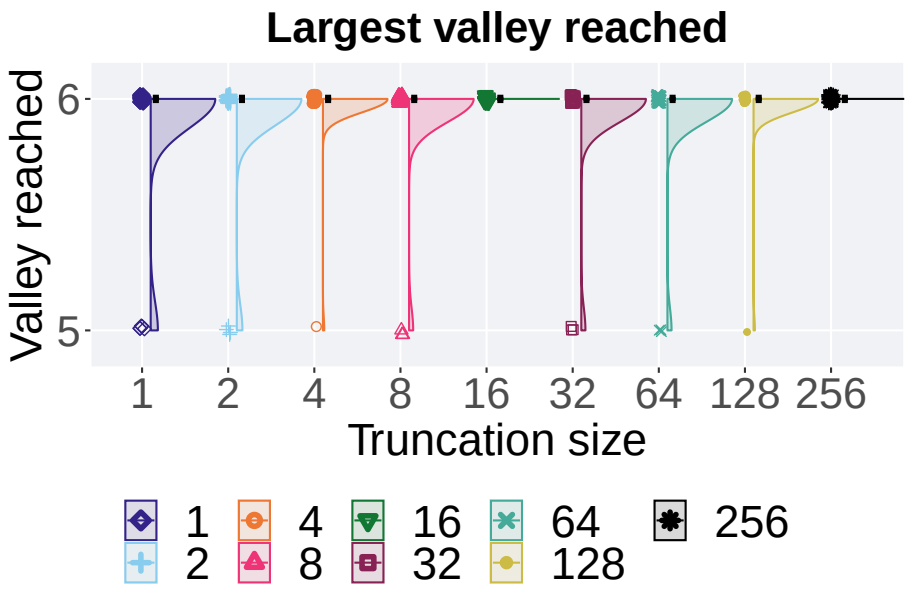
2.2.3 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'exp' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = T, y = val, color = T, fill = T, shape = T)) +
```

```
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  scale_y_continuous(
    name="Valley reached",
    limits=c(4.9,6.1),
    breaks = c(5,6)
  ) +
  scale_x_discrete(
    name="Truncation size"
  )+
  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)
)
```



2.2.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df,acro == 'exp' &  var == 'ele_big_peak')
valleys$T = factor(valleys$T, levels = TR_LIST)
valleys %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0     5      6  5.9      6     0
## 2 2        50      0     5      6  5.92     6     0
## 3 4        50      0     5      6  5.98     6     0
## 4 8        50      0     5      6  5.94     6     0
## 5 16       50      0     6      6  6        6     0
## 6 32       50      0     5      6  5.94     6     0
## 7 64       50      0     5      6  5.94     6     0
## 8 128      50      0     5      6  5.98     6     0
## 9 256      50      0     6      6  6        6     0
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 12.008, df = 8, p-value = 0.1508
```

2.3 Ordered exploitation results

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

2.3.1 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.

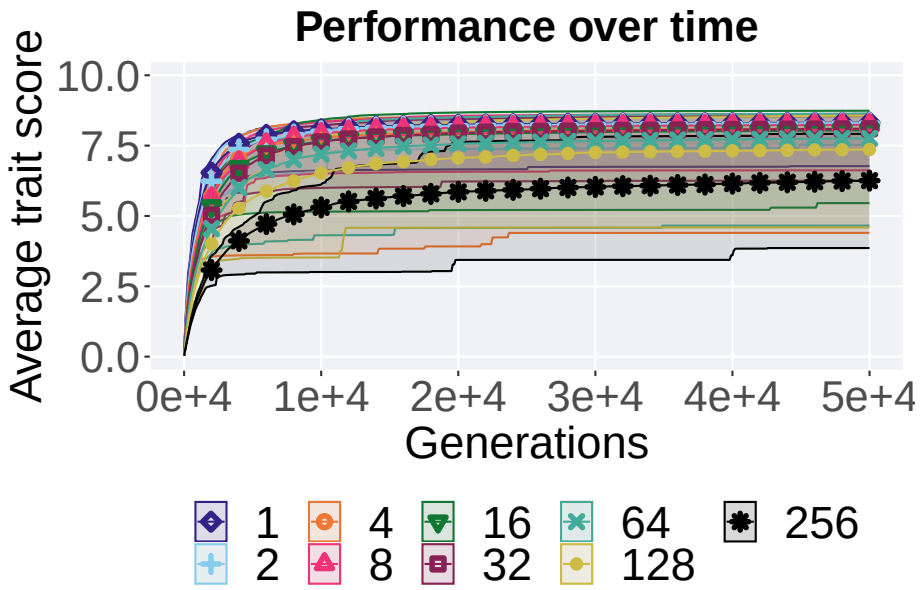
```
lines = filter(over_time_df, acro == 'ord') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape
  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, 10)
  ) +
  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()) +
  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")
  )

over_time_plot
```



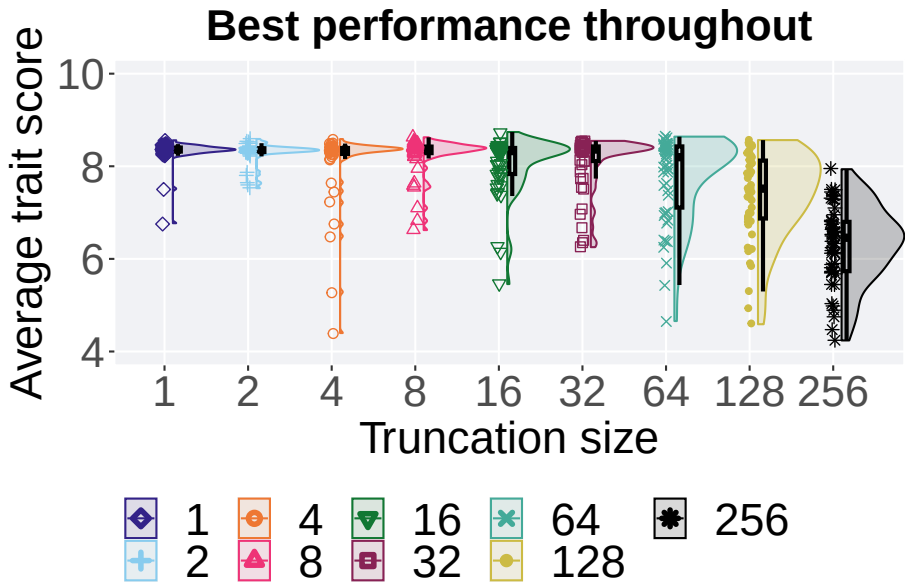
2.3.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'ord' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = T, y = val / DIMENSIONALITY, color = T, fill = T, shape = T)) +
  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, positio
  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(4, 10)
  ) +
  scale_x_discrete(
    name="Truncation size"
  )+
  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)
)
```



2.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'ord' &  var == 'pop_fit_max')
performance$T = factor(performance$T, levels = TR_LIST)
performance %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max    IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
```

```
## 1 1        50      0  6.77   8.36  8.32  8.56 0.0867
## 2 2        50      0  7.53   8.35  8.30  8.59 0.0808
## 3 4        50      0  4.40   8.36  8.10  8.59 0.128
## 4 8        50      0  6.62   8.37  8.24  8.63 0.142
## 5 16       50      0  5.46   8.33  8.05  8.74 0.563
## 6 32       50      0  6.25   8.36  8.10  8.55 0.316
## 7 64       50      0  4.66   8.20  7.75  8.64 1.32
## 8 128      50      0  4.59   7.52  7.36  8.57 1.25
## 9 256      50      0  3.86   6.42  6.26  7.94 1.06
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 148.77, df = 8, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$T
##
##     1       2       4       8       16      32      64      128
## 2   1.00000 -       -       -       -       -       -       -
## 4   1.00000 1.00000 -       -       -       -       -       -
## 8   1.00000 1.00000 1.00000 -       -       -       -       -
## 16  1.00000 1.00000 1.00000 1.00000 -       -       -       -
## 32  1.00000 1.00000 1.00000 1.00000 1.00000 -       -       -
## 64  0.91020 1.00000 1.00000 0.95152 1.00000 1.00000 -       -
## 128 6.0e-08 8.7e-08 3.7e-05 1.3e-06 0.00104 0.00013 0.34001 -
## 256 6.6e-16 3.4e-16 8.6e-13 2.9e-15 2.9e-13 2.9e-13 2.9e-09 1.7e-06
##
## P value adjustment method: bonferroni
```

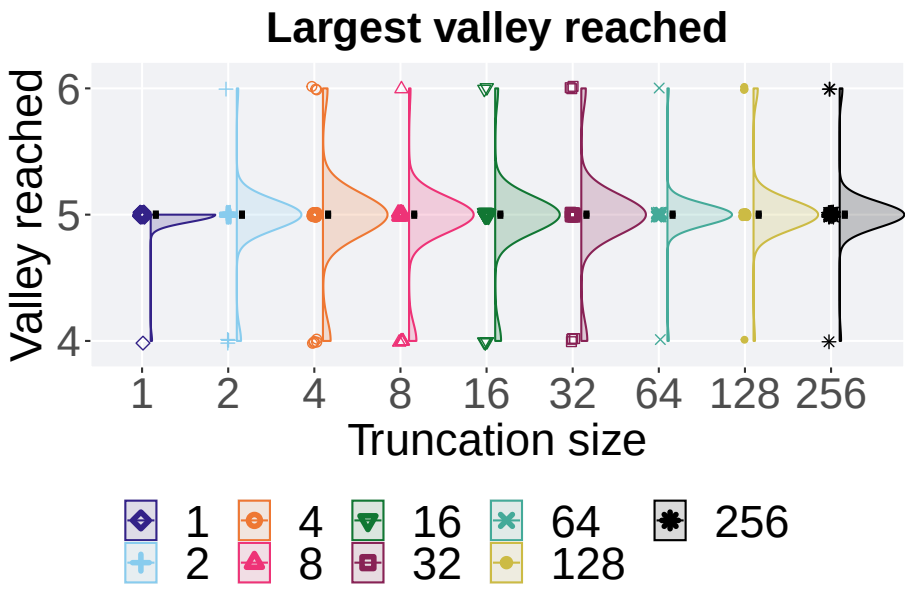
2.3.3 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'ord' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = T, y = val, color = T, fill = T, shape = T)) +
```

```
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(3.9,6.1),
    breaks = c(4,5,6)
  ) +
  scale_x_discrete(
    name="Truncation size"
  )+
  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)
)
```



2.3.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'ord' & var == 'ele_big_peak')
valleys$T = factor(valleys$T, levels = TR_LIST)
valleys %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0     4      5  4.98     5     0
## 2 2        50      0     4      5  4.96     6     0
## 3 4        50      0     4      5  4.96     6     0
## 4 8        50      0     4      5  4.92     6     0
## 5 16       50      0     4      5  4.96     6     0
## 6 32       50      0     4      5  4.98     6     0
## 7 64       50      0     4      5  5        6     0
## 8 128      50      0     4      5  5.04     6     0
## 9 256      50      0     4      5  5.02     6     0
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 5.7698, df = 8, p-value = 0.673
```

2.4 Contradictory objectives resutls

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

2.4.1 Activation gene coverage over time

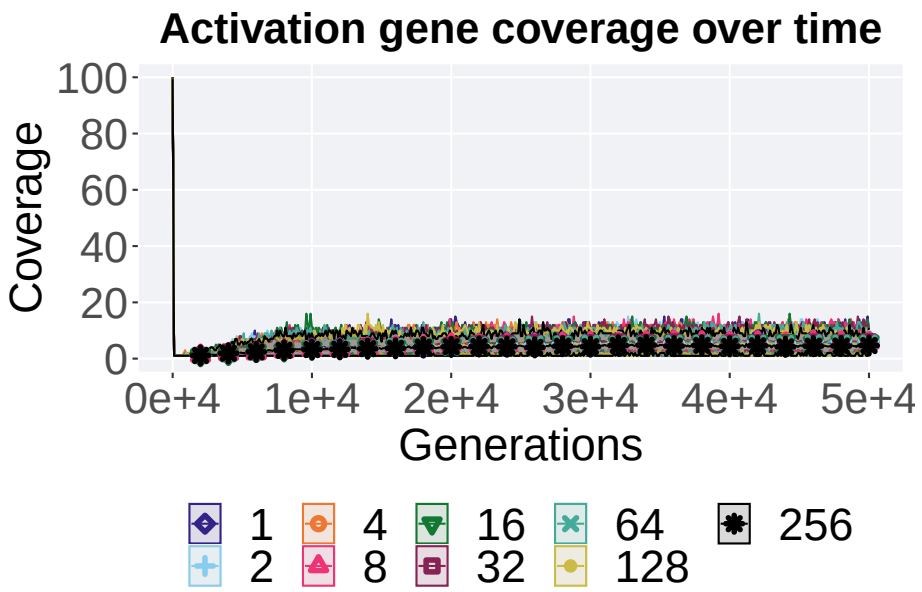
Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  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()) +
  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")
  )
```



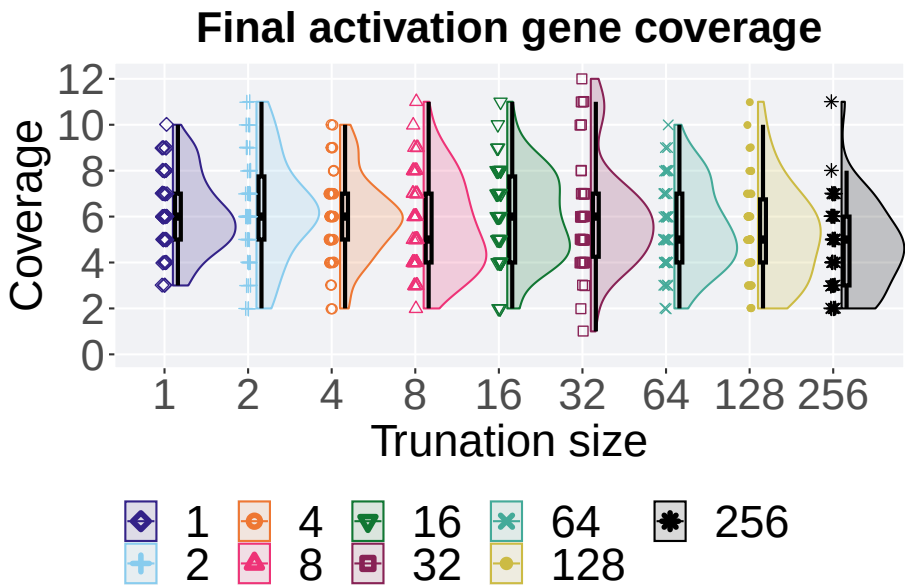
2.4.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  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, positio
  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, 12.1),
    breaks=seq(0,12,2)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  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)
)
```



2.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$T = factor(act_coverage$T, levels = TR_LIST)
act_coverage %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 1         50      0     3      6  6.1     10  2
## 2 2         50      0     2      6  6.28    11  2.75
## 3 4         50      0     2      6  5.98    10  2
## 4 8         50      0     2      5  5.68    11  3
## 5 16        50      0     2      6  5.94    11  3.75
## 6 32        50      0     1      6  5.96    12  2.75
## 7 64        50      0     2      5  5.54    10  3
## 8 128       50      0     2      5  5.28    11  2.75
## 9 256       50      0     2      5  4.66    11  3
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 24.29, df = 8, p-value = 0.002049
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$T, p.adjust.method = "bonferr
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$T
##
##     1      2      4      8      16     32     64     128
## 2   1.0000 -      -      -      -      -      -      -
## 4   1.0000 1.0000 -      -      -      -      -      -
## 8   1.0000 1.0000 1.0000 -      -      -      -      -
## 16  1.0000 1.0000 1.0000 1.0000 -      -      -      -
## 32  1.0000 1.0000 1.0000 1.0000 1.0000 -      -      -
## 64  1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 -      -
## 128 1.0000 0.8123 1.0000 1.0000 1.0000 1.0000 1.0000 -
## 256 0.0053 0.0079 0.0194 0.6738 0.0578 0.1290 0.8827 1.0000
##
## P value adjustment method: bonferroni
```

2.4.3 Satisfactory trait coverage over time

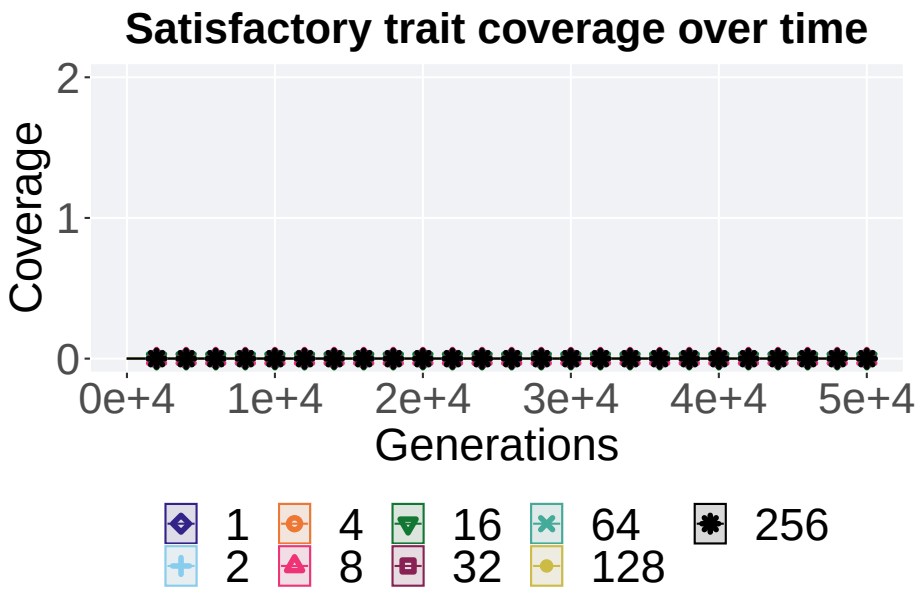
Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  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, 2),
    breaks=c(0,1,2)
  ) +
  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()) +
  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")
  )
```



2.4.4 Final satisfactory trait coverage

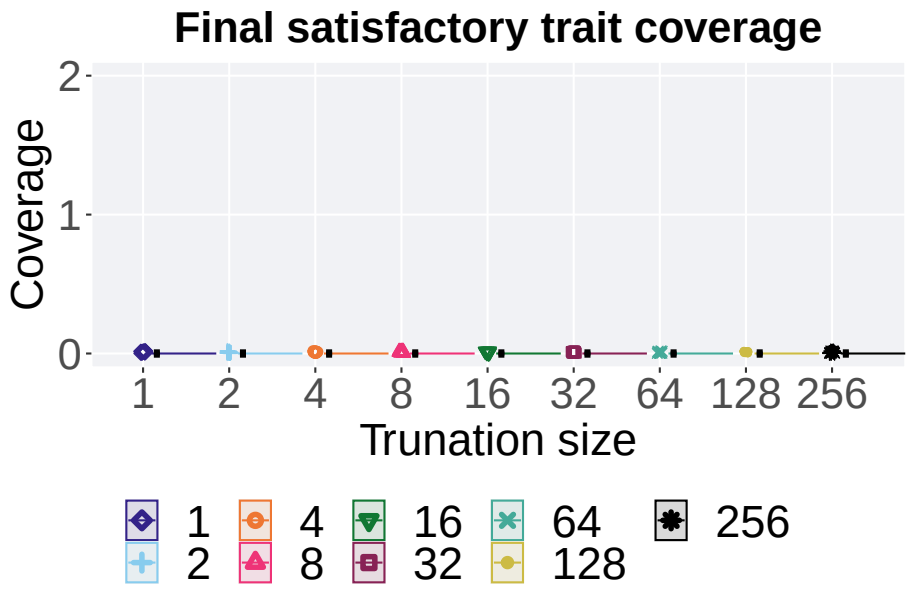
Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = T, y = pop_uni_obj, color = T, fill = T, shape = T)) +
  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, positio
  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, 2),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  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 236 rows containing missing values (`geom_point()`).
```



2.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$T = factor(sat_coverage$T, levels = TR_LIST)
sat_coverage %>%
  group_by(T) %>%
  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)
  )
```

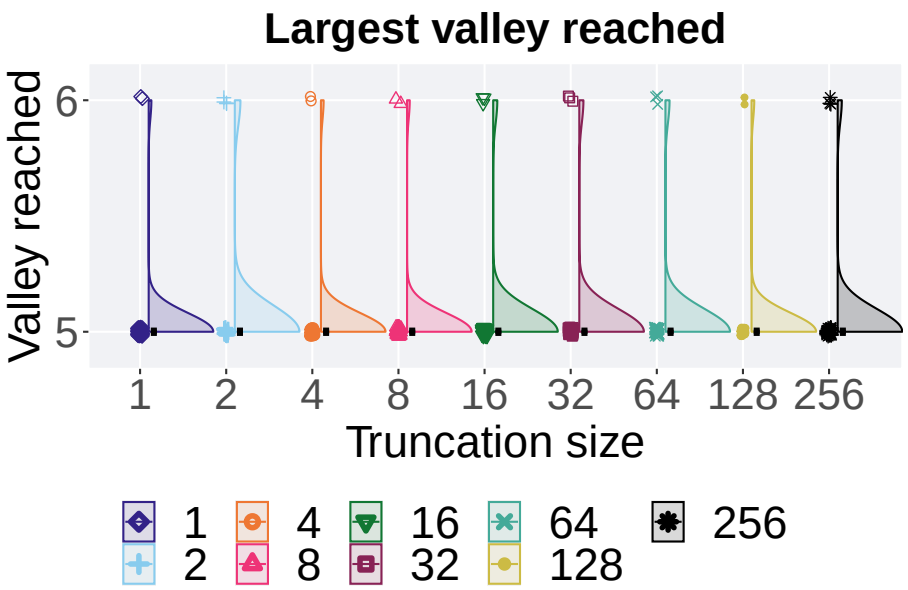
```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 1        50      0     0      0     0     0     0
## 2 2        50      0     0      0     0     0     0
## 3 4        50      0     0      0     0     0     0
## 4 8        50      0     0      0     0     0     0
## 5 16       50      0     0      0     0     0     0
## 6 32       50      0     0      0     0     0     0
## 7 64       50      0     0      0     0     0     0
## 8 128      50      0     0      0     0     0     0
## 9 256      50      0     0      0     0     0     0
```

2.4.5 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, acro == 'con' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = T, y = val, color = T, fill = T, shape = T)) +
  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, positio
  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(4.9,6.1),
    breaks = c(5,6)
  ) +
  scale_x_discrete(
    name="Truncation size"
  )+
  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)
)
```



2.4.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'con' & var == 'ele_big_peak')
valleys$T = factor(valleys$T, levels = TR_LIST)
valleys %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0     5      5  5.04     6     0
## 2 2        50      0     5      5  5.08     6     0
## 3 4        50      0     5      5  5.04     6     0
## 4 8        50      0     5      5  5.04     6     0
## 5 16       50      0     5      5  5.06     6     0
```

```
## 6 32       50      0     5      5  5.06     6     0
## 7 64       50      0     5      5  5.06     6     0
## 8 128      50      0     5      5  5.04     6     0
## 9 256      50      0     5      5  5.06     6     0
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 1.581, df = 8, p-value = 0.9913
```

2.5 Multi-path exploration results

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

2.5.1 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.

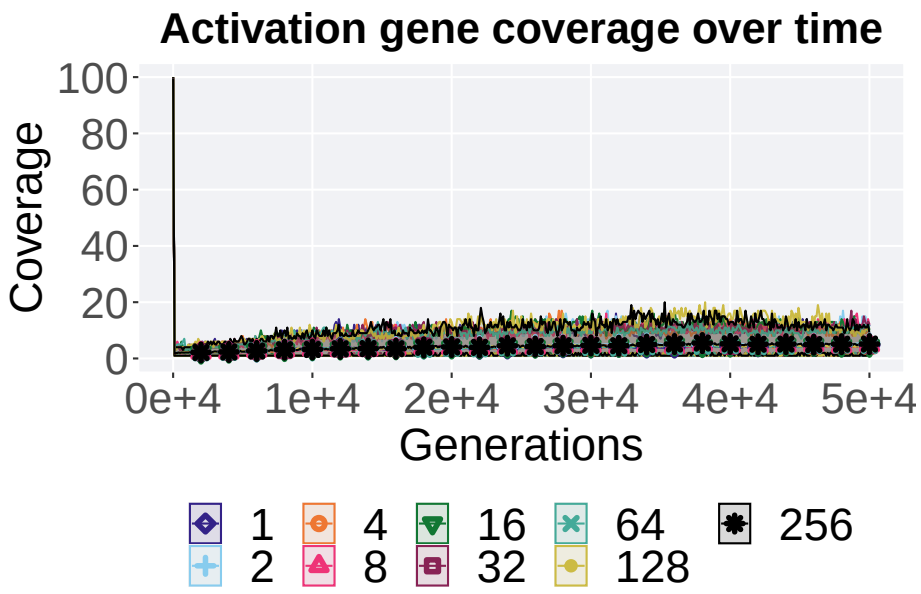
```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



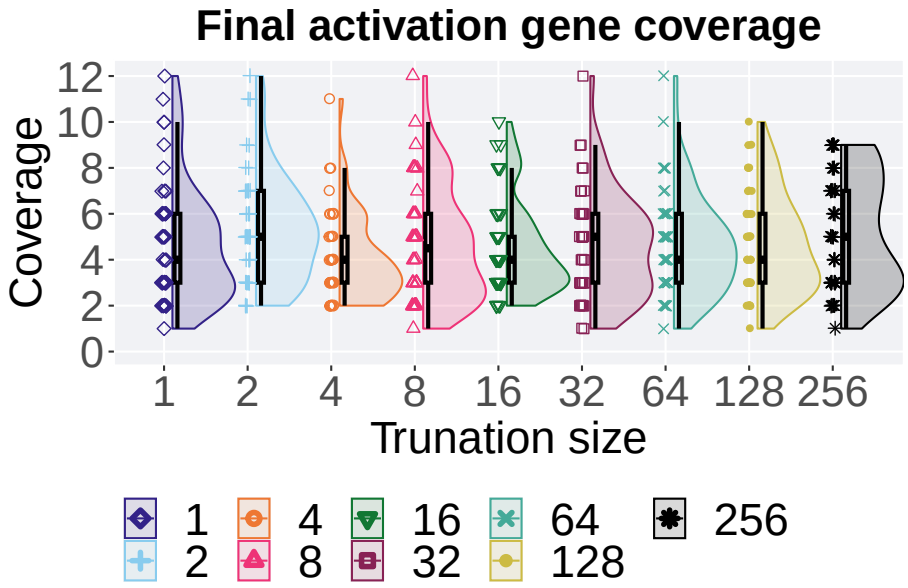
2.5.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
```

```
  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, positio
  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, 12.1),
    breaks=seq(0,12,2)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  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)
)
```



2.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$T = factor(act_coverage$T, levels = TR_LIST)
act_coverage %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 1        50      0     1    4    4.68    12     3
## 2 2        50      0     2    5    5.34    12     4
## 3 4        50      0     2    4    4.04    11     2
## 4 8        50      0     1    4.5  4.7     12     3
## 5 16       50      0     2    4    4.46    10     2
## 6 32       50      0     1    5    4.58    12     3
## 7 64       50      0     1    4    4.64    12     3
## 8 128      50      0     1    4    4.66    10     3
## 9 256      50      0     1    5    5.12     9     4
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 10.094, df = 8, p-value = 0.2585
```

2.5.3 Performance over time

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

```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(T, gen) %>%
```

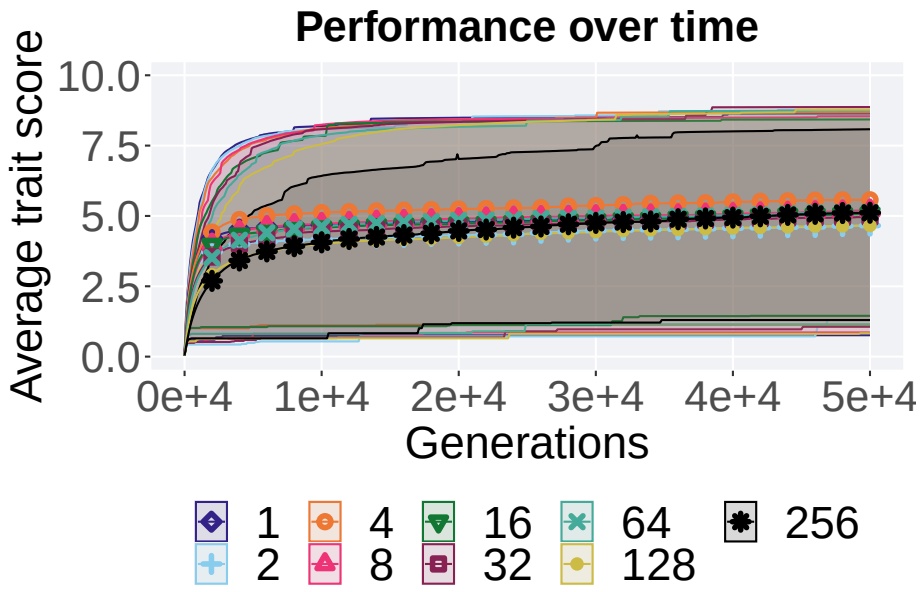
```
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 10)
  ) +
  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()) +
  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")
  )

over_time_plot
```



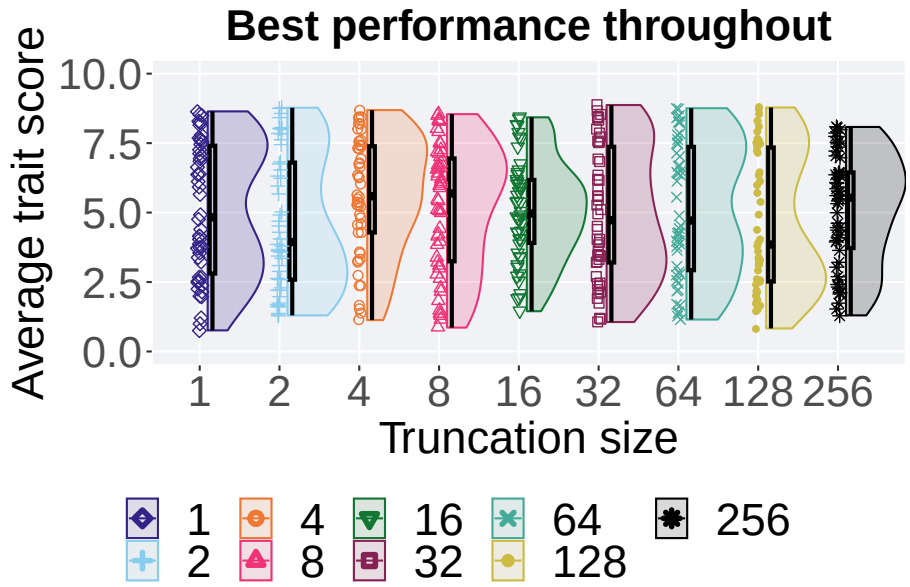
2.5.4 Best performance throughout

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

```
plot = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = T, y = val / DIMENSIONALITY, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 10)
  ) +
  scale_x_discrete(
    name="Truncation size"
  )+
  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)
)
```



2.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max')
performance$T = factor(performance$T, levels = TR_LIST)
performance %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 1        50      0 0.760   4.83  5.09  8.64  4.60
## 2 2        50      0 1.30    3.94  4.66  8.77  4.22
## 3 4        50      0 1.13    5.59  5.57  8.69  3.11
## 4 8        50      0 0.860   5.69  5.20  8.54  3.70
## 5 16       50      0 1.45    4.97  5.08  8.43  2.27
## 6 32       50      0 1.06    4.72  4.96  8.87  4.17
## 7 64       50      0 1.15    4.71  5.13  8.76  4.44
## 8 128      50      0 0.830   3.85  4.66  8.78  4.83
## 9 256      50      0 1.3     5.54  5.12  8.08  2.72
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 5.6719, df = 8, p-value = 0.6839
```

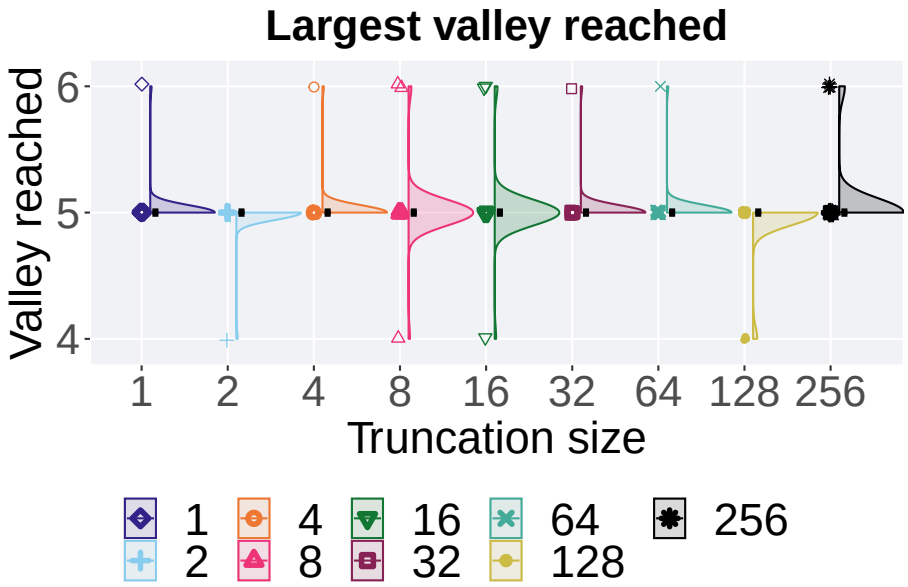
2.5.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'mpe' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = T, y = val, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(3.9,6.1),
    breaks = c(4,5,6)
  ) +
  scale_x_discrete(
    name="Truncation size"
  )+
  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)
)
```



2.5.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'mpe' & var == 'ele_big_peak')
valleys$T = factor(valleys$T, levels = TR_LIST)
valleys %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 1        50      0     5      5  5.02     6     0
## 2 2        50      0     4      5  4.98     5     0
## 3 4        50      0     5      5  5.02     6     0
## 4 8        50      0     4      5  5.02     6     0
## 5 16       50      0     4      5  5.02     6     0
## 6 32       50      0     5      5  5.02     6     0
## 7 64       50      0     5      5  5.02     6     0
## 8 128      50      0     4      5  4.94     5     0
## 9 256      50      0     5      5  5.08     6     0
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 13.997, df = 8, p-value = 0.08184
```

Chapter 3

Tournament selection

Results for the tournament selection parameter sweep on the diagnostics with valleys.

3.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME-MVC/tor.csv', sep = "", collapse = NULL), head
over_time_df$T <- factor(over_time_df$T, levels = TS_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST-MVC/tor.csv', sep = "", collapse = NULL), header = TRUE,
best_df$T <- factor(best_df$T, levels = TS_LIST)
```

3.2 Exploitation rate results

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

3.2.1 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 = filter(over_time_df, acro == 'exp') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

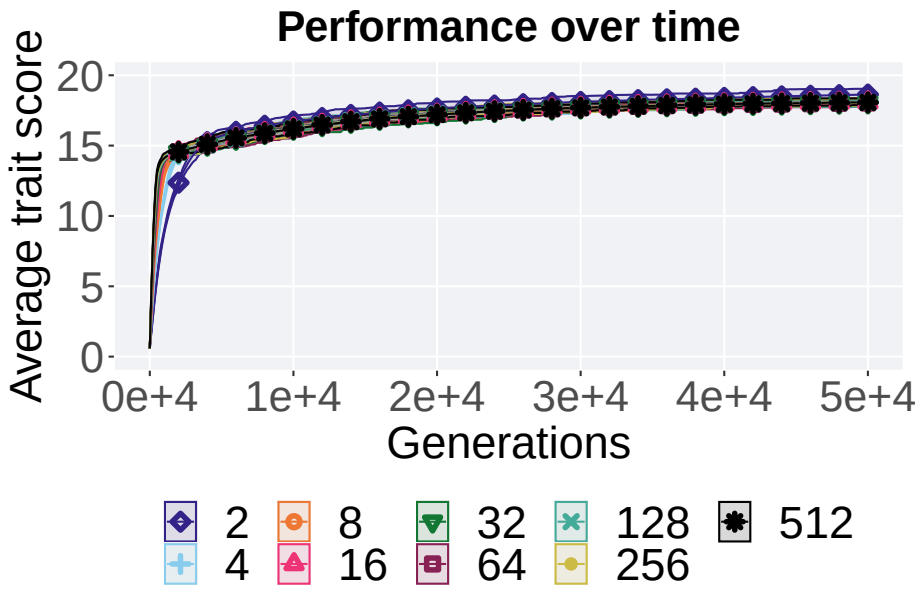
```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape
  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(0,20)
  ) +
  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()) +
  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")
  )

over_time_plot
```



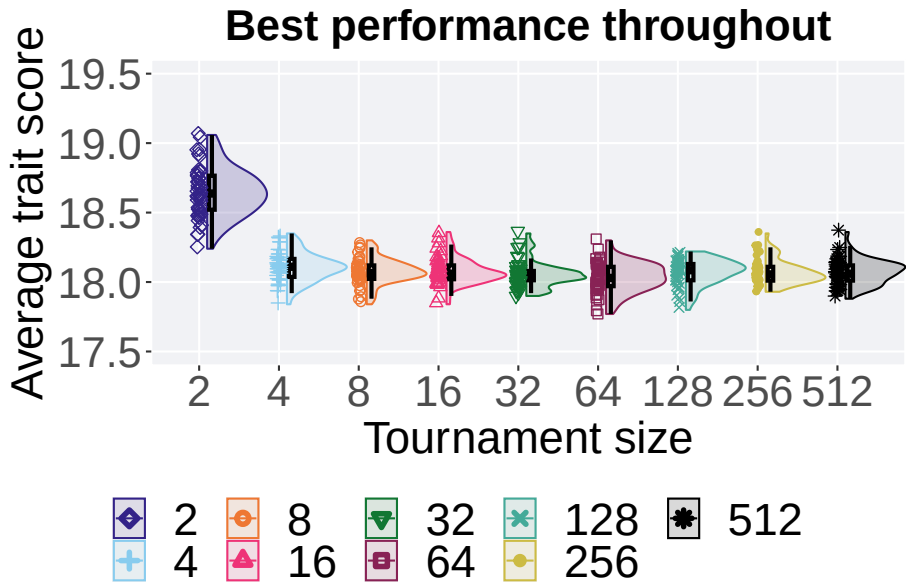
3.2.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'exp' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = T, y = val / DIMENSIONALITY, color = T, fill = T, shape = T)) +
  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, positio
  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(17.5, 19.5)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  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)
)
```



3.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'exp' &  var == 'pop_fit_max')
performance$T = factor(performance$T, levels = TS_LIST)
performance %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max    IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
```

```
## 1 2        50      0  18.2   18.6  18.6  19.1 0.246
## 2 4        50      0  17.8   18.1  18.1  18.3 0.130
## 3 8        50      0  17.8   18.1  18.1  18.3 0.0975
## 4 16       50      0  17.8   18.1  18.1  18.4 0.100
## 5 32       50      0  17.9   18.0  18.1  18.3 0.0675
## 6 64       50      0  17.8   18.0  18.0  18.3 0.140
## 7 128      50      0  17.8   18.1  18.1  18.2 0.117
## 8 256      50      0  17.9   18.1  18.1  18.3 0.0999
## 9 512      50      0  17.9   18.1  18.1  18.4 0.110
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 144.89, df = 8, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$T
##
##     2       4     8     16    32    64    128   256
## 4   3.9e-16 -     -     -     -     -     -     -
## 8   3.2e-16 1.000 -     -     -     -     -     -
## 16  3.6e-16 1.000 1.000 -     -     -     -     -
## 32  3.4e-16 0.188 1.000 1.000 -     -     -     -
## 64  2.8e-16 0.058 1.000 1.000 1.000 -     -     -
## 128 2.5e-16 1.000 1.000 1.000 1.000 1.000 -     -
## 256 3.2e-16 1.000 1.000 1.000 1.000 1.000 1.000 -
## 512 3.2e-16 1.000 1.000 1.000 1.000 1.000 1.000 1.000
##
## P value adjustment method: bonferroni
```

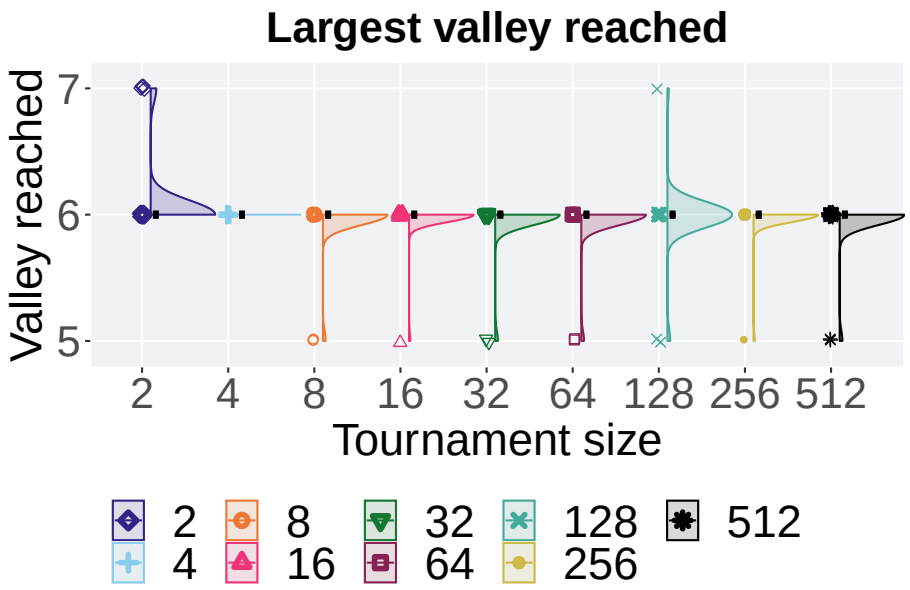
3.2.3 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'exp' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = T, y = val, color = T, fill = T, shape = T)) +
```

```
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(4.9,7.1),
    breaks = c(5,6,7)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  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)
)
```



3.2.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'exp' & var == 'ele_big_peak')
valleys$T = factor(valleys$T, levels = TS_LIST)
valleys %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0     6      6  6.08     7     0
## 2 4        50      0     6      6  6        6     0
## 3 8        50      0     5      6  5.96     6     0
## 4 16       50      0     5      6  5.98     6     0
## 5 32       50      0     5      6  5.96     6     0
## 6 64       50      0     5      6  5.96     6     0
## 7 128      50      0     5      6  5.98     7     0
## 8 256      50      0     5      6  5.98     6     0
## 9 512      50      0     5      6  5.96     6     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 15.545, df = 8, p-value = 0.04938
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
```

```
## data:  valleys$val and valleys$T
##
##     2    4    8    16   32   64   128  256
## 4   1.00 -    -    -    -    -    -    -
## 8   0.53 1.00 -    -    -    -    -    -
## 16  0.91 1.00 1.00 -    -    -    -    -
## 32  0.53 1.00 1.00 1.00 -    -    -    -
## 64  0.53 1.00 1.00 1.00 1.00 -    -    -
## 128 1.00 1.00 1.00 1.00 1.00 1.00 -    -
## 256 0.91 1.00 1.00 1.00 1.00 1.00 1.00 -
## 512 0.53 1.00 1.00 1.00 1.00 1.00 1.00 1.00
##
## P value adjustment method: bonferroni
```

3.3 Ordered exploitation results

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

3.3.1 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 = filter(over_time_df, acro == 'ord') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

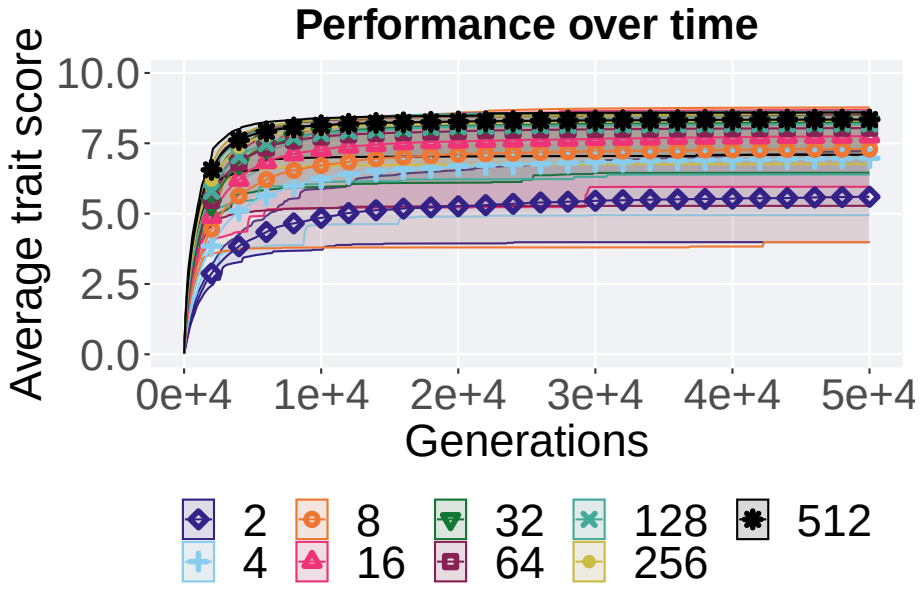
```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape
  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, 10)
  ) +
  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()) +
  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")
  )

over_time_plot
```



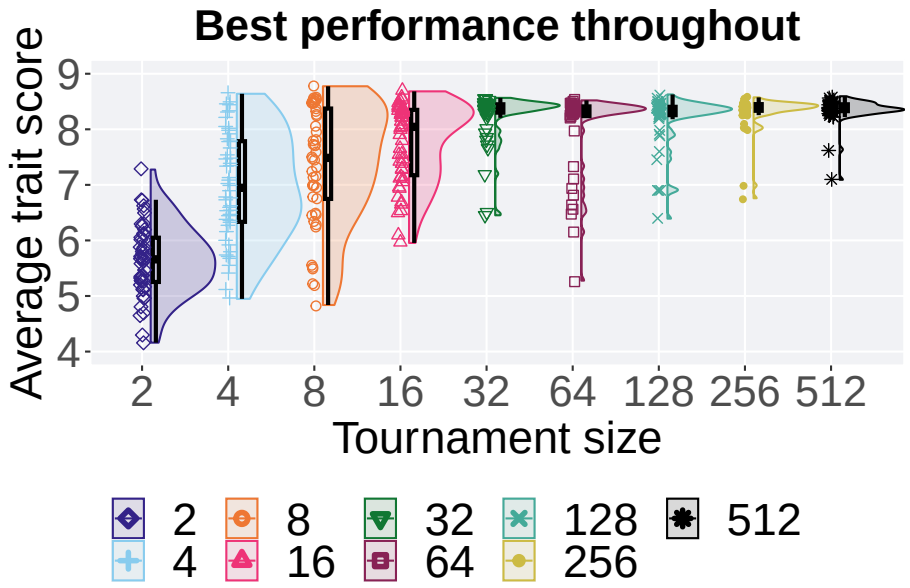
3.3.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'ord' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = T, y = val / DIMENSIONALITY, color = T, fill = T, shape = T)) +
```

```
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Average trait score",
    limits=c(4, 9)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  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)
)
```



3.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'ord' &  var == 'pop_fit_max')
performance$T = factor(performance$T, levels = TS_LIST)
performance %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0  4.00   5.65  5.61  7.28 0.840
## 2 4        50      0  4.95   6.95  6.97  8.64 1.45
## 3 8        50      0  3.99   7.48  7.29  8.77 1.82
## 4 16       50      0  5.95   8.04  7.75  8.68 1.18
## 5 32       50      0  6.45   8.39  8.24  8.57 0.151
## 6 64       50      0  5.28   8.35  8.04  8.52 0.175
## 7 128      50      0  6.39   8.35  8.18  8.63 0.150
## 8 256      50      0  6.76   8.39  8.31  8.58 0.120
## 9 512      50      0  7.09   8.37  8.34  8.60 0.124
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 198.69, df = 8, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
```

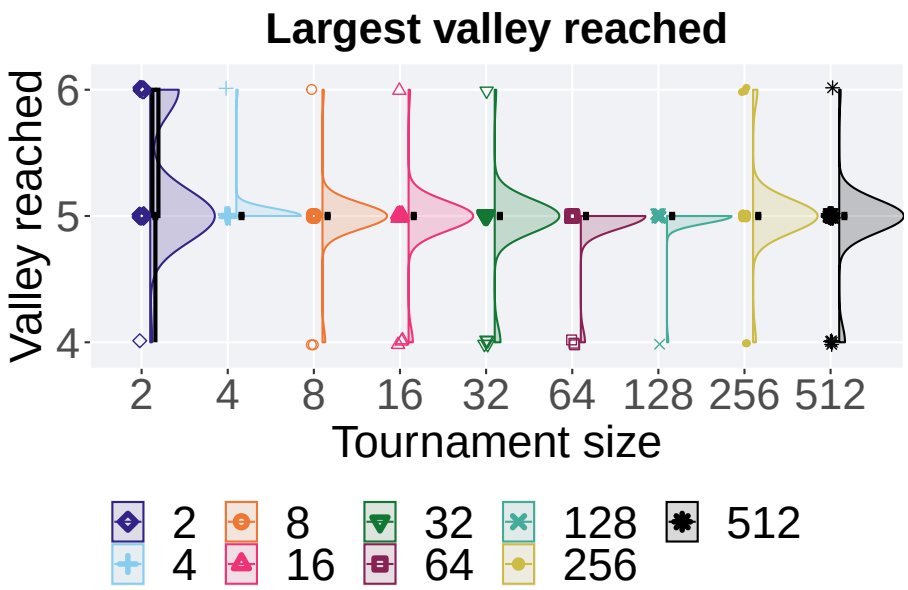
```
## data:  performance$val and performance$T
##
##     2       4       8       16      32      64      128     256
## 4   7.6e-09 -       -       -       -       -       -       -
## 8   2.7e-09 1.00000 -       -       -       -       -       -
## 16  5.1e-15 0.00201 1.00000 -       -       -       -       -
## 32  5.5e-16 4.4e-09 0.00320 0.00108 -       -       -       -
## 64  9.7e-15 2.2e-06 0.22098 0.34689 0.99449 -       -       -
## 128 4.4e-16 2.1e-08 0.01684 0.05029 1.00000 1.00000 -       -
## 256 2.9e-16 2.9e-10 0.00114 0.00011 1.00000 0.32009 1.00000 -
## 512 2.7e-16 9.6e-11 0.00037 8.8e-05 1.00000 1.00000 1.00000 1.00000
##
## P value adjustment method: bonferroni
```

3.3.3 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, acro == 'ord' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = T, y = val, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(3.9,6.1),
    breaks = c(4,5,6)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  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)
)
```



3.3.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'ord' & var == 'ele_big_peak')
valleys$T = factor(valleys$T, levels = TS_LIST)
valleys %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0     4      5  5.28     6     1
## 2 4        50      0     5      5  5.02     6     0
## 3 8        50      0     4      5  4.98     6     0
## 4 16       50      0     4      5  4.96     6     0
## 5 32       50      0     4      5  4.94     6     0
```

```
## 6 64       50      0     4      5  4.94     5     0
## 7 128      50      0     4      5  4.98     5     0
## 8 256      50      0     4      5  5.02     6     0
## 9 512      50      0     4      5  4.94     6     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 48.274, df = 8, p-value = 8.755e-08
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$T
##
##     2      4      8      16     32     64     128    256
## 4   0.0209 -      -      -      -      -      -      -
## 8   0.0098 1.0000 -      -      -      -      -      -
## 16  0.0066 1.0000 1.0000 -      -      -      -      -
## 32  0.0044 1.0000 1.0000 1.0000 -      -      -      -
## 64  0.0017 1.0000 1.0000 1.0000 1.0000 -      -      -
## 128 0.0037 1.0000 1.0000 1.0000 1.0000 1.0000 -      -
## 256 0.0885 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 -
## 512 0.0044 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
##
## P value adjustment method: bonferroni
```

3.4 Contradictory objectives resutls

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

3.4.1 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.

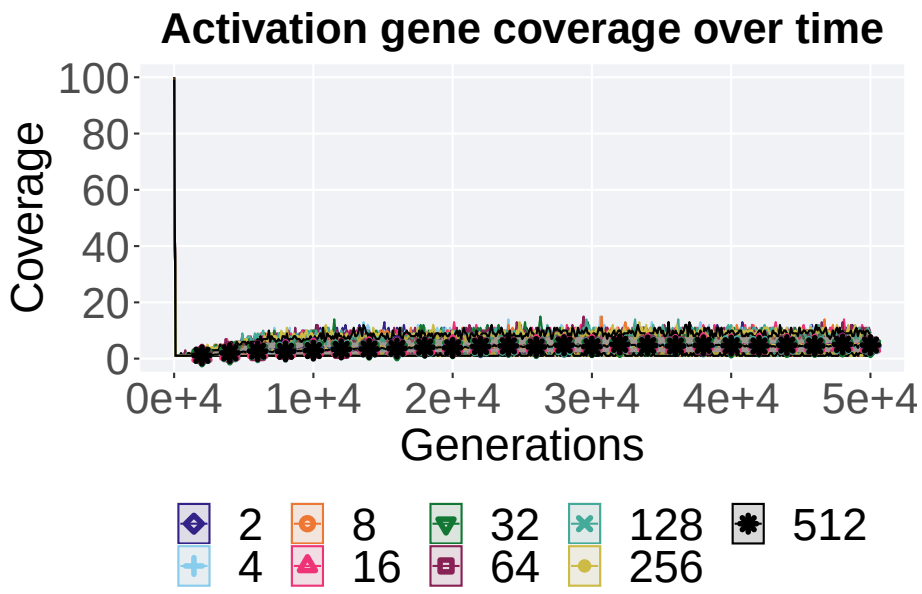
Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



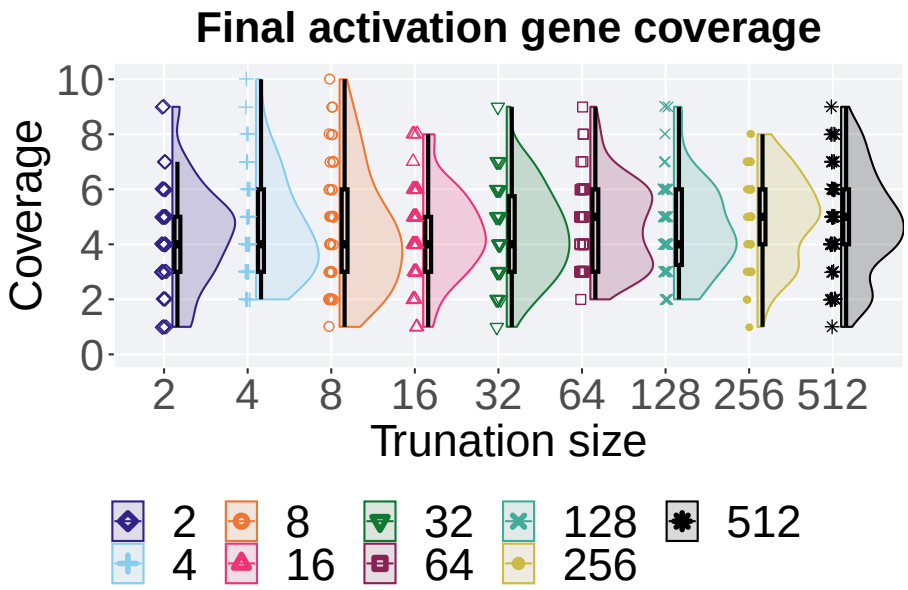
3.4.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 10.1),
    breaks=seq(0,10,2)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  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)
)
```



3.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$T = factor(act_coverage$T, levels = TS_LIST)
act_coverage %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 2        50      0     1      4  4.32     9  2
## 2 4        50      0     2      4  4.6     10  3
## 3 8        50      0     1      4  4.54    10  3
## 4 16       50      0     1      4  4.42     8  2
## 5 32       50      0     1      4  4.42     9  2.75
## 6 64       50      0     2      5  4.76     9  3
## 7 128      50      0     2      4  4.6      9  2.75
## 8 256      50      0     1      5  4.92     8  2
## 9 512      50      0     1      5  4.82     9  2
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 6.0815, df = 8, p-value = 0.6381
```

3.4.3 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.

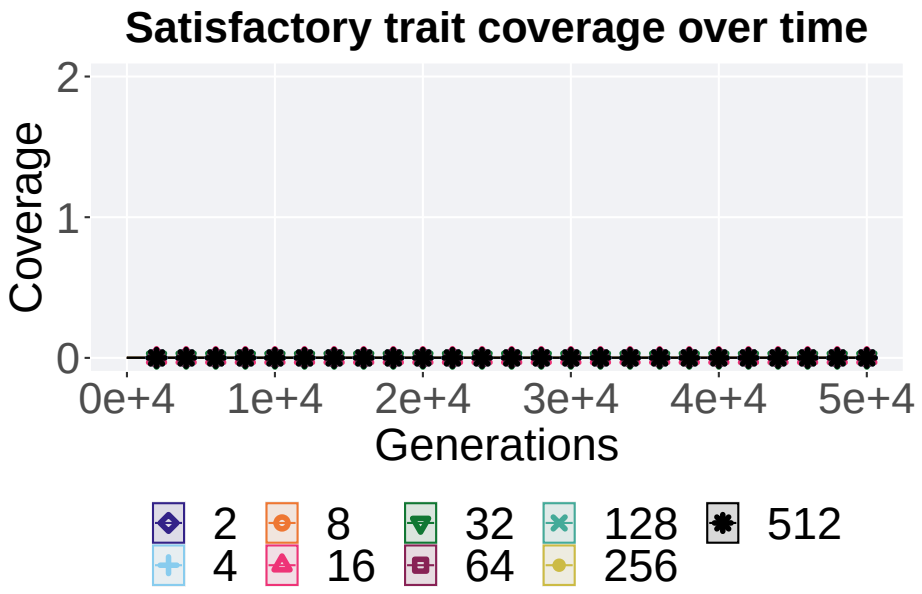
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  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, 2),
    breaks=c(0,1,2)
  ) +
  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()) +
  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")
  )
```



3.4.4 Final satisfactory trait coverage

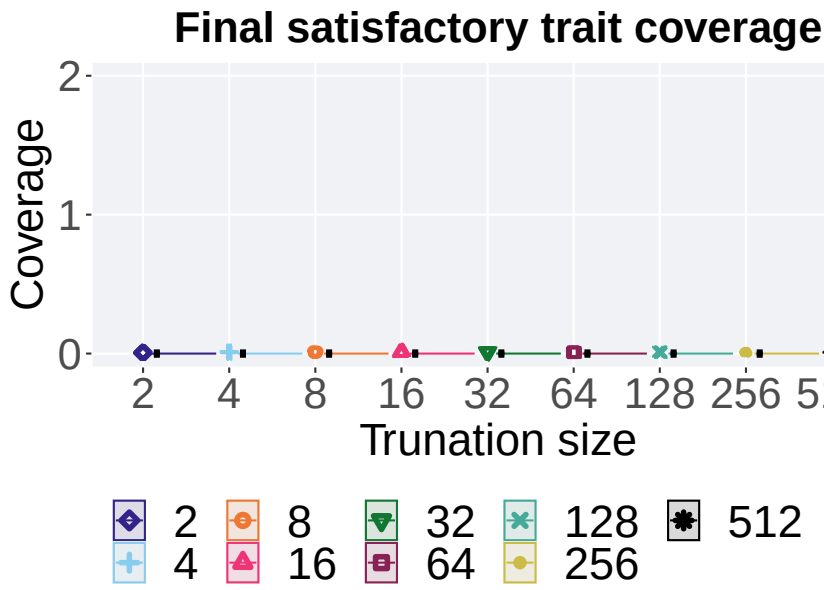
Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = T, y = pop_uni_obj, color = T, fill = T, shape = T)) +
  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, positio
```

```
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  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 231 rows containing missing values (`geom_point()`).
```



3.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$T = factor(sat_coverage$T, levels = TS_LIST)
sat_coverage %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 2        50      0     0      0     0     0     0
## 2 4        50      0     0      0     0     0     0
## 3 8        50      0     0      0     0     0     0
## 4 16       50      0     0      0     0     0     0
## 5 32       50      0     0      0     0     0     0
## 6 64       50      0     0      0     0     0     0
## 7 128      50      0     0      0     0     0     0
## 8 256      50      0     0      0     0     0     0
## 9 512      50      0     0      0     0     0     0
```

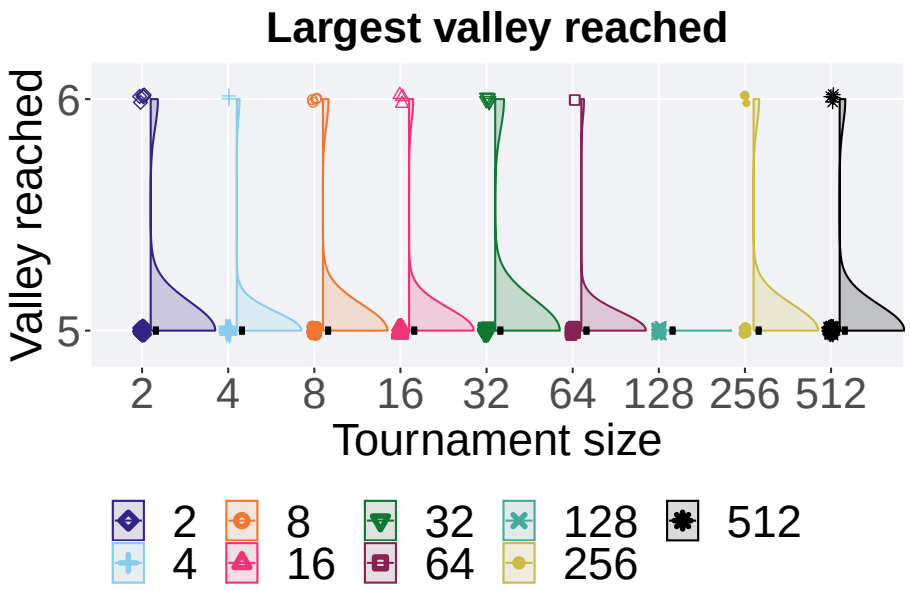
3.4.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'con' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = T, y = val, color = T, fill = T, shape = T)) +
  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, positio
  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(4.9,6.1),
    breaks = c(5,6)
  ) +
  scale_x_discrete(
```

```
    name="Tournament size"
  )+
  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)
)
```



3.4.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'con' & var == 'ele_big_peak')
valleys$T = factor(valleys$T, levels = TS_LIST)
valleys %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0     5      5  5.1      6     0
## 2 4        50      0     5      5  5.04     6     0
## 3 8        50      0     5      5  5.08     6     0
## 4 16       50      0     5      5  5.06     6     0
## 5 32       50      0     5      5  5.12     6     0
## 6 64       50      0     5      5  5.04     6     0
## 7 128      50      0     5      5  5        5     0
## 8 256      50      0     5      5  5.08     6     0
## 9 512      50      0     5      5  5.08     6     0
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 8.3386, df = 8, p-value = 0.4011
```

3.5 Multi-path exploration results

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

3.5.1 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.

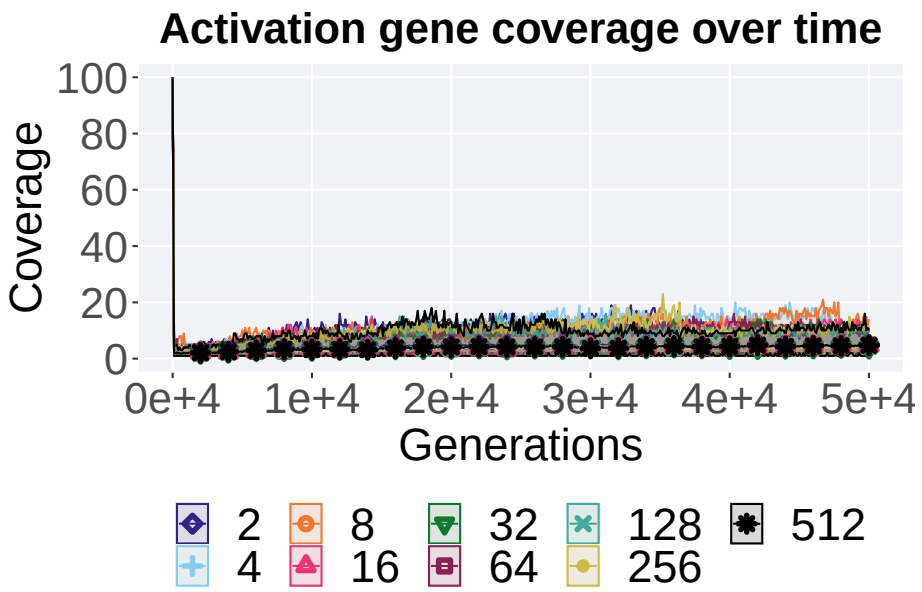
```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(T, gen) %>%
```

```
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  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()) +
  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")
  )
```



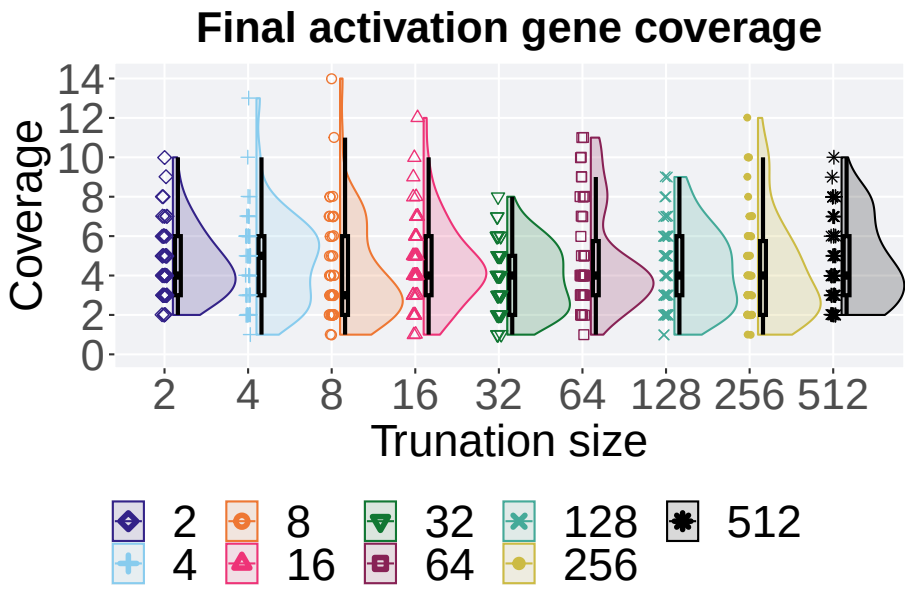
3.5.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  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, positio
  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, 14.1),
    breaks=seq(0,14,2)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  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)
)
```



3.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$T = factor(act_coverage$T, levels = TS_LIST)
act_coverage %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 2        50      0     2      4  4.66    10  3
## 2 4        50      0     1      5  4.76    13  3
## 3 8        50      0     1      3  4.26    14  4
## 4 16       50      0     1      4  4.66    12  3
## 5 32       50      0     1      4  3.86     8  3
## 6 64       50      0     1      4  4.78    11  2.75
## 7 128      50      0     1      4  4.36     9  3
## 8 256      50      0     1      4  4.34    12  3.75
## 9 512      50      0     2      4  4.76    10  3
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 9.3181, df = 8, p-value = 0.3162
```

3.5.3 Performance over time

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

```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

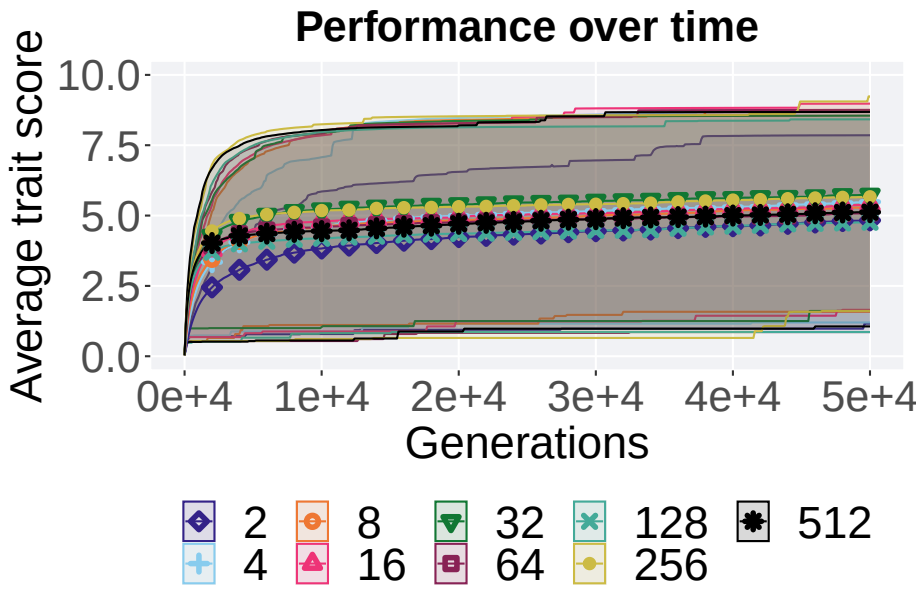
```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 10)
  ) +
  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()) +
  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")
  )

over_time_plot
```



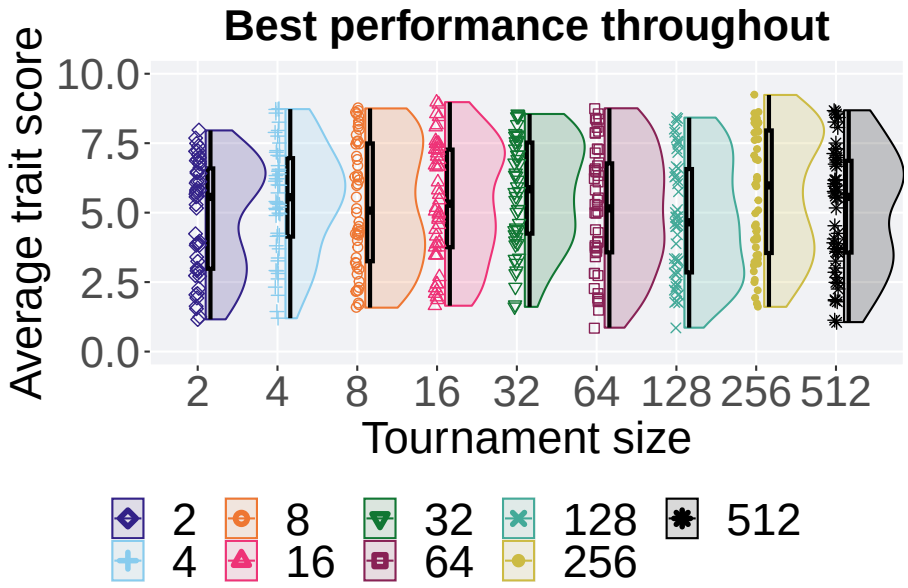
3.5.4 Best performance throughout

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

```
plot = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = T, y = val / DIMENSIONALITY, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
```

```
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  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, 10)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  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)
)
```



3.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max')
performance$T = factor(performance$T, levels = TS_LIST)
performance %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0 1.16    5.57  4.85  7.96  3.61
## 2 4        50      0 1.20    5.56  5.49  8.73  2.83
## 3 8        50      0 1.58    5.07  5.29  8.75  4.24
## 4 16       50      0 1.65    5.32  5.39  8.98  3.52
## 5 32       50      0 1.61    5.84  5.73  8.55  3.28
## 6 64       50      0 0.860   5.15  5.09  8.76  3.21
## 7 128      50      0 0.860   4.65  4.76  8.42  3.72
## 8 256      50      0 1.61    5.98  5.67  9.24  4.41
## 9 512      50      0 1.06    5.57  5.12  8.68  3.30
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 9.8288, df = 8, p-value = 0.2772
```

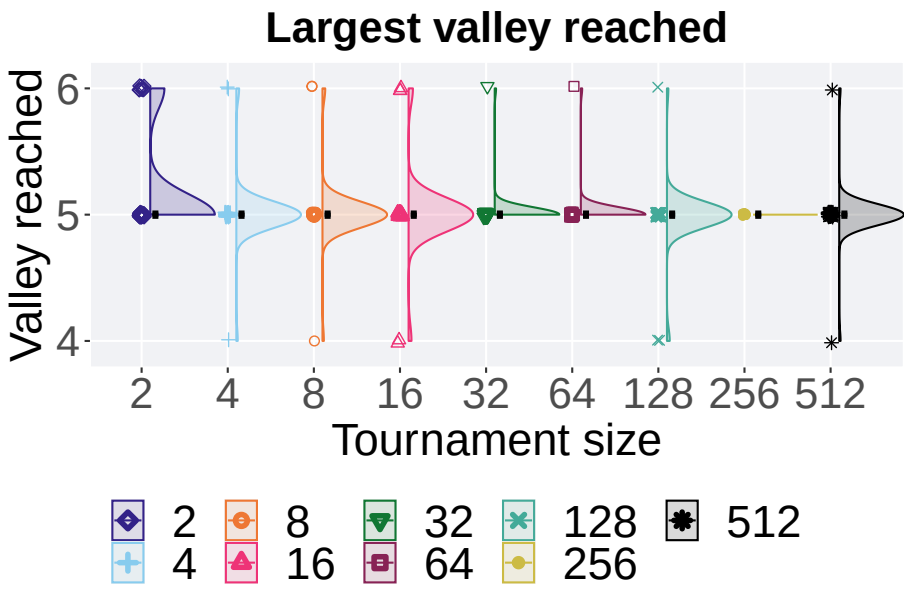
3.5.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'mpe' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = T, y = val, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
```

```
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  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(3.9,6.1),
    breaks = c(4,5,6)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  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)
)
```



3.5.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df,acro == 'mpe' & var == 'ele_big_peak')
valleys$T = factor(valleys$T, levels = TS_LIST)
valleys %>%
  group_by(T) %>%
  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: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0     5      5  5.18     6     0
## 2 4        50      0     4      5  5.02     6     0
## 3 8        50      0     4      5  5.02     6     0
## 4 16       50      0     4      5  5.02     6     0
## 5 32       50      0     5      5  5.02     6     0
## 6 64       50      0     5      5  5.02     6     0
## 7 128      50      0     4      5  4.98     6     0
## 8 256      50      0     5      5  5        5     0
## 9 512      50      0     4      5  5        6     0
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 23.215, df = 8, p-value = 0.003099
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
```

```
## data:  valleys$val and valleys$T
##
##     2     4     8     16    32    64    128   256
## 4   0.612 -     -     -     -     -     -     -
## 8   0.612 1.000 -     -     -     -     -     -
## 16  1.000 1.000 1.000 -     -     -     -     -
## 32  0.293 1.000 1.000 1.000 -     -     -     -
## 64  0.293 1.000 1.000 1.000 1.000 -     -     -
## 128 0.117 1.000 1.000 1.000 1.000 1.000 -     -
## 256 0.065 1.000 1.000 1.000 1.000 1.000 1.000 -
## 512 0.186 1.000 1.000 1.000 1.000 1.000 1.000 1.000
##
## P value adjustment method: bonferroni
```

Chapter 4

Genotypic fitness sharing

Results for the genotypic fitness sharing parameter sweep on the diagnostics with valleys.

4.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME-MVC/gfs.csv', sep = "", collapse = NULL), head
over_time_df$Sigma <- factor(over_time_df$Sigma, levels = FS_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST-MVC/gfs.csv', sep = "", collapse = NULL), header = TRUE,
best_df$Sigma <- factor(best_df$Sigma, levels = FS_LIST)
```

4.2 Exploitation rate results

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

4.2.1 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 = filter(over_time_df, acro == 'exp') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

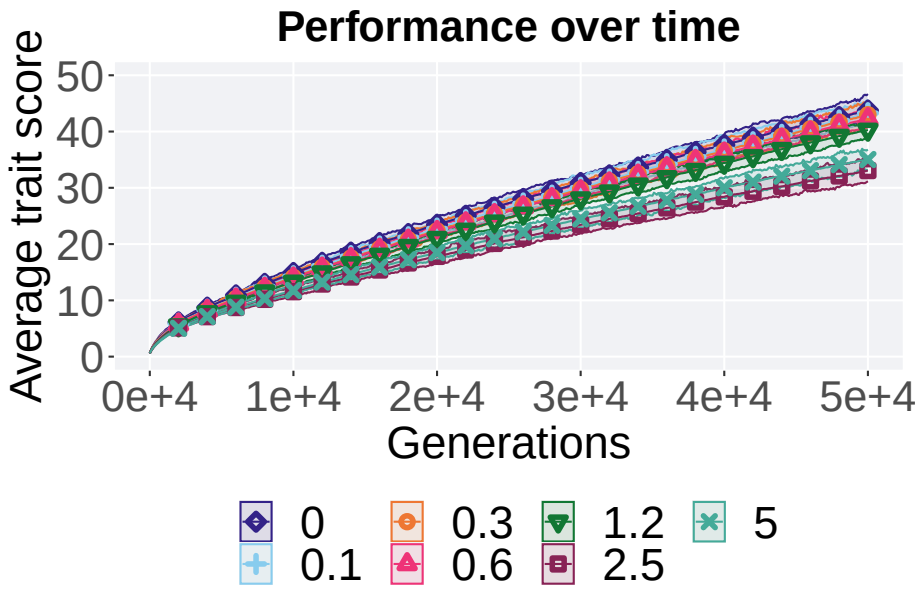
```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color =
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Average trait score",
    limits = c(0,50)
  ) +
  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()) +
  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")
  )

over_time_plot
```



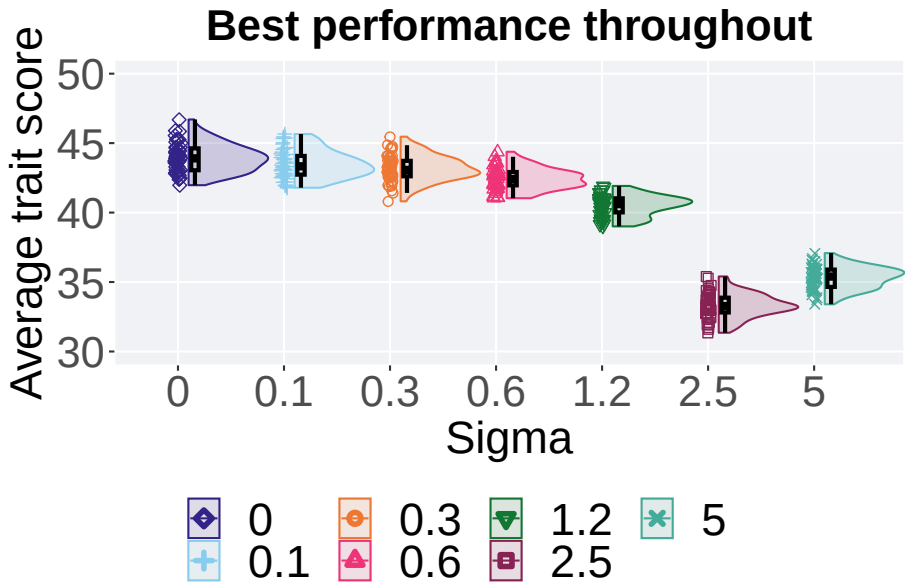
4.2.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'exp' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  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, positio
  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(30,50)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'exp' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels = FS_LIST)
performance %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 0        50      0  42.0   43.9  43.9  46.7 1.58
## 2 0.1      50      0  41.8   43.3  43.5  45.7 1.35
## 3 0.3      50      0  40.8   43.1  43.2  45.5 1.11
## 4 0.6      50      0  41.0   42.4  42.4  44.4 0.963
## 5 1.2      50      0  39.0   40.6  40.5  41.9 1.01
## 6 2.5      50      0  31.3   33.3  33.3  35.4 1.07
## 7 5        50      0  33.4   35.4  35.3  37.1 1.25
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 294.25, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 0.6669  -       -       -       -       -
## 0.3 0.0125  1.0000  -       -       -       -
## 0.6 1.5e-09 1.5e-05 0.0024  -       -       -
## 1.2 < 2e-16 < 2e-16 1.1e-15 6.0e-15 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.7e-13
##
## P value adjustment method: bonferroni
```

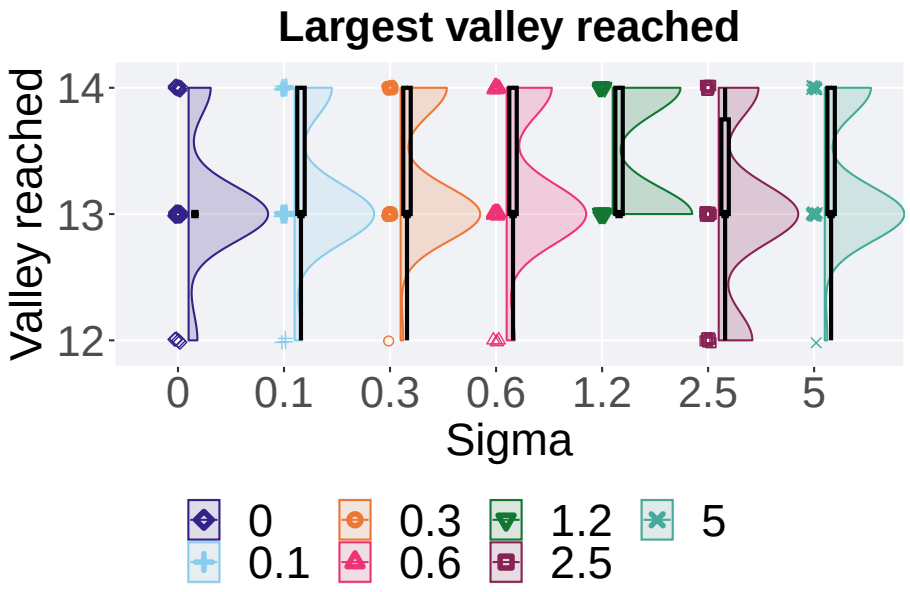
4.2.3 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'exp' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  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, positio
  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(11.9,14.1),
    breaks = c(11,12,13,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.2.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'exp' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = FS_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0    12     13  13.1    14  0
## 2 0.1      50      0    12     13  13.2    14  1
## 3 0.3      50      0    12     13  13.3    14  1
## 4 0.6      50      0    12     13  13.3    14  1
## 5 1.2      50      0    13     13  13.5    14  1
## 6 2.5      50      0    12     13  13.0    14  0.75
## 7 5        50      0    12     13  13.3    14  1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 16.366, df = 6, p-value = 0.01192
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0     0.1   0.3   0.6   1.2   2.5
## 0.1 1.000 -     -     -     -     -
## 0.3 0.869 1.000 -     -     -     -
```

```
## 0.6 1.000 1.000 1.000 -     -     -
## 1.2 0.042 1.000 1.000 1.000 -     -
## 2.5 1.000 1.000 0.589 1.000 0.046 -
## 5   0.869 1.000 1.000 1.000 1.000 0.589
##
## P value adjustment method: bonferroni
```

4.3 Ordered exploitation results

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

4.3.1 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 = filter(over_time_df, acro == 'ord') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

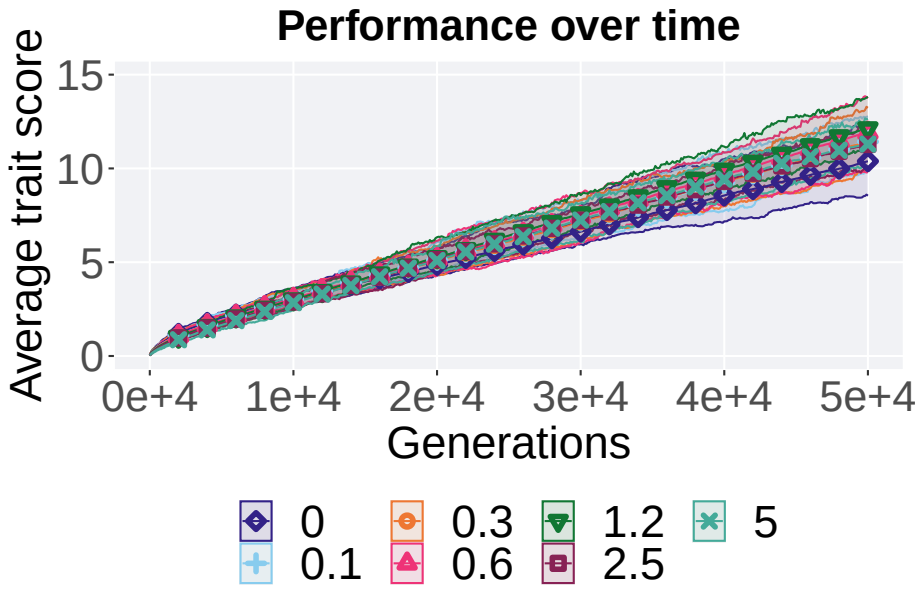
```
## `summarise()` has grouped output by 'Sigma'. You can override using the
## `.groups` argument.
```

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

  ) +
  scale_shape_manual(values=SHAPE)+
```

```
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  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")
  )

over_time_plot
```



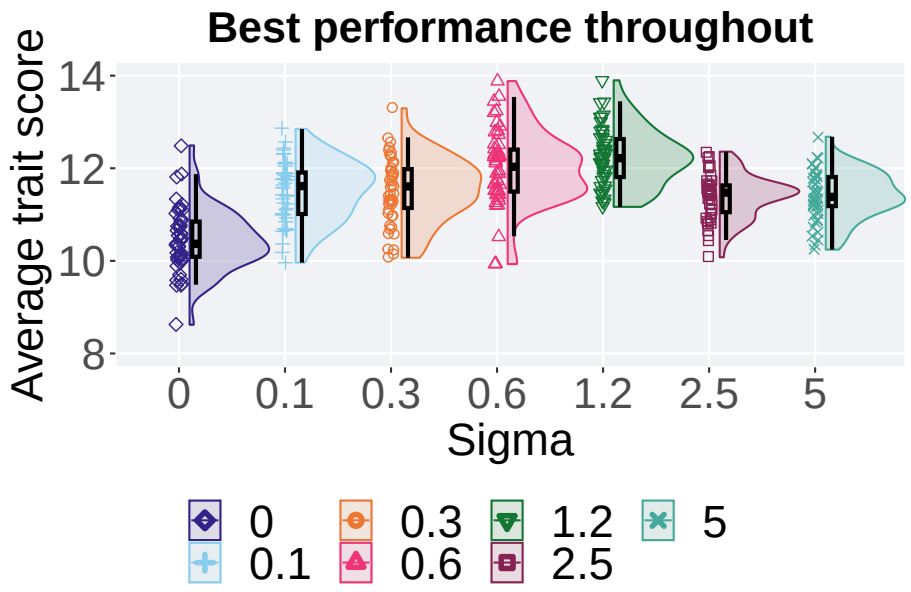
4.3.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'ord' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  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, positio
  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(8,14)
  ) +
```

```
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'ord' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels = FS_LIST)
performance %>%
  group_by(Sigma) %>%
```

```
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  8.62   10.4  10.5  12.5 0.761
## 2 0.1      50      0  9.96   11.6  11.5  12.8 0.898
## 3 0.3      50      0 10.1    11.6  11.5  13.3 0.840
## 4 0.6      50      0  9.93   12.0  12.0  13.9 0.908
## 5 1.2      50      0 11.2    12.2  12.2  13.9 0.821
## 6 2.5      50      0 10.1    11.5  11.4  12.4 0.579
## 7 5        50      0 10.2    11.4  11.4  12.7 0.629
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 128.96, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 5.3e-09 -       -       -       -       -
## 0.3 6.1e-09 1.00000 -       -       -       -
## 0.6 4.5e-12 0.04909 0.06661 -       -       -
## 1.2 8.0e-15 3.3e-05 0.00010 1.00000 -       -
## 2.5 1.9e-09 1.00000 1.00000 0.00035 2.6e-08 -
## 5   2.0e-09 1.00000 1.00000 0.00056 8.8e-08 1.00000
```

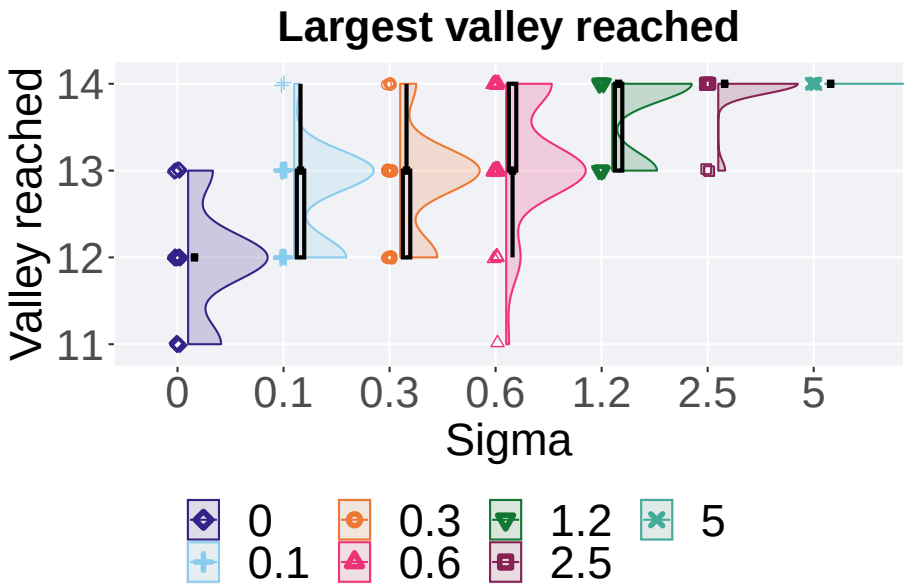
```
##
## P value adjustment method: bonferroni
```

4.3.3 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, acro == 'ord' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(10.9,14.1),
    breaks = c(11,12,13,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.3.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'ord' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = FS_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0    11     12  11.9    13     0
## 2 0.1      50      0    12     13  12.7    14     1
## 3 0.3      50      0    12     13  12.8    14     1
## 4 0.6      50      0    11     13  13.2    14     1
## 5 1.2      50      0    13     14  13.6    14     1
```

```
## 6 2.5      50      0    13     14  13.9    14     0
## 7 5        50      0    14     14  14      14     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 231.15, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 7.9e-06 -       -       -       -       -
## 0.3 1.1e-07 1.0000  -       -       -       -
## 0.6 6.4e-11 0.0013  0.1424  -       -       -
## 1.2 1.0e-15 1.2e-10 9.6e-08 0.0093  -       -
## 2.5 < 2e-16 3.3e-16 9.9e-14 1.6e-08 0.0165  -
## 5   < 2e-16 < 2e-16 3.9e-16 3.8e-11 6.8e-05 0.9092
##
## P value adjustment method: bonferroni
```

4.4 Contradictory objectives results

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

4.4.1 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.

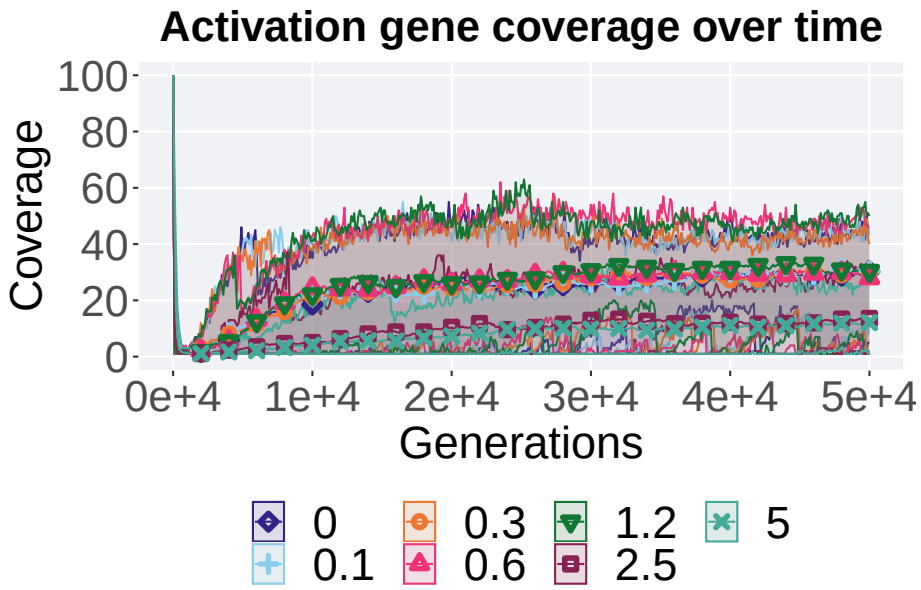
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
```

```
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



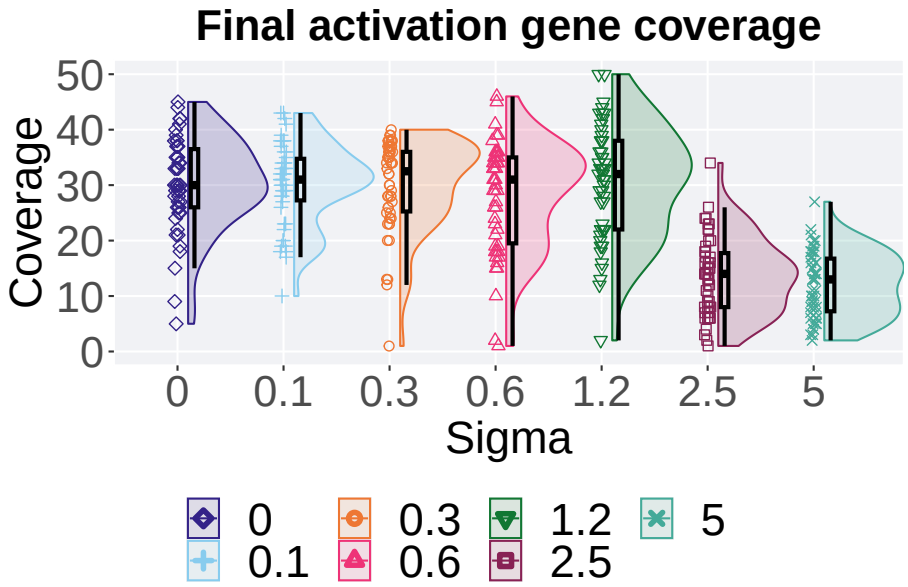
4.4.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 50.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = FS_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0        50      0     5   30    30.2    45 10.5
## 2 0.1      50      0    10   31    30.3    43  7.5
## 3 0.3      50      0     1   32.5  30      40 10.8
## 4 0.6      50      0     1   31    27.9    46 15.5
## 5 1.2      50      0     2   32    30.4    50 16
## 6 2.5      50      0     1   14    13.6    34  9.75
## 7 5        50      0     2   13    12.1    27  9.5
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 157.28, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.met
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1       -       -       -       -       -
## 0.3 1       1       -       -       -       -
## 0.6 1       1       1       -       -       -
## 1.2 1       1       1       1       -       -
## 2.5 2.7e-12 7.4e-13 6.8e-12 2.1e-09 9.8e-11 -
## 5   1.5e-13 2.0e-14 4.5e-13 1.1e-10 1.8e-12 1
##
## P value adjustment method: bonferroni
```

4.4.3 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.

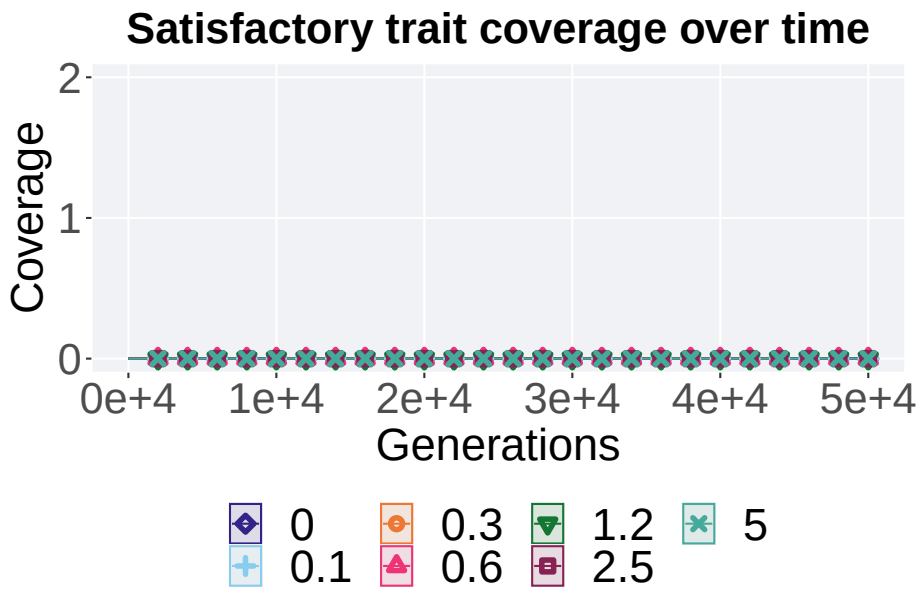
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
```

```
    max = max(pop_uni_obj)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  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()) +
  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")
  )
```



4.4.4 Final satisfactory trait coverage

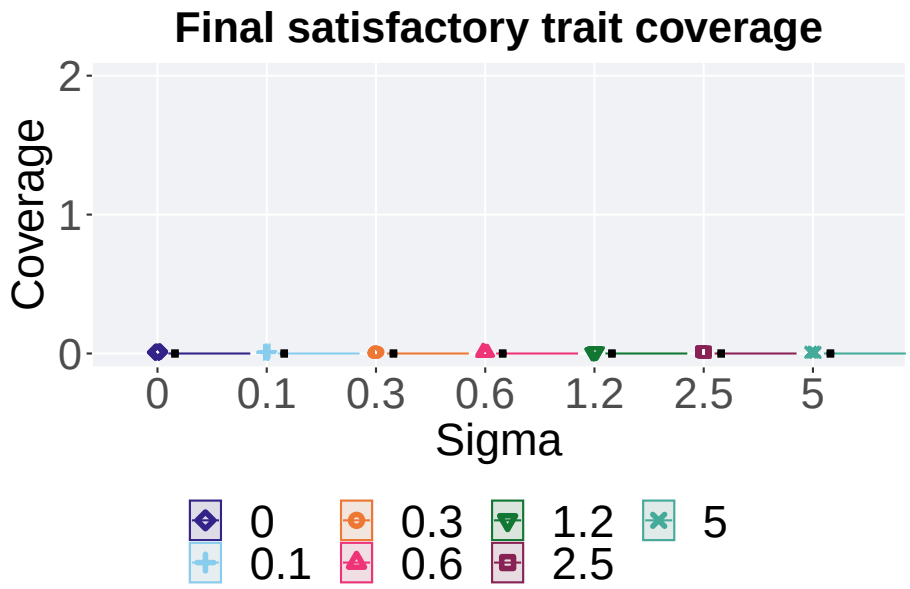
Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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 175 rows containing missing values (`geom_point()`).
```



4.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$Sigma = factor(sat_coverage$Sigma, levels = FS_LIST)
sat_coverage %>%
  group_by(Sigma) %>%
  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)
  )
```

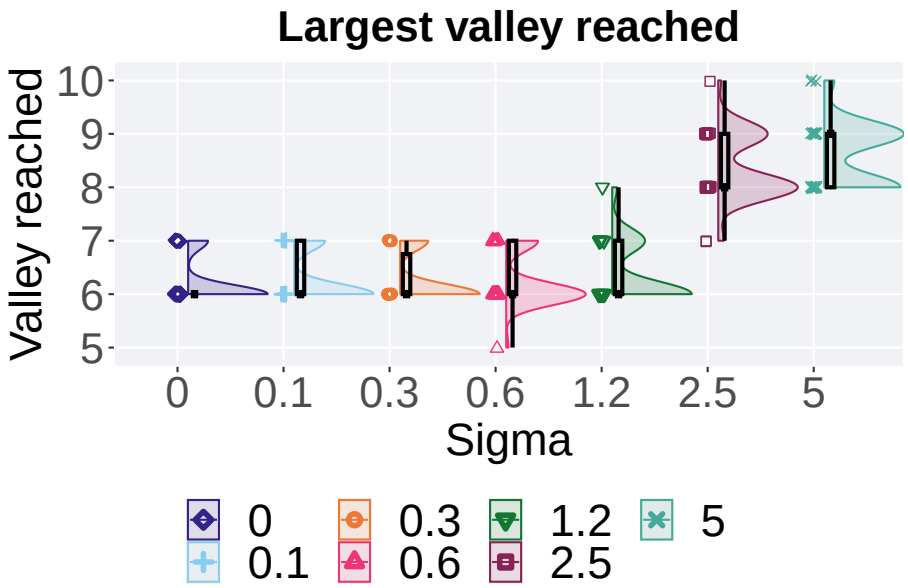
```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 0        50      0     0      0     0     0     0
## 2 0.1      50      0     0      0     0     0     0
## 3 0.3      50      0     0      0     0     0     0
## 4 0.6      50      0     0      0     0     0     0
## 5 1.2      50      0     0      0     0     0     0
## 6 2.5      50      0     0      0     0     0     0
## 7 5        50      0     0      0     0     0     0
```

4.4.5 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, acro == 'con' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(4.9,10.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.4.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'con' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = FS_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0     6      6  6.2      7  0
## 2 0.1      50      0     6      6  6.28     7  1
## 3 0.3      50      0     6      6  6.26     7  0.75
## 4 0.6      50      0     5      6  6.26     7  1
## 5 1.2      50      0     6      6  6.36     8  1
```

```
## 6 2.5      50      0     7      8  8.36    10  1
## 7 5        50      0     8      9  8.6     10  1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 248.66, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0      0.1    0.3    0.6    1.2    2.5
## 0.1 1      -      -      -      -      -
## 0.3 1      1      -      -      -      -
## 0.6 1      1      1      -      -      -
## 1.2 1      1      1      1      -      -
## 2.5 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## 5   <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 1
##
## P value adjustment method: bonferroni
```

4.5 Multi-path exploration results

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

4.5.1 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.

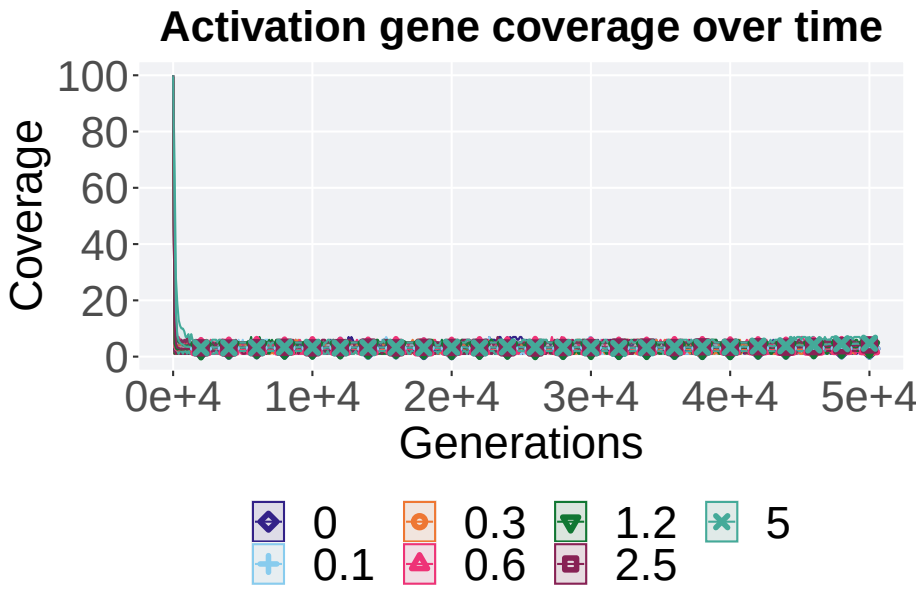
```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(Sigma, gen) %>%
```

```
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



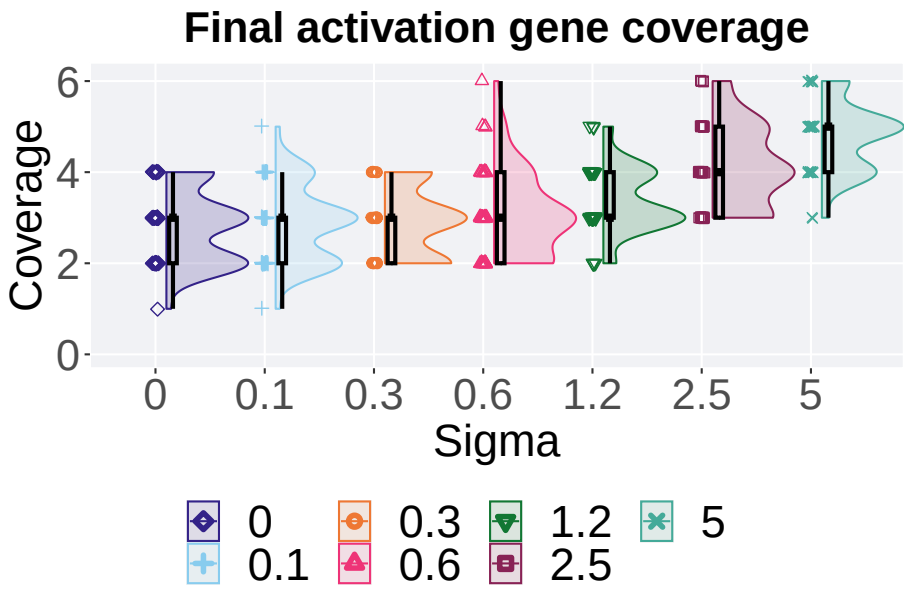
4.5.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 6.1),
    breaks=seq(0,6,2)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = FS_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 0        50      0     1      3  2.8      4     1
## 2 0.1      50      0     1      3  2.86     5     1
## 3 0.3      50      0     2      3  2.9      4     1
## 4 0.6      50      0     2      3  3.08     6     2
## 5 1.2      50      0     2      3  3.38     5     1
## 6 2.5      50      0     3      4  4.14     6     2
## 7 5        50      0     3      5  4.76     6     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 138.2, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.met
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1.0000  -       -       -       -       -
## 0.3 1.0000  1.0000  -       -       -       -
## 0.6 1.0000  1.0000  1.0000  -       -       -
## 1.2 0.0128  0.0347  0.0682  0.7788  -       -
## 2.5 1.7e-08 6.2e-08 7.8e-08 1.1e-05 0.0012  -
## 5   4.8e-14 1.6e-13 8.5e-14 4.3e-11 9.6e-11 0.0129
##
## P value adjustment method: bonferroni
```

4.5.3 Performance over time

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

```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
```

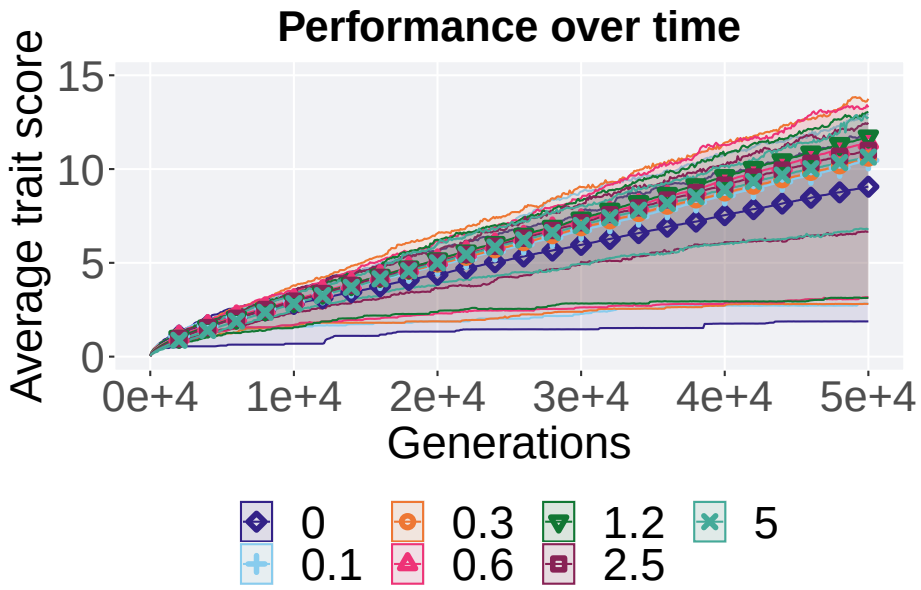
```
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, sha
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 15)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  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")
  )

over_time_plot
```



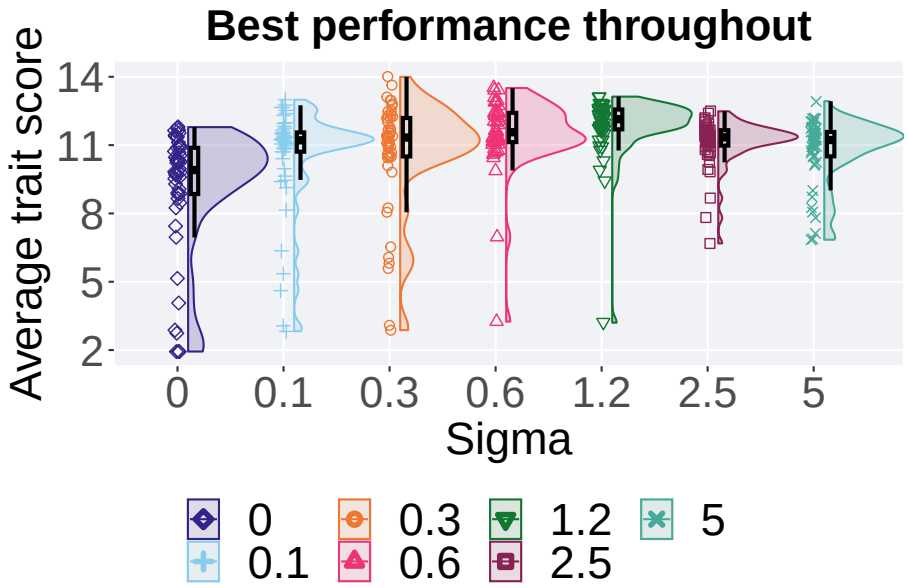
4.5.4 Best performance throughout

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

```
plot = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Average trait score",
    limits=c(1.9, 14.1),
    breaks=c(2,5,8,11,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels =FS_LIST)
performance %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  1.94   9.90  9.12  11.8 2.03
## 2 0.1      50      0  2.83  11.2  10.5   13.0 0.846
## 3 0.3      50      0  2.88  11.3  10.7   14.0 1.69
## 4 0.6      50      0  3.25  11.6  11.5   13.5 1.29
## 5 1.2      50      0  3.20  12.1  11.8   13.1 0.870
## 6 2.5      50      0  6.68  11.3  11.1   12.5 0.677
## 7 5        50      0  6.85  11.2  10.8   12.9 1.07
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 80.554, df = 6, p-value = 2.745e-15
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bo
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 0.00157 -       -       -       -       -
## 0.3 0.00067 1.00000 -       -       -       -
## 0.6 1.6e-08 0.16193 1.00000 -       -       -
## 1.2 6.8e-12 8.0e-05 0.01514 1.00000 -       -
## 2.5 2.3e-06 1.00000 1.00000 0.58011 8.3e-06 -
## 5   0.00075 1.00000 1.00000 0.08218 3.1e-06 1.00000
##
## P value adjustment method: bonferroni
```

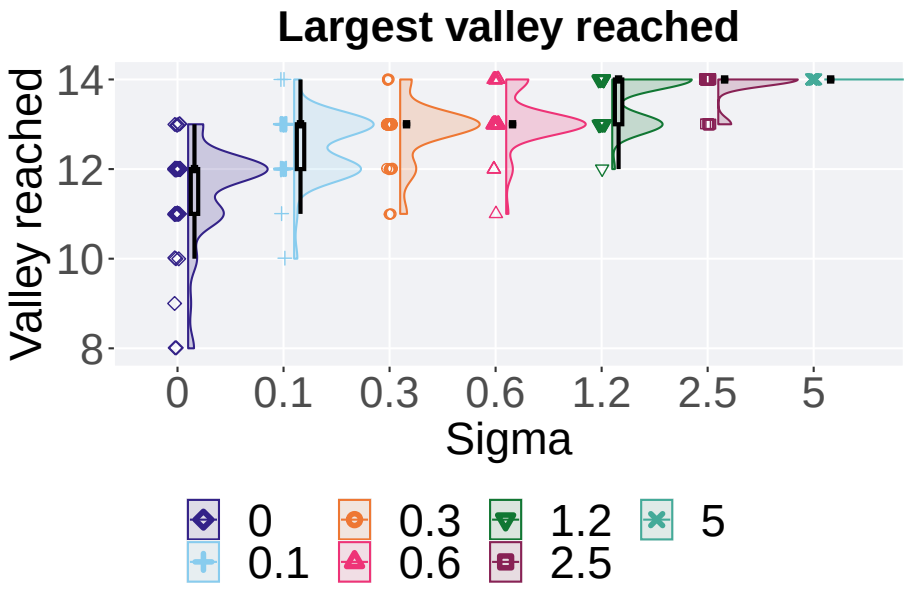
4.5.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'mpe' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
```

```
  scale_y_continuous(
    name="Valley reached",
    limits=c(7.9,14.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



4.5.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'mpe' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = FS_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0     8     12  11.5    13     1
## 2 0.1      50      0    10     13  12.5    14     1
## 3 0.3      50      0    11     13  12.8    14     0
## 4 0.6      50      0    11     13  13.1    14     0
## 5 1.2      50      0    12     14  13.6    14     1
## 6 2.5      50      0    13     14  13.9    14     0
## 7 5        50      0    14     14  14      14     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 238.55, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 2.2e-06 -       -       -       -       -
## 0.3 4.4e-10 0.19650 -       -       -       -
```

```
## 0.6 2.1e-13 0.00022 0.88310 -       -       -
## 1.2 6.9e-16 4.6e-10 1.2e-06 0.00092 -       -
## 2.5 < 2e-16 5.9e-15 1.2e-12 9.9e-10 0.07000 -
## 5   < 2e-16 < 2e-16 < 2e-16 2.1e-14 1.4e-05 0.13628
##
## P value adjustment method: bonferroni
```

Chapter 5

Phenotypic fitness sharing

Results for the phenotypic fitness sharing parameter sweep on the diagnostics with valleys.

5.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME-MVC/pfs.csv', sep = "", collapse = NULL), head
over_time_df$Sigma <- factor(over_time_df$Sigma, levels = FS_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST-MVC/pfs.csv', sep = "", collapse = NULL), header = TRUE,
best_df$Sigma <- factor(best_df$Sigma, levels = FS_LIST)
```

5.2 Exploitation rate results

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

5.2.1 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 = filter(over_time_df, acro == 'exp') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

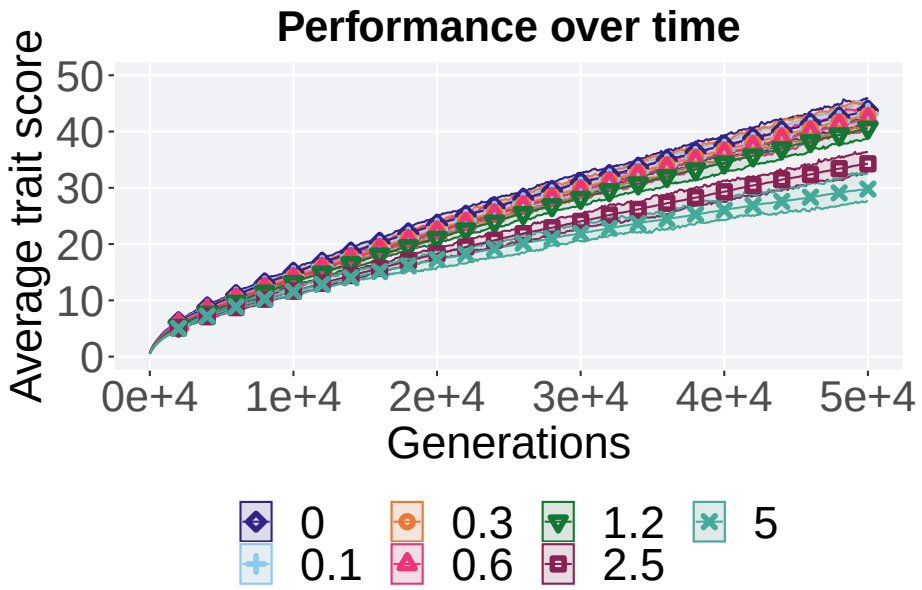
```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color =
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Average trait score",
    limits = c(0,50)
  ) +
  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()) +
  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")
  )

over_time_plot
```



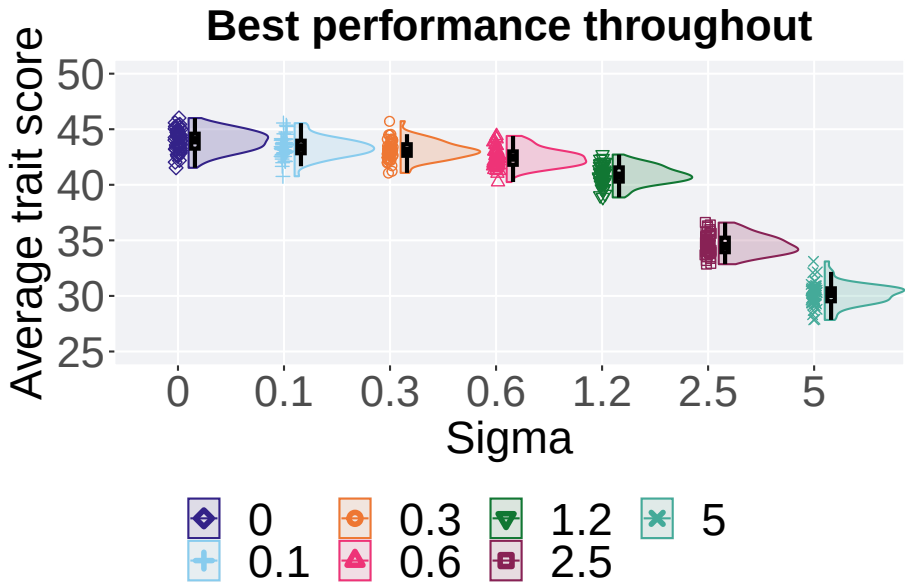
5.2.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'exp' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  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, positio
  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(25,50)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'exp' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels = FS_LIST)
performance %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 0        50      0  41.5   44.0  43.9  46.0  1.34
## 2 0.1      50      0  40.8   43.4  43.4  45.5  1.13
## 3 0.3      50      0  41.1   43.1  43.1  45.7  1.04
## 4 0.6      50      0  40.2   42.4  42.5  44.4  1.20
## 5 1.2      50      0  38.9   40.8  40.9  42.7  1.27
## 6 2.5      50      0  32.9   34.5  34.6  36.6  1.31
## 7 5        50      0  27.8   30.2  30.1  33.1  1.16
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 290.52, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 0.08040 -       -       -       -       -
## 0.3 0.00051 1.00000 -       -       -       -
## 0.6 4.4e-09 8.2e-05 0.00565 -       -       -
## 1.2 5.2e-16 3.8e-15 1.0e-14 5.6e-11 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

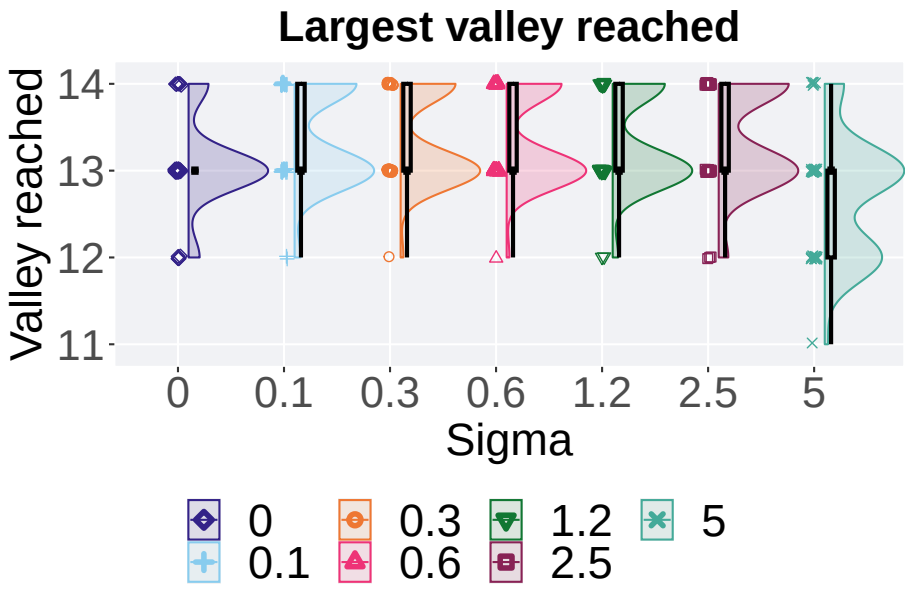
5.2.3 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'exp' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  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, positio
  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(10.9,14.1),
    breaks = c(10,11,12,13,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.2.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'exp' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = FS_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0    12     13  13.1    14     0
## 2 0.1      50      0    12     13  13.4    14     1
## 3 0.3      50      0    12     13  13.4    14     1
## 4 0.6      50      0    12     13  13.4    14     1
## 5 1.2      50      0    12     13  13.3    14     1
## 6 2.5      50      0    12     13  13.4    14     1
## 7 5        50      0    11     13  12.7    14     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 43.54, df = 6, p-value = 9.117e-08
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 0.15268 -       -       -       -       -
## 0.3 0.14162 1.00000 -       -       -       -
```

```
## 0.6 0.14162 1.00000 1.00000 -       -       -
## 1.2 0.39290 1.00000 1.00000 1.00000 -       -
## 2.5 0.16234 1.00000 1.00000 1.00000 1.00000 -
## 5   0.09970 9.4e-05 6.3e-05 6.3e-05 0.00024 0.00014
##
## P value adjustment method: bonferroni
```

5.3 Ordered exploitation results

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

5.3.1 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 = filter(over_time_df, acro == 'ord') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

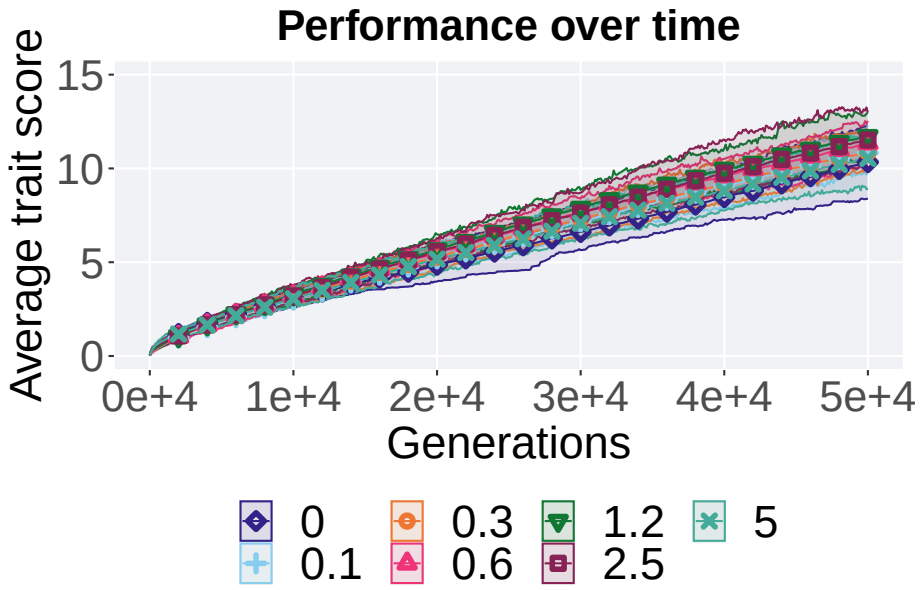
```
## `summarise()` has grouped output by 'Sigma'. You can override using the
## `.groups` argument.
```

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

  ) +
  scale_shape_manual(values=SHAPE)+
```

```
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  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")
  )

over_time_plot
```



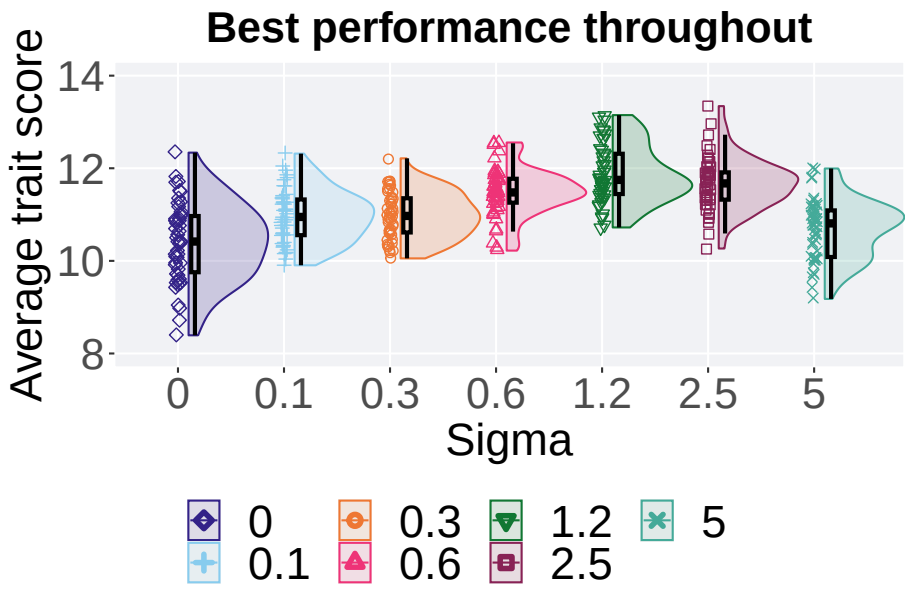
5.3.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'ord' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  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, positio
  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(8,14)
  ) +
```

```
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'ord' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels = FS_LIST)
performance %>%
  group_by(Sigma) %>%
```

```
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  8.39   10.4  10.4  12.3 1.21
## 2 0.1      50      0  9.90   10.9  11.0  12.3 0.771
## 3 0.3      50      0 10.1    11.0  11.0  12.2 0.739
## 4 0.6      50      0 10.2    11.5  11.5  12.6 0.517
## 5 1.2      50      0 10.7    11.7  11.9  13.1 0.870
## 6 2.5      50      0 10.3    11.7  11.7  13.3 0.590
## 7 5        50      0  9.18   10.8  10.7  12.0 1.01
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 140.32, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 0.01089 -       -       -       -       -
## 0.3 0.00397 1.00000 -       -       -       -
## 0.6 9.5e-09 0.00020 0.00017 -       -       -
## 1.2 9.3e-12 1.7e-08 6.6e-09 0.07200 -       -
## 2.5 1.6e-10 2.1e-06 1.0e-06 1.00000 1.00000 -
## 5   1.00000 0.71439 0.38977 5.5e-08 2.5e-11 6.5e-10
```

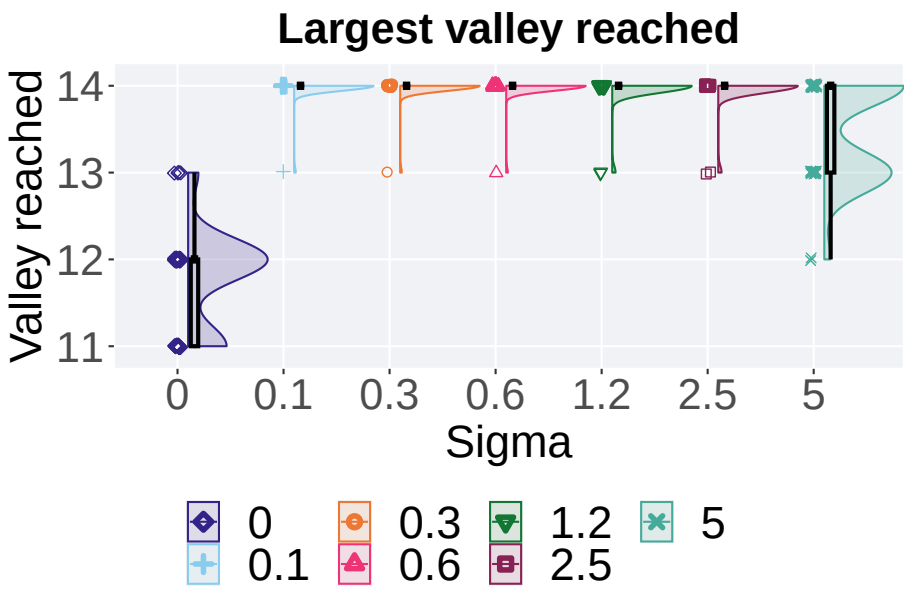
```
##
## P value adjustment method: bonferroni
```

5.3.3 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, acro == 'ord' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(10.9,14.1),
    breaks = c(11,12,13,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.3.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'ord' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = FS_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0    11     12  11.8    13     1
## 2 0.1      50      0    13     14  14.0    14     0
## 3 0.3      50      0    13     14  14.0    14     0
## 4 0.6      50      0    13     14  14.0    14     0
## 5 1.2      50      0    13     14  14.0    14     0
```

```
## 6 2.5      50      0    13     14  14.0    14     0
## 7 5        50      0    12     14  13.5    14     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 265.8, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1       -       -       -       -
## 0.6 < 2e-16 1       1       -       -       -
## 1.2 < 2e-16 1       1       1       -       -
## 2.5 < 2e-16 1       1       1       1       -
## 5   1.3e-15 2.8e-06 2.8e-06 2.8e-06 1.3e-05 1.3e-05
##
## P value adjustment method: bonferroni
```

5.4 Contradictory objectives results

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

5.4.1 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.

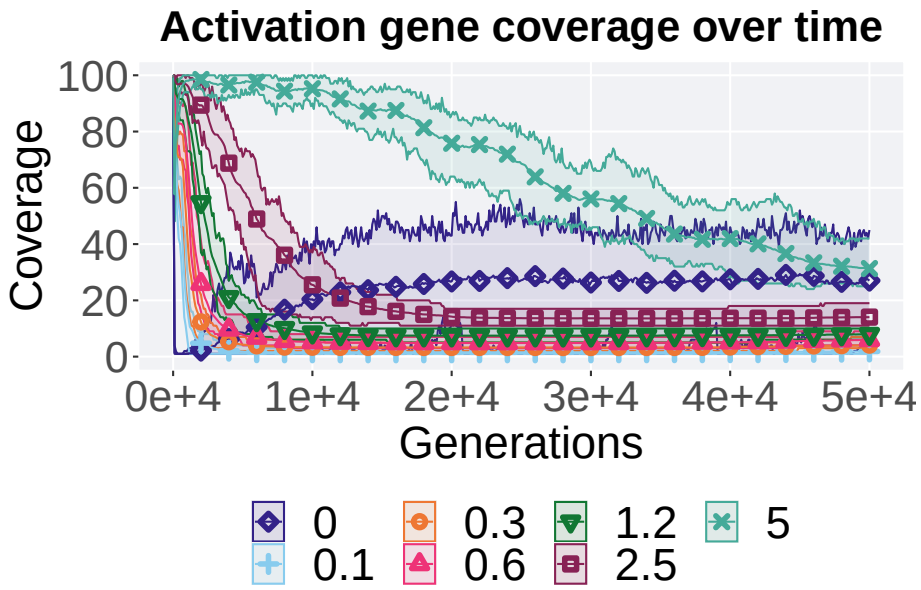
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
```

```
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



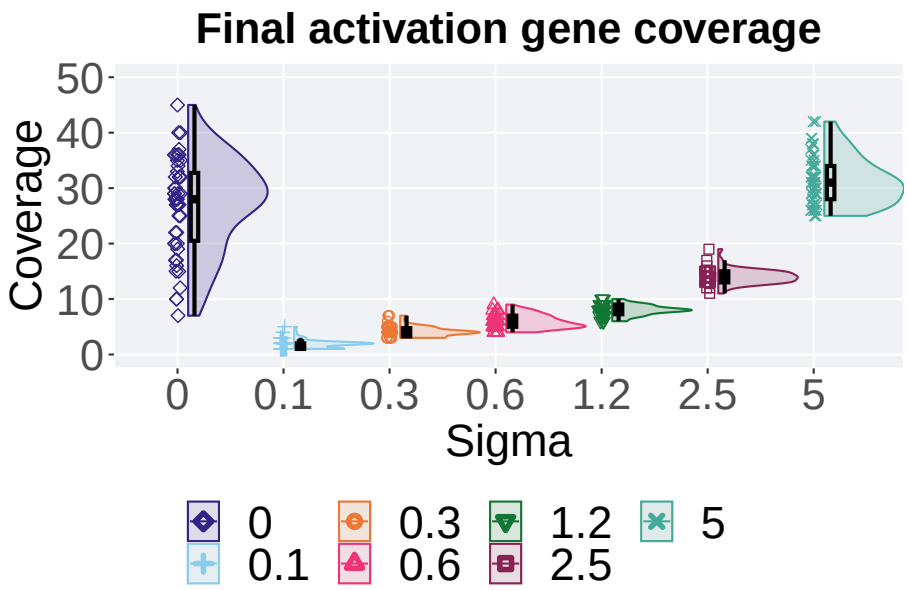
5.4.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 50.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

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

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



5.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = FS_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0        50      0     7     28 27.0     45 12.2
## 2 0.1      50      0     1      2  1.84     5  1
## 3 0.3      50      0     3      4  4.1      7  1.5
## 4 0.6      50      0     4      6  5.92     9  2
## 5 1.2      50      0     6      8  8       10  1.75
## 6 2.5      50      0    11     14 14.0     19  2
## 7 5        50      0    25     31 31.3     42  6
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 325.94, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.met
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 6.8e-15 -       -       -       -
## 0.6 < 2e-16 < 2e-16 4.9e-10 -       -       -
## 1.2 1.2e-15 < 2e-16 < 2e-16 2.6e-10 -       -
## 2.5 2.4e-11 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   0.43    < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

5.4.3 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.

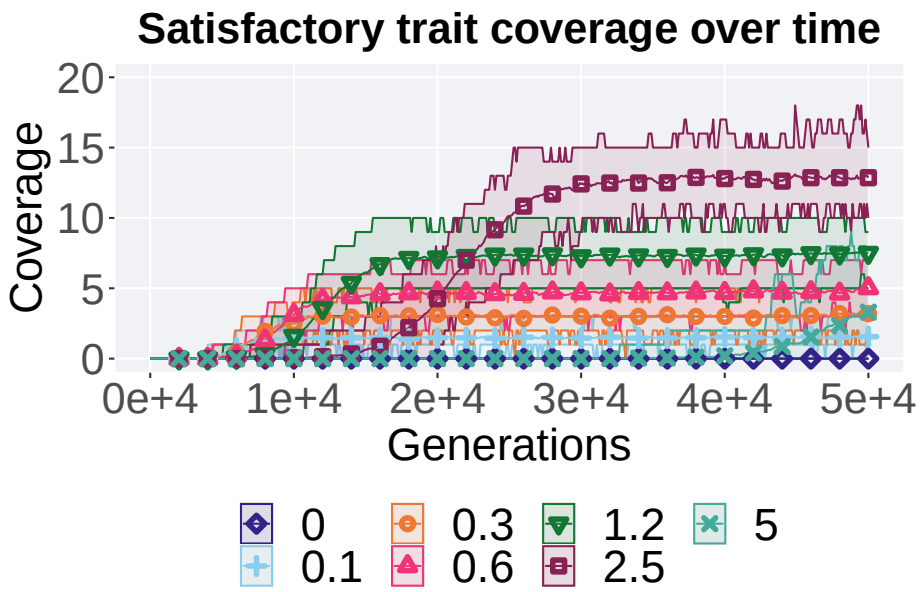
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
```

```
    max = max(pop_uni_obj)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 20)
  ) +
  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()) +
  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")
  )
```



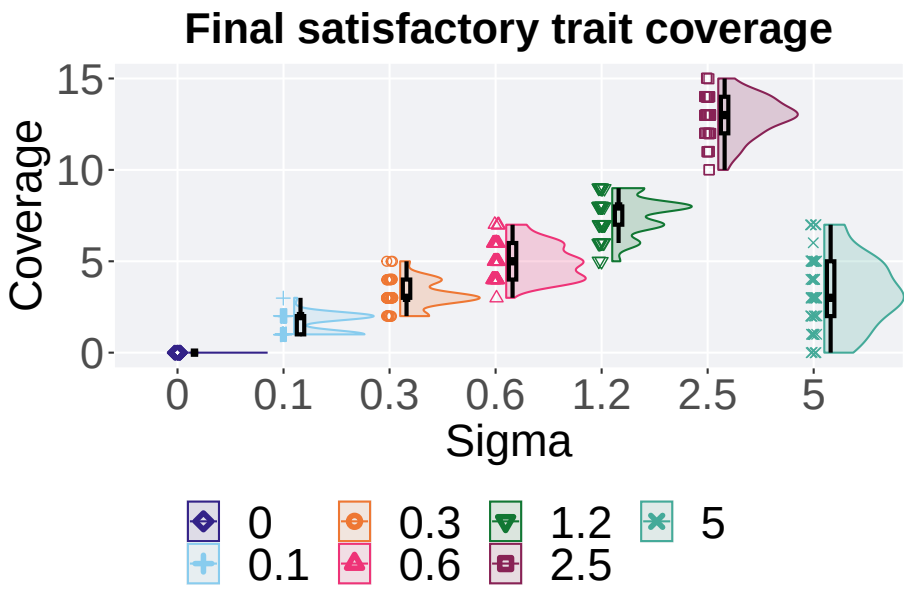
5.4.4 Final satisfactory trait coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  scale_y_continuous(
    name="Coverage",
    limits=c(-0.1, 15.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$Sigma = factor(sat_coverage$Sigma, levels = FS_LIST)
sat_coverage %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0        50      0     0      0  0        0     0
## 2 0.1      50      0     1      2  1.56     3     1
## 3 0.3      50      0     2      3  3.2      5     1
## 4 0.6      50      0     3      5  5.02     7     2
## 5 1.2      50      0     5      8  7.5      9     1
## 6 2.5      50      0    10     13 12.9     15     2
## 7 5        50      0     0      3  3.3      7     3
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(pop_uni_obj ~ Sigma, data = sat_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = 313.86, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = sat_coverage$pop_uni_obj, g = sat_coverage$Sigma, p.adjust.met
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  sat_coverage$pop_uni_obj and sat_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 3.5e-14 -       -       -       -
## 0.6 < 2e-16 < 2e-16 4.5e-12 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 2.1e-13 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 4.7e-06 1       2.3e-05 7.3e-15 < 2e-16
##
## P value adjustment method: bonferroni
```

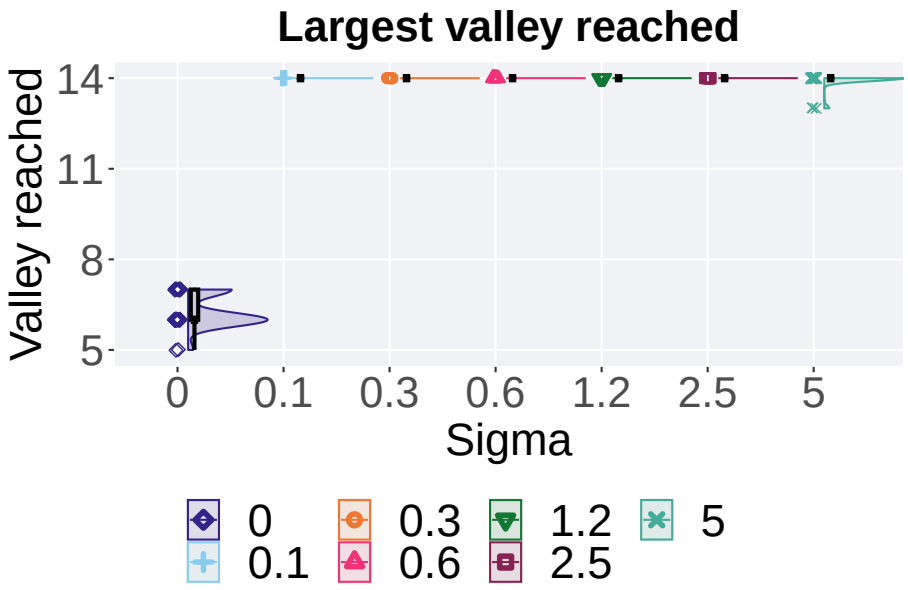
5.4.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'con' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
```

```
    name="Valley reached",
    limits=c(4.9,14.1),
    breaks=c(5,8,11,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.4.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'con' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = FS_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0     5      6   6.3     7     1
## 2 0.1      50      0    14     14  14      14     0
## 3 0.3      50      0    14     14  14      14     0
## 4 0.6      50      0    14     14  14      14     0
## 5 1.2      50      0    14     14  14      14     0
## 6 2.5      50      0    14     14  14      14     0
## 7 5        50      0    13     14  13.9    14     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 331.14, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0      0.1 0.3 0.6 1.2 2.5
## 0.1 <2e-16 -   -   -   -   -
## 0.3 <2e-16 -   -   -   -   -
```

```
## 0.6 <2e-16 -   -   -   -   -
## 1.2 <2e-16 -   -   -   -   -
## 2.5 <2e-16 -   -   -   -   -
## 5   <2e-16 0.9 0.9 0.9 0.9 0.9
##
## P value adjustment method: bonferroni
```

5.5 Multi-path exploration results

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

5.5.1 Activation gene coverage over time

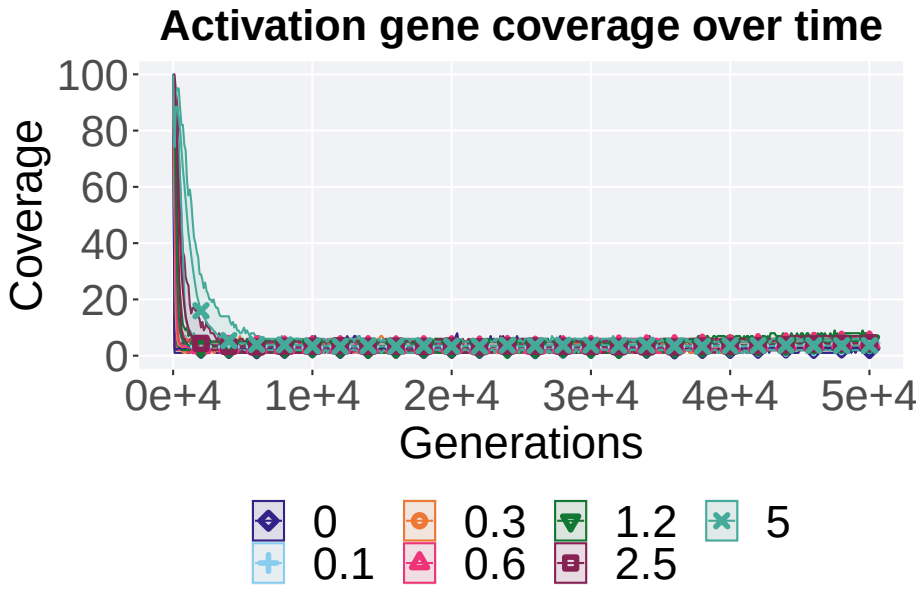
Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



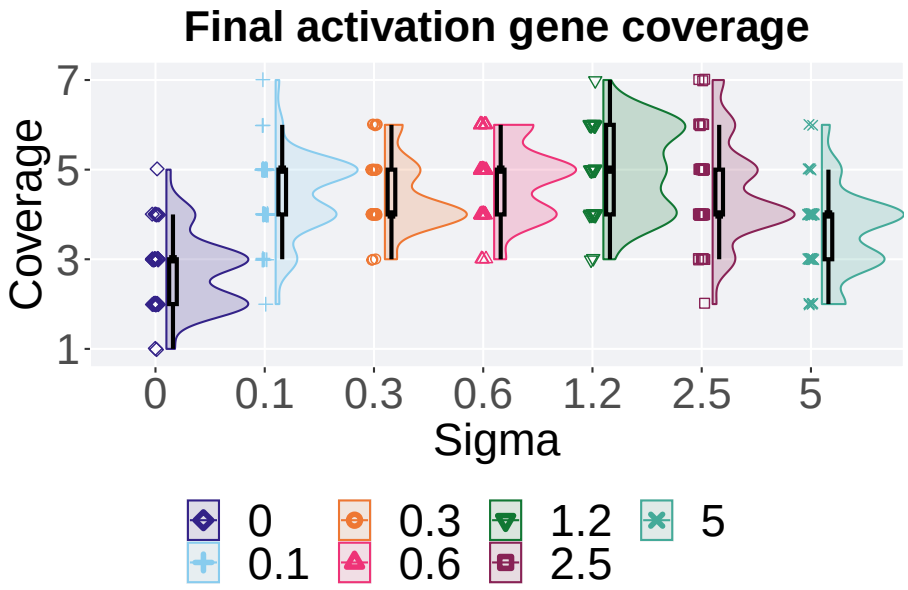
5.5.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  scale_y_continuous(
    name="Coverage",
    limits=c(0.9, 7.1),
    breaks=c(1,3,5,7)
```

```
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = FS_LIST)
act_coverage %>%
```

```
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 0        50      0     1      3  2.7      5     1
## 2 0.1      50      0     2      5  4.44     7     1
## 3 0.3      50      0     3      4  4.4      6     1
## 4 0.6      50      0     3      5  4.76     6     1
## 5 1.2      50      0     3      5  5        7     2
## 6 2.5      50      0     2      4  4.5      7     1
## 7 5        50      0     2      4  3.62     6     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 129.93, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.met
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1.3e-11 -       -       -       -       -
## 0.3 1.8e-12 1.00000 -       -       -       -
## 0.6 9.0e-14 1.00000 0.45482 -       -       -
## 1.2 3.0e-14 0.12465 0.03295 1.00000 -       -
## 2.5 3.1e-11 1.00000 1.00000 1.00000 0.29397 -
```

```
## 5   9.3e-05 0.00053 0.00084 1.2e-06 9.1e-08 0.00181
##
## P value adjustment method: bonferroni
```

5.5.3 Performance over time

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

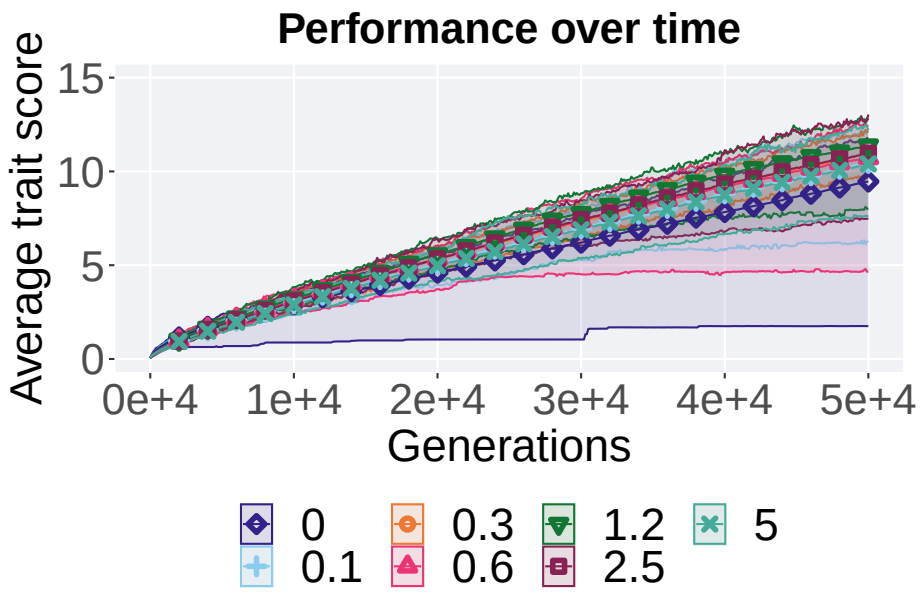
```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, sha
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 15)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  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")
  )
```

```
over_time_plot
```

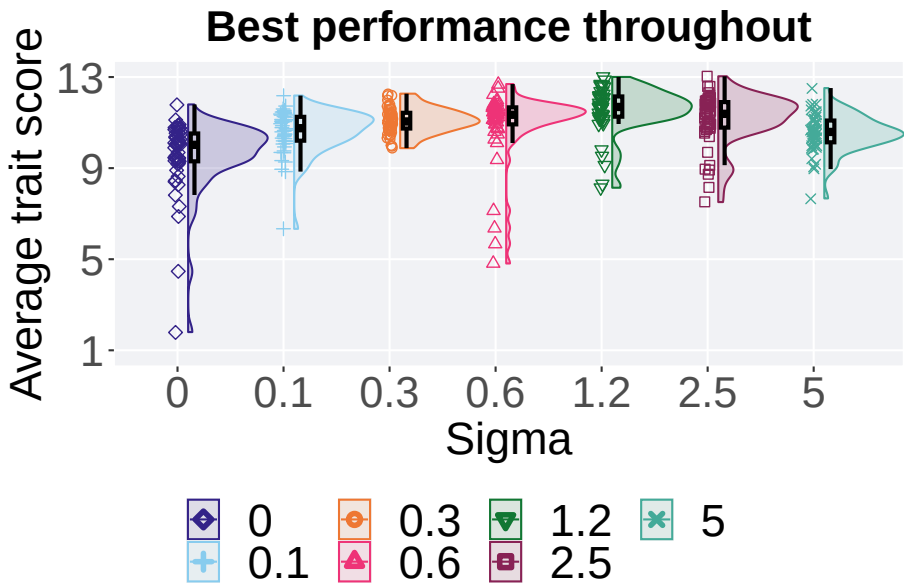


5.5.4 Best performance throughout

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

```
plot = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Average trait score",
    limits=c(0.9, 13.1),
    breaks=c(1,5,9,13)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels = FS_LIST)
performance %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  1.80   10.0  9.53  11.8 1.22
## 2 0.1      50      0  6.33   10.8 10.6   12.2 1.05
## 3 0.3      50      0  9.88   11.0 11.1   12.3 0.690
## 4 0.6      50      0  4.81   11.3 10.9   12.7 0.747
## 5 1.2      50      0  8.14   11.7 11.5   13.0 0.889
## 6 2.5      50      0  7.51   11.4 11.1   13.0 1.12
## 7 5        50      0  7.67   10.6 10.5   12.5 0.956
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 108.01, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bo
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 0.00012 -       -       -       -       -
## 0.3 1.7e-10 0.19052 -       -       -       -
## 0.6 9.9e-09 0.00777 1.00000 -       -       -
## 1.2 1.9e-11 8.4e-07 0.00101 0.04375 -       -
## 2.5 3.6e-08 0.01477 1.00000 1.00000 0.45978 -
## 5   0.00262 1.00000 0.00346 0.00087 2.3e-07 0.00140
##
## P value adjustment method: bonferroni
```

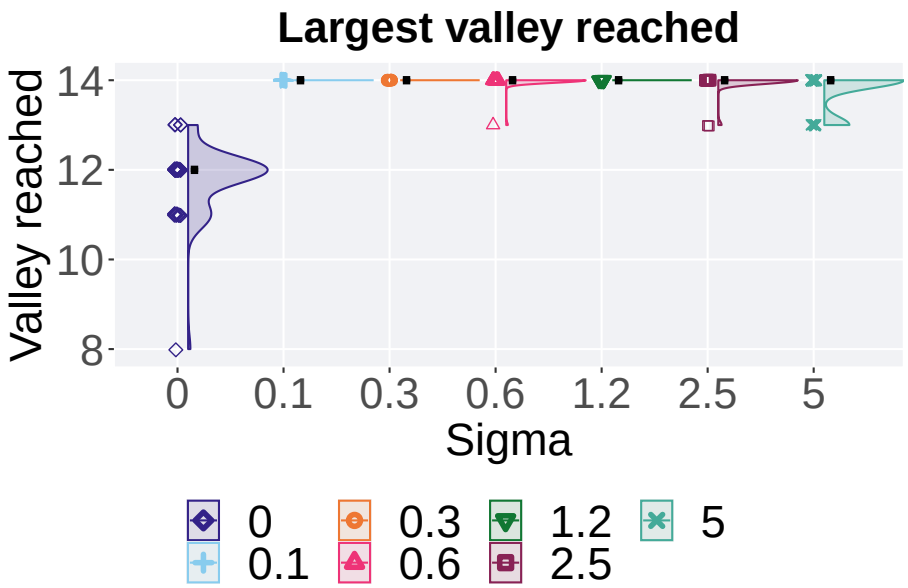
5.5.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'mpe' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
```

```
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  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(7.9,14.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



5.5.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'mpe' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = FS_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0     8     12  11.8    13     0
## 2 0.1      50      0    14     14  14      14     0
## 3 0.3      50      0    14     14  14      14     0
## 4 0.6      50      0    13     14  14.0    14     0
## 5 1.2      50      0    14     14  14      14     0
## 6 2.5      50      0    13     14  14.0    14     0
## 7 5        50      0    13     14  13.8    14     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 288.11, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
```

```
##     0      0.1    0.3    0.6    1.2    2.5
## 0.1 <2e-16 -      -      -      -      -
## 0.3 <2e-16 -      -      -      -      -
## 0.6 <2e-16 1.0000 1.0000 -      -      -
## 1.2 <2e-16 -      -      1.0000 -      -
## 2.5 <2e-16 1.0000 1.0000 1.0000 1.0000 -
## 5   <2e-16 0.0044 0.0044 0.0209 0.0044 0.0758
##
## P value adjustment method: bonferroni
```

Chapter 6

Nondominated sorting

Results for the nondominated sorting parameter sweep on the diagnostics with valleys.

6.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME-MVC/nds.csv', sep = "", collapse = NULL), head
over_time_df$Sigma <- factor(over_time_df$Sigma, levels = ND_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST-MVC/nds.csv', sep = "", collapse = NULL), header = TRUE,
best_df$Sigma <- factor(best_df$Sigma, levels = ND_LIST)
```

6.2 Exploitation rate results

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

6.2.1 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 = filter(over_time_df, acro == 'exp') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

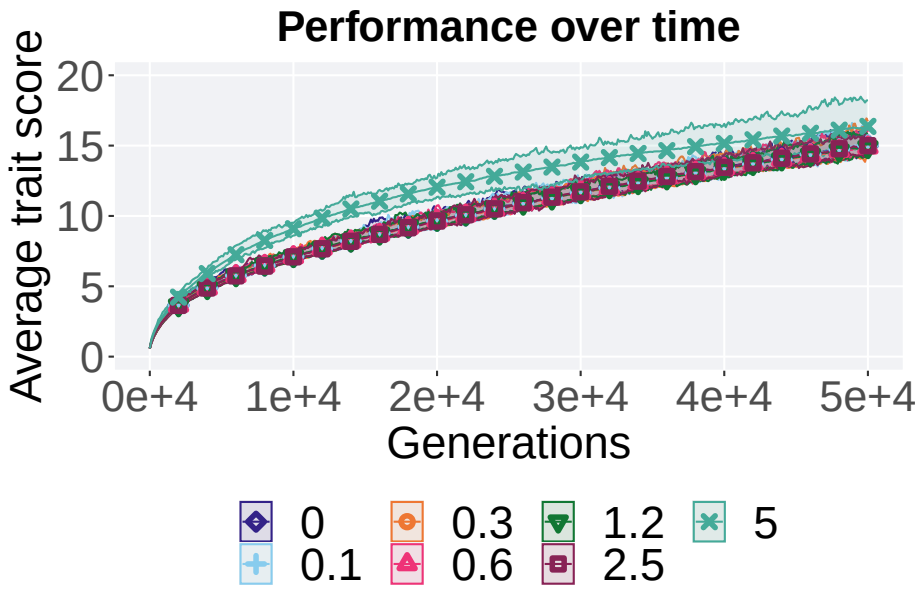
```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, 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(0,20)
  ) +
  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()) +
  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")
  )

over_time_plot
```



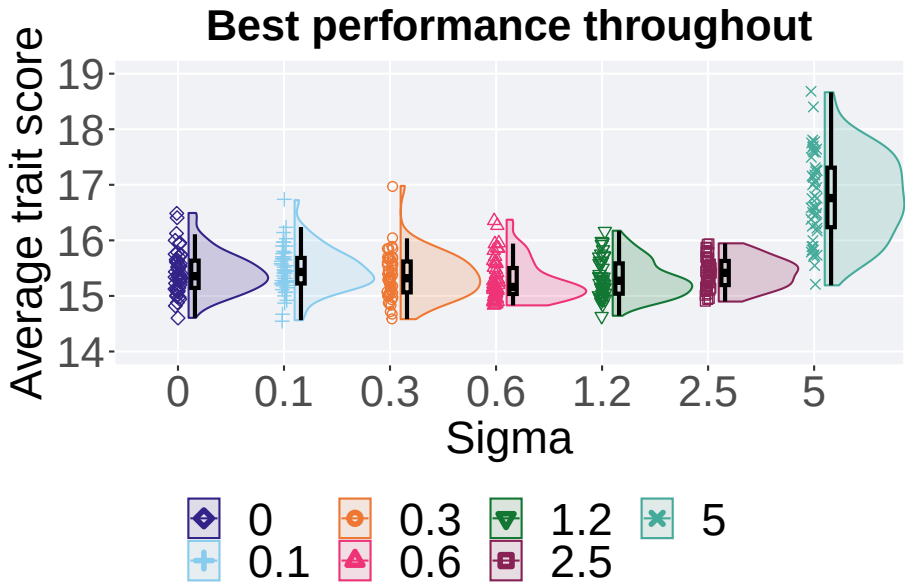
6.2.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'exp' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  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, positio
  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(14,19)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'exp' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels = ND_LIST)
performance %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 0        50      0  14.6   15.4  15.4  16.5 0.491
## 2 0.1      50      0  14.6   15.4  15.5  16.7 0.460
## 3 0.3      50      0  14.6   15.3  15.4  17.0 0.556
## 4 0.6      50      0  14.8   15.2  15.3  16.4 0.467
## 5 1.2      50      0  14.6   15.3  15.3  16.2 0.546
## 6 2.5      50      0  14.9   15.4  15.4  15.9 0.434
## 7 5        50      0  15.2   16.8  16.8  18.7 1.07
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 116.31, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1.00    -       -       -       -       -
## 0.3 1.00    1.00    -       -       -       -
## 0.6 0.55    0.12    1.00    -       -       -
## 1.2 1.00    0.83    1.00    1.00    -       -
## 2.5 1.00    1.00    1.00    0.15    1.00    -
## 5   2.5e-13 7.3e-13 7.5e-14 3.1e-14 2.3e-14 2.9e-14
##
## P value adjustment method: bonferroni
```

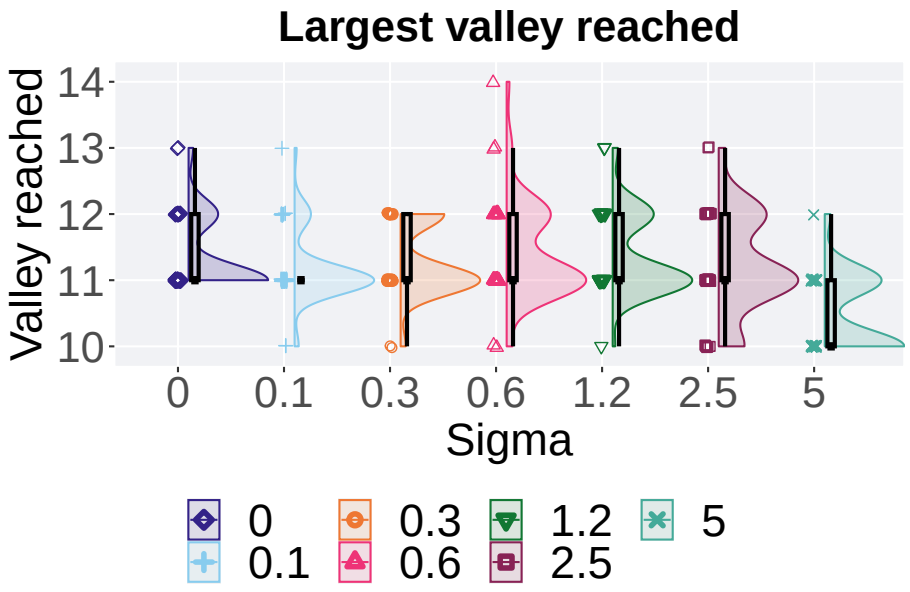
6.2.3 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'exp' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  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, positio
  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(9.9,14.1),
    breaks = c(10,11,12,13,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.2.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'exp' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = ND_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0    11     11  11.3    13     1
## 2 0.1      50      0    10     11  11.2    13     0
## 3 0.3      50      0    10     11  11.3    12     1
## 4 0.6      50      0    10     11  11.4    14     1
## 5 1.2      50      0    10     11  11.4    13     1
## 6 2.5      50      0    10     11  11.2    13     1
## 7 5        50      0    10     10  10.5    12     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 67.685, df = 6, p-value = 1.219e-12
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1.00    -       -       -       -       -
## 0.3 1.00    1.00    -       -       -       -
```

```
## 0.6 1.00    0.84    1.00    -       -       -
## 1.2 1.00    0.89    1.00    1.00    -       -
## 2.5 1.00    1.00    1.00    1.00    1.00    -
## 5   8.1e-09 7.0e-07 5.5e-08 2.5e-08 1.0e-08 3.1e-05
##
## P value adjustment method: bonferroni
```

6.3 Ordered exploitation results

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

6.3.1 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 = filter(over_time_df, acro == 'ord') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

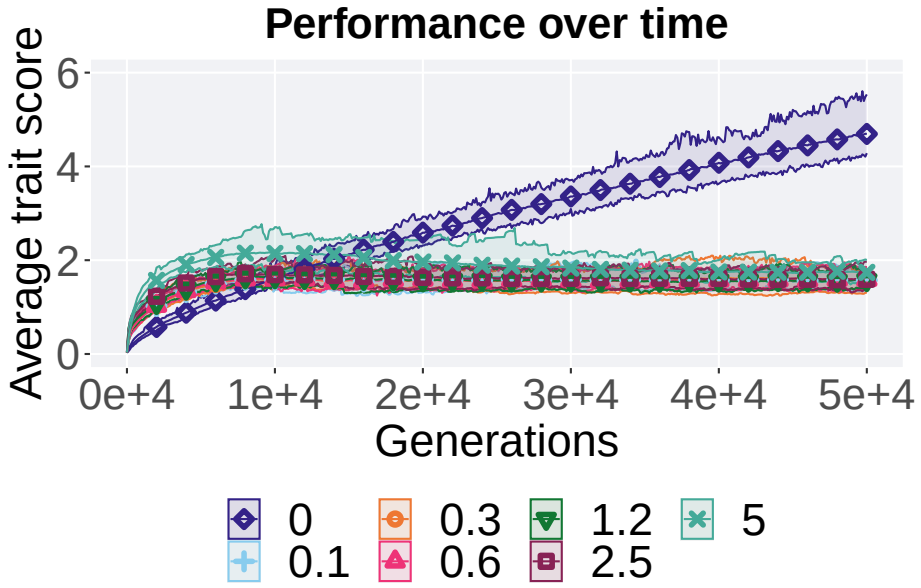
```
## `summarise()` has grouped output by 'Sigma'. You can override using the
## `.groups` argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, 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(0,6)
  ) +
  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()) +
  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")
  )

over_time_plot
```



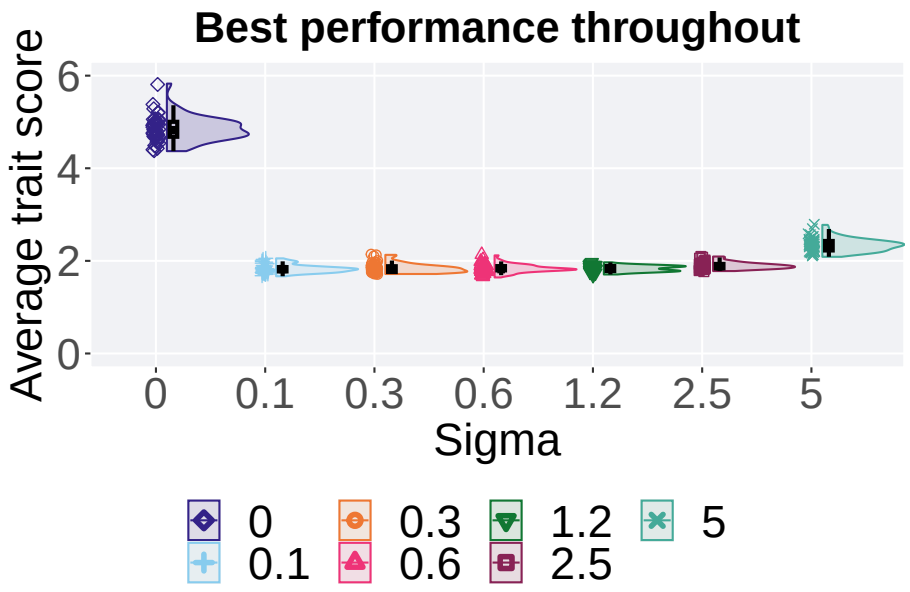
6.3.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'ord' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  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, positio
  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(0,6)
  ) +
```

```
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'ord' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels = ND_LIST)
performance %>%
  group_by(Sigma) %>%
```

```
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max    IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 0        50      0  4.37   4.82  4.85  5.83 0.323
## 2 0.1      50      0  1.67   1.82  1.83  2.05 0.0929
## 3 0.3      50      0  1.72   1.83  1.84  2.13 0.125
## 4 0.6      50      0  1.64   1.82  1.84  2.12 0.0923
## 5 1.2      50      0  1.71   1.84  1.83  1.97 0.106
## 6 2.5      50      0  1.78   1.88  1.90  2.09 0.0952
## 7 5        50      0  2.09   2.34  2.34  2.78 0.205
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 233.29, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1.00000 -       -       -       -
## 0.6 < 2e-16 1.00000 1.00000 -       -       -
## 1.2 < 2e-16 1.00000 1.00000 1.00000 -       -
## 2.5 < 2e-16 0.00025 0.00387 0.00169 0.00111 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
```

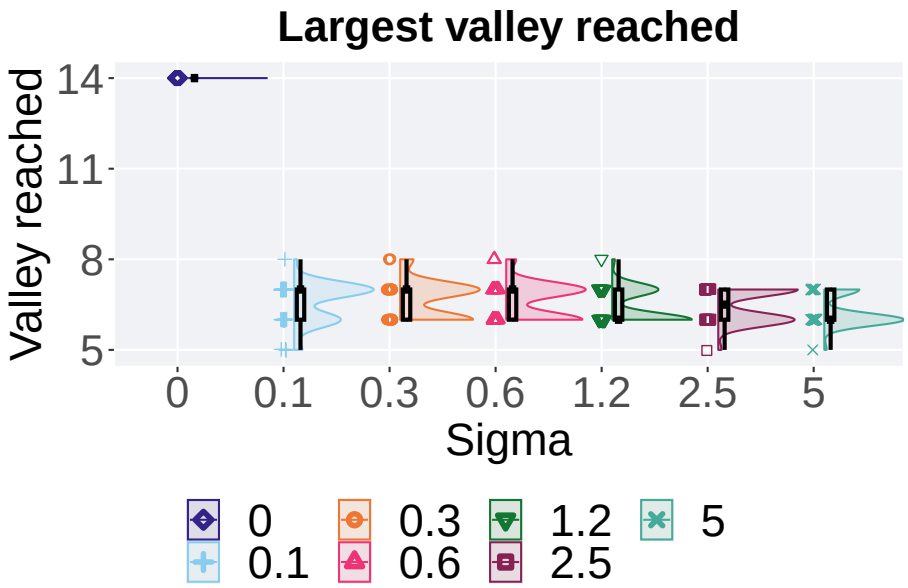
```
##
## P value adjustment method: bonferroni
```

6.3.3 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, acro == 'ord' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(4.9,14.1),
    breaks = c(5,8,11,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.3.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'ord' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = ND_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0    14   14    14       14     0
## 2 0.1      50      0     5    7     6.56     8     1
## 3 0.3      50      0     6    7     6.64     8     1
## 4 0.6      50      0     6    7     6.6      8     1
## 5 1.2      50      0     6    6     6.4      8     1
```

```
## 6 2.5      50      0     5    6.5  6.48     7     1
## 7 5        50      0     5    6    6.28     7     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 158.07, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0      0.1   0.3   0.6   1.2   2.5
## 0.1 <2e-16 -     -     -     -     -
## 0.3 <2e-16 1.000 -     -     -     -
## 0.6 <2e-16 1.000 1.000 -     -     -
## 1.2 <2e-16 1.000 1.000 1.000 -     -
## 2.5 <2e-16 1.000 1.000 1.000 1.000 -
## 5   <2e-16 0.171 0.083 0.172 1.000 1.000
##
## P value adjustment method: bonferroni
```

6.4 Contradictory objectives results

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

6.4.1 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.

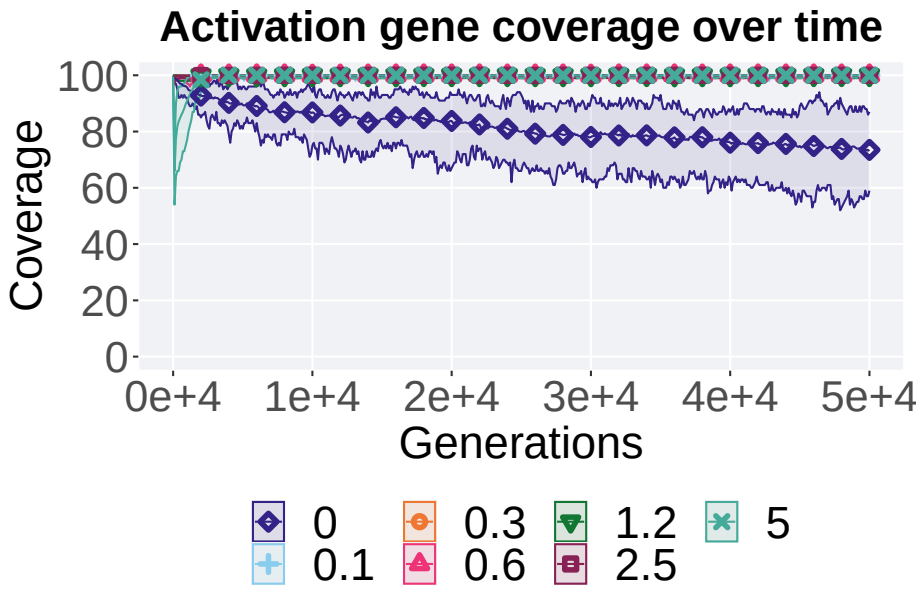
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
```

```
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



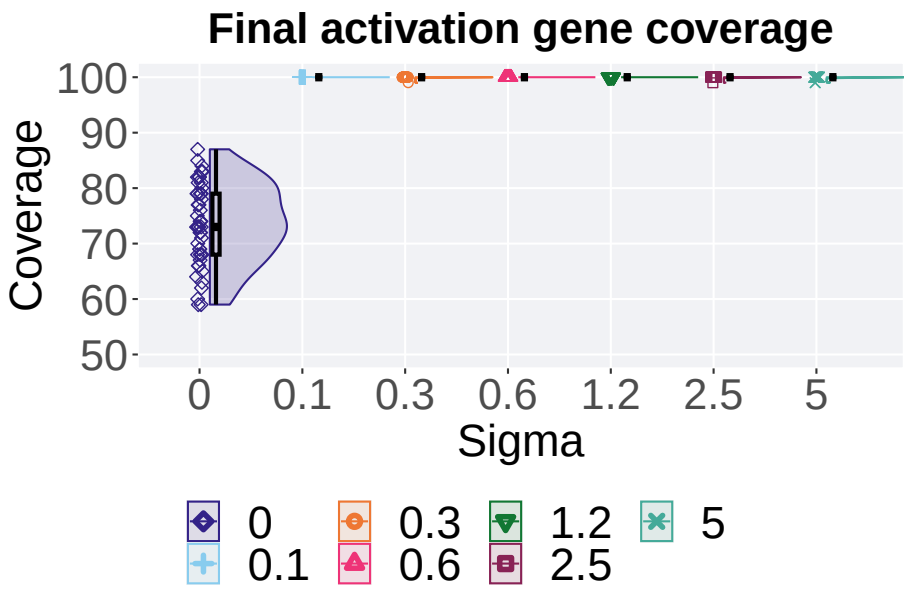
6.4.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(50, 100.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = ND_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0        50      0    59     73  73.5    87    11
## 2 0.1      50      0   100    100 100     100     0
## 3 0.3      50      0    99    100 100.    100     0
## 4 0.6      50      0   100    100 100     100     0
## 5 1.2      50      0   100    100 100     100     0
## 6 2.5      50      0    99    100 100.    100     0
## 7 5        50      0    99    100 100.    100     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 324.85, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.met
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0      0.1 0.3 0.6 1.2 2.5
## 0.1 <2e-16 -   -   -   -   -
## 0.3 <2e-16 1   -   -   -   -
## 0.6 <2e-16 -   1   -   -   -
## 1.2 <2e-16 -   1   -   -   -
## 2.5 <2e-16 1   1   1   1   -
## 5   <2e-16 1   1   1   1   1
##
## P value adjustment method: bonferroni
```

6.4.3 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.

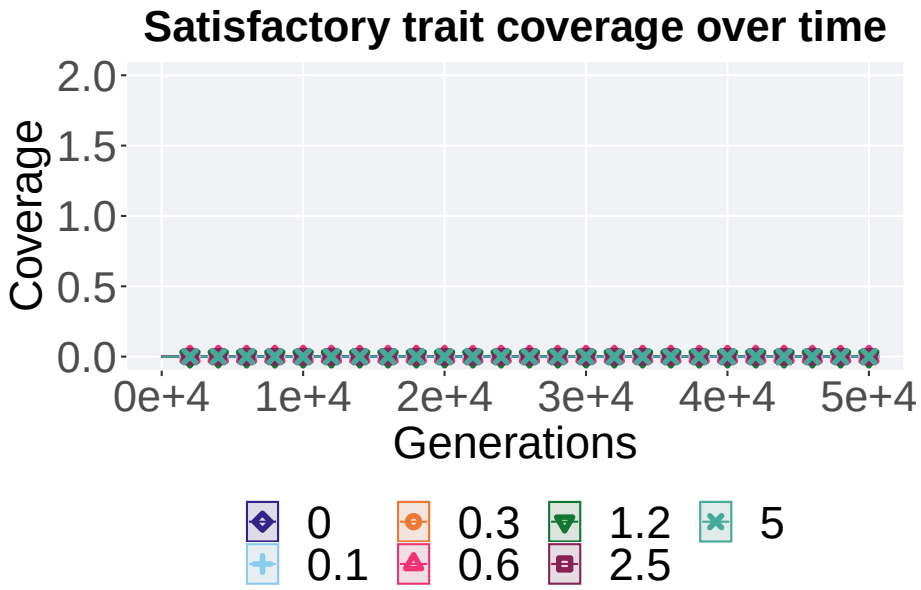
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
```

```
    max = max(pop_uni_obj)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2)
  ) +
  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()) +
  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")
  )
```



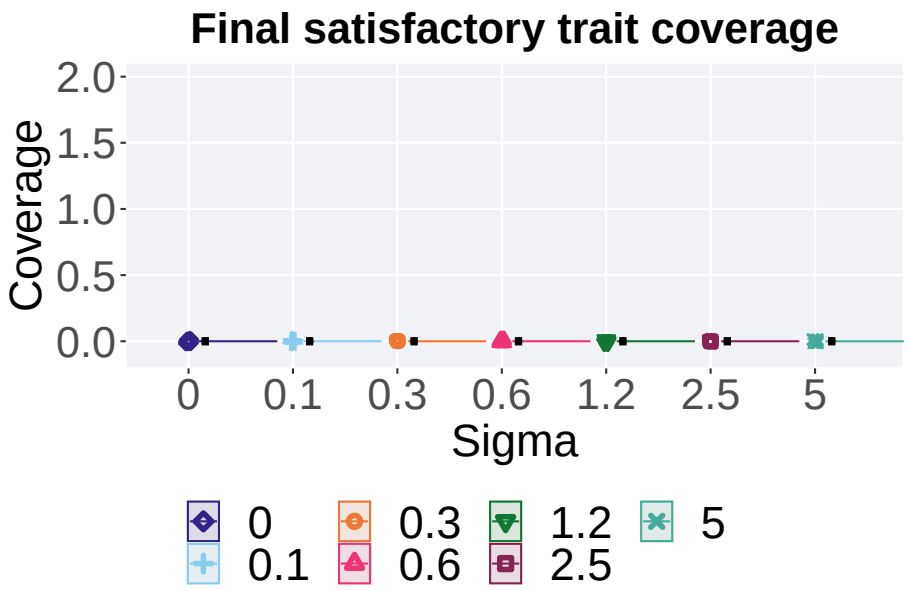
6.4.4 Final satisfactory trait coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  scale_y_continuous(
    name="Coverage",
    limits=c(-0.1, 2)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$Sigma = factor(sat_coverage$Sigma, levels = ND_LIST)
sat_coverage %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

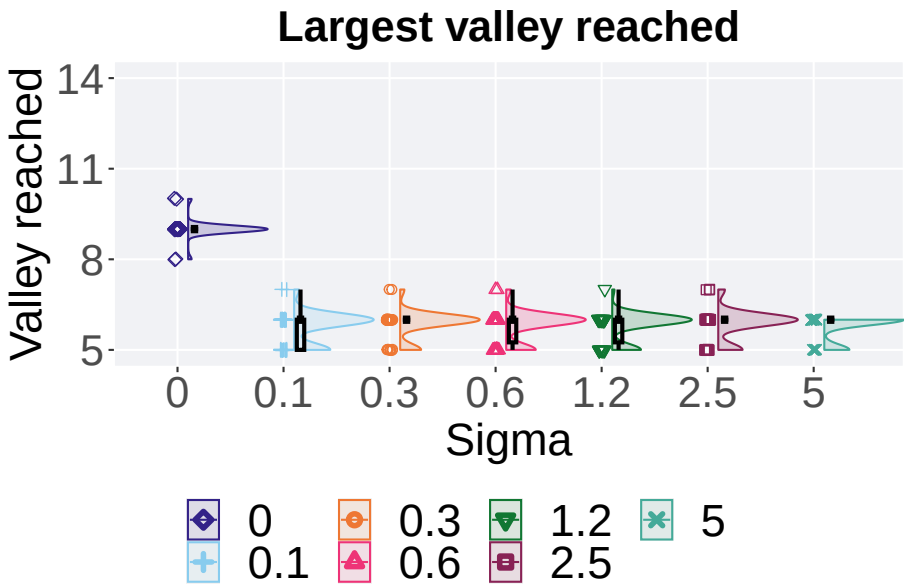
```
## 1 0       50      0     0     0     0     0     0
## 2 0.1     50      0     0     0     0     0     0
## 3 0.3     50      0     0     0     0     0     0
## 4 0.6     50      0     0     0     0     0     0
## 5 1.2     50      0     0     0     0     0     0
## 6 2.5     50      0     0     0     0     0     0
## 7 5       50      0     0     0     0     0     0
```

6.4.5 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, acro == 'con' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(4.9,14.1),
    breaks=c(5,8,11,14)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.4.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'con' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = ND_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0     8      9  9       10  0
## 2 0.1      50      0     5      6  5.74     7  1
## 3 0.3      50      0     5      6  5.84     7  0
## 4 0.6      50      0     5      6  5.78     7  0.75
## 5 1.2      50      0     5      6  5.76     7  0.75
```

```
## 6 2.5      50      0     5      6  5.84     7  0
## 7 5        50      0     5      6  5.76     6  0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 171.34, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0      0.1 0.3 0.6 1.2 2.5
## 0.1 <2e-16 -   -   -   -   -
## 0.3 <2e-16 1   -   -   -   -
## 0.6 <2e-16 1   1   -   -   -
## 1.2 <2e-16 1   1   1   -   -
## 2.5 <2e-16 1   1   1   1   -
## 5   <2e-16 1   1   1   1   1
##
## P value adjustment method: bonferroni
```

6.5 Multi-path exploration results

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

6.5.1 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.

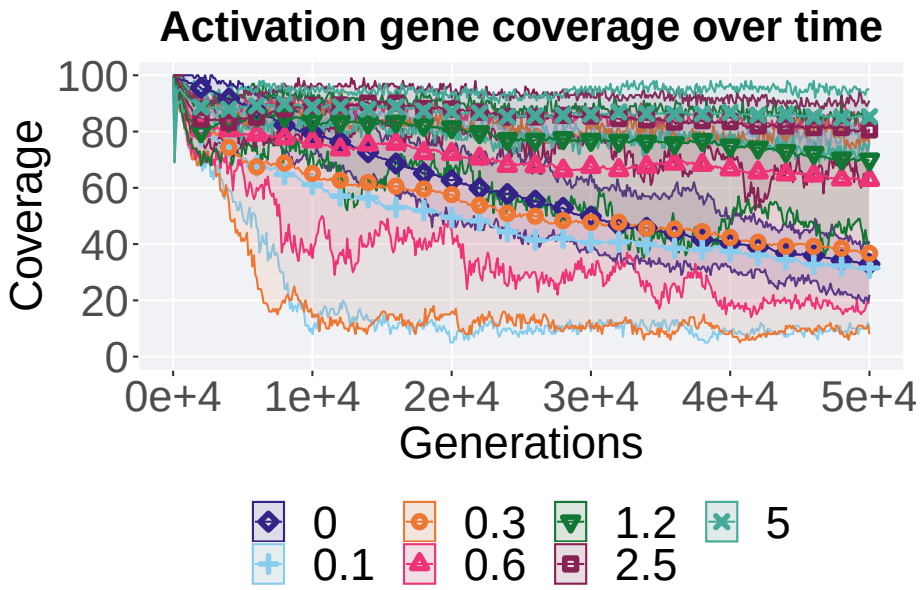
```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(Sigma, gen) %>%
```

```
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



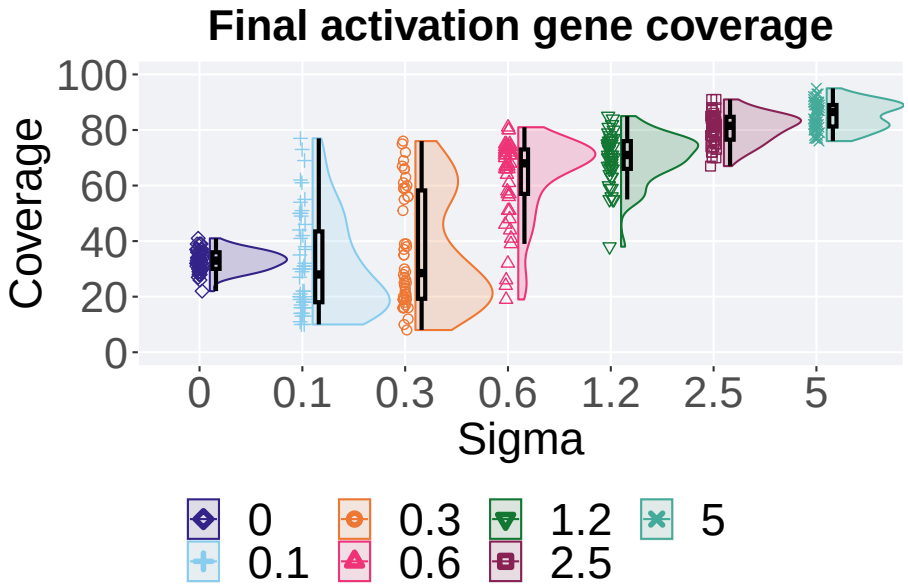
6.5.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  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="Sigma"
  )+
  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)
)
```



6.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = ND_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  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: 7 x 8
```

```
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 0        50      0    22   33    32.9    41  6
## 2 0.1      50      0    10   28    31.5    77 25.5
## 3 0.3      50      0     8   28.5  36.7    76 39
## 4 0.6      50      0    19   68    62.9    81 16
## 5 1.2      50      0    38   71    70.0    85 10
## 6 2.5      50      0    67   81.5  80.6    91  8.25
## 7 5        50      0    76   86.5  85.3    95  7.75
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 268, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.me
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1.0000  -       -       -       -       -
## 0.3 1.0000  1.0000  -       -       -       -
## 0.6 3.5e-12 3.0e-10 1.0e-07 -       -       -
## 1.2 < 2e-16 1.1e-13 2.1e-11 0.3489  -       -
## 2.5 < 2e-16 4.6e-16 6.8e-16 3.7e-12 1.5e-08 -
## 5   < 2e-16 < 2e-16 < 2e-16 7.7e-16 5.1e-14 0.0033
##
## P value adjustment method: bonferroni
```

6.5.3 Performance over time

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

```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
```

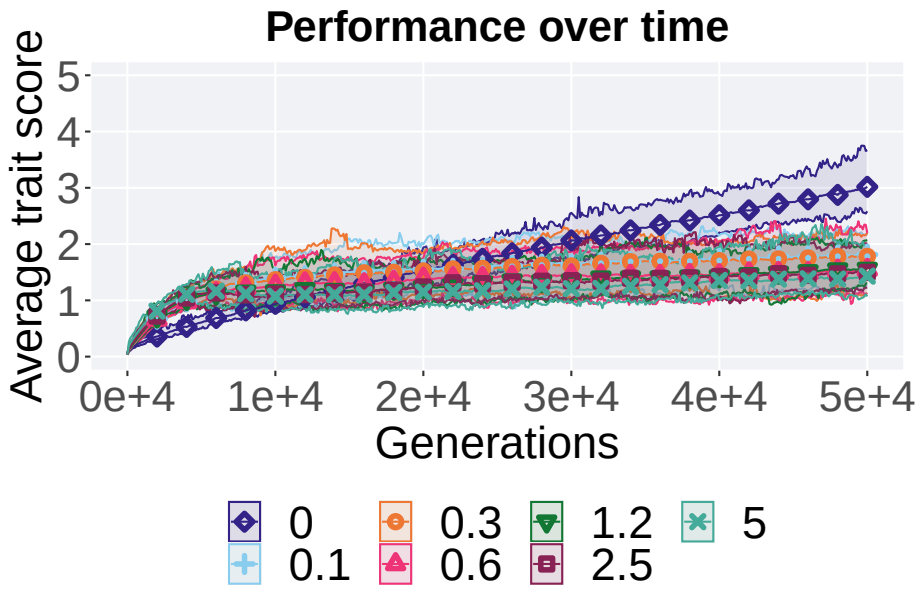
```
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, sha
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 5)
  ) +
  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()) +
  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")
  )

over_time_plot
```



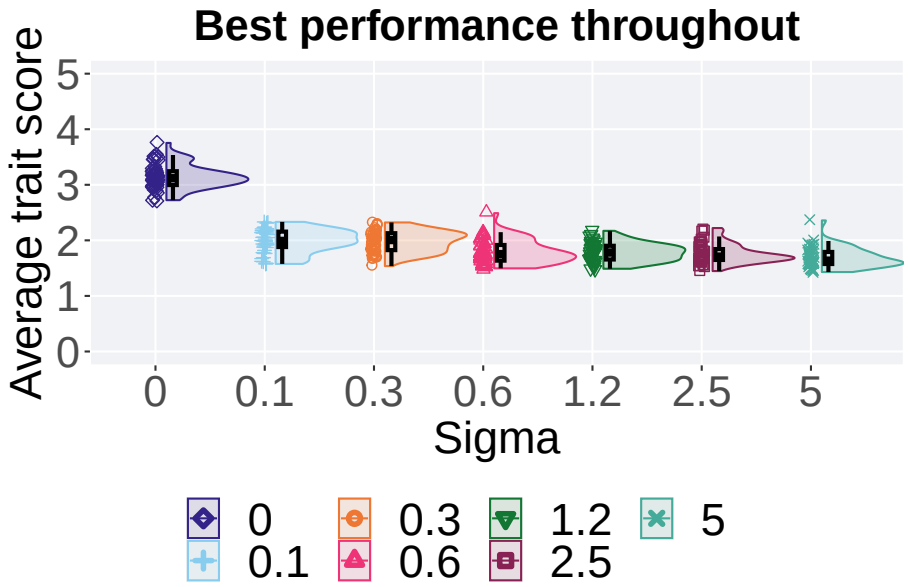
6.5.4 Best performance throughout

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

```
plot = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Average trait score",
    limits=c(0,5)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max')
performance$Sigma = factor(performance$Sigma, levels = ND_LIST)
performance %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 0        50      0  2.72   3.12  3.14  3.75 0.249
## 2 0.1      50      0  1.58   1.99  1.99  2.33 0.277
## 3 0.3      50      0  1.54   2.01  1.98  2.32 0.305
## 4 0.6      50      0  1.50   1.75  1.79  2.49 0.282
## 5 1.2      50      0  1.49   1.78  1.78  2.17 0.247
## 6 2.5      50      0  1.45   1.71  1.76  2.22 0.196
## 7 5        50      0  1.43   1.63  1.68  2.36 0.217
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 186.67, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$Sigma, p.adjust.method = "bo
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1.00000 -       -       -       -
## 0.6 < 2e-16 0.00026 0.00029 -       -       -
## 1.2 < 2e-16 3.2e-05 6.3e-05 1.00000 -       -
## 2.5 < 2e-16 9.6e-06 2.2e-06 1.00000 1.00000 -
## 5   < 2e-16 8.3e-09 5.8e-09 0.18672 0.05257 0.50321
##
## P value adjustment method: bonferroni
```

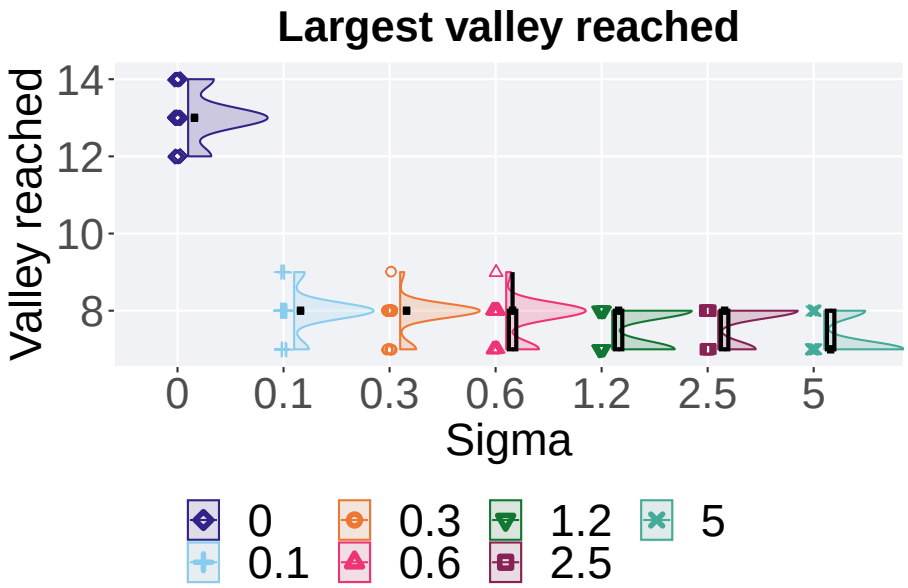
6.5.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'mpe' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
```

```
    name="Valley reached",
    limits=c(6.9,14.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  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)
)
```



6.5.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'mpe' & var == 'ele_big_peak')
valleys$Sigma = factor(valleys$Sigma, levels = ND_LIST)
valleys %>%
  group_by(Sigma) %>%
  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: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0    12     13 13.0     14     0
## 2 0.1      50      0     7      8  7.96     9     0
## 3 0.3      50      0     7      8  7.88     9     0
## 4 0.6      50      0     7      8  7.76     9     1
## 5 1.2      50      0     7      8  7.56     8     1
## 6 2.5      50      0     7      8  7.68     8     1
## 7 5        50      0     7      7  7.34     8     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 187.01, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$Sigma, p.adjust.method = "bonferroni"
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1.0000  -       -       -       -
```

```
## 0.6 < 2e-16 1.0000  1.0000  -      -      -
## 1.2 < 2e-16 0.0046  0.0282  1.0000 -      -
## 2.5 < 2e-16 0.1364  0.7287  1.0000 1.0000 -
## 5   < 2e-16 1.1e-06 6.4e-06 0.0022 0.5899 0.0152
##
## P value adjustment method: bonferroni
```

Chapter 7

Novelty search

Results for the novelty search parameter sweep on the diagnostics with valleys.

7.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME-MVC/nov.csv', sep = "", collapse = NULL), head
over_time_df$K <- factor(over_time_df$K, levels = NS_LIST)
over_time_df$uni_str_pos = over_time_df$uni_str_pos + over_time_df$arc_acti_gene - over_time_df$c

best_df <- read.csv(paste(DATA_DIR,'BEST-MVC/nov.csv', sep = "", collapse = NULL), header = TRUE,
best_df$K <- factor(best_df$K, levels = NS_LIST)
```

7.2 Exploitation rate results

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

7.2.1 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 = filter(over_time_df, acro == 'exp') %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

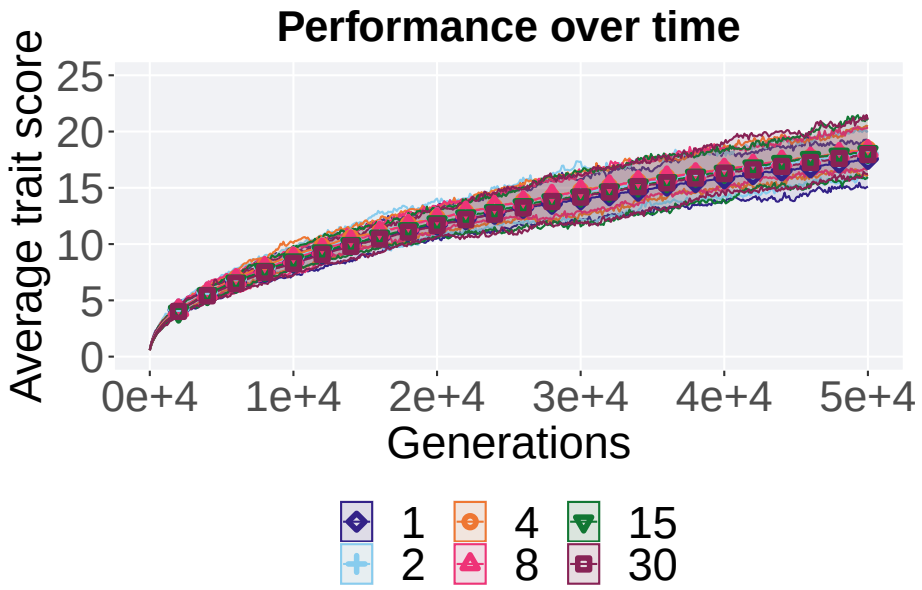
```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape
  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(0,25)
  ) +
  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()) +
  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")
  )

over_time_plot
```



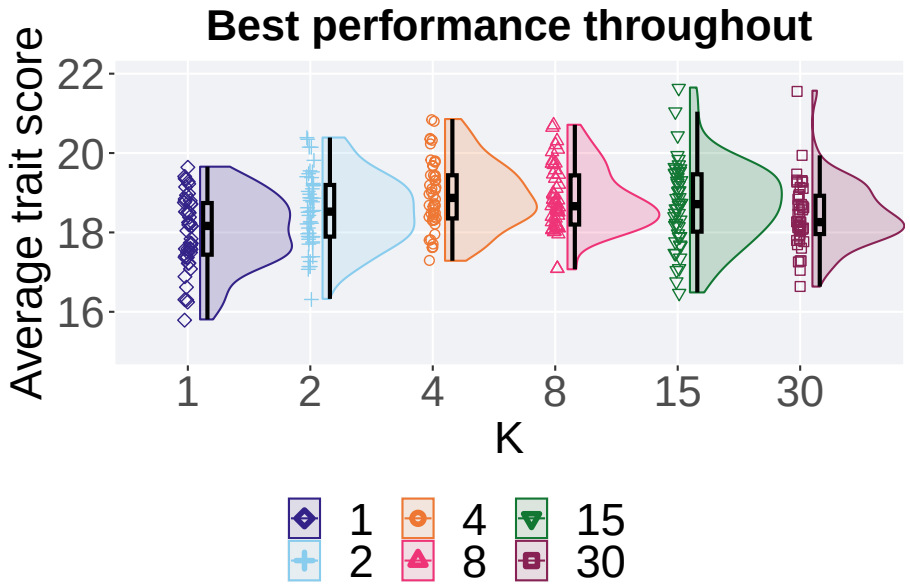
7.2.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'exp' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = K, y = val / DIMENSIONALITY, color = K, fill = K, shape = K)) +
  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, positio
  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(15,22)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
)
```



7.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'exp' &  var == 'pop_fit_max')
performance$K = factor(performance$K, levels = NS_LIST)
performance %>%
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 1        50      0  15.8   18.2  18.0  19.7 1.30
## 2 2        50      0  16.3   18.5  18.5  20.4 1.30
## 3 4        50      0  17.3   18.9  18.9  20.9 1.09
## 4 8        50      0  17.1   18.7  18.8  20.7 1.24
## 5 15       50      0  16.5   18.7  18.7  21.7 1.45
## 6 30       50      0  16.6   18.3  18.4  21.6 0.965
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 27.662, df = 5, p-value = 4.237e-05
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$K, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$K
##
##    1       2       4       8       15
## 2  0.19048 -       -       -       -
## 4  0.00014 0.65631 -       -       -
## 8  0.00204 1.00000 1.00000 -       -
## 15 0.02198 1.00000 1.00000 1.00000 -
## 30 0.91878 1.00000 0.03427 0.17619 1.00000
##
## P value adjustment method: bonferroni
```

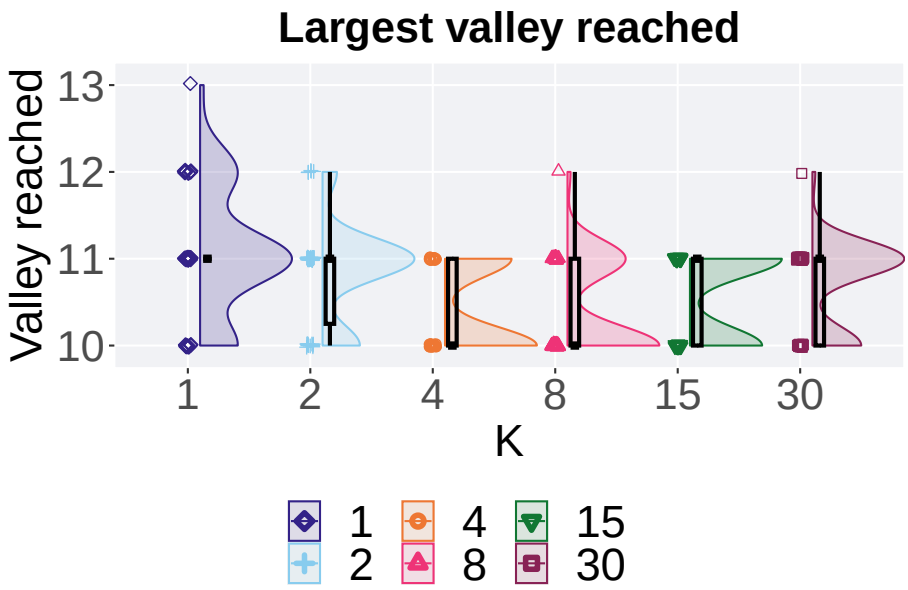
7.2.3 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'exp' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = K, y = val, color = K, fill = K, shape = K)) +
  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, positio
  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(9.9,13.1),
```

```
    breaks = c(10,11,12,13)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
)
```



7.2.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'exp' & var == 'ele_big_peak')
valleys$K = factor(valleys$K, levels = NS_LIST)
```

```
valleys %>%
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0    10     11  11.0    13  0
## 2 2        50      0    10     11  10.8    12  0.75
## 3 4        50      0    10     10  10.4    11  1
## 4 8        50      0    10     10  10.4    12  1
## 5 15       50      0    10     11  10.6    11  1
## 6 30       50      0    10     11  10.7    12  1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 36.731, df = 5, p-value = 6.781e-07
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$K, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$K
##
##    1       2       4       8       15
## 2  1.00000 -       -       -       -
## 4  0.00019 0.00612 -       -       -
## 8  0.00020 0.00596 1.00000 -       -
## 15 0.00946 0.27532 1.00000 1.00000 -
## 30 0.14860 1.00000 0.20761 0.18174 1.00000
```

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

7.3 Ordered exploitation results

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

7.3.1 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 = filter(over_time_df, acro == 'ord') %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

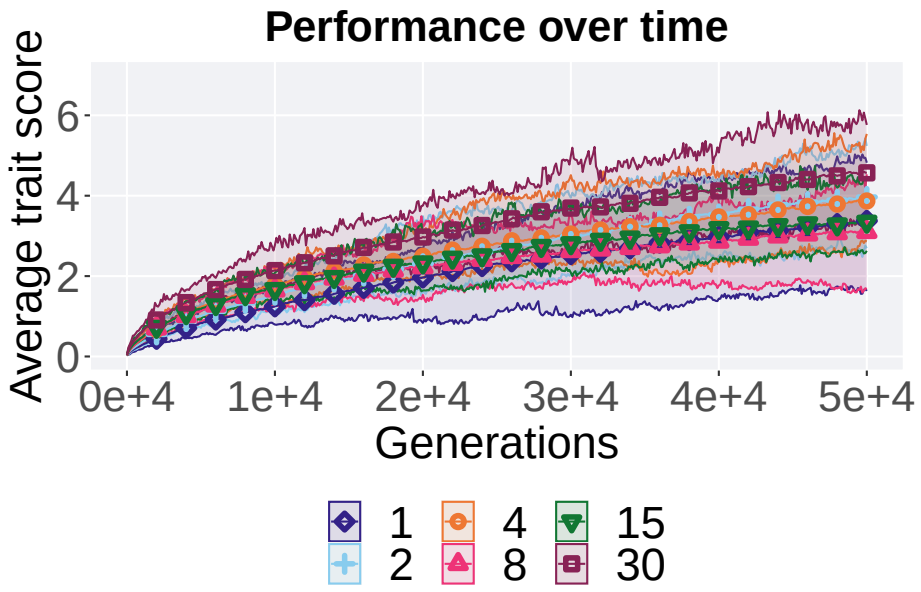
```
## `summarise()` has grouped output by 'K'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape
  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(0,7)
  ) +
  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()) +
```

```
  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")
  )

over_time_plot
```



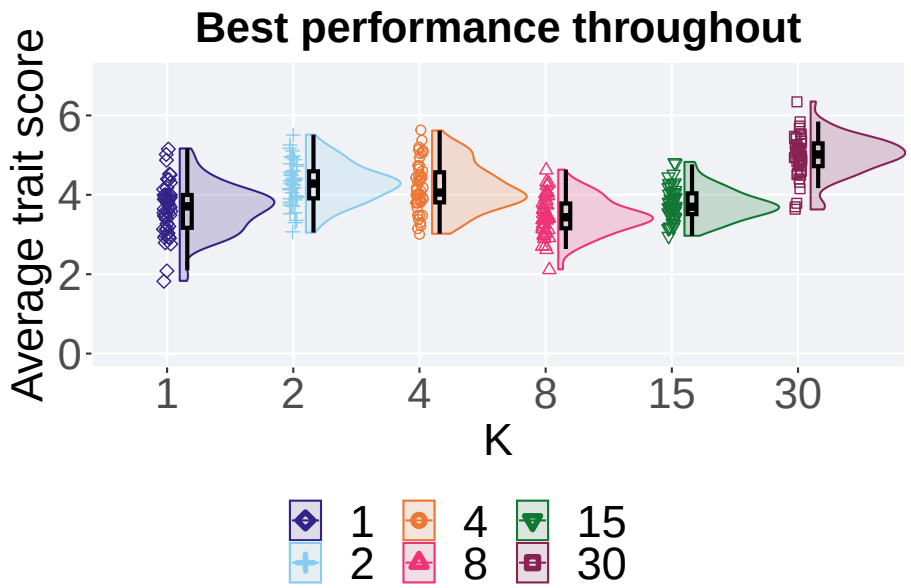
7.3.2 Best performance throughout

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

```
plot = filter(best_df, acro == 'ord' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = K, y = val / DIMENSIONALITY, color = K, fill = K, shape = K)) +
  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, positio
  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(0,7)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
)
```



7.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'ord' &  var == 'pop_fit_max')
performance$K = factor(performance$K, levels = NS_LIST)
performance %>%
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0  1.83   3.71  3.65  5.17 0.827
## 2 2        50      0  3.05   4.28  4.28  5.52 0.682
## 3 4        50      0  3.02   4.06  4.19  5.62 0.758
## 4 8        50      0  2.12   3.44  3.47  4.64 0.625
## 5 15       50      0  2.97   3.70  3.77  4.83 0.525
## 6 30       50      0  3.63   5.02  4.98  6.35 0.575
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 134.98, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$K, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

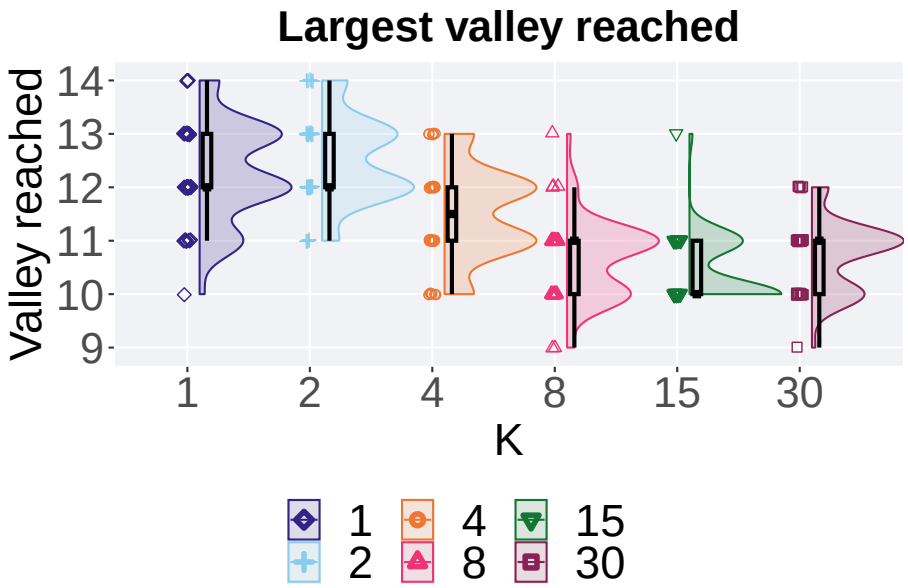
```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$K
##
##    1       2       4       8       15
## 2  2.7e-05 -       -       -       -
## 4  0.0016  1.0000  -       -       -
## 8  1.0000  1.8e-09 4.4e-07 -       -
## 15 1.0000  1.1e-05 0.0038  0.0195  -
## 30 6.8e-13 1.7e-07 3.2e-07 2.5e-15 8.8e-14
##
## P value adjustment method: bonferroni
```

7.3.3 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, acro == 'ord' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = K, y = val, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(8.9,14.1),
    breaks = c(9,10,11,12,13,14)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
)
```



7.3.3.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'ord' & var == 'ele_big_peak')
valleys$K = factor(valleys$K, levels = NS_LIST)
valleys %>%
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0    10   12    12.3    14     1
## 2 2        50      0    11   12    12.5    14     1
## 3 4        50      0    10   11.5  11.5    13     1
## 4 8        50      0     9   11    10.7    13     1
## 5 15       50      0    10   10    10.4    13     1
```

```
## 6 30       50      0     9   11    10.7    12     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 162.64, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$K, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$K
##
##    1       2       4       8       15
## 2  1.00000 -       -       -       -
## 4  0.00118 3.1e-06 -       -       -
## 8  1.4e-11 3.4e-14 3.8e-05 -       -
## 15 7.6e-14 8.6e-16 2.0e-08 0.61124 -
## 30 2.0e-11 2.5e-14 0.00017 1.00000 0.07759
##
## P value adjustment method: bonferroni
```

7.4 Contradictory objectives results

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

7.4.1 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.

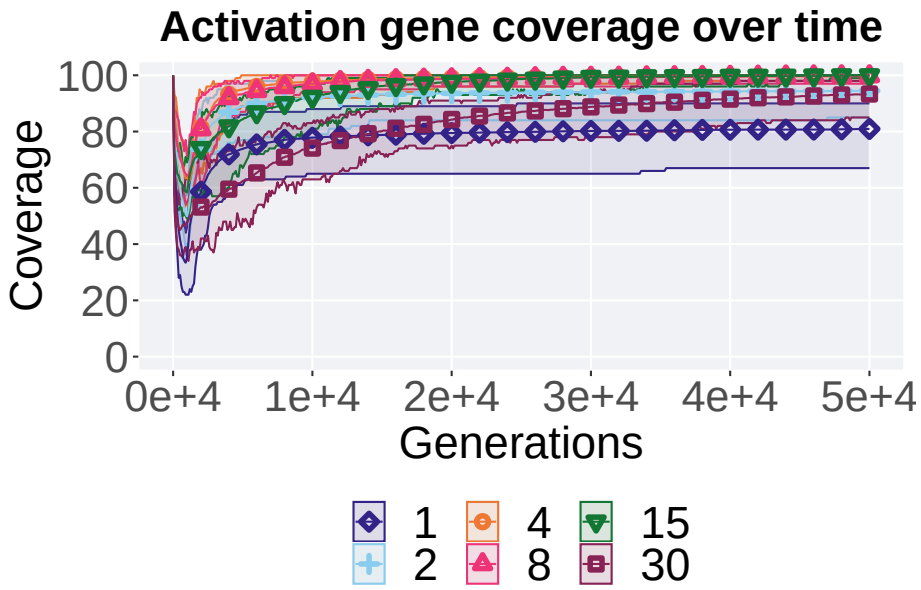
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
```

```
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



7.4.2 Final activation gene coverage

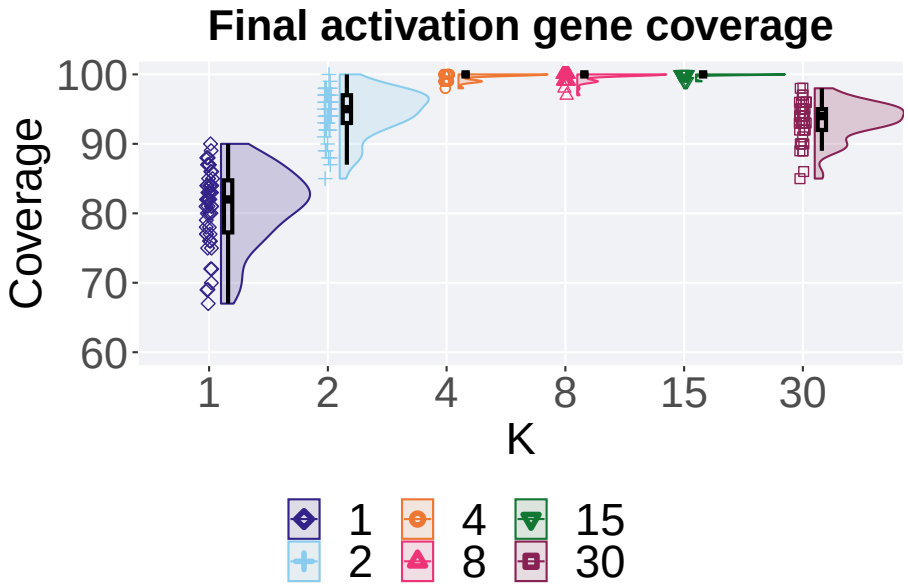
Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = K, y = uni_str_pos, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(60,100)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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 67 rows containing missing values (`geom_point()`).
```



7.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$K = factor(act_coverage$K, levels = NS_LIST)
act_coverage %>%
  group_by(K) %>%
  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: 6 x 8
```

```
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 1        50      0    67     82  80.9    90   7.5
## 2 2        50      0    85     95  94.5   100   4
## 3 4        50      0    98    100  99.7   100   0
## 4 8        50      0    97    100  99.7   100   0
## 5 15       50      0    99    100  99.9   100   0
## 6 30       50      0    85     94  93.3    98   3
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by K
## Kruskal-Wallis chi-squared = 262.41, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$K, p.adjust.method
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$K
##
##    1       2       4       8       15
## 2  5.6e-16 -       -       -       -
## 4  < 2e-16 7.3e-16 -       -       -
## 8  < 2e-16 8.5e-16 1.00    -       -
## 15 < 2e-16 < 2e-16 0.17    0.31    -
## 30 5.1e-16 0.42    < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

7.4.3 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.

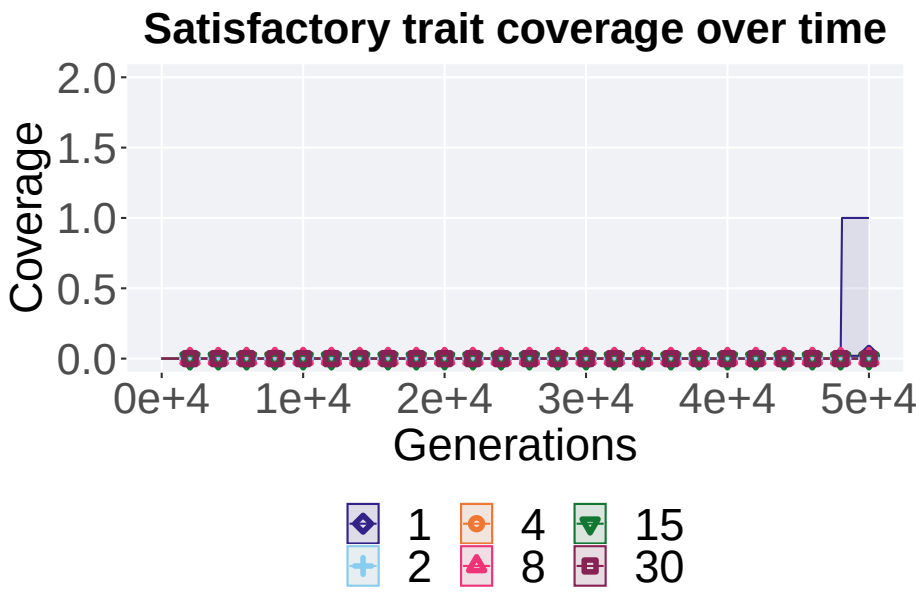
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
```

```
    max = max(pop_uni_obj)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2)
  ) +
  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()) +
  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")
  )
```



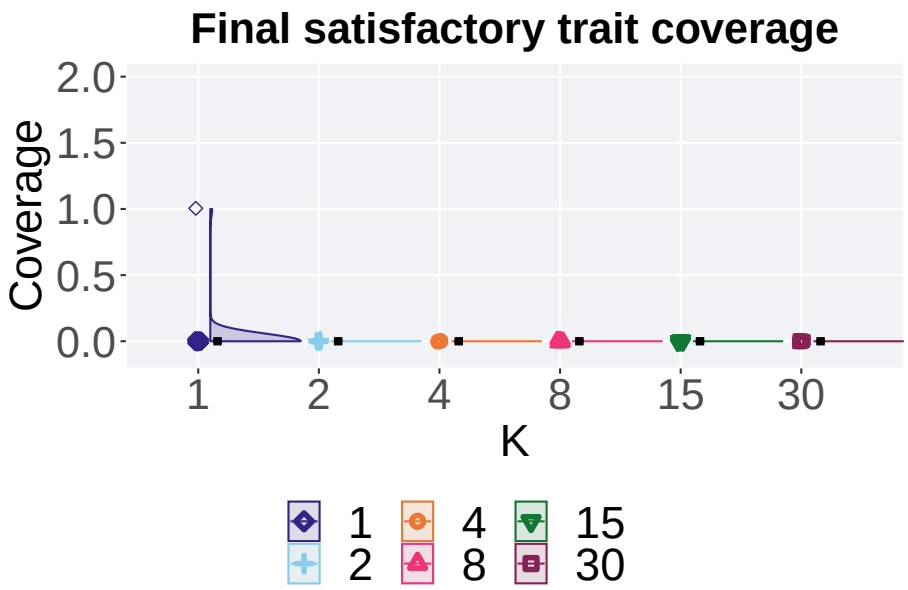
7.4.4 Final satisfactory trait coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = K, y = pop_uni_obj, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(-0.1, 2)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
)
```



7.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$K = factor(sat_coverage$K, levels = NS_LIST)
sat_coverage %>%
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 1        50      0     0      0  0.02     1     0
## 2 2        50      0     0      0  0        0     0
## 3 4        50      0     0      0  0        0     0
## 4 8        50      0     0      0  0        0     0
## 5 15       50      0     0      0  0        0     0
## 6 30       50      0     0      0  0        0     0
```

Kruskal–Wallis test illustrates evidence of **no statistical differences**.

```
kruskal.test(pop_uni_obj ~ K, data = sat_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by K
## Kruskal-Wallis chi-squared = 5, df = 5, p-value = 0.4159
```

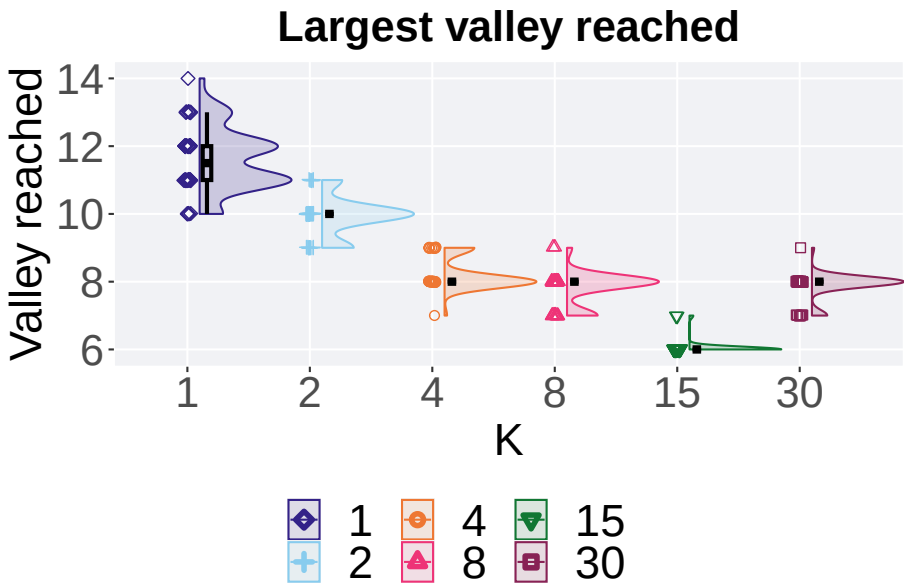
7.4.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'con' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = K, y = val, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Valley reached",
    limits=c(5.9,14.1),
    breaks=c(6,8,10,12,14)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
```

```
)
```



7.4.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'con' & var == 'ele_big_peak')
valleys$K = factor(valleys$K, levels = NS_LIST)
valleys %>%
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0    10   11.5 11.6     14     1
## 2 2        50      0     9   10    9.92    11     0
## 3 4        50      0     7    8    8.22     9     0
```

```
## 4 8       50      0     7    8    7.8      9     0
## 5 15      50      0     6    6    6.04     7     0
## 6 30      50      0     7    8    7.88     9     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 273.17, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$K, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$K
##
##    1       2       4       8       15
## 2  3.3e-13 -       -       -       -
## 4  < 2e-16 < 2e-16 -       -       -
## 8  < 2e-16 < 2e-16 0.00089 -       -
## 15 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 30 < 2e-16 < 2e-16 0.00301 1.00000 < 2e-16
##
## P value adjustment method: bonferroni
```

7.5 Multi-path exploration results

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

7.5.1 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.

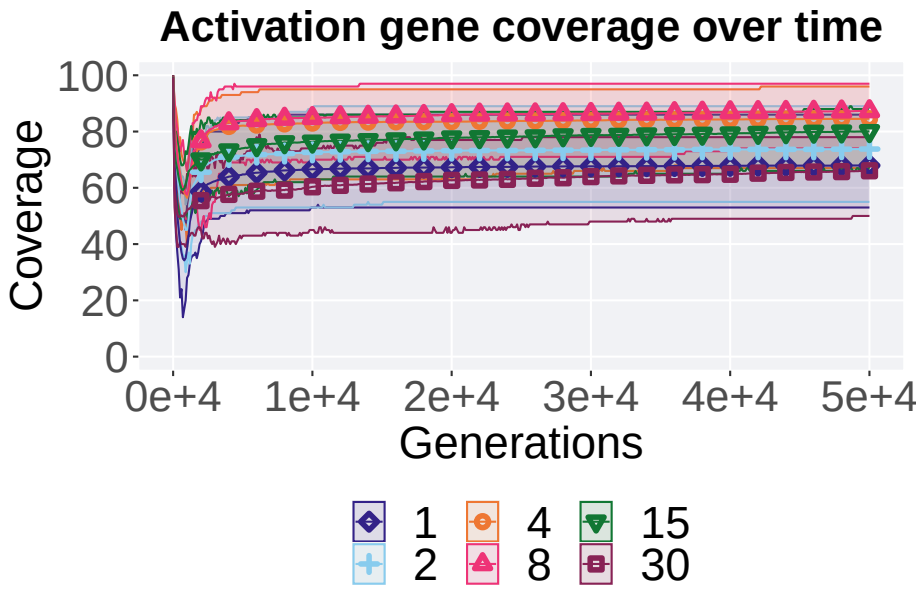
```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(K, gen) %>%
```

```
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="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()) +
  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")
  )
```



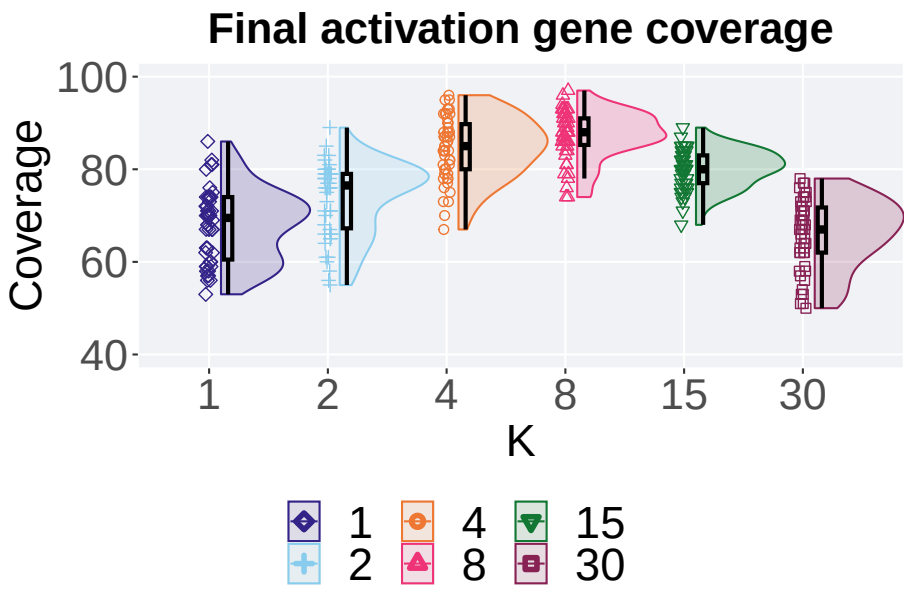
7.5.2 Final activation gene coverage

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

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = K, y = uni_str_pos, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(40, 100)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
)
```



7.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$K = factor(act_coverage$K, levels = NS_LIST)
act_coverage %>%
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 1        50      0    53   69.5  67.9    86 13.5
## 2 2        50      0    55   76.5  73.8    89 11.8
## 3 4        50      0    67   85    84.5    96  9.75
## 4 8        50      0    74   88    87.3    97  5.75
## 5 15       50      0    68   80    80.1    89  6
## 6 30       50      0    50   67    66.1    78  9.75
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by K
## Kruskal-Wallis chi-squared = 186.99, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$K, p.adjust.method
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$K
##
##    1       2       4       8       15
## 2  0.00498 -       -       -       -
## 4  5.1e-13 2.1e-08 -       -       -
## 8  2.7e-15 2.0e-12 0.55005 -       -
## 15 1.8e-11 0.00017 0.00300 7.0e-09 -
## 30 1.00000 4.6e-05 8.7e-15 3.1e-16 2.5e-14
##
## P value adjustment method: bonferroni
```

7.5.3 Performance over time

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

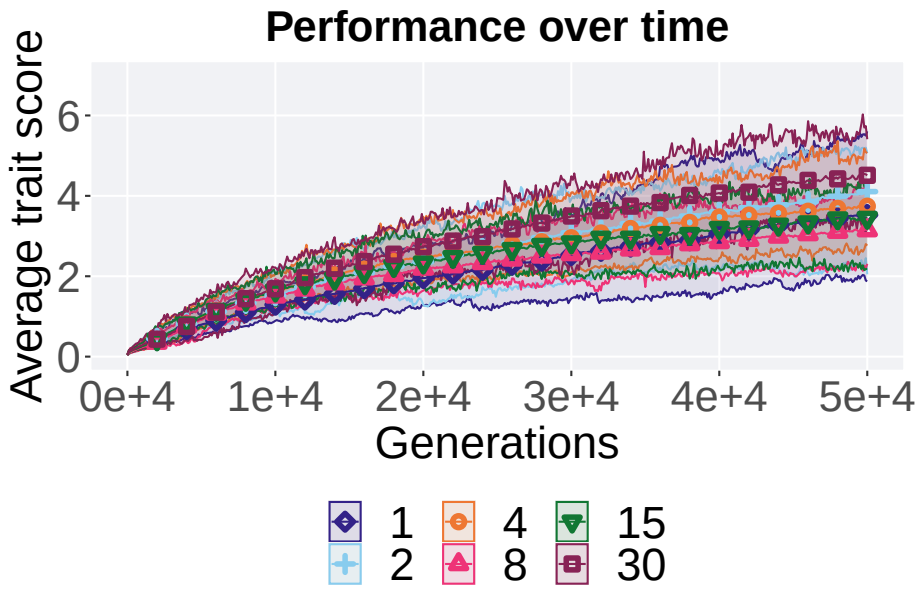
```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 7)
  ) +
  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()) +
  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")
  )

over_time_plot
```



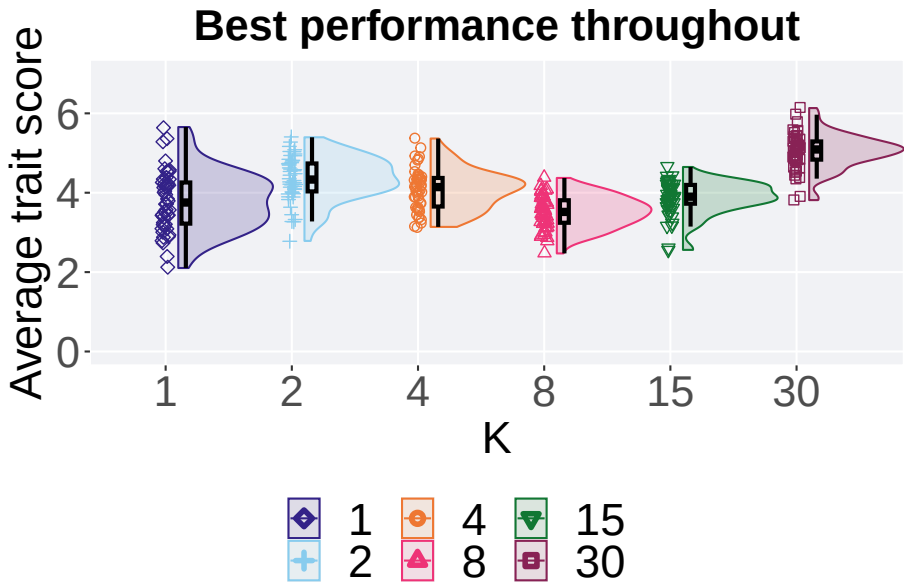
7.5.4 Best performance throughout

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

```
plot = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max') %>%
  ggplot(., aes(x = K, y = val / DIMENSIONALITY, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = (
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Average trait score",
    limits=c(0,7)
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
)
```



7.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, acro == 'mpe' &  var == 'pop_fit_max')
performance$K = factor(performance$K, levels = NS_LIST)
performance %>%
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 1        50      0  2.10   3.75  3.79  5.65 1.04
## 2 2        50      0  2.79   4.33  4.36  5.40 0.704
## 3 4        50      0  3.14   4.14  4.09  5.37 0.720
## 4 8        50      0  2.47   3.52  3.51  4.37 0.568
## 5 15       50      0  2.56   3.90  3.87  4.65 0.473
## 6 30       50      0  3.81   5.09  5.04  6.13 0.458
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 138.11, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = performance$val, g = performance$K, p.adjust.method = "bonferr
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$K
##
##    1       2       4       8       15
## 2  0.00094 -       -       -       -
## 4  0.37925 0.15967 -       -       -
## 8  0.60413 3.0e-10 3.0e-06 -       -
## 15 1.00000 3.1e-05 0.62456 0.00030 -
## 30 6.3e-12 9.1e-08 9.0e-12 4.2e-16 5.4e-15
##
## P value adjustment method: bonferroni
```

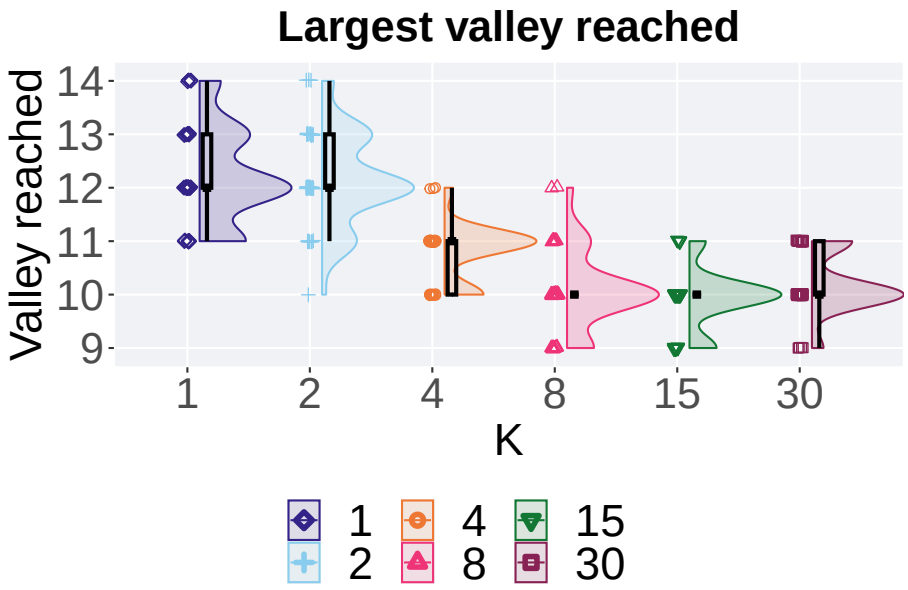
7.5.5 Largest valley reached throughout

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

```
plot = filter(best_df, acro == 'mpe' & var == 'ele_big_peak') %>%
  ggplot(., aes(x = K, y = val, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  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
  scale_y_continuous(
    name="Valley reached",
    limits=c(8.9,14.1)
```

```
  ) +
  scale_x_discrete(
    name="K"
  )+
  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)
)
```



7.5.5.1 Stats

Summary statistics for the largest valley crossed.

```
valleys = filter(best_df, acro == 'mpe' & var == 'ele_big_peak')
valleys$K = factor(valleys$K, levels = NS_LIST)
valleys %>%
```

```
  group_by(K) %>%
  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: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0    11     12 12.2     14     1
## 2 2        50      0    10     12 12.2     14     1
## 3 4        50      0    10     11 10.8     12     1
## 4 8        50      0     9     10 10.1     12     0
## 5 15       50      0     9     10  9.92    11     0
## 6 30       50      0     9     10 10.2     11     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 201.02, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = valleys$val, g = valleys$K, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  valleys$val and valleys$K
##
##    1       2       4       8    15
## 2  1.00    -       -       -    -
## 4  4.2e-12 7.4e-12 -       -    -
## 8  2.9e-15 5.9e-15 2.9e-06 -    -
## 15 < 2e-16 < 2e-16 5.3e-09 1.00 -
## 30 1.1e-15 2.6e-15 5.9e-05 1.00 0.25
##
```

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