

# Diagnostics Supplemental Material

Jose Guadalupe Hernandez

2022-11-18



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	About our supplemental material . . . . .	5
1.2	Contributing authors . . . . .	6
1.3	Research overview . . . . .	6
1.4	Experimental setup . . . . .	6
<b>2</b>	<b>Exploitation rate results</b>	<b>11</b>
2.1	Analysis dependencies . . . . .	11
2.2	Setup . . . . .	11
2.3	Performance over time . . . . .	12
2.4	Best performance throughout . . . . .	13
2.5	Generation satisfactory solution found . . . . .	16
<b>3</b>	<b>Ordered exploitation results</b>	<b>19</b>
3.1	Analysis dependencies . . . . .	19
3.2	Setup . . . . .	19
3.3	Performance over time . . . . .	20
3.4	Best performance throughout . . . . .	21
3.5	Generation satisfactory solution found . . . . .	24
3.6	Streaks . . . . .	26
<b>4</b>	<b>Contradictory objectives results</b>	<b>31</b>
4.1	Analysis dependencies . . . . .	31
4.2	Setup . . . . .	31
4.3	Satisfactory trait coverage . . . . .	32
4.4	Activation gene coverage . . . . .	39
4.5	Nondominated sorting split . . . . .	43
<b>5</b>	<b>Multi-path exploration results</b>	<b>53</b>
5.1	Analysis dependencies . . . . .	53
5.2	Setup . . . . .	53
5.3	Performance . . . . .	54
5.4	Activation gene coverage . . . . .	61

<b>6 Multi-valley crossing results</b>	<b>67</b>
6.1 Analysis dependencies . . . . .	67
6.2 Setup . . . . .	67
6.3 Performance . . . . .	68
6.4 Best gene value . . . . .	75
<b>7 Truncation selection</b>	<b>83</b>
7.1 Exploitation rate results . . . . .	84
7.2 Ordered exploitation results . . . . .	88
7.3 Contraditory objectives diagnostic . . . . .	93
7.4 Multi-path exploration results . . . . .	102
<b>8 Tournament selection</b>	<b>115</b>
8.1 Exploitation rate results . . . . .	116
8.2 Ordered exploitation results . . . . .	120
8.3 Contraditory objectives diagnostic . . . . .	125
8.4 Multi-path exploration results . . . . .	134
<b>9 Genotypic fitness sharing</b>	<b>145</b>
9.1 Exploitation rate results . . . . .	146
9.2 Ordered exploitation results . . . . .	150
9.3 Contraditory objectives diagnostic . . . . .	153
9.4 Multi-path exploration results . . . . .	162
9.5 Multi-valley crossing results . . . . .	173
<b>10 Phenotypic fitness sharing</b>	<b>187</b>
10.1 Exploitation rate results . . . . .	188
10.2 Ordered exploitation results . . . . .	192
10.3 Contraditory objectives diagnostic . . . . .	196
10.4 Multi-path exploration results . . . . .	206
10.5 Multi-valley crossing results . . . . .	216
<b>11 Nondominated sorting</b>	<b>231</b>
11.1 Exploitation rate results . . . . .	232
11.2 Ordered exploitation results . . . . .	236
11.3 Contraditory objectives diagnostic . . . . .	240
11.4 Multi-path exploration results . . . . .	252
11.5 Multi-valley crossing results . . . . .	263
<b>12 Novelty Search</b>	<b>277</b>
12.1 Exploitation rate results . . . . .	278
12.2 Ordered exploitation results . . . . .	282
12.3 Contraditory objectives diagnostic . . . . .	286
12.4 Multi-path exploration results . . . . .	296

# Chapter 1

## Introduction

This is the supplemental material associated with our 2022 ECJ contribution entitled, *A suite of diagnostic metrics for characterizing selection schemes*. Preprint [here](#).

### 1.1 About our supplemental material

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

Our supplemental material includes the following paper figures and statistics:

- Exploitation rate results (Section 2)
- Ordered exploitation results (Section 3)
- Contradictory objectives results (Section 4)
- Multi-path exploration results (Section 5)
- Multi-valley crossing results (Section 6)

Additionally, our supplemental material includes the results from parameter tuning selection schemes:

- Truncation selection (Section 7)
- Tournament selection sharing (Section 8)
- Genotypic fitness sharing (Section 9)
- Phenotypic fitness sharing (Section 10)
- Nondominated sorting (Section 11)
- Novelty search (Section 12)

## 1.2 Contributing authors

- Jose Guadalupe Hernandez
- Alexander Lalejini
- Charles Ofria

## 1.3 Research overview

### Abstract:

Evolutionary algorithms typically consist of multiple interacting components, where each component influences an algorithm’s problem-solving abilities. Understanding how each component of an evolutionary algorithm influences problem-solving success can improve our ability to target particular problem domains. Benchmark suites provide insights into an evolutionary algorithm’s problem-solving capabilities, but benchmarking problems often have complex search space topologies, making it difficult to isolate and test an algorithm’s strengths and weaknesses. Our work focuses on diagnosing selection schemes, which identify individuals to contribute genetic material to the next generation, thus driving an evolutionary algorithm’s search strategy. We introduce four diagnostics for empirically testing the strengths and weaknesses of selection schemes: the exploitation rate diagnostic, ordered exploitation rate diagnostic, contradictory objectives diagnostic, and the multi-path exploration diagnostic. Each diagnostic is a handcrafted search space designed to isolate and measure the relative exploitation and exploration characteristics of selection schemes. Here, we use our diagnostics to evaluate six population selection methods: truncation selection, tournament selection, fitness sharing, lexicase selection, nondominated sorting, and novelty search. Expectedly, tournament and truncation selection excelled at gradient exploitation but poorly explored search spaces, while novelty search excelled at exploration but failed to exploit gradients. Fitness sharing performed poorly across all diagnostics, suggesting poor overall exploitation and exploration abilities. Nondominated sorting was best for maintaining diverse populations comprised of individuals inhabiting multiple optima, but struggled to effectively exploit gradients. Lexicase selection balanced search space exploration without sacrificing exploitation, generally performing well across diagnostics. Our work demonstrates the value of diagnostics for building a deeper understanding of selection schemes, which can then be used to improve or develop new selection methods.

## 1.4 Experimental setup

Setting up required variables variables.

```
library(ggplot2)
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

# variables used throughout
TRAITS = 100
TSIZE = 26
ORDER = c('Truncation (tru)', 'Tournament (tor)', 'Lexicase (lex)', 'Genotypic Fitness Sharing (gfs')
ACRON = tolower(c('TRU', 'TOR', 'LEX', 'GFS', 'PFS', 'NDS', 'NOV', 'RAN'))
SHAPE = c(5,3,1,2,6,0,4,20,1)
PARAM = c('8', '8', '0.0', '0.3', '0.3', '0.3', '15', '1')
cb_palette <- c('#332288', '#88CCEE', '#EE7733', '#EE3377', '#117733', '#882255', '#44AA99', '#CCBB44',
GENERATIONS = 50000

# selection scheme parameters
TR_LIST = c(1, 2, 4, 8, 16, 32, 64, 128, 256)
TS_LIST = c(2, 4, 8, 16, 32, 64, 128, 256)
FS_LIST = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0)
ND_LIST = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0)
NS_LIST = c(1, 2, 4, 8, 15, 30)

# theme that graphs will follow
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=18),
  legend.text=element_text(size=14),
  axis.title = element_text(size=18),
  axis.text = element_text(size=16),
  legend.position="bottom",
  panel.background = element_rect(fill = "#f1f2f5",
                                    colour = "white",
                                    size = 0.5, linetype = "solid")
)

## Warning: The `size` argument of `element_rect()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.

```

```

# cross comparison data frames
cc_over_time <- read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL')
colnames(cc_over_time)[colnames(cc_over_time) == "Selection.Scheme"] = 'Selection\nnScheme'
cc_over_time$`Selection\nnScheme` <- factor(cc_over_time$`Selection\nnScheme`, levels = 0:1)
cc_over_time$uni_str_pos = cc_over_time$uni_str_pos + cc_over_time$arc_acti_gene - cc_over_time$arc_acti_gene

cc_best = read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL')
cc_best$acron <- factor(cc_best$acron, levels = ACRON)
colnames(cc_best)[colnames(cc_best) == "Selection.Scheme"] = 'Selection\nnScheme'

cc_ssf = read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL')
cc_ssf$acron <- factor(cc_ssf$acron, levels = ACRON)
colnames(cc_over_time)[colnames(cc_over_time) == "Selection.Scheme"] = 'Selection\nnScheme'
cc_ssf[is.na(cc_ssf)] <- 59999

cc_end <- filter(cc_over_time, gen == 50000)
cc_end$acron <- factor(cc_end$acron, levels = ACRON)

# selection scheme data frames
ss_over_time <- read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL')
colnames(ss_over_time)[colnames(ss_over_time) == "Selection.Scheme"] = 'Selection\nnScheme'
ss_over_time$uni_str_pos = ss_over_time$uni_str_pos + ss_over_time$arc_acti_gene - ss_over_time$arc_acti_gene

ss_best <- read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL')
colnames(ss_best)[colnames(ss_best) == "Selection.Scheme"] = 'Selection\nnScheme'

ss_ssf <- read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL')
colnames(ss_ssf)[colnames(ss_ssf) == "Selection.Scheme"] = 'Selection\nnScheme'

## genotypic fitness sharing data frames
gfs_ot <- filter(ss_over_time, acron == 'gfs')
gfs_ot$Sigma <- factor(gfs_ot$trt, levels = FS_LIST)
gfs_best <- filter(ss_best, acron == 'gfs')
gfs_best$Sigma <- factor(gfs_best$trt, levels = FS_LIST)
gfs_end <- filter(gfs_ot, gen == 50000)

## phenotypic fitness sharing data frames
pfs_ot <- filter(ss_over_time, acron == 'pfs')
pfs_ot$Sigma <- factor(pfs_ot$trt, levels = FS_LIST)
pfs_best <- filter(ss_best, acron == 'pfs')
pfs_best$Sigma <- factor(pfs_best$trt, levels = FS_LIST)
pfs_end <- filter(pfs_ot, gen == 50000)

## nodominated sorting data frames
nds_ot <- filter(ss_over_time, acron == 'nds')

```

```

nds_ot$Sigma <- factor(nds_ot$trt, levels = ND_LIST)
nds_best <- filter(ss_best, acron == 'nds')
nds_best$Sigma <- factor(nds_best$trt, levels = ND_LIST)
nds_end <- filter(nds_ot, gen == 50000)

## novelty search data frames
nov_ot <- filter(ss_over_time, acron == 'nov' & trt != 0)
nov_ot$K <- factor(nov_ot$trt, levels = NS_LIST)
nov_best <- filter(ss_best, acron == 'nov' & trt != 0)
nov_best$K <- factor(nov_best$trt, levels = NS_LIST)
nov_end <- filter(nov_ot, gen == 50000)

## tournament data frames
tor_ot <- filter(ss_over_time, acron == 'tor' & trt != 1)
tor_ot$T <- factor(tor_ot$trt, levels = TS_LIST)
tor_best <- filter(ss_best, acron == 'tor' & trt != 1)
tor_best$T <- factor(tor_best$trt, levels = TS_LIST)
tor_end <- filter(tor_ot, gen == 50000)
tor_ssf <- filter(ss_ssf, acron == 'tor' & trt != 1)
tor_ssf$T <- factor(tor_ssf$trt, levels = TS_LIST)
tor_ssf[is.na(tor_ssf)] <- 59999

## truncation data frames
tru_ot <- filter(ss_over_time, acron == 'tru')
tru_ot$T <- factor(tru_ot$trt, levels = TR_LIST)
tru_best <- filter(ss_best, acron == 'tru')
tru_best$T <- factor(tru_best$trt, levels = TR_LIST)
tru_end <- filter(tru_ot, gen == 50000)
tru_ssf <- filter(ss_ssf, acron == 'tru')
tru_ssf$T <- factor(tru_ssf$trt, levels = TR_LIST)
tru_ssf[is.na(tru_ssf)] <- 59999

```

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        Patched
## major         4
## minor        2.2
## year         2022
## month        11

```

```
## day          10
## svn rev     83330
## language    R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname    Innocent and Trusting
```

# Chapter 2

## Exploitation rate results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

### 2.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

### 2.2 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         Patched
## major          4
## minor          2.2
## year           2022
## month          11
```

```
## day          10
## svn rev     83330
## language    R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname    Innocent and Trusting
```

## 2.3 Performance over time

Best performance in a population over time.

```
exploitation_rate = filter(cc_over_time, diagnostic == 'exploitation_rate')
lines = exploitation_rate %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

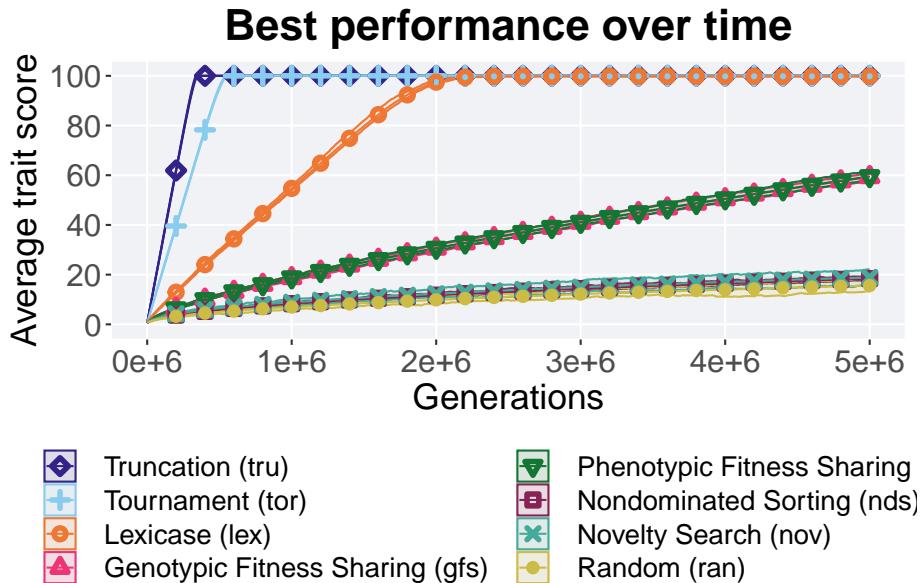
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = `Selection\nScheme`, fill = Selection))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
```

```

guides(
  shape=guide_legend(ncol=2, title.position = "bottom"),
  color=guide_legend(ncol=2, title.position = "bottom"),
  fill=guide_legend(ncol=2, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title=element_blank()
)

ot

```



## 2.4 Best performance throughout

Best performance throughout 50,000 generations.

```

plot = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate') %>%
  ggplot(., aes(x = acron, y = val / TRAITS, color = acron, fill = acron, shape = acron)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous()

```

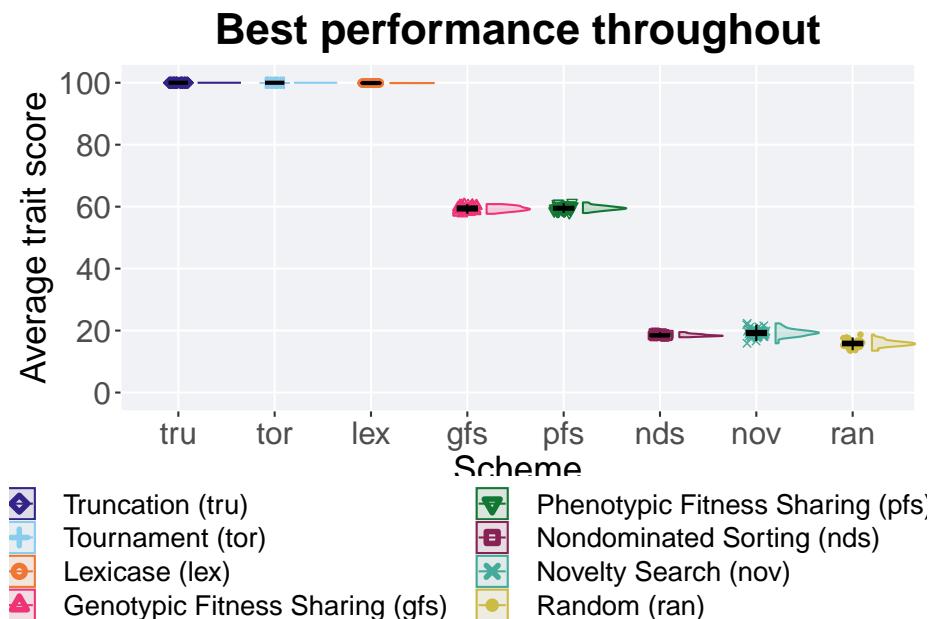
```

name="Average trait score",
limits=c(-1, 101),
breaks=seq(0,100, 20),
labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_discrete(
  name="Scheme"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Best performance throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```

## Warning: Using the `size` aesthetic with geom\_polygon was deprecated in ggplot2 3.0.0.  
 ## i Please use the `linewidth` aesthetic instead.



### 2.4.1 Stats

Summary statistics for the performance of the best performance throughout 50,000 generations.

```
performance = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')
performance$acron = factor(performance$acron, levels = c('tru', 'tor', 'lex', 'gfs', 'pfs', 'nov')
performance%>%
group_by(acron) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(val)),
  min = min(val / TRAITS, na.rm = TRUE),
  median = median(val / TRAITS, na.rm = TRUE),
  mean = mean(val / TRAITS, na.rm = TRUE),
  max = max(val / TRAITS, na.rm = TRUE),
  IQR = IQR(val / TRAITS, na.rm = TRUE)
)

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 tru     50      0 100    100    100    100    0
## 2 tor     50      0 100    100    100    100    0
## 3 lex     50      0 99.9   99.9   99.9   99.9  0.0137
## 4 gfs     50      0 57.7    59.3   59.4   60.8  1.31
## 5 pfs     50      0 58.0    59.5   59.5   61.4  0.908
## 6 nov     50      0 15.9    19.2   19.3   22.3  1.34
## 7 nds     50      0 17.9    18.4   18.5   19.5  0.516
## 8 ran     50      0 13.5    15.9   15.9   18.7  1.15
```

Kruskal–Wallis test provides evidence of statistical differences among best performance found throughout 50,000 generations.

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

##
##  Kruskal-Wallis rank sum test
##
## data: val by acron
## Kruskal-Wallis chi-squared = 384.91, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

```
##
```

```

##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$val and performance$acron
##
##    tru      tor      lex      gfs      pfs      nov      nds
## tor 1e+00   -       -       -       -       -       -
## lex < 2e-16 < 2e-16   -       -       -       -       -
## gfs < 2e-16 < 2e-16 < 2e-16   -       -       -       -
## pfs < 2e-16 < 2e-16 < 2e-16 1e+00   -       -       -
## nov < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16   -       -
## nds < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 6e-04   -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.9e-15 7.9e-16
##
## P value adjustment method: bonferroni

```

## 2.5 Generation satisfactory solution found

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

```

ssf = filter(cc_ssf, diagnostic == 'exploitation_rate')
schemes = data.frame()

for (i in 1:8) {
  if(i == 3)
  {
    schemes = rbind(schemes, filter(ssf, acron == ACRON[i]))
    next
  }
  schemes = rbind(schemes, filter(ssf, acron == ACRON[i] & trt == PARAM[i]))
}

plot <- ggplot(schemes, aes(x = acron, y = generation, color = acron, fill = acron, shape = acron))
plot + geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6", "Fail"))
) +
  scale_x_discrete(
    name="Scheme"

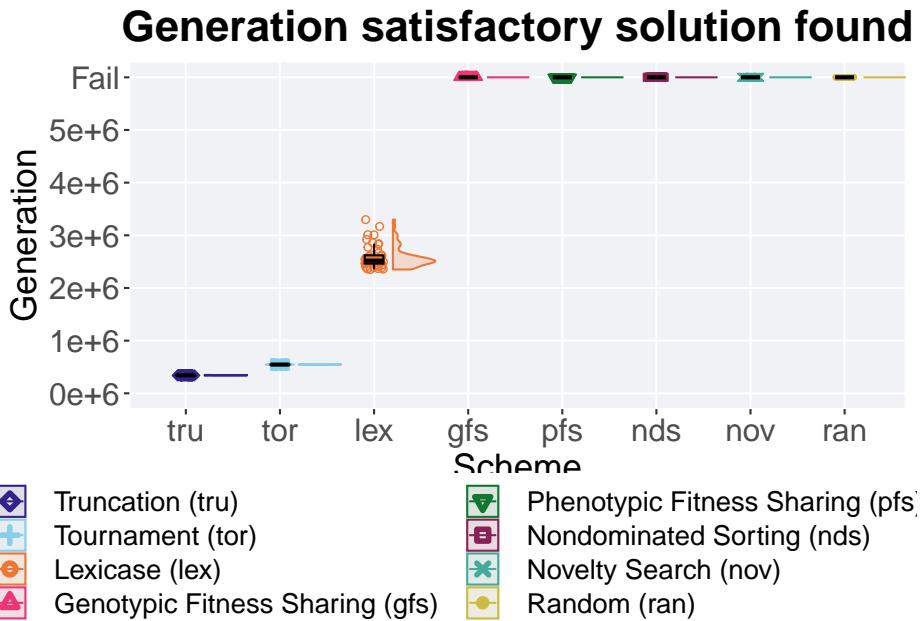
```

```

) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



### 2.5.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```

schemes <- filter(schemes, acron == 'tru' | acron == 'tor' | acron == 'lex')
group_by(schemes, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),

```

```

min = min(generation, na.rm = TRUE),
median = median(generation, na.rm = TRUE),
mean = mean(generation, na.rm = TRUE),
max = max(generation, na.rm = TRUE),
IQR = IQR(generation, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   acron count na_cnt   min median   mean   max     IQR
##   <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 tru      50     0  3357   3420  3421.  3481   34.2
## 2 tor      50     0  5403   5457  5453.  5519   51.8
## 3 lex      50     0 23514  25190  25857. 32980  1581

```

Kruskal–Wallis test provides evidence of difference amoung selection schemes the first generation a satisfactory solution is found throughout the 50,000 generations.

```
kruskal.test(generation ~ acron, data = schemes)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: generation by acron
## Kruskal-Wallis chi-squared = 132.46, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the first generation a satisfactory solution is found throughout the 50,000 generations.

```

pairwise.wilcox.test(x = schemes$generation, g = schemes$acron, p.adjust.method = "bonf"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

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

```

# Chapter 3

## Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

### 3.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

### 3.2 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         Patched
## major          4
## minor          2.2
## year          2022
## month         11
```

```
## day          10
## svn rev     83330
## language    R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname    Innocent and Trusting
```

### 3.3 Performance over time

Best performance in a population over time.

```
ordered_exploitation = filter(cc_over_time, diagnostic == 'ordered_exploitation')
lines = ordered_exploitation %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

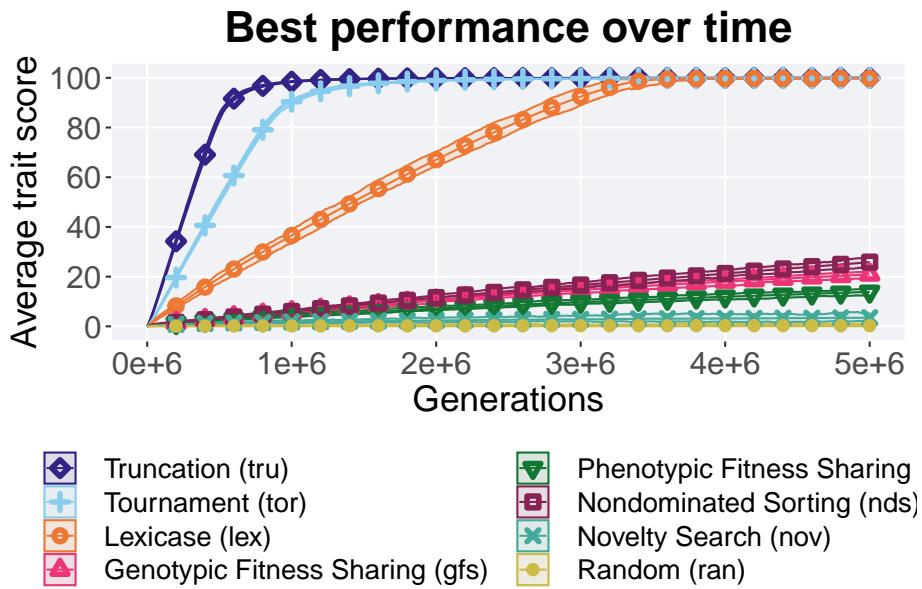
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = `Selection\nScheme`, fill = Selection))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
```

```

guides(
  shape=guide_legend(ncol=2, title.position = "bottom"),
  color=guide_legend(ncol=2, title.position = "bottom"),
  fill=guide_legend(ncol=2, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title=element_blank()
)

```

ot



### 3.4 Best performance throughout

Best performance throughout 50,000 generations.

```

plot = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation') %>%
  ggplot(., aes(x = acron, y = val / TRAITS, color = acron, fill = acron, shape = acron)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous()

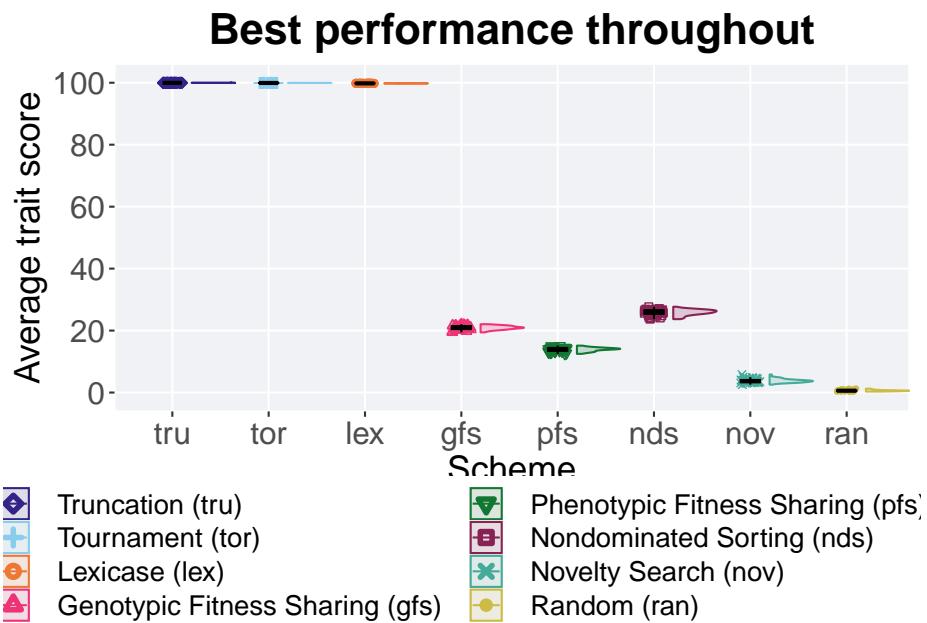
```

```

name="Average trait score",
limits=c(-1, 101),
breaks=seq(0,100, 20),
labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_discrete(
  name="Scheme"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Best performance throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



### 3.4.1 Stats

Summary statistics for the performance of the best performance throughout 50,000 generations.

```
performance = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')
performance$acron = factor(performance$acron, levels = c('tru', 'tor', 'lex', 'nds', 'gfs', 'pfs',
performance%>%
group_by(acron) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(val)),
  min = min(val / TRAITS, na.rm = TRUE),
  median = median(val / TRAITS, na.rm = TRUE),
  mean = mean(val / TRAITS, na.rm = TRUE),
  max = max(val / TRAITS, na.rm = TRUE),
  IQR = IQR(val / TRAITS, na.rm = TRUE)
)

## # A tibble: 8 x 8
##   acron count na_cnt     min   median    mean     max     IQR
##   <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 tru      50     0 100.    100.    100.    100.    0.00208
## 2 tor      50     0 99.9    99.9    99.9    99.9    0.00445
## 3 lex      50     0 99.8    99.8    99.8    99.8    0.0207
## 4 nds      50     0 23.7    26.0    25.9    27.7    1.17
## 5 gfs      50     0 19.4    21.0    20.9    22.1    0.970
## 6 pfs      50     0 12.5    14.1    13.9    15.1    0.871
## 7 nov      50     0 2.55    3.70    3.80    5.82    0.718
## 8 ran      50     0 0.319   0.598   0.634   1.26    0.240
```

Kruskal–Wallis test provides evidence of statistical differences among best performance found throughout 50,000 generations.

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

```
##
## Kruskal-Wallis rank sum test
##
## data: val by acron
## Kruskal-Wallis chi-squared = 392.77, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

```
##
```

```

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

```

### 3.5 Generation satisfactory solution found

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

```

ssf = filter(cc_ss, diagnostic == 'ordered_exploitation')
schemes = data.frame()

for (i in 1:8) {
  if(i == 3)
  {
    schemes = rbind(schemes, filter(ssf, acron == ACRON[i]))
    next
  }
  schemes = rbind(schemes, filter(ssf, acron == ACRON[i] & trt == PARAM[i]))
}

plot <- ggplot(schemes, aes(x = acron, y = generation, color = acron, fill = acron, shape = acron))
plot + geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6", "Fail"))
) +
  scale_x_discrete(
    name="Scheme"

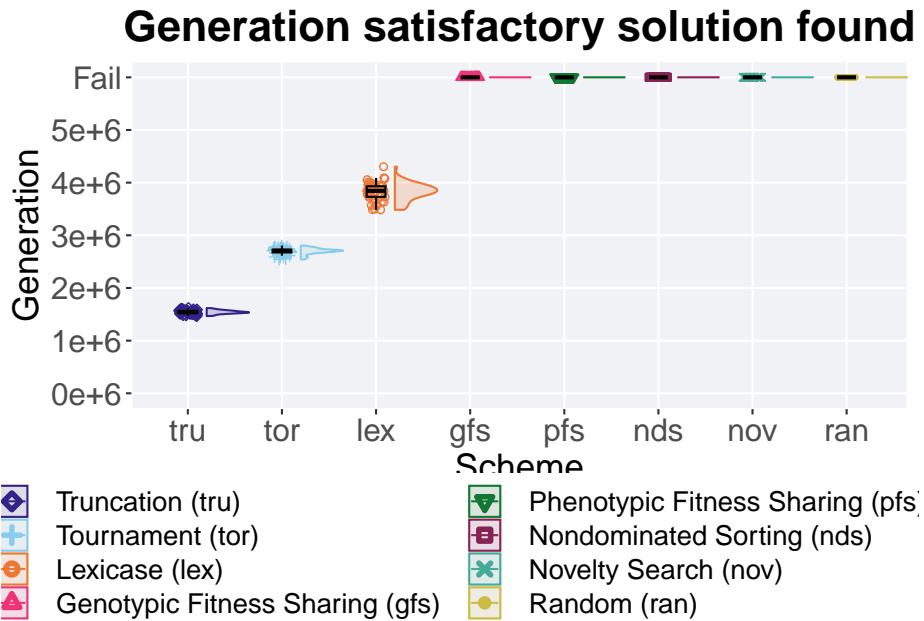
```

```

) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



### 3.5.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```

schemes <- filter(schemes, acron == 'tru' | acron == 'tor' | acron == 'lex')
group_by(schemes, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),

```

```

min = min(generation, na.rm = TRUE),
median = median(generation, na.rm = TRUE),
mean = mean(generation, na.rm = TRUE),
max = max(generation, na.rm = TRUE),
IQR = IQR(generation, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   acron count na_cnt  min median  mean  max   IQR
##   <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 tru     50     0 14676 15406. 15424. 16175 396.
## 2 tor     50     0 25421 27072. 26970. 28050 528.
## 3 lex     50     0 34842 38440. 38265. 43020 2006.

```

Kruskal–Wallis test provides evidence of difference amoung selection schemes the first generation a satisfactory solution is found throughout the 50,000 generations..

```
kruskal.test(generation ~ acron, data = schemes)
```

```

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

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the first generation a satisfactory solution is found throughout the 50,000 generations.

```

pairwise.wilcox.test(x = schemes$generation, g = schemes$acron, p.adjust.method = "bonf"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

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

```

## 3.6 Streaks

### 3.6.1 Over time

Best streak in a population over time.

```

streaks_df = filter(ordered_exploitation, acron == 'pfs' | acron == 'nds' | acron == 'gfs' | acron == 'nondominated')
streaks_df$`Selection\`nScheme` <- factor(streaks_df$`Selection\`nScheme`, levels = c('Nondominated', 'pfs', 'nds', 'gfs'))

lines = streaks_df %>%
  group_by(`Selection\`nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_str_max),
    mean = mean(pop_str_max),
    max = max(pop_str_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

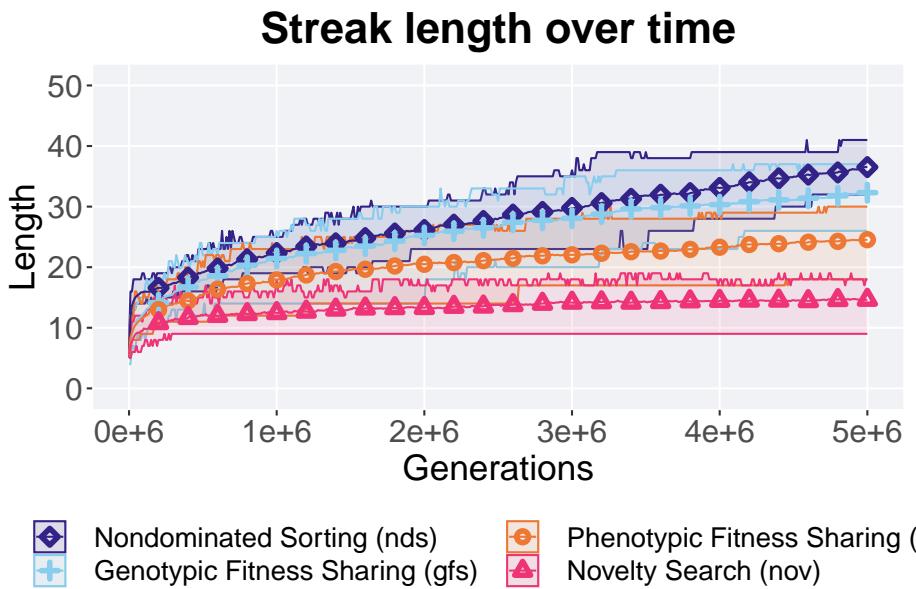
ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\`nScheme`, fill = `Selection\`nScheme`), color = `Selection\`nScheme`)
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Length",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Streak length over time") +
  p_theme +
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.title.position = "top"
  )

```

```

    legend.justification="center",
    legend.title=element_blank()
)
ot

```



### 3.6.2 End of 50,000 generations

Best streak in the population at the end of 50,000 generations.

```

end <- filter(streaks_df, gen == 50000)
end = filter(end, acron == 'pfs' | acron == 'nds' | acron == 'gfs' | acron == 'nov')
end$acron <- factor(end$acron, levels = c('nds', 'gfs', 'pfs', 'nov'))

plot = ggplot(end, aes(x = acron, y = pop_str_max, color = acron, fill = acron, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Length",
    limits=c(-1, 51),
    breaks=seq(0, 50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_discrete(
    name="Algorithm"
  )

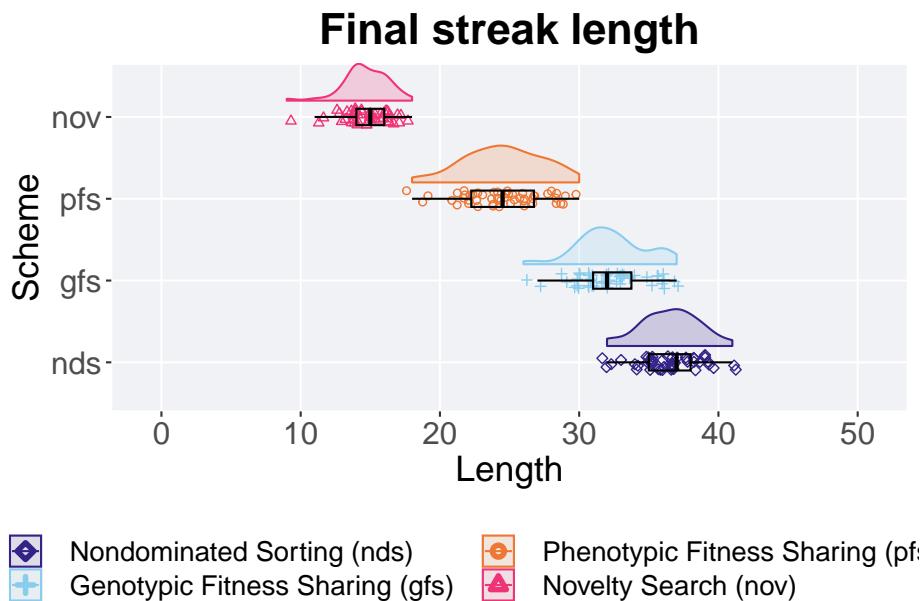
```

```

    name="Scheme"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
coord_flip() +
p_theme

plot_grid(
  plot +
  ggtitle("Final streak length") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



### 3.6.2.1 Stats

Summary statistics for best streak in the population at the end of 50,000 generations.

```

group_by(end, acron) %>%
  dplyr::summarise(

```

```

count = n(),
na_cnt = sum(is.na(pop_str_max)),
min = min(pop_str_max, na.rm = TRUE),
median = median(pop_str_max, na.rm = TRUE),
mean = mean(pop_str_max, na.rm = TRUE),
max = max(pop_str_max, na.rm = TRUE),
IQR = IQR(pop_str_max, na.rm = TRUE)
)

## # A tibble: 4 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <int> <dbl> <dbl> <int> <dbl>
## 1 nds     50     0    32    37    36.5    41     3
## 2 gfs     50     0    26    32    32.3    37    2.75
## 3 pfs     50     0    18    24.5   24.5    30    4.5
## 4 nov     50     0     9    15    14.7    18     2

```

Kruskal–Wallis test provides evidence of difference among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_str_max ~ acron, data = end)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_str_max by acron
## Kruskal-Wallis chi-squared = 178.91, df = 3, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best performance in the population at the end of 50,000 generations.

```

pairwise.wilcox.test(x = end$pop_str_max, g = end$acron, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_str_max and end$acron
##
##      nds     gfs     pfs
## gfs 2e-11   -     -
## pfs <2e-16 <2e-16   -
## nov <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

# Chapter 4

## Contradictory objectives results

Here we present the results for the **satisfactory trait coverage** and **activation gene coverage** generated by each selection scheme replicate on the contradictory objectives diagnostic. Note both of these values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100. Satisfactory trait coverage refers to the count of unique satisfied traits in a given population; this gives us a range of integers between 0 and 100.

### 4.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(Pupillometry)
```

### 4.2 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      Patched
## major       4
## minor      2.2
## year       2022
## month      11
## day        10
## svn rev    83330
## language   R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname   Innocent and Trusting

```

## 4.3 Satisfactory trait coverage

Satisfactory trait coverage analysis.

### 4.3.1 Coverage over time

Satisfactory trait coverage over time.

```

contradictory_objectives = filter(cc_over_time, diagnostic == 'contradictory_objectives')
lines = contradictory_objectives %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill = `Selection\nScheme`),
            geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
            geom_line(size = 0.5) +
            geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
            scale_y_continuous(
              name="Coverage",
              limits=c(-1, 101),
              breaks=seq(0,100, 20),
              labels=c("0", "20", "40", "60", "80", "100")
            ) +
            scale_x_continuous(
              name="Generations",
              limits=c(0, 50000),

```

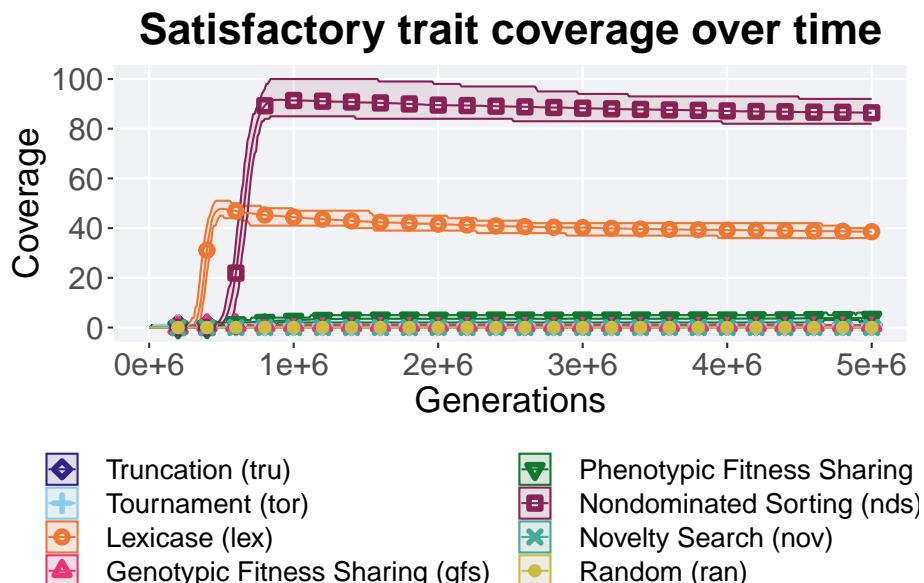
```

breaks=c(0, 10000, 20000, 30000, 40000, 50000),
labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
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 +
guides(
  shape=guide_legend(ncol=2, title.position = "bottom"),
  color=guide_legend(ncol=2, title.position = "bottom"),
  fill=guide_legend(ncol=2, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title=element_blank()
)

ot

```

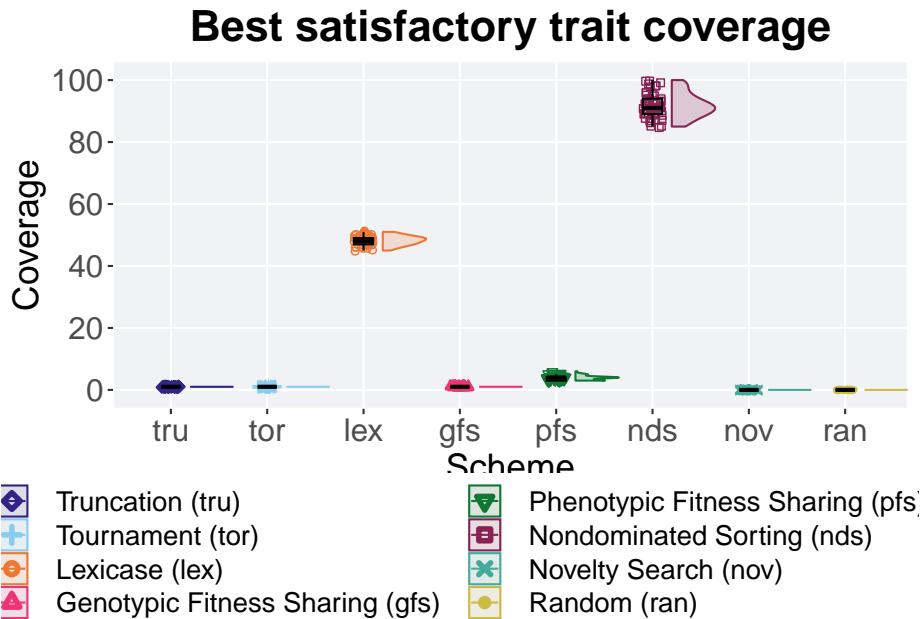


### 4.3.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(cc_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives')
plot = ggplot(best, aes(x = acron, y = val, color = acron, fill = acron, shape = acron))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100"))
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```



#### 4.3.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

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

## # A tibble: 8 x 8
##   acron count na_cnt  min median  mean  max  IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 nds     50     0    85    91  91.8   100     5
## 2 lex     50     0    45    48  48.2    51     2
## 3 pfs     50     0     3     4  3.84     6     1
## 4 gfs     50     0     1     1     1      1     0
## 5 tor     50     0     1     1     1      1     0
## 6 tru     50     0     1     1     1      1     0
```

```
## 7 nov      50      0      0      0  0      0      0
## 8 ran      50      0      0      0  0      0      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage throughout 50,000 generations.

```
kruskal.test(val ~ acron, data = best)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: val by acron
## Kruskal-Wallis chi-squared = 396.67, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage throughout 50,000 generations.

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

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

### 4.3.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

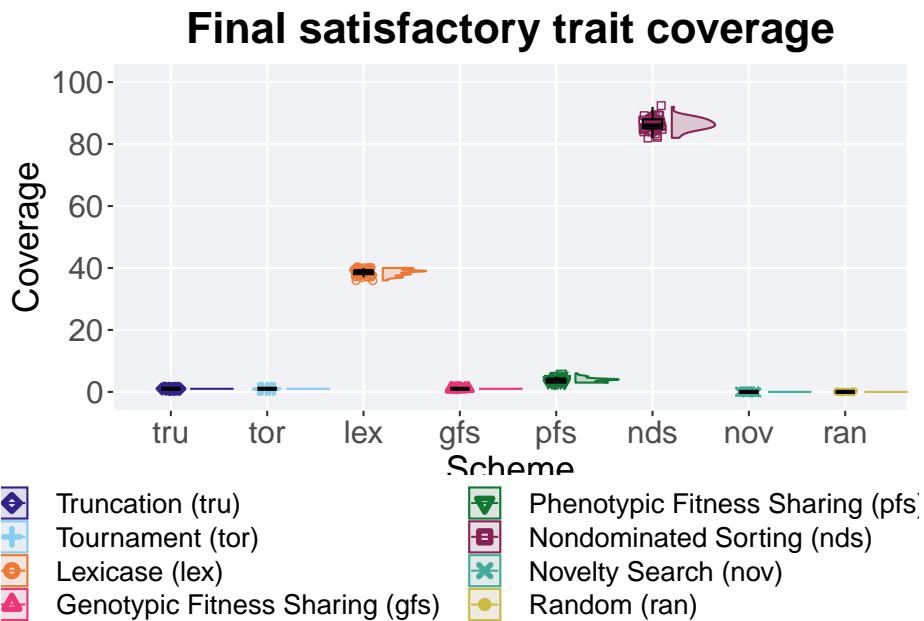
```
end = filter(cc_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = acron, y = pop_uni_obj, color = acron, fill = acron, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_discrete(
  name="Scheme"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Final satisfactory trait coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 4.3.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```

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

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <int> <dbl>
## 1 nds     50     0     82     86  86.4     92     3
## 2 lex     50     0     36     39  38.6     40     1
## 3 pfs     50     0      3      4   3.82      6     1
## 4 gfs     50     0      1      1     1       1     0
## 5 tor     50     0      1      1     1       1     0
## 6 tru     50     0      1      1     1       1     0
## 7 nov     50     0      0      0     0       0     0
## 8 ran     50     0      0      0     0       0     0

```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```

kruskal.test(pop_uni_obj ~ acron, data = end)

##
##  Kruskal-Wallis rank sum test
##
##  data:  pop_uni_obj by acron
##  Kruskal-Wallis chi-squared = 396.7, df = 7, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage in the population at the end of 50,000 generations.

pairwise.wilcox.test(x = end$pop_uni_obj, g = end$acron, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
##  data:  end$pop_uni_obj and end$acron
##
##    nds    lex    pfs    gfs    tor    tru    nov
##  lex <2e-16 -     -     -     -     -     -

```

```

## pfs <2e-16 <2e-16 - - - -
## gfs <2e-16 <2e-16 - - - -
## tor <2e-16 <2e-16 <2e-16 1 - - -
## tru <2e-16 <2e-16 <2e-16 1 1 - -
## nov <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## ran <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 1
##
## P value adjustment method: bonferroni

```

## 4.4 Activation gene coverage

Activation gene coverage analysis.

### 4.4.1 Over time coverage

Activation gene coverage over time.

```

contradictory_objectives = filter(cc_over_time, diagnostic == 'contradictory_objectives')
lines = contradictory_objectives %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill = `Selection\nScheme`, col
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

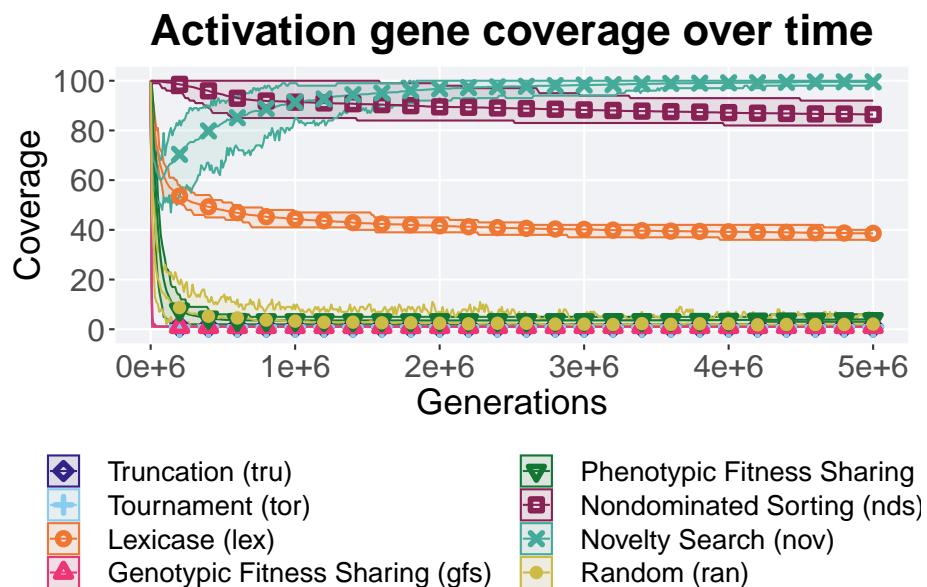
```

```

) +
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 +
guides(
  shape=guide_legend(ncol=2, title.position = "bottom"),
  color=guide_legend(ncol=2, title.position = "bottom"),
  fill=guide_legend(ncol=2, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title=element_blank()
)

```

ot

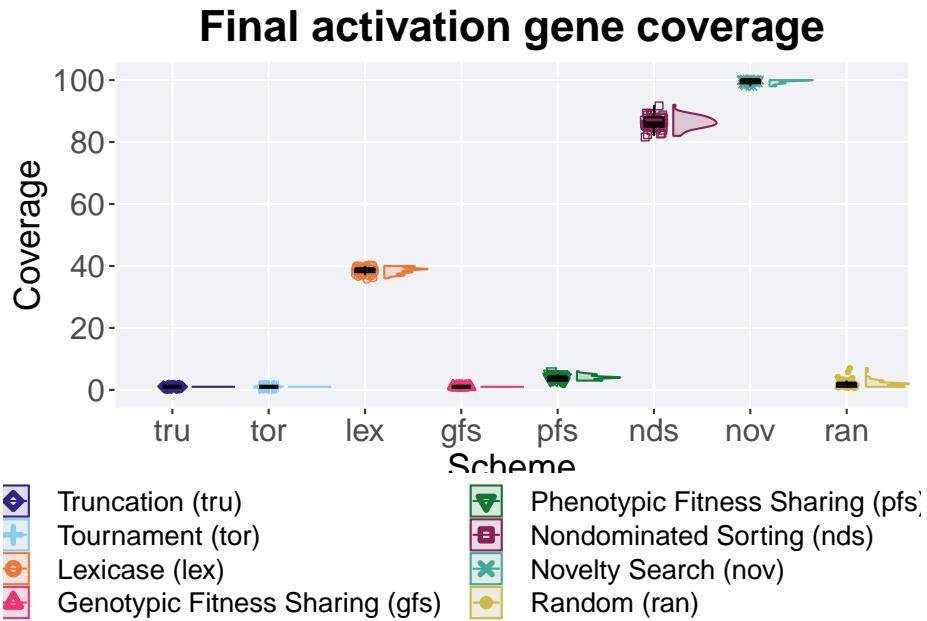


#### 4.4.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
end = filter(cc_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = acron, y = uni_str_pos, color = acron, fill = acron, shape = acron)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
  ggtitle("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```



#### 4.4.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
end$acron = factor(end$acron, levels = c('nov', 'nds', 'lex', 'pfs', 'ran', 'gfs', 'tor'))
group_by(end, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <int> <dbl>
## 1 nov     50     0     98  100  99.6   100     1
## 2 nds     50     0     82   86  86.4    92     3
## 3 lex     50     0     36   39  38.6    40     1
## 4 pfs     50     0      3    4  3.98     6     1
## 5 ran     50     0      1    2  2.06     7     1
## 6 gfs     50     0      1    1    1      1     0
```

```
## 7 tor      50      0      1      1  1      1      0
## 8 tru      50      0      1      1  1      1      0
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage in the population at the end of 50,000 generations.

```
kruskal.test(uni_str_pos ~ acron, data = end)

##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by acron
## Kruskal-Wallis chi-squared = 384.23, df = 7, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage in the population at the end of 50,000 generations.

pairwise.wilcox.test(x = end$uni_str_pos, g = end$acron, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$uni_str_pos and end$acron
##
##      nov     nds     lex     pfs     ran     gfs   tor
## nds < 2e-16 -      -      -      -      -      -
## lex < 2e-16 < 2e-16 -      -      -      -      -
## pfs < 2e-16 < 2e-16 < 2e-16 -      -      -      -
## ran < 2e-16 < 2e-16 < 2e-16 2.9e-12 -      -      -
## gfs < 2e-16 < 2e-16 < 2e-16 < 2e-16 3.0e-10 -      -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 3.0e-10 1      -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 3.0e-10 1      1
##
## P value adjustment method: bonferroni
```

## 4.5 Nondominated sorting split

Here analyze the satisfactory trait coverage and activation gene coverage results for nondominated sorting, nondominated front ranking (no fitness sharing between fronts), and phenotypic fitness sharing.

### 4.5.1 Coverage over time

Satisfactory trait coverage over time.

```
nds <- filter(nds_ot, Sigma == 0.3 & diagnostic == 'contradictory_objectives')
nds$'Selection\nScheme' = 'Nondominated sorting (nds)'
```

```

nfr <- filter(nds_ot, Sigma == 0.0 & diagnostic == 'contradictory_objectives')
nfr$acron = 'nfr'
nfr$'Selection\nScheme' = 'Nondominated front ranking (nfr)'

pfs <- filter(pfs_ot, Sigma == 0.3 & diagnostic == 'contradictory_objectives')
pfs$'Selection\nScheme' = 'Phenotypic fitness sharing (pfs)'

contradictory_objectives <- rbind(nds,nfr,pfs)
contradictory_objectives$`Selection\nScheme` <- factor(contradictory_objectives$`Selection\nScheme`)

lines = contradictory_objectives %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )

## `summarise()` has grouped output by 'Selection Scheme'. You can override using
## the ` `.groups` argument.
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill = `Selection\nScheme`),
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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+

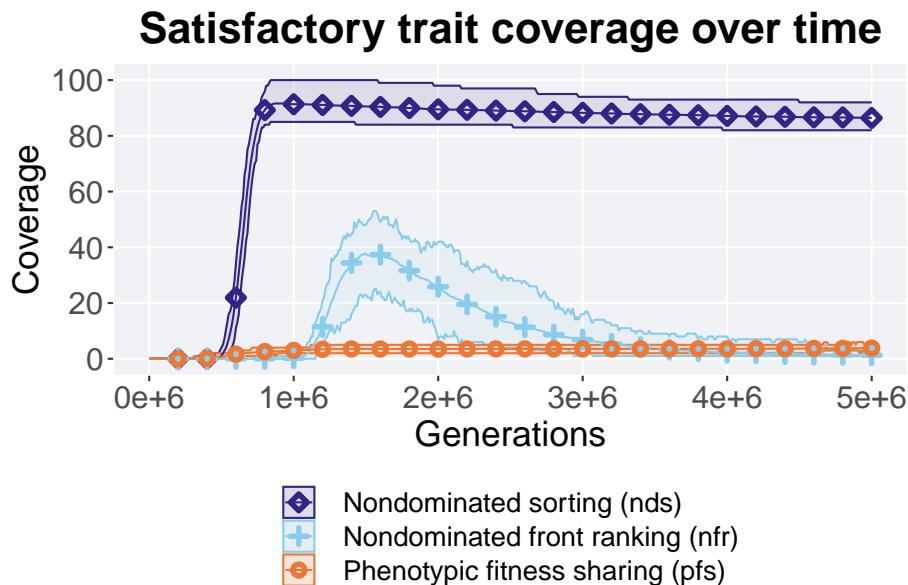
```

```

guides(
  shape=guide_legend(ncol=1, title.position = "bottom"),
  color=guide_legend(ncol=1, title.position = "bottom"),
  fill=guide_legend(ncol=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title=element_blank()
)

```

ot



#### 4.5.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```

nds <- filter(nds_best, diagnostic == 'contradictory_objectives' & col == 'pop_uni_obj' & Sigma ==
nds$'Selection\nScheme' = 'Nondominated sorting (nds)'
nds$acron = 'nds'

nfr <- filter(nds_best, diagnostic == 'contradictory_objectives' & col == 'pop_uni_obj' & Sigma ==
nfr$'Selection\nScheme' = 'Nondominated front ranking (nfr)'
nfr$acron = 'nfr'

```

```

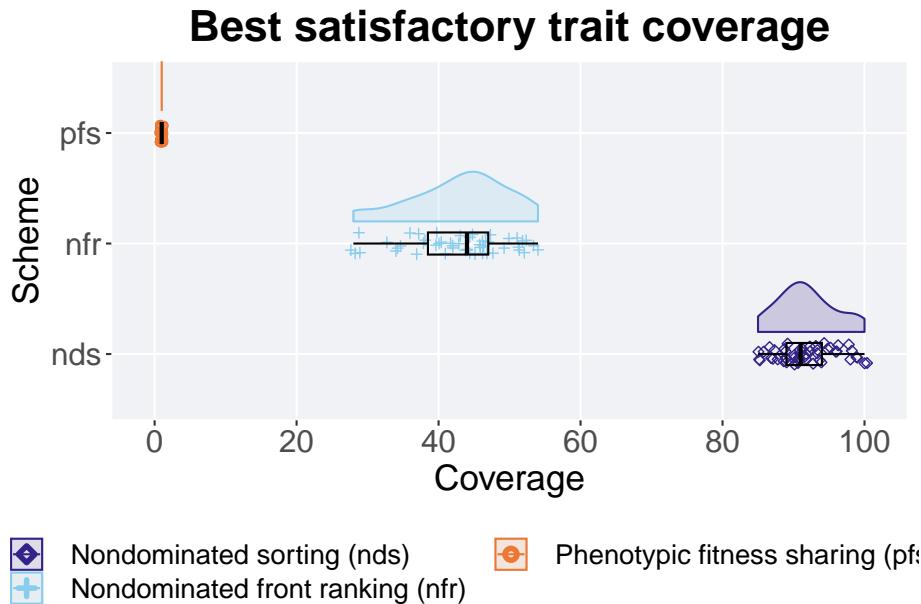
pfs <- filter(pfs_best, diagnostic == 'contradictory_objectives' & col == 'pop_uni_obj')
pfs$'Selection\nScheme' = 'Phenotypic fitness sharing (pfs)'
pfs$acron = 'pfs'

best <- rbind(nds, nfr, pfs)
best$`Selection\nScheme` <- factor(best$`Selection\nScheme`, levels = c('Nondominated '))

plot = ggplot(best, aes(x = acron, y = val, color = acron, fill = acron, shape = acron))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.1) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  guides(fill = "none", color = 'none', shape = 'none') +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100"))
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  coord_flip() +
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(1.3,.32),
  label_size = TSIZE
)

```



#### 4.5.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```
group_by(best, acron) %>%
  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: 3 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <chr>  <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 nds      50       0     85     91    91.8    100     5
## 2 nfr      50       0     28     44    42.8     54    8.5
## 3 pfs      50       0      1      1      1       1     0
```

Kruskal–Wallis test provides evidence of difference among best satisfactory trait coverage throughout 50,000 generations

```

kruskal.test(val ~ acron, data = best)

##
## Kruskal-Wallis rank sum test
##
## data: val by acron
## Kruskal-Wallis chi-squared = 137.61, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
best satisfactory trait coverage throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$acron
##
##      nds     nfr
## nfr <2e-16 -
## pfs <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

### 4.5.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```

nds <- filter(nds_end, Sigma == 0.3 & diagnostic == 'contradictory_objectives')
nds$'Selection\nScheme' = 'Nondominated sorting (nds)'

nfr <- filter(nds_end, Sigma == 0.0 & diagnostic == 'multipath_exploration')
nfr$acron = 'nfr'
nfr$'Selection\nScheme' = 'Nondominated front ranking (nfr)'

pfs <- filter(pfs_end, Sigma == 0.3 & diagnostic == 'multipath_exploration')
pfs$'Selection\nScheme' = 'Phenotypic fitness sharing (pfs)'

end <- rbind(nds, nfr, pfs)
end$acron <- factor(end$acron, levels = c('nds', 'nfr', 'pfs'))

plot = ggplot(end, aes(x = acron, y = pop_uni_obj, color = acron, fill = acron, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous()

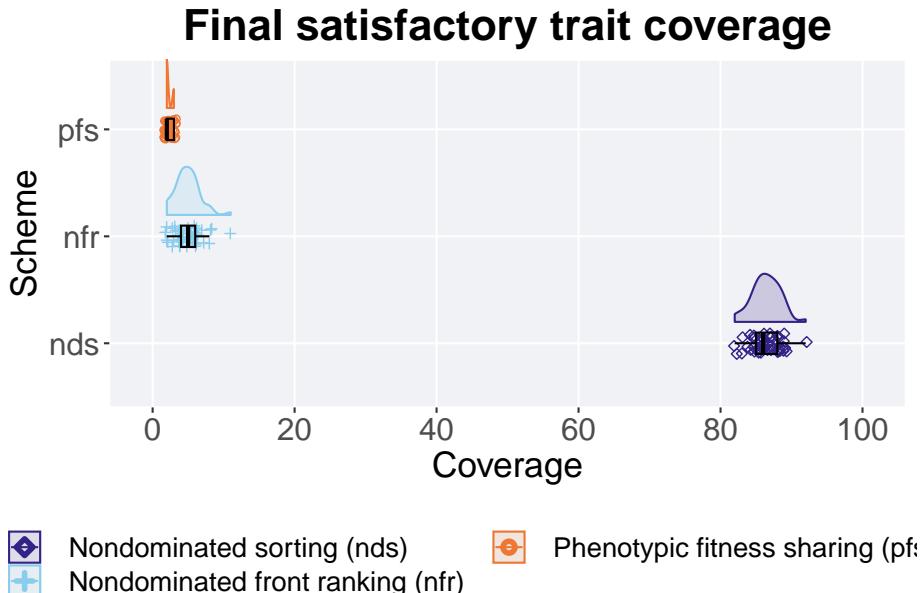
```

```

name="Coverage",
limits=c(-1, 101),
breaks=seq(0,100, 20),
labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_discrete(
  name="Scheme"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
coord_flip() +
p_theme

plot_grid(
  plot +
  ggtitle("Final satisfactory trait coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 4.5.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
group_by(end, acron) %>%
  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: 3 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <int> <dbl>
## 1 nds     50     0     82     86  86.4     92     3
## 2 nfr     50     0      2      5   4.92     11     2
## 3 pfs     50     0      2      2   2.28     3     1
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_uni_obj ~ acron, data = end)

##
## Kruskal-Wallis rank sum test
##
## data: pop_uni_obj by acron
## Kruskal-Wallis chi-squared = 127.35, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_uni_obj and end$acron
##
##   nds     nfr
## nfr <2e-16 -
## pfs <2e-16 1e-14
##
```

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



# Chapter 5

## Multi-path exploration results

Here we present the results for the **best performances** and **activation gene coverage** generated by each selection scheme replicate on the contradictory objectives diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that activation gene coverage values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100.

### 5.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(Pupilometry)
```

### 5.2 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      Patched
## major       4
## minor      2.2
## year       2022
## month      11
## day        10
## svn rev    83330
## language   R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname   Innocent and Trusting

```

## 5.3 Performance

Performance analysis.

### 5.3.1 Over time

Best performance in a population over time.

```

multipath_exploration = filter(cc_over_time, diagnostic == 'multipath_exploration')
lines = multipath_exploration %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = `Selection\nScheme`, fill = `Sel
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),

```

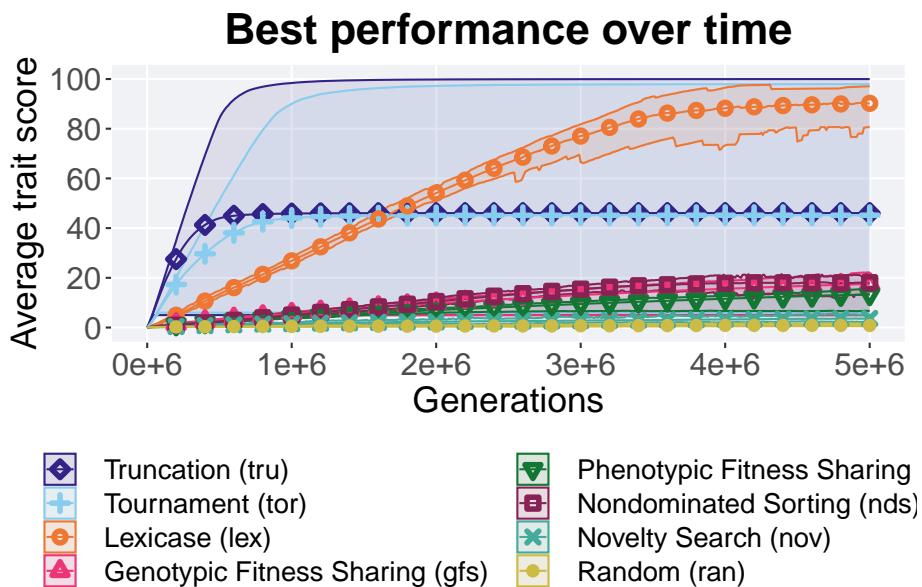
```

breaks=c(0, 10000, 20000, 30000, 40000, 50000),
labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme+
guides(
shape=guide_legend(ncol=2, title.position = "bottom"),
color=guide_legend(ncol=2, title.position = "bottom"),
fill=guide_legend(ncol=2, title.position = "bottom")
) +
theme(
legend.position = "bottom",
legend.box="vertical",
legend.justification="center",
legend.title=element_blank()
)
)

ot

```

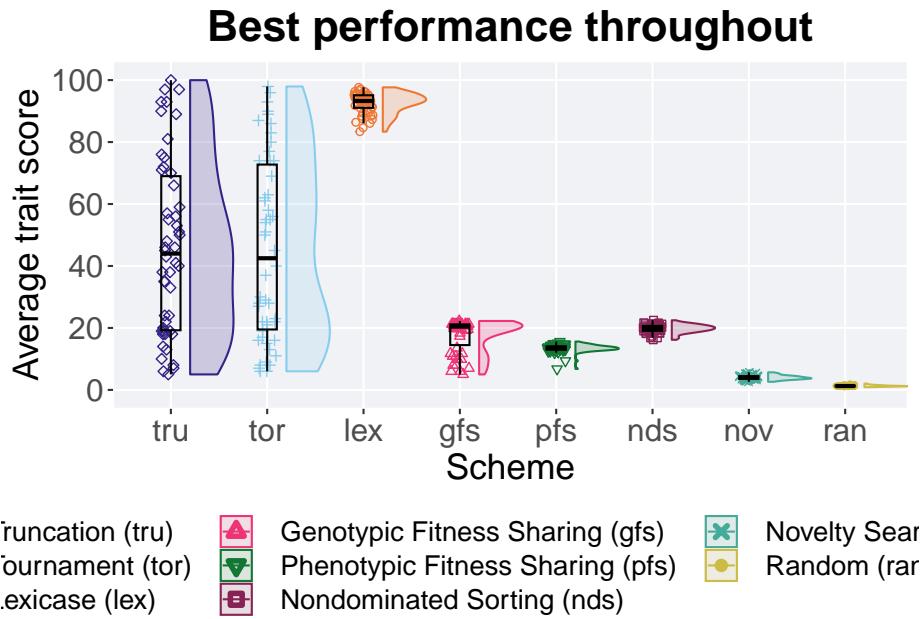


### 5.3.2 Best performance throughout

Best performance throughout 50,000 generations.

```
best = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')
plot = ggplot(best, aes(x = acron, y = val / TRAITS, color = acron, fill = acron, shape = acron))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 101),
    breaks = seq(0, 100, 20),
    labels = c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name = "Scheme"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .55),
  label_size = TSIZE
)
```



#### 5.3.2.1 Stats

Summary statistics for the best performance throughout 50,000 generations.

```
best$acron <- factor(best$acron, levels = c('lex','tor','tru','nds','gfs','pfs','nov','ran'))
group_by(best, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
    IQR = IQR(val, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max     IQR
##   <fct> <int>   <int> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 lex      50       0 8335.  9325.  9251.  9766.   405.
## 2 tor      50       0  600.   4250.  4519.  9794.  5324.
## 3 tru      50       0  500.   4400.  4606.  9997.  4974.
## 4 nds      50       0 1617.   1987.  1976.  2248.   159.
## 5 gfs      50       0  499.   2043.  1760.  2224.   669.
## 6 pfs      50       0  676.   1354.  1339.  1561.   110.
## 7 nov      50       0  262.   389.   401.   568.   86.0
```

```
## 8 ran      50      0   87.0  125.  128.  204.  28.8
```

Kruskal–Wallis test provides evidence of difference among best performances throughout 50,000 generations.

```
kruskal.test(val ~ acron, data = best)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: val by acron  
## Kruskal-Wallis chi-squared = 329.88, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best performance throughout 50,000 generations.

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

```
##  
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction  
##  
## data: best$val and best$acron  
##  
##    lex     tor     tru     nds     gfs     pfs     nov  
## tor 3.0e-13 -      -      -      -      -      -  
## tru 1.1e-11 1.00000 -      -      -      -      -  
## nds < 2e-16 0.00047 0.00027 -      -      -      -  
## gfs < 2e-16 2.3e-05 1.6e-05 1.00000 -      -      -  
## pfs < 2e-16 3.1e-08 6.9e-10 < 2e-16 0.00015 -      -  
## nov < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -  
## ran < 2e-16 -  
##  
## P value adjustment method: bonferroni
```

### 5.3.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

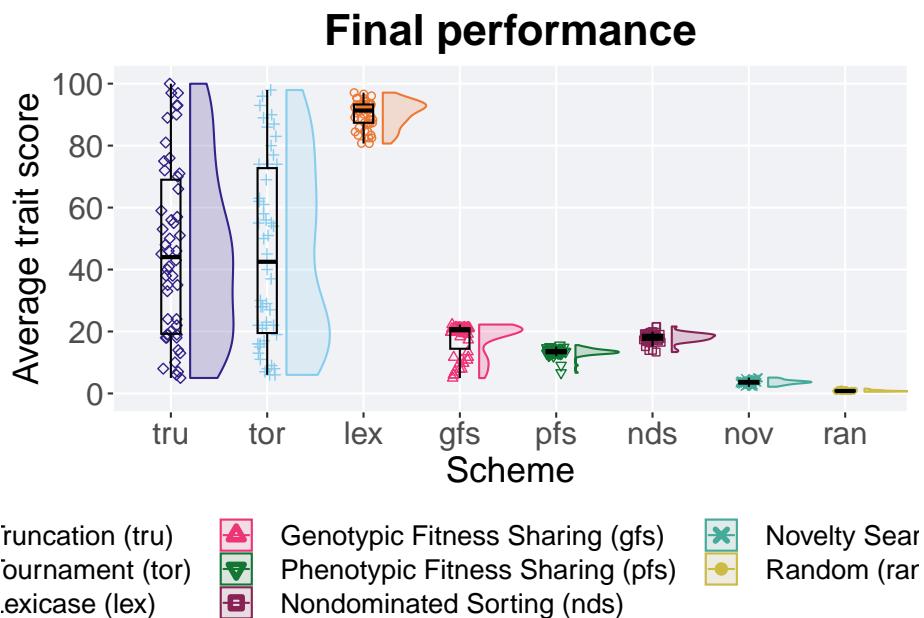
```
end = filter(cc_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = acron, y = pop_fit_max / TRAITS, color = acron, fill = acron))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 101),
    breaks = seq(0, 100, 20),
    labels = c("0", "20", "40", "60", "80", "100")
```

```

) +
scale_x_discrete(
  name="Scheme"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Final performance") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



### 5.3.3.1 Stats

Summary statistics for best performance in the population at the end of 50,000 generations.

```

end$acron <- factor(end$acron, levels = c('lex','tor','tru','nds','gfs','pfs','nov','ran'))
group_by(end, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max, na.rm = TRUE),
    median = median(pop_fit_max, na.rm = TRUE),
    mean = mean(pop_fit_max, na.rm = TRUE),
    max = max(pop_fit_max, na.rm = TRUE),
    IQR = IQR(pop_fit_max, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 lex     50     0 8067.  9132.  9022.  9709.  591.
## 2 tor     50     0  600.  4250.  4519.  9794.  5324.
## 3 tru     50     0  500.  4400.  4606.  9997.  4974.
## 4 nds     50     0 1337.  1812.  1803.  2161.  165.
## 5 gfs     50     0  496.  2039.  1756.  2224.  668.
## 6 pfs     50     0  667.  1350.  1333.  1560.  104.
## 7 nov     50     0  216.   366.   364.   512.   85.9
## 8 ran     50     0  55.3  78.5  84.0  156.   29.9

```

Kruskal–Wallis test provides evidence of difference among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ acron, data = end)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by acron
## Kruskal-Wallis chi-squared = 330.05, df = 7, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best performance in the population at the end of 50,000 generations.

```

pairwise.wilcox.test(x = end$pop_fit_max, g = end$acron, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$acron
##
##      lex    tor    tru    nds    gfs    pfs    nov
## tor 3.9e-12 -     -     -     -     -     -

```

```

## tru 7.1e-11 1.00000 - - - -
## nds < 2e-16 8.2e-05 9.3e-07 - - - -
## gfs < 2e-16 2.2e-05 1.6e-05 1.00000 - - - -
## pfs < 2e-16 3.0e-08 6.6e-10 3.0e-15 0.00015 - - -
## nov < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni

```

## 5.4 Activation gene coverage

Activation gene coverage analysis.

### 5.4.1 Over time coverage

Activation gene coverage over time.

```

multipath_exploration = filter(cc_over_time, diagnostic == 'multipath_exploration')
lines = multipath_exploration %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill = `Selection\nScheme`, col
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

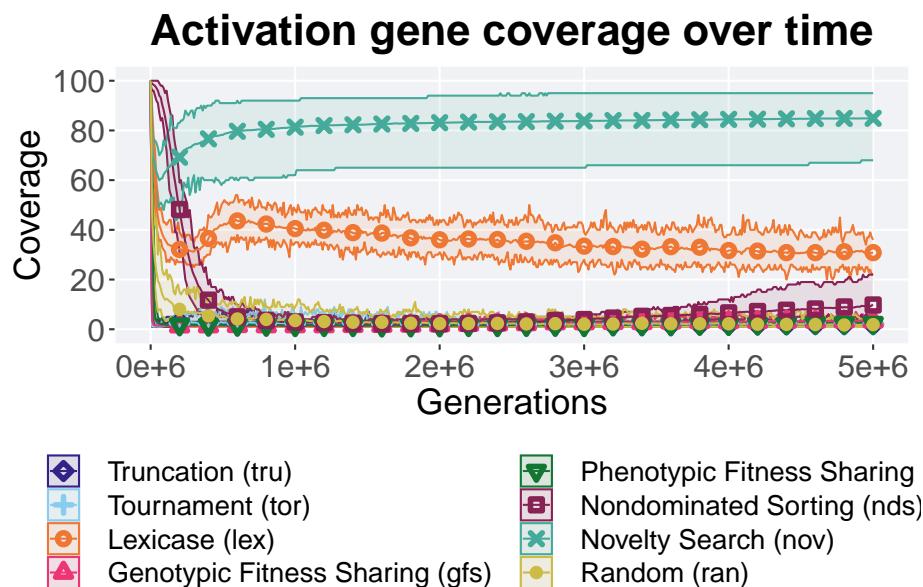
```

```

) +
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 +
guides(
  shape=guide_legend(ncol=2, title.position = "bottom"),
  color=guide_legend(ncol=2, title.position = "bottom"),
  fill=guide_legend(ncol=2, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title=element_blank()
)

```

ot

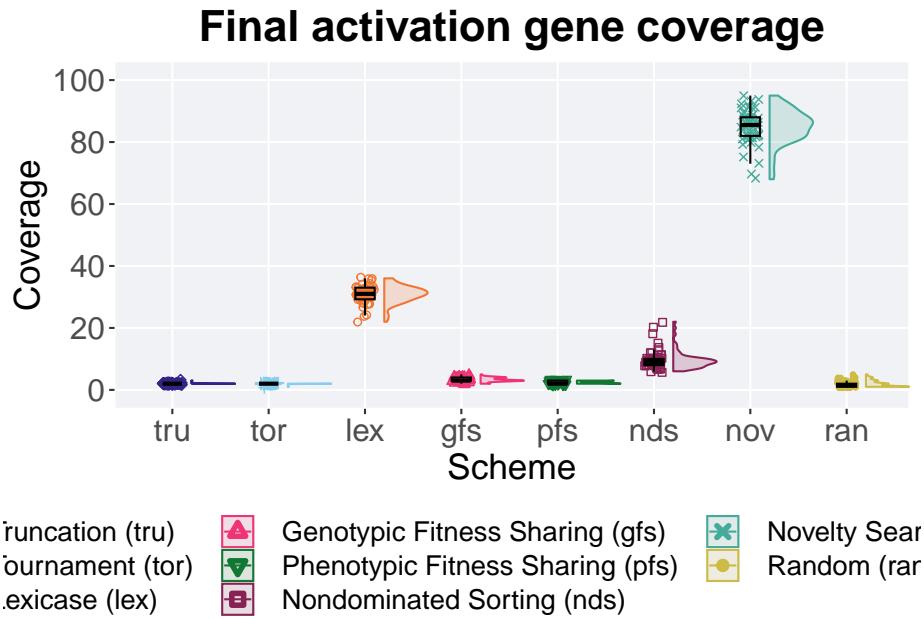


#### 5.4.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
end = filter(cc_end, diagnostic == 'multipath_exploration')
best = ggplot(end, aes(x = acron, y = uni_str_pos, color = acron, fill = acron, shape = acron)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  best +
  ggtitle("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```



#### 5.4.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
end$acron <- factor(end$acron, levels = c('nov', 'lex', 'nds', 'gfs', 'pfs', 'tor', 'tru', 'ran'))
group_by(end, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <int> <dbl>
## 1 nov     50     0     68   85.5  84.9    95    6
## 2 lex     50     0     22    31    30.8    36   3.75
## 3 nds     50     0      6     9    9.76    22    2
## 4 gfs     50     0      2     3    3.24     5    1
## 5 pfs     50     0      2     2    2.46     3    1
## 6 tor     50     0      1     2    1.98     2    0
```

```
## 7 tru      50      0      2      2     2.02      3      0
## 8 ran      50      0      1     1.5    1.86      5      1
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage in the population at the end of 50,000 generations.

```
kruskal.test(uni_str_pos ~ acron, data = end)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by acron
## Kruskal-Wallis chi-squared = 351.29, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage in the population at the end of 50,000 generations.

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

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$uni_str_pos and end$acron
##
##    nov    lex    nds    gfs    pfs    tor    tru
## lex < 2e-16 -      -      -      -      -      -
## nds < 2e-16 < 2e-16 -      -      -      -      -
## gfs < 2e-16 < 2e-16 < 2e-16 -      -      -      -
## pfs < 2e-16 < 2e-16 < 2e-16 7.8e-07 -      -      -
## tor < 2e-16 < 2e-16 < 2e-16 4.2e-16 6.3e-07 -      -
## tru < 2e-16 < 2e-16 < 2e-16 1.4e-15 4.3e-06 1.00000 -
## ran < 2e-16 < 2e-16 < 2e-16 1.1e-08 0.00073 0.20446 0.10598
##
## P value adjustment method: bonferroni
```



# Chapter 6

## Multi-valley crossing results

Here we present the results for the **best performances** and **best gene value** generated by each selection scheme replicate on the multi-valley crossing diagnostic. Best performance found refers to the largest average trait score found in a given population. Best gene value refers to maximum gene value found in the population; this gives us a range of values in [0.0, 100.0].

### 6.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

### 6.2 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         Patched
## major          4
## minor          2.2
## year          2022
```

```

## month      11
## day        10
## svn rev    83330
## language   R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname   Innocent and Trusting

```

## 6.3 Performance

Performance analysis.

### 6.3.1 Over time

Best performance in a population over time.

```

multivalley_crossing = filter(cc_over_time, diagnostic == 'multivalley_crossing')
lines = multivalley_crossing %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = `Selection\nScheme`, fill = Sel
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +

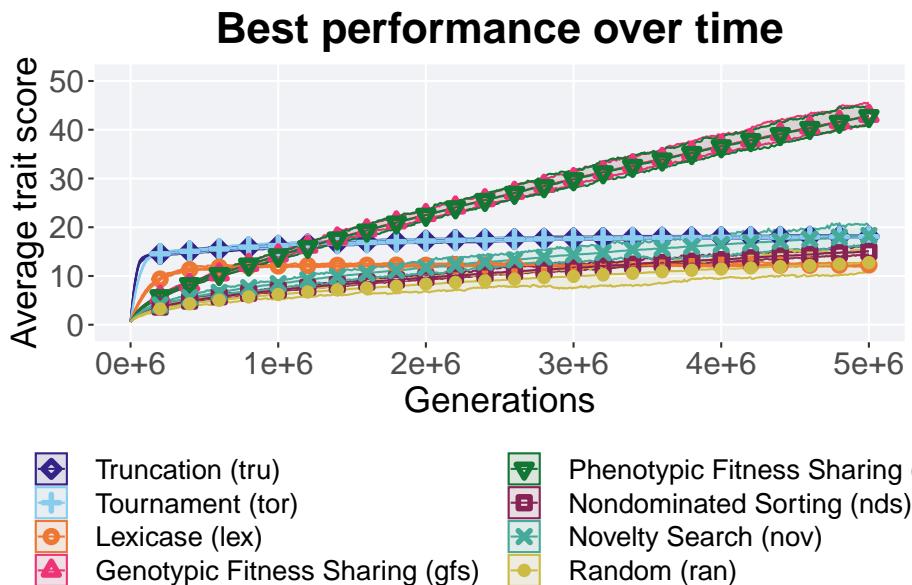
```

```

scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme +
guides(
  shape=guide_legend(ncol=2, title.position = "bottom"),
  color=guide_legend(ncol=2, title.position = "bottom"),
  fill=guide_legend(ncol=2, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title=element_blank()
)

```

ot



### 6.3.2 Best performance throughout

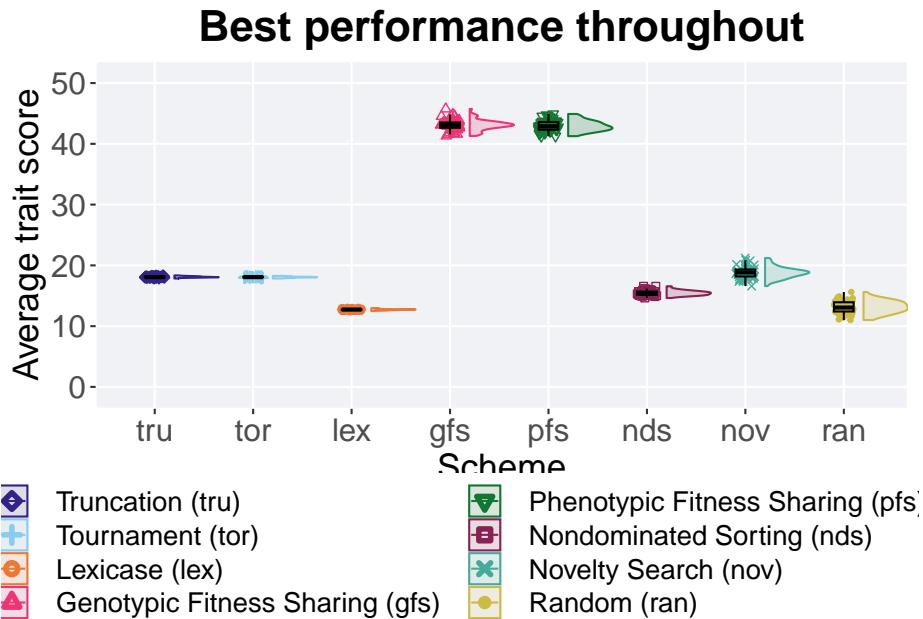
Best performance throughout 50,000 generations.

```

best = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'multivalley_crossing')
plot = ggplot(best, aes(x = acron, y = val / TRAITS, color = acron, fill = acron, shape = acron))

```

```
geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +  
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +  
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +  
  scale_y_continuous(  
    name = "Average trait score",  
    limits = c(-1, 51),  
    breaks = seq(0, 50, 10),  
    labels = c("0", "10", "20", "30", "40", "50")  
  ) +  
  scale_x_discrete(  
    name = "Scheme"  
  ) +  
  scale_shape_manual(values = SHAPE) +  
  scale_colour_manual(values = cb_palette) +  
  scale_fill_manual(values = cb_palette) +  
  p_theme  
  
plot_grid(  
  plot +  
    ggtitle("Best performance throughout") +  
    theme(legend.position = "none"),  
  legend,  
  nrow = 2,  
  rel_heights = c(2, .55),  
  label_size = TSIZE  
)
```



#### 6.3.2.1 Stats

Summary statistics for the best performance throughout 50,000 generations.

```
best$acron <- factor(best$acron, levels = c('pfs', 'gfs', 'nov', 'tor', 'tru', 'nds', 'ran', 'lex'))
group_by(best, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
    IQR = IQR(val, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   acron count na_cnt   min   median   mean   max   IQR
##   <fct> <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 pfs     50      0  4127.  4288.  4296.  4492. 126.
## 2 gfs     50      0  4130.  4311.  4310.  4575.  87.6
## 3 nov     50      0  1659.  1883.  1880.  2121. 114.
## 4 tor     50      0  1784.  1806.  1806.  1827.   9
## 5 tru     50      0  1784.  1807.  1809.  1835. 14.0
## 6 nds     50      0  1463.  1544.  1547.  1656. 54.5
## 7 ran     50      0  1100.  1306.  1314.  1562. 152.
```

```
## # 8 lex      50      0 1248. 1273. 1272. 1296. 11.5
```

Kruskal–Wallis test provides evidence of difference among best performances throughout 50,000 generations.

```
kruskal.test(val ~ acron, data = best)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: val by acron  
## Kruskal-Wallis chi-squared = 369.74, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best performance throughout 50,000 generations.

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

```
##  
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction  
##  
## data: best$val and best$acron  
##  
##    pfs     gfs     nov     tor     tru     nds     ran  
## gfs 1.00   -     -     -     -     -     -  
## nov < 2e-16 < 2e-16 -     -     -     -     -  
## tor < 2e-16 < 2e-16 4.3e-06 -     -     -     -  
## tru < 2e-16 < 2e-16 9.2e-06 1.00   -     -     -  
## nds < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -     -  
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 7.9e-16 -  
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 0.24  
##  
## P value adjustment method: bonferroni
```

### 6.3.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

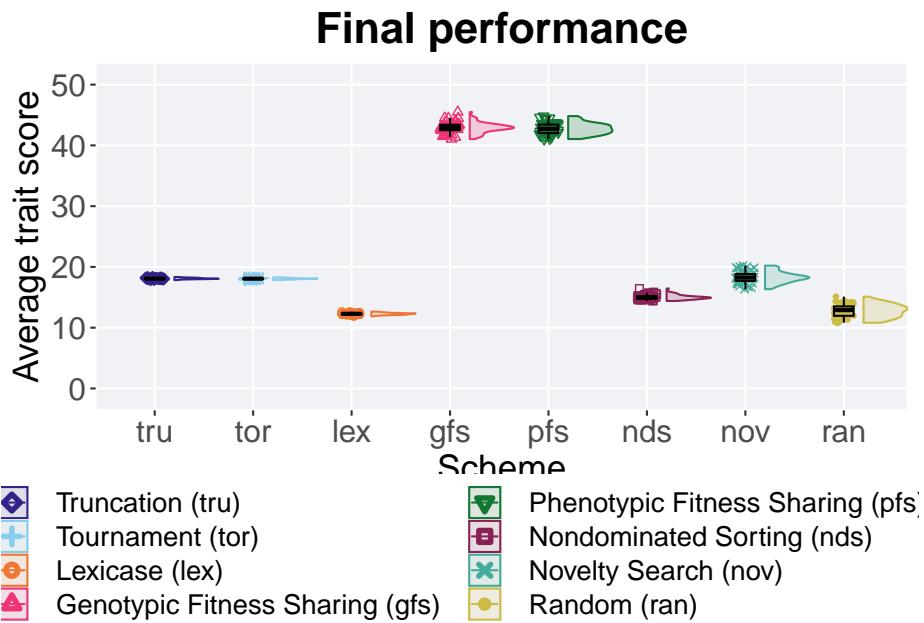
```
end = filter(cc_end, diagnostic == 'multivalley_crossing')  
plot = ggplot(end, aes(x = acron, y = pop_fit_max / TRAITS, color = acron, fill = acron)) +  
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +  
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +  
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +  
  scale_y_continuous(  
    name = "Average trait score",  
    limits = c(-1, 51),  
    breaks = seq(0, 50, 10),  
    labels = c("0", "10", "20", "30", "40", "50")
```

```

) +
  scale_x_discrete(
    name="Scheme"
) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 6.3.3.1 Stats

Summary statistics for best performance in the population at the end of 50,000 generations.

```

end$acron <- factor(end$acron, levels = c('pfs','gfs','nov','tor','tru','nds','ran','le
group_by(end, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max, na.rm = TRUE),
    median = median(pop_fit_max, na.rm = TRUE),
    mean = mean(pop_fit_max, na.rm = TRUE),
    max = max(pop_fit_max, na.rm = TRUE),
    IQR = IQR(pop_fit_max, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 pfs      50     0 4105. 4272. 4275. 4482. 140.
## 2 gfs      50     0 4104. 4291. 4290. 4549. 90.7
## 3 nov      50     0 1634. 1825. 1829. 2021. 114.
## 4 tor      50     0 1784. 1806. 1806. 1827.  9
## 5 tru      50     0 1784. 1807. 1809. 1835  14.0
## 6 nds      50     0 1440. 1496. 1502. 1644. 42.9
## 7 ran      50     0 1081. 1289. 1274. 1511. 154.
## 8 lex      50     0 1186. 1229. 1227. 1265. 19.0

```

Kruskal–Wallis test provides evidence of difference among best performances in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ acron, data = end)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by acron
## Kruskal-Wallis chi-squared = 365.54, df = 7, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best performances in the population at the end of 50,000 generations.

```

pairwise.wilcox.test(x = end$pop_fit_max, g = end$acron, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$acron
##
##   pfs     gfs     nov     tor     tru     nds     ran
##   gfs 1.00   -     -     -     -     -     -

```

```

## nov < 2e-16 < 2e-16 -      -      -      -      -
## tor < 2e-16 < 2e-16 0.49   -      -      -      -      -
## tru < 2e-16 < 2e-16 0.94   1.00   -      -      -      -
## nds < 2e-16 < 2e-16 < 2e-16 < 2e-16 -      -      -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 8.4e-16 -
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 0.53
##
## P value adjustment method: bonferroni

```

## 6.4 Best gene value

Best gene value analysis.

### 6.4.1 Over time gene value

Best gene value over time.

```

multivalley_crossing = filter(cc_over_time, diagnostic == 'multivalley_crossing')

lines = multivalley_crossing %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_max_gene),
    mean = mean(pop_max_gene),
    max = max(pop_max_gene)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill = `Selection\nScheme`, col
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Max gene value",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
scale_shape_manual(values=SHAPE)+  

scale_colour_manual(values = cb_palette) +  

scale_fill_manual(values = cb_palette) +  

ggtitle("Best gene value over time") +  

p_theme+  

guides(  

shape=guide_legend(ncol=2, title.position = "bottom"),  

color=guide_legend(ncol=2, title.position = "bottom"),  

fill=guide_legend(ncol=2, title.position = "bottom")
) +
theme(  

legend.position = "bottom",  

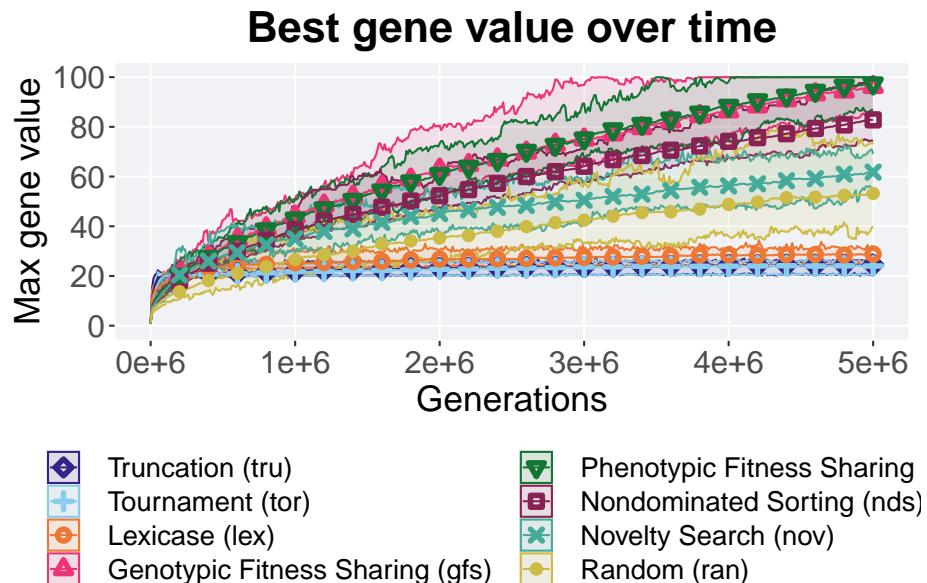
legend.box="vertical",  

legend.justification="center",  

legend.title=element_blank()
)

ot

```

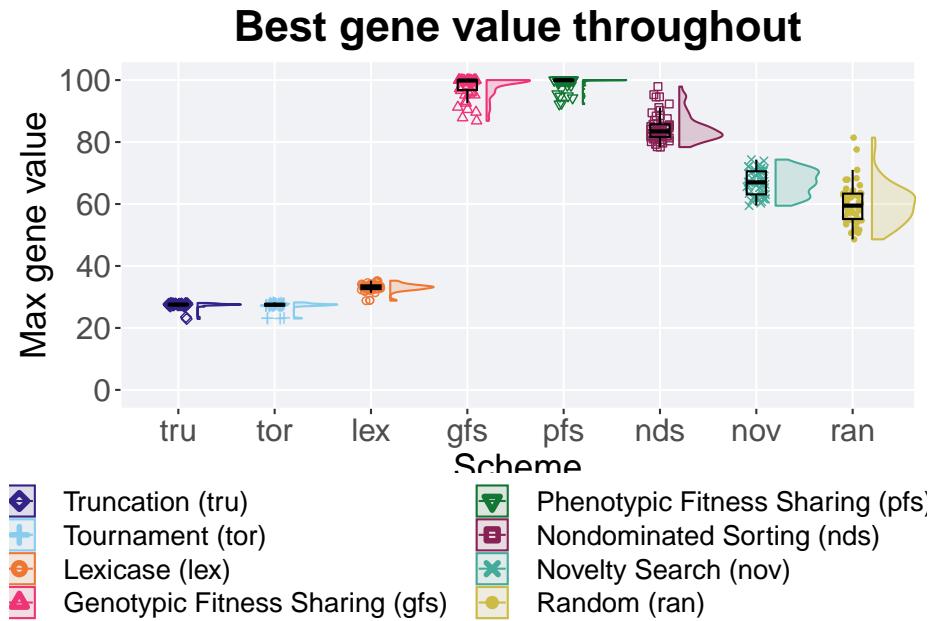


#### 6.4.2 Best gene value throughout

Best gene value throughout 50,000 generations.

```
best = filter(cc_best, col == 'pop_max_gene' & diagnostic == 'multivalley_crossing')
plot = ggplot(best, aes(x = acron, y = val, color = acron, fill = acron, shape = acron)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Max gene value",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
  ggtitle("Best gene value throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```



#### 6.4.2.1 Stats

Summary statistics for the best gene value throughout 50,000 generations.

```
best$acron <- factor(best$acron, levels = c('pfs','gfs','nds','nov','ran','lex','tor','
group_by(best, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
    IQR = IQR(val, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 pfs     50     0  92.3  100.   99.1 100   0.195
## 2 gfs     50     0  86.8  99.9  97.9 100   3.20 
## 3 nds     50     0  78.4  83.4  84.6 97.9  4.07 
## 4 nov     50     0  59.5  67.0  66.9 74.3  7.45 
## 5 ran     50     0  48.6  59.5  60.0 81.4  8.20 
## 6 lex     50     0  28.8  33.2  33.0 35.2  1.20 
## 7 tor     50     0  23.0  27.5  27.2 28.2  0.456
```

```
## 8 tru      50      0  22.9  27.5  27.4  28.1 0.321
```

Kruskal–Wallis test provides evidence of difference among best gene value throughout 50,000 generations.

```
kruskal.test(val ~ acron,data = best)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: val by acron
## Kruskal-Wallis chi-squared = 380.02, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best gene value throughout 50,000 generations.

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

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$acron
##
##    pfs     gfs     nds     nov     ran     lex     tor
## gfs 0.9      -      -      -      -      -      -
## nds 3.1e-16 3.2e-15 -      -      -      -      -
## nov < 2e-16 < 2e-16 < 2e-16 -      -      -      -
## ran < 2e-16 < 2e-16 < 2e-16 2.6e-07 -      -      -
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -      -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.0
##
## P value adjustment method: bonferroni
```

### 6.4.3 End of 50,000 generations

Best gene value in the population at the end of 50,000 generations.

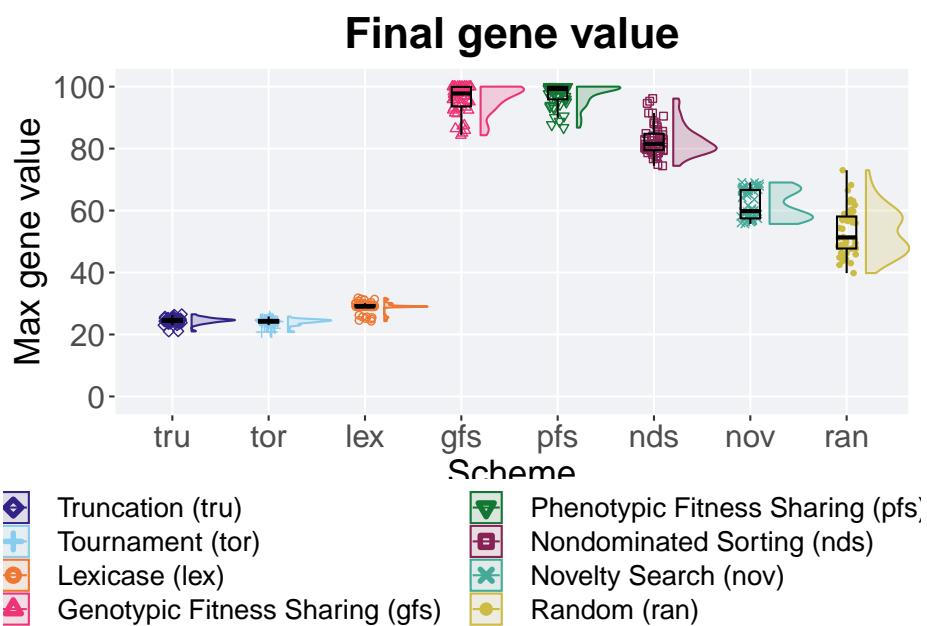
```
end = filter(cc_end, diagnostic == 'multivalley_crossing')
end_p = ggplot(end, aes(x = acron, y = pop_max_gene, color = acron, fill = acron, shape = acron))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Max gene value",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
```

```

) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  end_p +
  ggtitle("Final gene value") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 6.4.3.1 Stats

Summary statistics for best gene values in the population at the end of 50,000 generations.

```

end$acron <- factor(end$acron, levels = c('pfs','gfs','nds','nov','ran','lex','tor','tru'))
group_by(end, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_max_gene)),
    min = min(pop_max_gene, na.rm = TRUE),
    median = median(pop_max_gene, na.rm = TRUE),
    mean = mean(pop_max_gene, na.rm = TRUE),
    max = max(pop_max_gene, na.rm = TRUE),
    IQR = IQR(pop_max_gene, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 pfs      50      0  86.8   99.2  97.4 100.   4.06
## 2 gfs      50      0  84.3   97.8  96.1 100.   6.29
## 3 nds      50      0  74.4   81.5   82.8  96.2  5.30
## 4 nov      50      0  55.7   59.8   61.8  69.1  9.18
## 5 ran      50      0  39.8   51.3   53.2  73.0 10.3
## 6 lex      50      0  24.3   29.0   28.8  31.8  0.501
## 7 tor      50      0  20.8   24.4   24.0  25.8  0.853
## 8 tru      50      0  21.0   24.6   24.5  26.5  0.962

```

Kruskal–Wallis test provides evidence of difference among best gene values in the population at the end of 50,000 generations.

```
kruskal.test(pop_max_gene ~ acron, data = end)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_max_gene by acron
## Kruskal-Wallis chi-squared = 377.84, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best gene values in the population at the end of 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_max_gene and end$acron
##
##   pfs     gfs     nds     nov     ran     lex     tor
##   gfs 1      -      -      -      -      -      -
```

```
## nds 2.5e-15 2.7e-14 -      -      -      -      -
## nov < 2e-16 < 2e-16 < 2e-16 -      -      -      -
## ran < 2e-16 < 2e-16 < 2e-16 5.2e-07 -      -      -
## lex < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -      -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.0e-15 -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 3.8e-14 1
##
## P value adjustment method: bonferroni
```

# Chapter 7

## Truncation selection

We present the results from our parameter sweep on truncation selection. 50 replicates are conducted for each truncation size T parameter value explored.

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

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       Patched
## major         4
## minor        2.2
## year        2022
## month        11
## day          10
## svn rev     83330
## language     R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname    Innocent and Trusting
```

## 7.1 Exploitation rate results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

### 7.1.1 Performance over time

Performance over time.

```
problem <- filter(tru_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

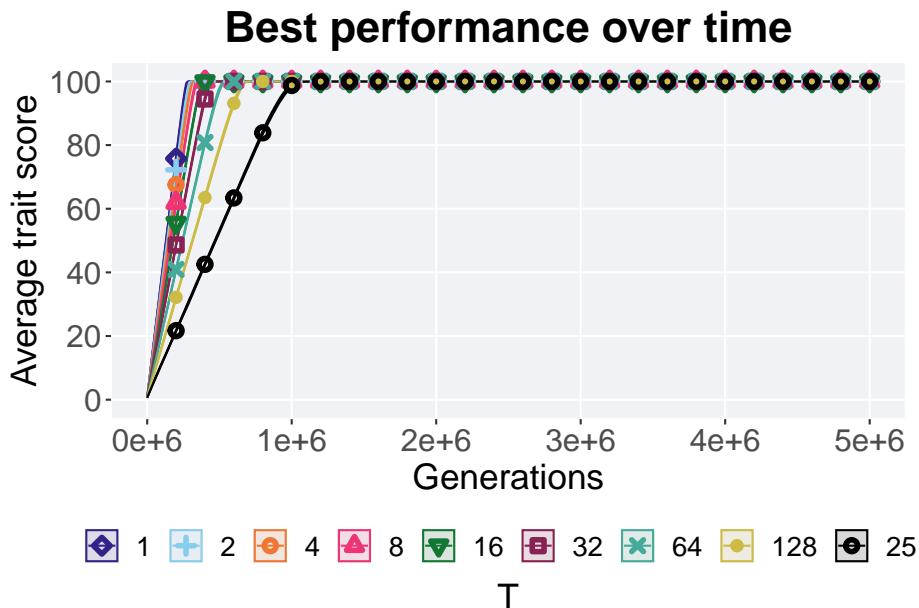
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
```

```

guides(
  shape=guide_legend(nrow=1, title.position = "bottom"),
  color=guide_legend(nrow=1, title.position = "bottom"),
  fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)

```

ot



### 7.1.2 Generation satisfactory solution found

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

```

ssf = filter(tru_ssf, diagnostic == 'exploitation_rate')

plot <- ggplot(ssf, aes(x = T, y = generation, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +

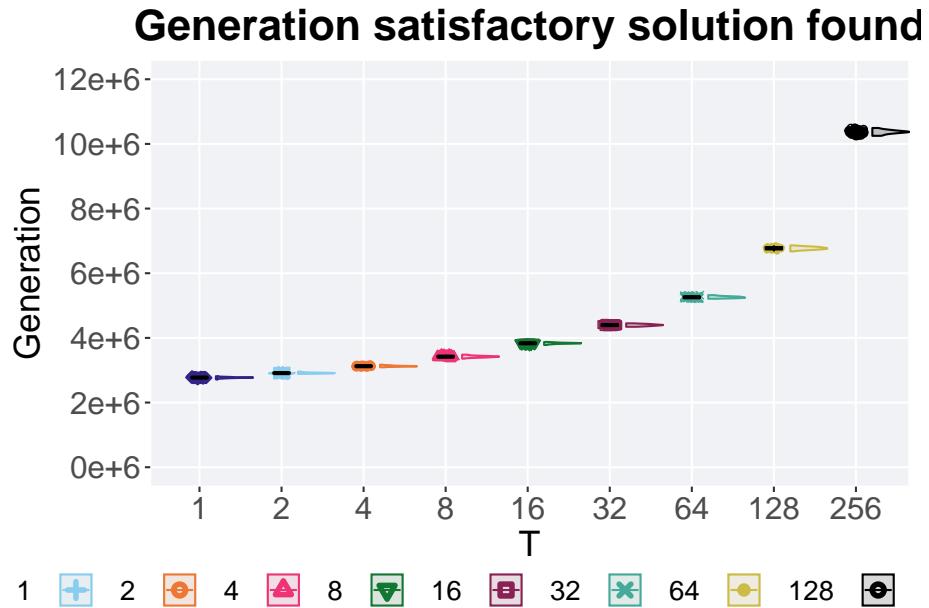
```

```

scale_shape_manual(values=SHAPE) +
scale_y_continuous(
  name="Generation",
  limits=c(0, 12000),
  breaks=c(0, 2000, 4000, 6000, 8000, 10000, 12000),
  labels=c("0e+6", "2e+6", "4e+6", "6e+6", "8e+6", "10e+6", "12e+6")
) +
scale_x_discrete(
  name="T"
) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Generation satisfactory solution found") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



### 7.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```
group_by(ssf, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
    mean = mean(generation, na.rm = TRUE),
    max = max(generation, na.rm = TRUE),
    IQR = IQR(generation, 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  2718  2772  2771.  2814  18.8
## 2 2       50     0  2887  2912  2912.  2955  18.2
## 3 4       50     0  3091  3125  3126.  3171  19
## 4 8       50     0  3357  3420  3421.  3481  34.2
## 5 16      50     0  3781  3834. 3833.  3873  20.8
## 6 32      50     0  4344  4396. 4396.  4450  41.2
## 7 64      50     0  5211  5256. 5259.  5322  38
## 8 128     50     0  6675  6773  6772.  6861  62
## 9 256     50     0 10250 10368. 10369. 10492 73.2
```

Kruskal–Wallis test provides evidence of significant differences among the generation a satisfactory solution is first found.

```
kruskal.test(generation ~ T,data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: generation by T
## Kruskal-Wallis chi-squared = 443.46, df = 8, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the generation a satisfactory solution is first found. .

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

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

```

## data: ssf$generation and ssf$T
##
##      1     2     4     8    16    32    64   128
## 2 <2e-16 - - - - - - -
## 4 <2e-16 <2e-16 - - - - - -
## 8 <2e-16 <2e-16 <2e-16 - - - - - -
## 16 <2e-16 <2e-16 <2e-16 <2e-16 - - - -
## 32 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 - - -
## 64 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 - -
## 128 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 - -
## 256 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 7.2 Ordered exploitation results

Here we present the results for best performances found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

### 7.2.1 Performance over time

Performance over time.

```

problem <- filter(tru_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

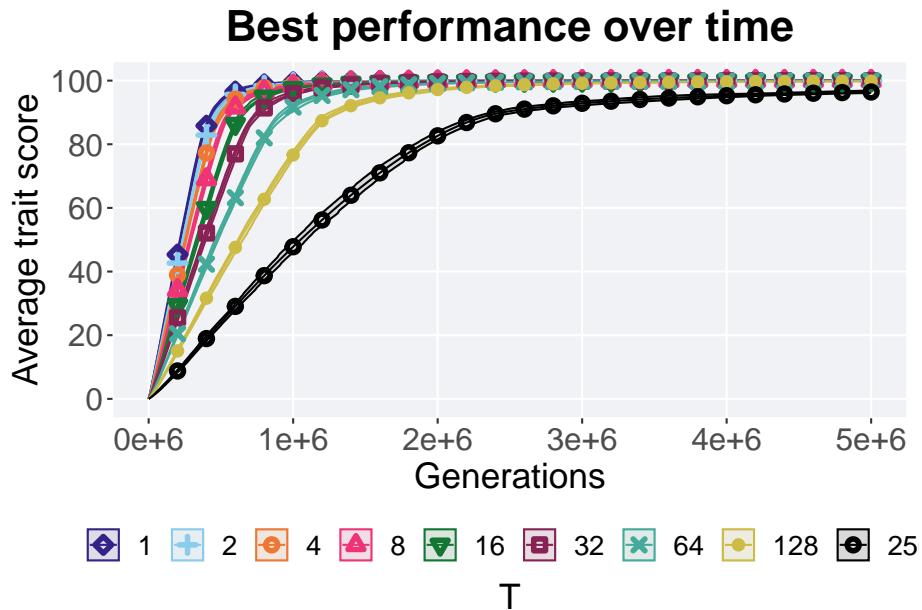
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),

```

```
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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme+
guides(
  shape=guide_legend(nrow=1, title.position = "bottom"),
  color=guide_legend(nrow=1, title.position = "bottom"),
  fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)

ot
```



### 7.2.2 Generation satisfactory solution found

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

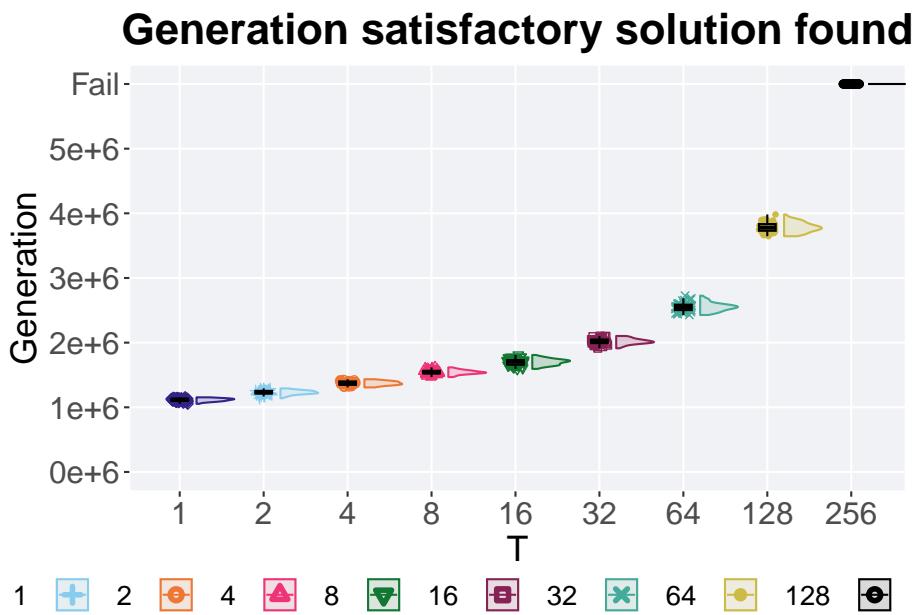
```
ssf = filter(tru_ssf, diagnostic == 'ordered_exploitation')

plot <- ggplot(ssf, aes(x = T, y = generation, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6", "Fail"))
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```

plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



### 7.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```

ssf = filter(ssf, T != 2)

group_by(ssf, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
    mean = mean(generation, na.rm = TRUE),
    max = max(generation, na.rm = TRUE),
    )

```

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

## # A tibble: 8 x 8
##   T     count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1      50    0 10539 11183 11152. 11542  327.
## 2 4      50    0 13078 13694 13719. 14341  519.
## 3 8      50    0 14676 15406. 15424. 16175  396.
## 4 16     50    0 15940 17059 16997. 18094  715.
## 5 32     50    0 19130 20178 20194. 21027  558.
## 6 64     50    0 24279 25498. 25504. 27262  842.
## 7 128    50    0 36460 37786. 37821 39825 1037.
## 8 256    50    0 59999 59999 59999 59999    0
```

Kruskal–Wallis test provides evidence of significant differences among the first generation a satisfactory solution is found throughout the 50,000 generations.

```
kruskal.test(generation ~ T, data = ssf)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: generation by T
## Kruskal-Wallis chi-squared = 393.51, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the first generation a satisfactory solution is found throughout the 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$generation and ssf$T
##
##   1     4     8     16    32    64   128
## 4 <2e-16 -     -     -     -     -     -
## 8 <2e-16 <2e-16 -     -     -     -     -
## 16 <2e-16 <2e-16 <2e-16 -     -     -     -
## 32 <2e-16 <2e-16 <2e-16 <2e-16 -     -     -
## 64 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -     -
## 128 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## 256 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 7.3 Contraditruy objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Satisfactruy trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

### 7.3.1 Satisfactruy trait coverage

Satisfactruy trait coverage analysis.

#### 7.3.1.1 Coverage over time

Satisfactruy trait coverage over time.

```
problem <- filter(tru_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.
points = filter(lines, gen %% 2000 == 0 & gen != 0)

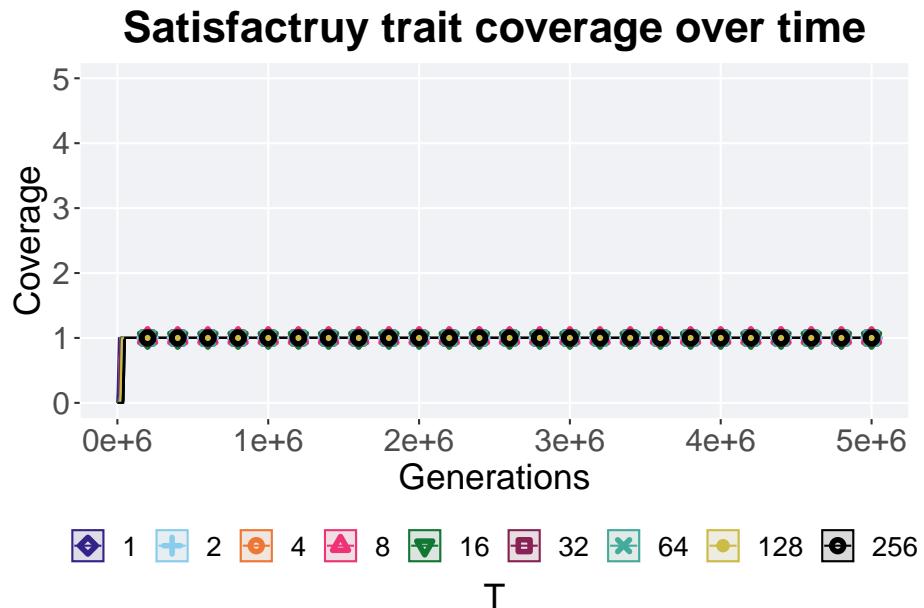
ot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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 +
guides(
  shape=guide_legend(nrow=1, title.position = "bottom"),
  color=guide_legend(nrow=1, title.position = "bottom"),
  fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)
)

ot

```



### 7.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```

best = filter(tru_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives')

plot = ggplot(best, aes(x = T, y = val, color = T, fill = T, shape = T)) +

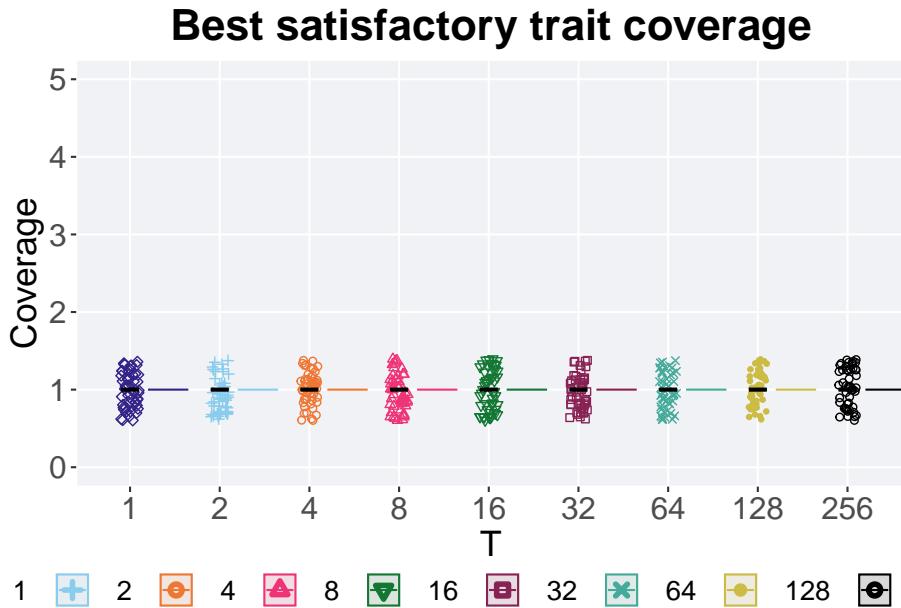
```

```

geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Coverage",
    limits = c(0, 5)
  ) +
  scale_x_discrete(
    name = "T"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .2),
  label_size = TSIZE
)

```



### 7.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```
group_by(best, 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     1     1     1     1     0
## 2 2       50      0     1     1     1     1     0
## 3 4       50      0     1     1     1     1     0
## 4 8       50      0     1     1     1     1     0
## 5 16      50      0     1     1     1     1     0
## 6 32      50      0     1     1     1     1     0
## 7 64      50      0     1     1     1     1     0
## 8 128     50      0     1     1     1     1     0
## 9 256     50      0     1     1     1     1     0
```

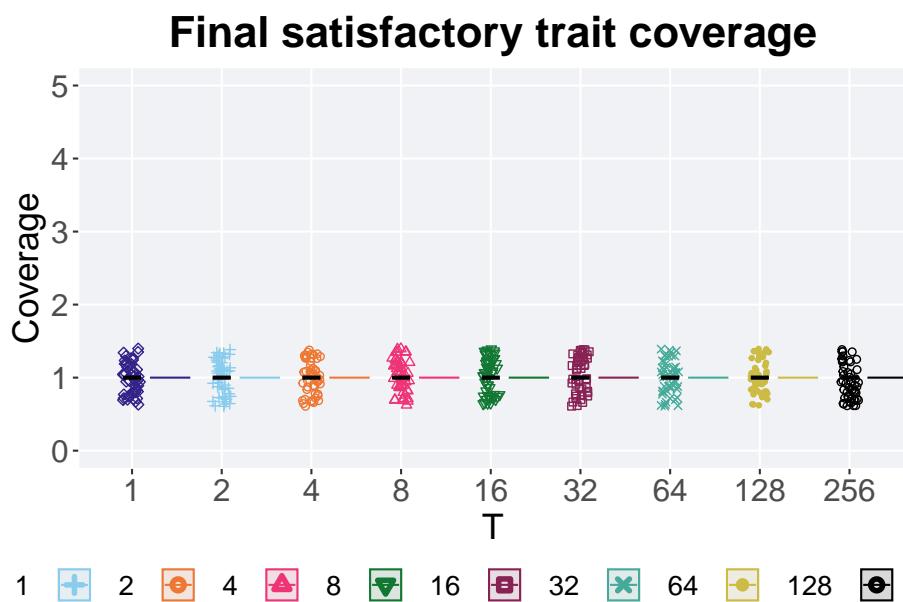
### 7.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
end = filter(tru_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = T, y = pop_uni_obj, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
```

```
p_theme
```

```
plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```



#### 7.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
group_by(end, 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> <dbl> <dbl> <dbl> <int> <dbl>
## 1 1      50     0     1     1     1     1     0
## 2 2      50     0     1     1     1     1     0
## 3 4      50     0     1     1     1     1     0
## 4 8      50     0     1     1     1     1     0
## 5 16     50     0     1     1     1     1     0
## 6 32     50     0     1     1     1     1     0
## 7 64     50     0     1     1     1     1     0
## 8 128    50     0     1     1     1     1     0
## 9 256    50     0     1     1     1     1     0

```

### 7.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

#### 7.3.2.1 Coverage over time

Activation gene coverage over time.

```

problem <- filter(tru_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

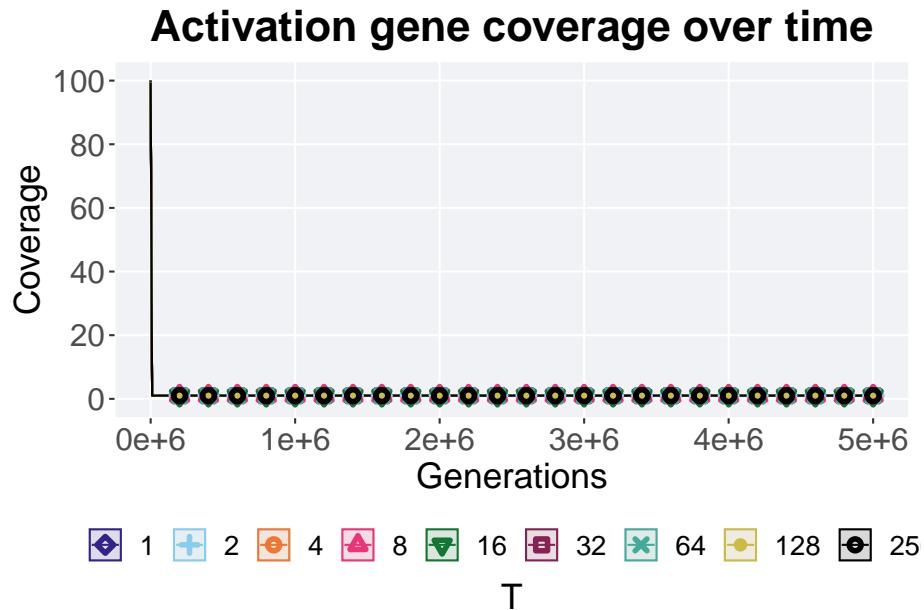
ot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
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 +
guides(
  shape=guide_legend(nrow=1, title.position = "bottom"),
  color=guide_legend(nrow=1, title.position = "bottom"),
  fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)

)
ot
```



#### 7.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

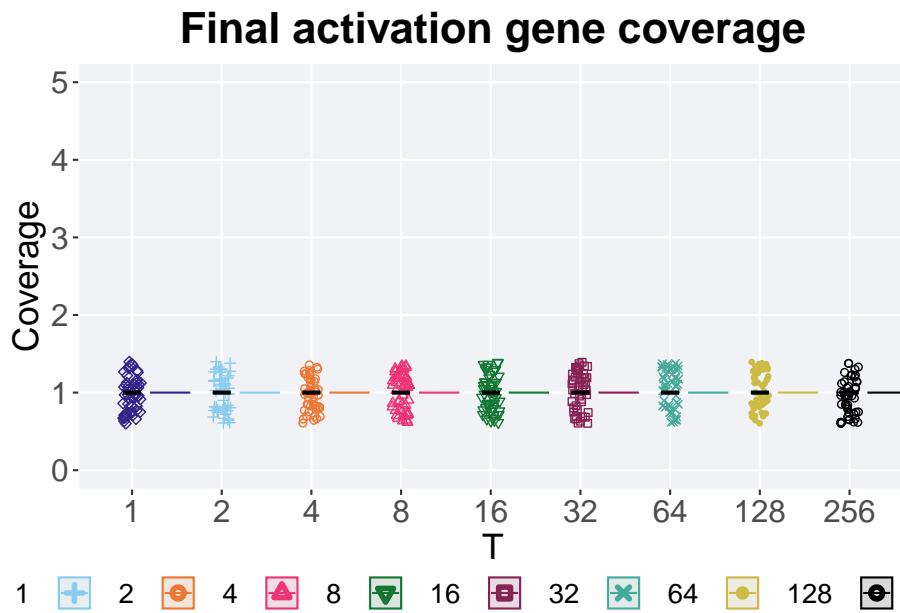
```
end = filter(tru_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
  ggtitle("Final activation gene coverage") +
  theme(legend.position="none"),
```

```

legend,
nrow=2,
rel_heights = c(2,.2),
label_size = TSIZE
)

```



#### 7.3.2.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```

group_by(end, 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> <dbl> <dbl> <dbl> <dbl>
## 1 1        50       0     1     1     1     1     0

```

```

## 2 2      50    0    1    1    1    1    0
## 3 4      50    0    1    1    1    1    0
## 4 8      50    0    1    1    1    1    0
## 5 16     50    0    1    1    1    1    0
## 6 32     50    0    1    1    1    1    0
## 7 64     50    0    1    1    1    1    0
## 8 128    50    0    1    1    1    1    0
## 9 256    50    0    1    1    1    1    0

```

## 7.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

### 7.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

#### 7.4.1.1 Performance over time

Performance over time.

```

problem <- filter(tru_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

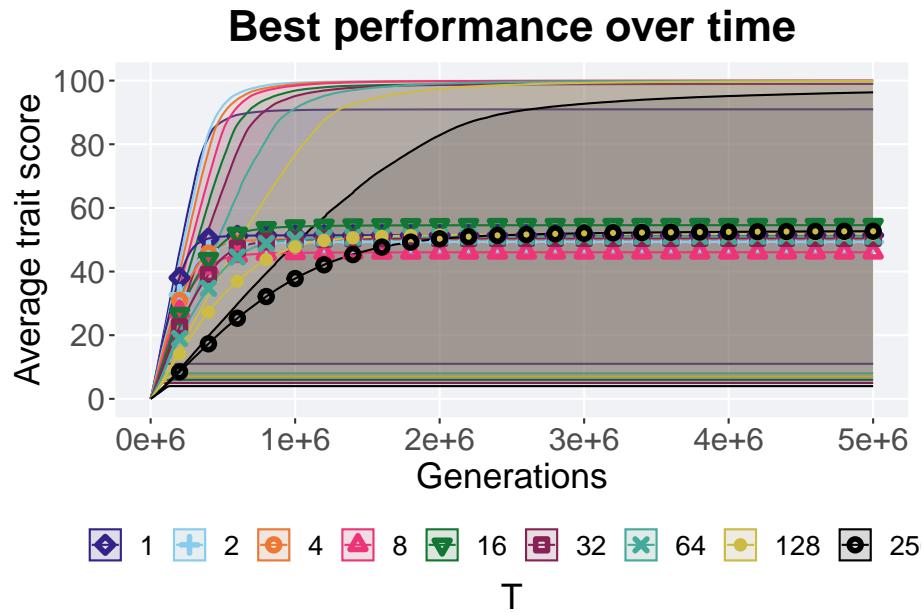
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",

```

```
limits=c(-1, 101),
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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme +
guides(
  shape=guide_legend(nrow=1, title.position = "bottom"),
  color=guide_legend(nrow=1, title.position = "bottom"),
  fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)

ot
```



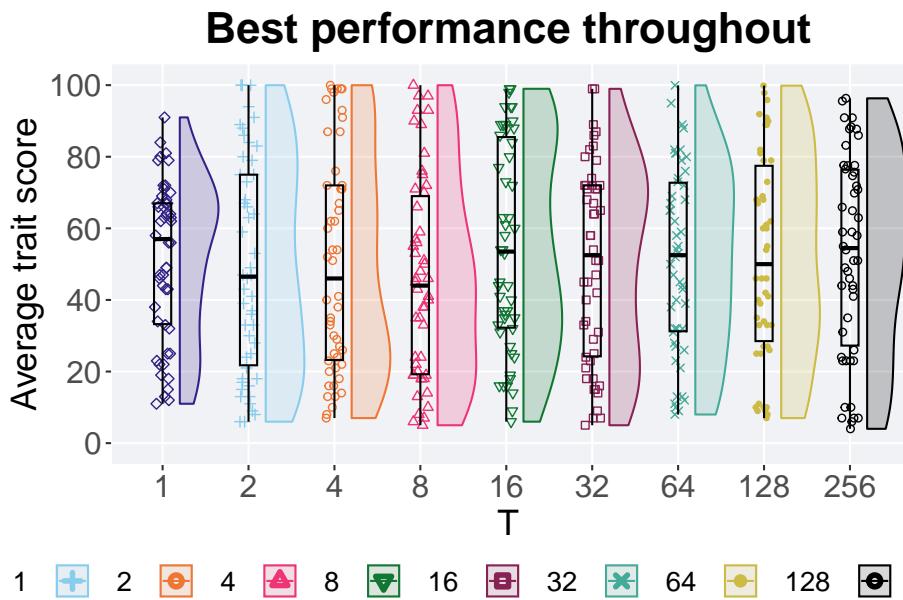
#### 7.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(tru_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = T, y = val / TRAITS, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100"))
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```
plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```



#### 7.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```
group_by(best, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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 11    57.0  51.4  91.0  33.7
## 2 2       50     0 6     46.5  49.3  100.   53.2
## 3 4       50     0 7.00  46.0  50.4  100.   48.7
## 4 8       50     0 5     44.0  46.1  100.   49.7
## 5 16      50     0 6     53.5  54.6  99.0  53.2
## 6 32      50     0 5     52.5  50.4  99.0  47.7
## 7 64      50     0 8.00  52.5  51.1  99.9  41.5
## 8 128     50     0 7     50.0  51.8  99.9  49.0
## 9 256     50     0 4     54.5  52.7  96.3  49.1
```

Kruskal–Wallis test provides evidence of no statistical differences among the best performing solution found throughout 50,000 generations.

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

##
##  Kruskal-Wallis rank sum test
##
## data: val by T
## Kruskal-Wallis chi-squared = 2.7539, df = 8, p-value = 0.9488
```

#### 7.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

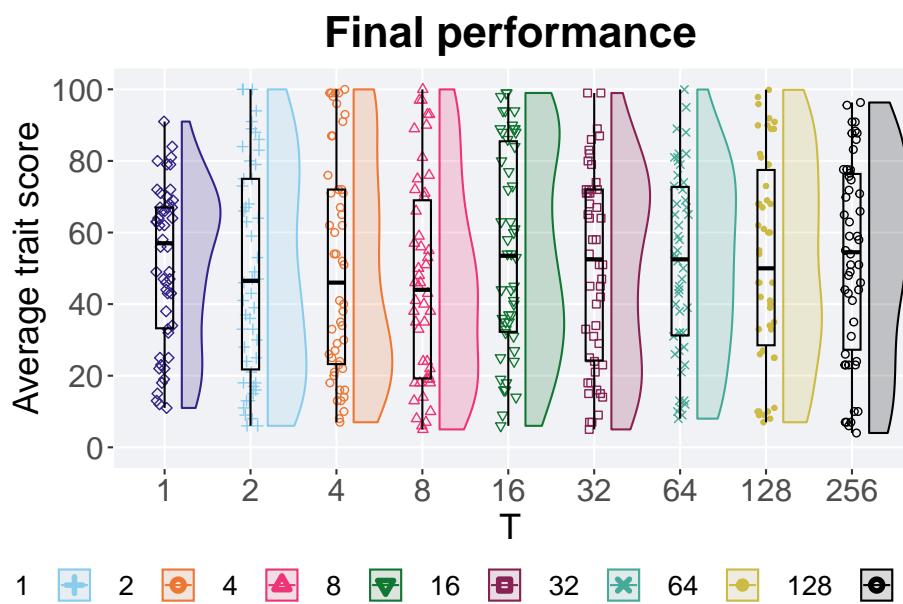
```
end = filter(tru_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = T, y = pop_fit_max / TRAITS, color = T, fill = T, shape = T))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 101),
    breaks = seq(0, 100, 20),
    labels = c("0", "20", "40", "60", "80", "100"))
  ) +
  scale_x_discrete(
    name = "T")
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
```

```

plot +
  ggtitle("Final performance") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 7.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```

group_by(end, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, 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 11    57.0  51.4  91.0  33.7
## 2 2      50     0  6    46.5  49.3 100.  53.2
## 3 4      50     0 7.00  46.0  50.4 100.  48.7
## 4 8      50     0  5    44.0  46.1 100.  49.7
## 5 16     50     0  6    53.5  54.6 99.0  53.2
## 6 32     50     0  5    52.5  50.4 99.0  47.7
## 7 64     50     0 8.00  52.5  51.1 99.9  41.5
## 8 128    50     0  7    50.0  51.8 99.9  49.0
## 9 256    50     0  4    54.5  52.7 96.3  49.1
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ T, data = end)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by T
## Kruskal-Wallis chi-squared = 2.7539, df = 8, p-value = 0.9488
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance in the population at the end of 50,000 generations.

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

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$T
##
##      1 2 4 8 16 32 64 128
## 2  1 - - - - - - -
## 4  1 1 - - - - - -
## 8  1 1 1 - - - - -
## 16 1 1 1 1 - - - -
## 32 1 1 1 1 1 - - -
## 64 1 1 1 1 1 1 - -
## 128 1 1 1 1 1 1 1 -
## 256 1 1 1 1 1 1 1 1
##
## P value adjustment method: bonferroni
```

### 7.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

#### 7.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(tru_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  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.

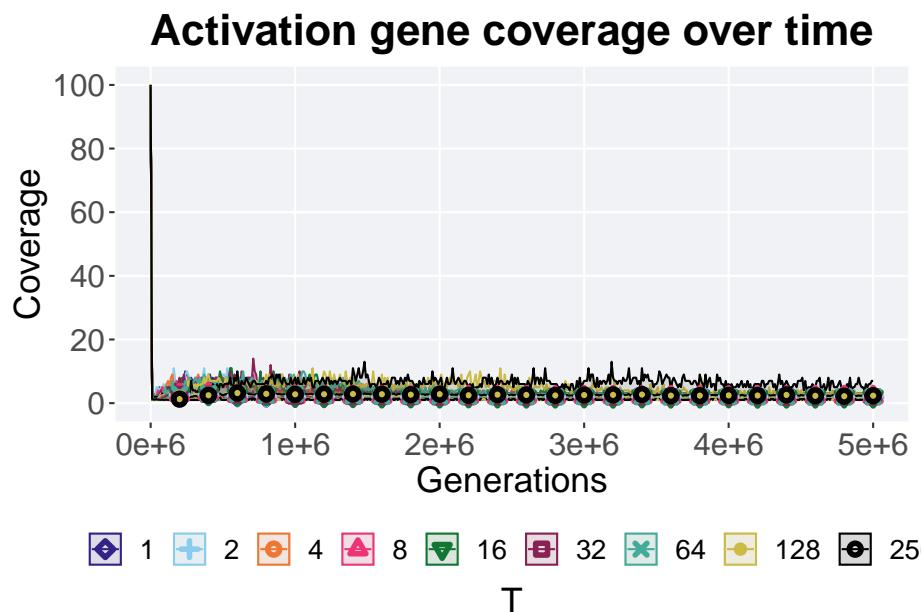
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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 +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )
```

```

    fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)
ot

```



#### 7.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```

end = filter(tru_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 10),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  )

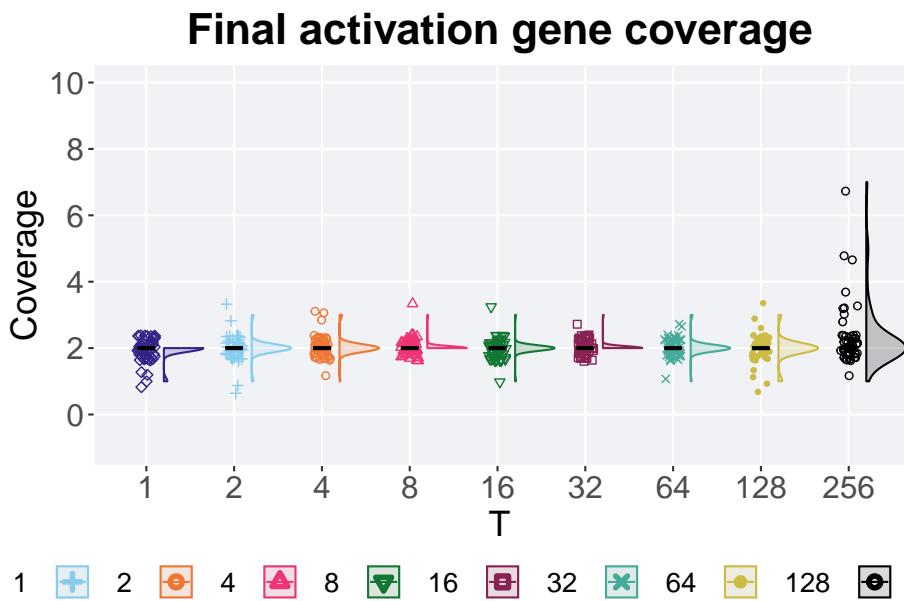
```

```

) +
  scale_x_discrete(
    name="T"
) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 7.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```

group_by(end, 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> <dbl> <dbl> <int> <dbl>
## 1 1        50     0     1     2  1.92     2     0
## 2 2        50     0     1     2     2     3     0
## 3 4        50     0     1     2  2.04     3     0
## 4 8        50     0     2     2  2.02     3     0
## 5 16       50     0     1     2     2     3     0
## 6 32       50     0     2     2  2.02     3     0
## 7 64       50     0     1     2  2.02     3     0
## 8 128      50     0     1     2  1.98     3     0
## 9 256      50     0     1     2  2.34     7     0

```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

```

kruskal.test(uni_str_pos ~ T, data = end)

##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 20.807, df = 8, p-value = 0.007679

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the activation gene coverage in the population at the end of 50,000 generations.

```

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

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$T
##
##      1     2     4     8     16    32    64   128
## 2 1.000 -    -    -    -    -    -    -    -

```

```
## 4  1.000 1.000 -    -    -    -    -  
## 8  0.911 1.000 1.000 -    -    -    -  
## 16 1.000 1.000 1.000 1.000 -    -    -  
## 32 0.911 1.000 1.000 1.000 1.000 -    -  
## 64 1.000 1.000 1.000 1.000 1.000 1.000 -  
## 128 1.000 1.000 1.000 1.000 1.000 1.000 1.000  
## 256 0.047 0.915 1.000 0.887 0.569 0.887 1.000 0.866  
##  
## P value adjustment method: bonferroni
```



# Chapter 8

## Tournament selection

We present the results from our parameter sweep on tournament selection. 50 replicates are conducted for each tournament size T parameter value explored.

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

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       Patched
## major         4
## minor        2.2
## year        2022
## month        11
## day          10
## svn rev     83330
## language     R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname    Innocent and Trusting
```

## 8.1 Exploitation rate results

Here we present the results for **best performances** found by each tournament selection value replicate on the exploitation rate diagnostic.

### 8.1.1 Performance over time

Performance over time.

```
problem <- filter(tor_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

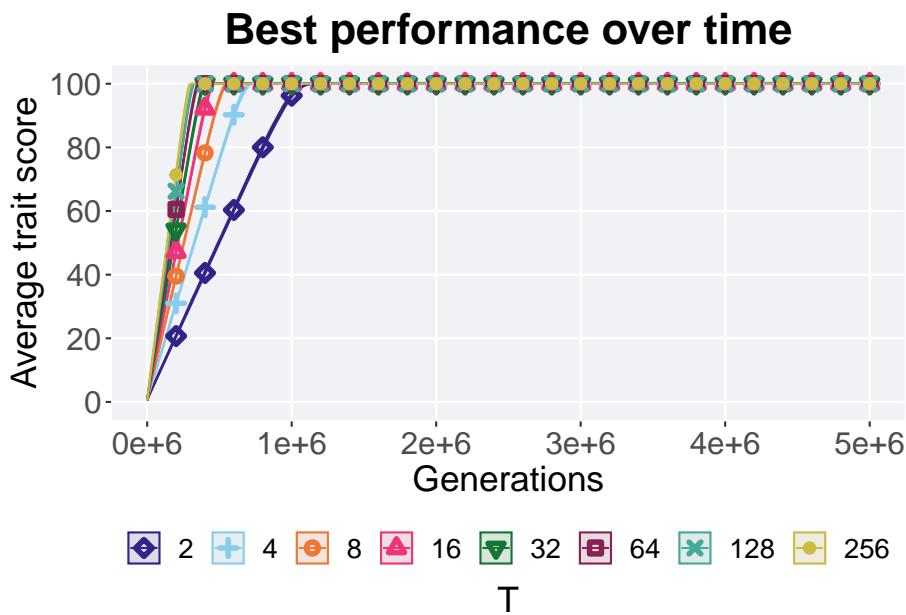
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )
}
```

```

color=guide_legend(nrow=1, title.position = "bottom"),
fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)

```

ot



#### 8.1.2 Generation satisfactory solution found

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

```

ssf = filter(ssf, diagnostic == 'exploitation_rate')

plot <- ggplot(ssf, aes(x = T, y = generation, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous()

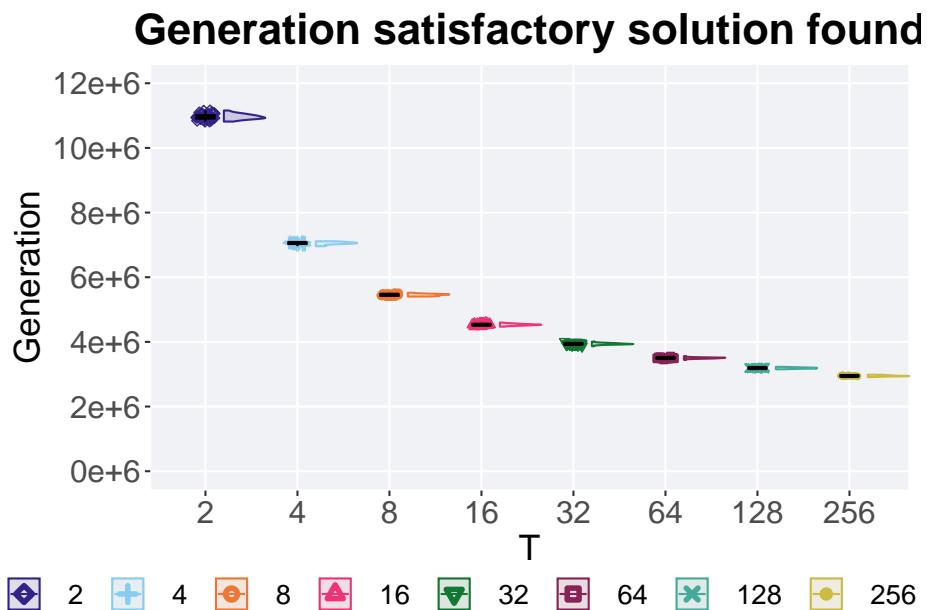
```

```

name="Generation",
limits=c(0, 12000),
breaks=c(0, 2000, 4000, 6000, 8000, 10000, 12000),
labels=c("0e+6", "2e+6", "4e+6", "6e+6", "8e+6", "10e+6", "12e+6")
) +
scale_x_discrete(
  name="T"
) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Generation satisfactory solution found") +
  theme(legend.position="none"),
legend,
nrow=2,
rel_heights = c(2,.2),
label_size = TSIZE
)

```



### 8.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
group_by(ssf, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
    mean = mean(generation, na.rm = TRUE),
    max = max(generation, na.rm = TRUE),
    IQR = IQR(generation, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   T      count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2        50     0 10815 10955 10966. 11157 102.
## 2 4        50     0  6960  7059  7052.  7113  48.8
## 3 8        50     0  5403  5457  5453.  5519  51.8
## 4 16       50     0  4471  4530. 4532.  4597  33.8
## 5 32       50     0  3865  3938  3939.  4009  31.8
## 6 64       50     0  3446  3504. 3502.  3556  24.2
## 7 128      50     0  3152  3192  3190.  3228  26.8
## 8 256      50     0  2912  2946. 2950.  2985  24.8
```

Kruskal–Wallis test provides evidence of significant differences among the generation a satisfactory solution is first found.

```
kruskal.test(generation ~ T,data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: generation by T
## Kruskal-Wallis chi-squared = 392.77, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the generation a satisfactory solution is first found. .

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$generation and ssf$T
```

```

##  

##      2      4      8     16     32     64    128  

## 4 <2e-16 - - - - - -  

## 8 <2e-16 <2e-16 - - - - - -  

## 16 <2e-16 <2e-16 <2e-16 - - - - - -  

## 32 <2e-16 <2e-16 <2e-16 <2e-16 - - - -  

## 64 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 - -  

## 128 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -  

## 256 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16  

##  

## P value adjustment method: bonferroni

```

## 8.2 Ordered exploitation results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

### 8.2.1 Performance over time

Performance over time.

```

problem <- filter(tor_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

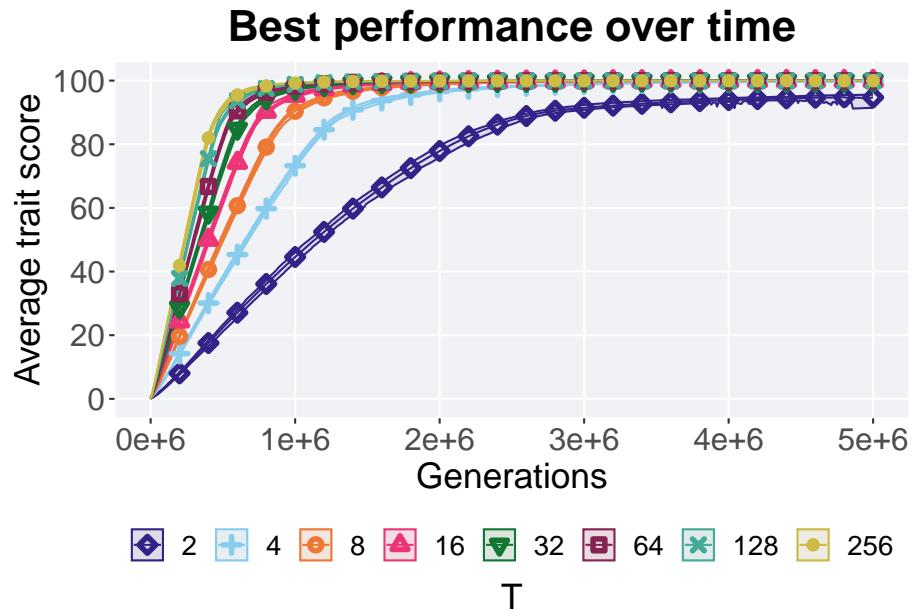
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape = T,
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6"))

) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.justification="center",
    legend.title.align=0.5
  )
)
ot
```



### 8.2.2 Generation satisfactory solution found

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

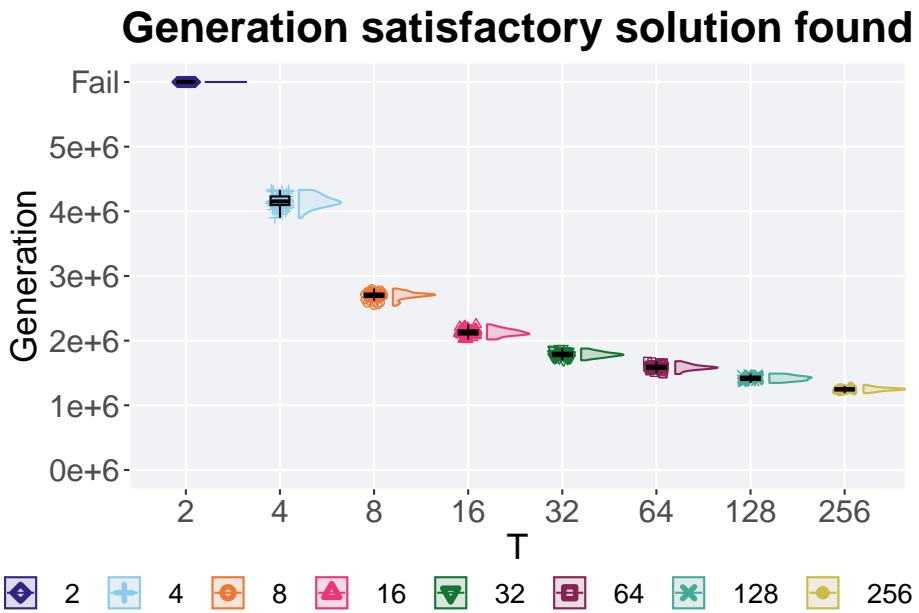
```
ssf = filter(tor_ssf, diagnostic == 'ordered_exploitation')

plot <- ggplot(ssf, aes(x = T, y = generation, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6", "Fail"))
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```

plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



### 8.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```

ssf = filter(ssf, T != 2)

group_by(ssf, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
    mean = mean(generation, na.rm = TRUE),
    max = max(generation, na.rm = TRUE),
    )

```

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

## # A tibble: 7 x 8
##   T      count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl>
## 1 4       50     0 38991 41526 41554. 43315 1332.
## 2 8       50     0 25421 27072. 26970. 28050  528.
## 3 16      50     0 20187 21202. 21249. 22514  638
## 4 32      50     0 16961 17854. 17873. 18782  523.
## 5 64      50     0 14837 15839 15872. 16821  489
## 6 128     50     0 13469 14214. 14191. 14903  525.
## 7 256     50     0 11883 12490 12485. 13162  317.
```

Kruskal–Wallis test provides evidence of significant differences among the first generation a satisfactory solution is found throughout the 50,000 generations.

```
kruskal.test(generation ~ T, data = ssf)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: generation by T
## Kruskal-Wallis chi-squared = 341.85, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the first generation a satisfactory solution is found throughout the 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$generation and ssf$T
##
##      4      8      16      32      64      128
## 8 <2e-16 -      -      -      -      -
## 16 <2e-16 <2e-16 -      -      -      -
## 32 <2e-16 <2e-16 <2e-16 -      -      -
## 64 <2e-16 <2e-16 <2e-16 <2e-16 -      -
## 128 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## 256 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 8.3 Contradictory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

### 8.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

#### 8.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
problem <- filter(tor_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

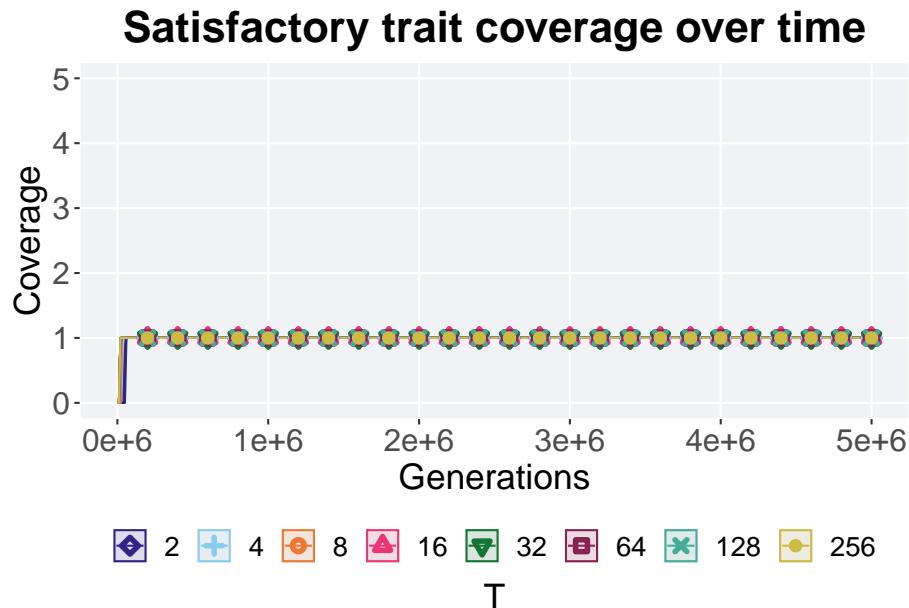
ot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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 +
guides(
  shape=guide_legend(nrow=1, title.position = "bottom"),
  color=guide_legend(nrow=1, title.position = "bottom"),
  fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)
)

ot

```



### 8.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```

best = filter(tor_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives')

plot = ggplot(best, aes(x = T, y = val, color = T, fill = T, shape = T)) +

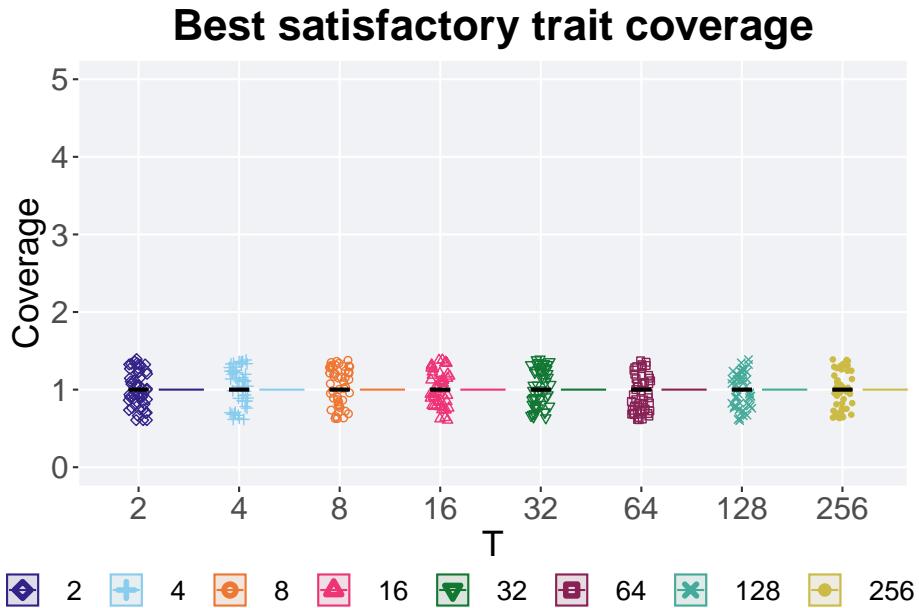
```

```

geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



### 8.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```
group_by(best, 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)
  )

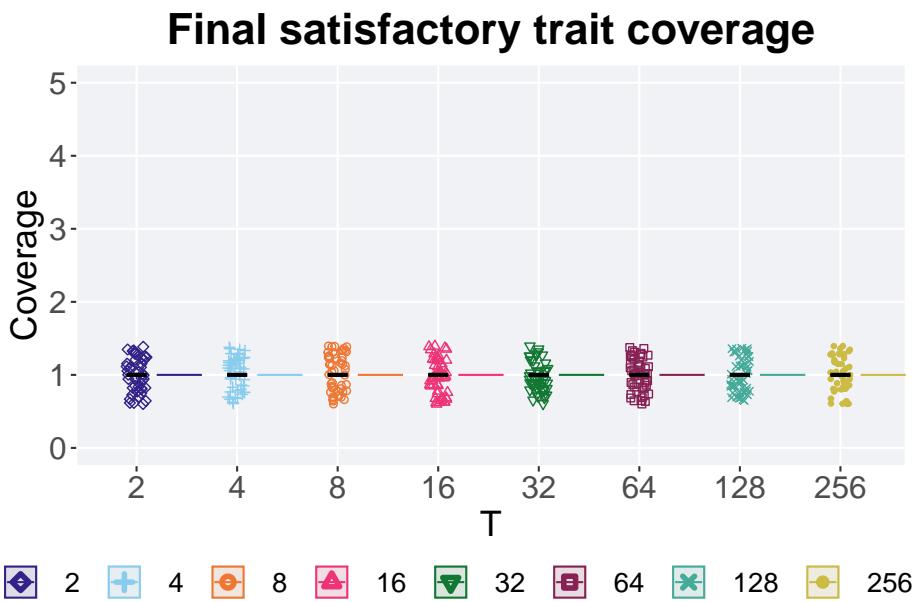
## # A tibble: 8 x 8
##   T     count na_cnt  min median  mean  max  IQR
##   <fct> <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 2       50      0     1     1     1     1     0
## 2 4       50      0     1     1     1     1     0
## 3 8       50      0     1     1     1     1     0
## 4 16      50    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

### 8.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
end = filter(tor_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = T, y = pop_uni_obj, color = T, fill = T), shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Coverage",
    limits = c(0, 5)
  ) +
  scale_x_discrete(
    name = "T"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```
plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```



#### 8.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

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

### 8.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

#### 8.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(tor_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
```

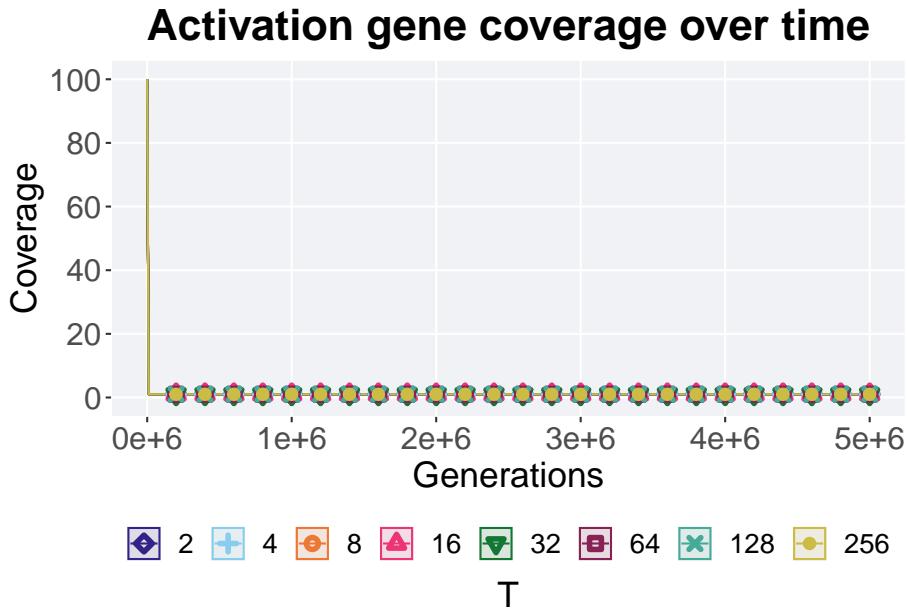
```

limits=c(0, 50000),
breaks=c(0, 10000, 20000, 30000, 40000, 50000),
labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
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+
guides(
shape=guide_legend(nrow=1, title.position = "bottom"),
color=guide_legend(nrow=1, title.position = "bottom"),
fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
legend.position = "bottom",
legend.box="vertical",
legend.justification="center",
legend.title.align=0.5
)
)

ot

```

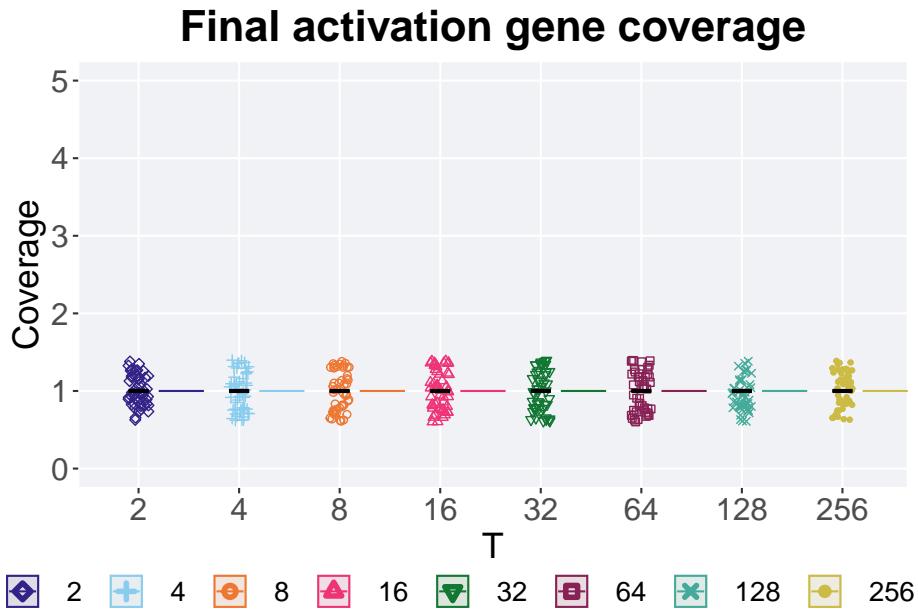


### 8.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
end = filter(tor_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Coverage",
    limits = c(0, 5)
  ) +
  scale_x_discrete(
    name = "T"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .2),
  label_size = TSIZE
)
```



#### 8.3.2.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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: 8 x 8
##   T      count  na_cnt  min  median  mean   max   IQR
##   <fct> <int>   <int> <dbl> <dbl> <dbl> <int> <dbl>
## 1 2        50       0     1      1      1     1      0
## 2 4        50       0     1      1      1     1      0
## 3 8        50       0     1      1      1     1      0
## 4 16       50       0     1      1      1     1      0
## 5 32       50       0     1      1      1     1      0
## 6 64       50       0     1      1      1     1      0
## 7 128      50       0     1      1      1     1      0
```

```
## 8 256      50      0      1      1      1      1      0
```

## 8.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

### 8.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

#### 8.4.1.1 Performance over time

Performance over time.

```
problem <- filter(tor_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
```

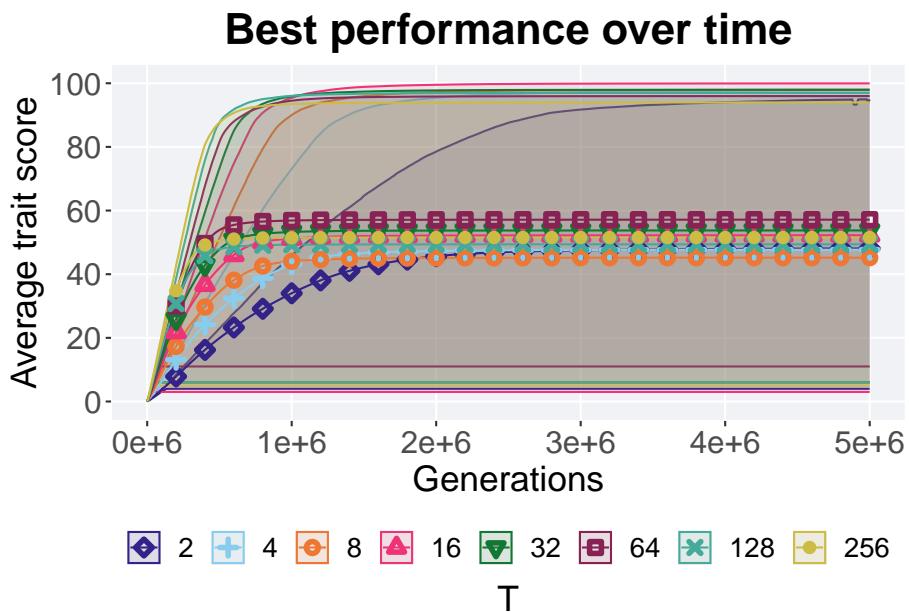
```

breaks=c(0, 10000, 20000, 30000, 40000, 50000),
labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme+
guides(
shape=guide_legend(nrow=1, title.position = "bottom"),
color=guide_legend(nrow=1, title.position = "bottom"),
fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
legend.position = "bottom",
legend.box="vertical",
legend.justification="center",
legend.title.align=0.5
)
)

ot

```



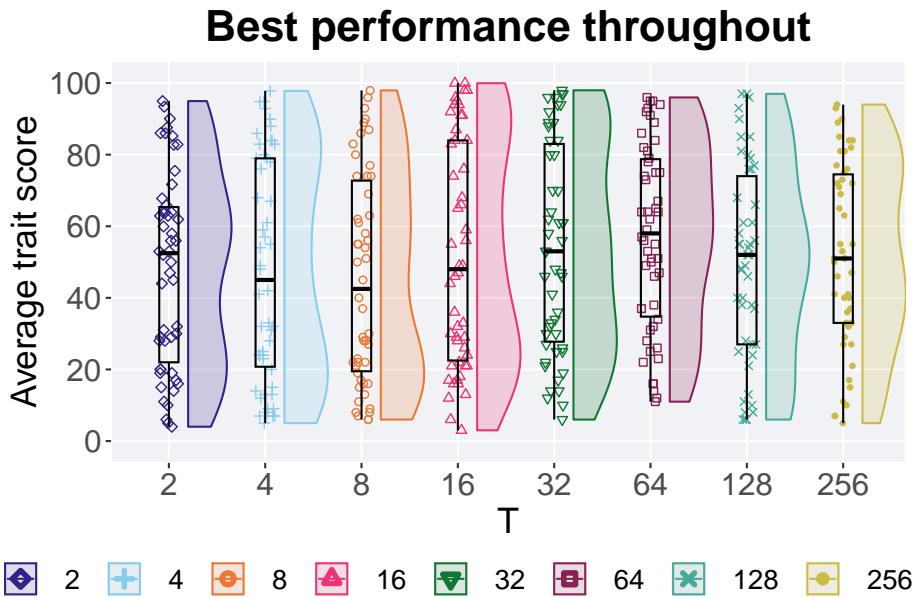
#### 8.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(tor_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = T, y = val / TRAITS, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 101),
    breaks = seq(0, 100, 20),
    labels = c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name = "T"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .3),
  label_size = TSIZE
)
```



#### 8.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```
group_by(best, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   T      count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2        50     0  4     52.5  48.1  95.0  43.4
## 2 4        50     0  5     45.0  47.7  97.8  58.2
## 3 8        50     0  6.00  42.5  45.2  97.9  53.2
## 4 16       50     0  3     48.0  52.2  100.   61.5
## 5 32       50     0  6     53.0  53.8  98.0  55.2
## 6 64       50     0  11    58.0  57.1  96.0  44.0
## 7 128      50     0  6.00  52.0  49.4  97.0  47.0
```

```
## 8 256      50      0 5      51.0 51.5 94.0 41.5
```

Kruskal–Wallis test provides evidence of no statistical differences among the best performing solution found throughout 50,000 generations.

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

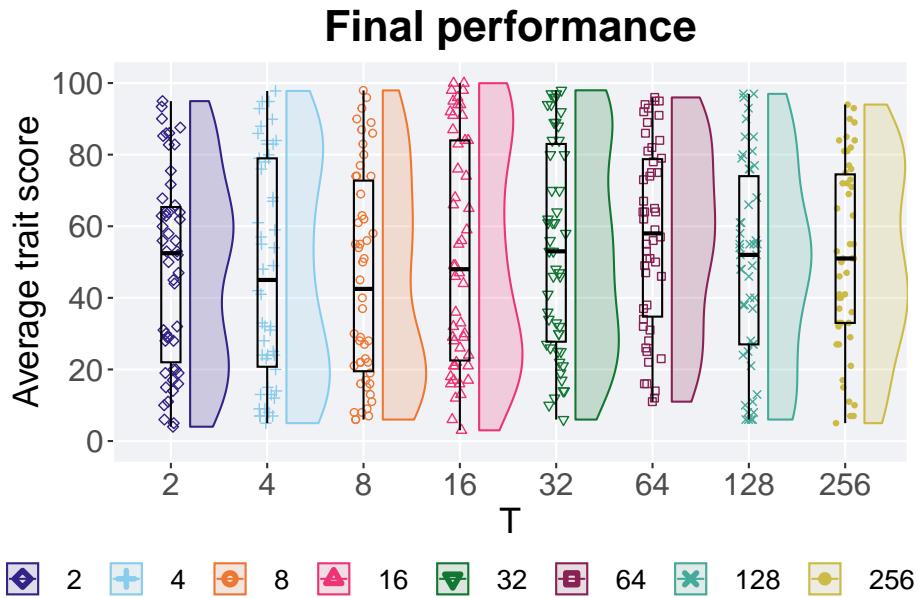
##
## Kruskal-Wallis rank sum test
##
## data: val by T
## Kruskal-Wallis chi-squared = 6.9356, df = 7, p-value = 0.4356
```

#### 8.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```
end = filter(tor_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = T, y = pop_fit_max / TRAITS, color = T, fill = T, shape = T))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 101),
    breaks = seq(0, 100, 20),
    labels = c("0", "20", "40", "60", "80", "100"))
  ) +
  scale_x_discrete(
    name = "T"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .3),
  label_size = TSIZE
)
```



#### 8.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, na.rm = TRUE)
  )

## # A tibble: 8 x 8
##   T      count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2        50     0  4     52.5  48.1  94.9  43.4
## 2 4        50     0  5     45.0  47.7  97.8  58.2
## 3 8        50     0  6.00  42.5  45.2  97.9  53.2
## 4 16       50     0  3     48.0  52.2  100.   61.5
## 5 32       50     0  6     53.0  53.8  98.0  55.2
## 6 64       50     0  11    58.0  57.1  96.0  44.0
## 7 128      50     0  6.00  52.0  49.4  97.0  47.0
```

```
## 8 256      50      0 5      51.0 51.5 94.0 41.5
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ T, data = end)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: pop_fit_max by T  
## Kruskal-Wallis chi-squared = 6.9356, df = 7, p-value = 0.4356
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance in the population at the end of 50,000 generations.

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

```
##  
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction  
##  
## data: end$pop_fit_max and end$T  
##  
##      2 4 8 16 32 64 128  
## 4   1 - - - - - -  
## 8   1 1 - - - - -  
## 16  1 1 1 - - - -  
## 32  1 1 1 1 - - -  
## 64  1 1 1 1 1 - -  
## 128 1 1 1 1 1 1 -  
## 256 1 1 1 1 1 1 1  
##  
## P value adjustment method: bonferroni
```

## 8.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

### 8.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(tor_ot, diagnostic == 'multipath_exploration')  
lines = problem %>%  
  group_by(T, gen) %>%  
  dplyr::summarise(  
    min = min(uni_str_pos),
```

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

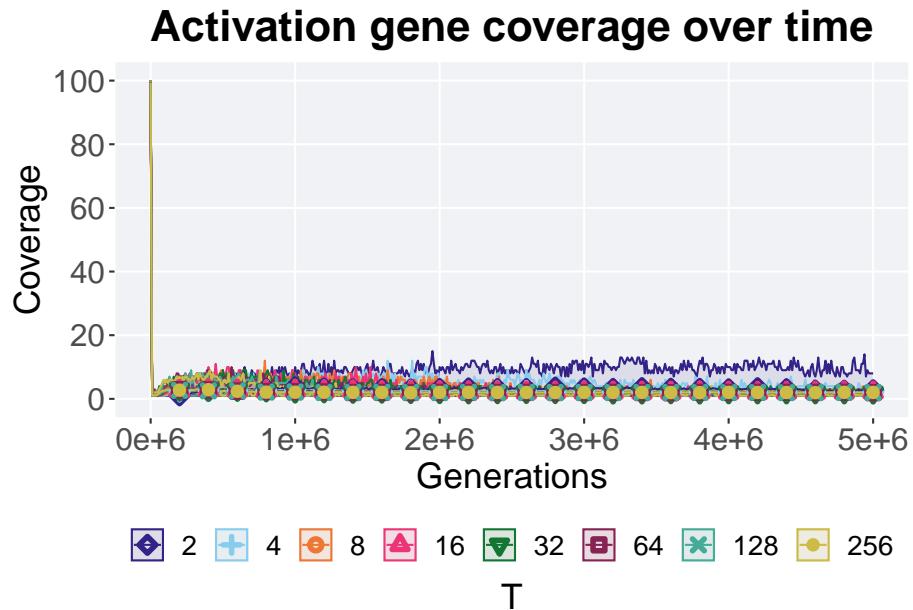
## `summarise()` has grouped output by 'T'. You can override using the `groups`#
## argument.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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 +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```



#### 8.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

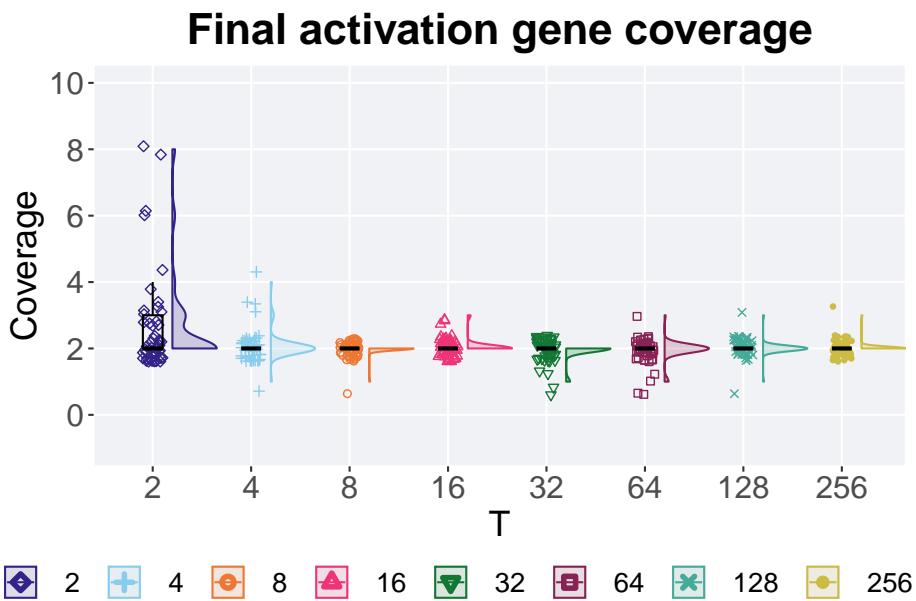
```
end = filter(tor_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 10),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10"))
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```

ggtile("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 8.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```

group_by(end, 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: 8 x 8
##   T      count  na_cnt  min  median  mean   max    IQR
##   <dbl>    <int>   <dbl> <dbl>   <dbl> <dbl> <dbl>   <dbl>
## 1 2       100000 100000 0.00  1.00  1.00  1.00  1.00
## 2 4       100000 100000 0.00  1.00  1.00  1.00  1.00
## 3 8       100000 100000 0.00  1.00  1.00  1.00  1.00
## 4 16      100000 100000 0.00  1.00  1.00  1.00  1.00
## 5 32      100000 100000 0.00  1.00  1.00  1.00  1.00
## 6 64      100000 100000 0.00  1.00  1.00  1.00  1.00
## 7 128     100000 100000 0.00  1.00  1.00  1.00  1.00
## 8 256     100000 100000 0.00  1.00  1.00  1.00  1.00

```

```
##   <fct> <int> <int> <int> <dbl> <dbl> <int> <dbl>
## 1 2      50     0     2     2  2.68    8     1
## 2 4      50     0     1     2  2.08    4     0
## 3 8      50     0     1     2  1.98    2     0
## 4 16     50     0     2     2  2.06    3     0
## 5 32     50     0     1     2  1.92    2     0
## 6 64     50     0     1     2  1.94    3     0
## 7 128    50     0     1     2     2     3     0
## 8 256    50     0     2     2  2.02    3     0
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by T
## Kruskal-Wallis chi-squared = 61.183, df = 7, p-value = 8.757e-11
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the activation gene coverage in the population at the end of 50,000 generations.

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

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$uni_str_pos and end$T
##
##   2      4      8      16     32     64     128
## 4 0.04907 -      -      -      -      -      -
## 8 0.00031 1.00000 -      -      -      -      -
## 16 0.02131 1.00000 1.00000 -      -      -      -
## 32 0.00011 0.58051 1.00000 0.24142 -      -      -
## 64 0.00044 1.00000 1.00000 0.99216 1.00000 -      -
## 128 0.00134 1.00000 1.00000 1.00000 1.00000 1.00000 -
## 256 0.00191 1.00000 1.00000 1.00000 0.70887 1.00000 1.00000
##
## P value adjustment method: bonferroni
```

## Chapter 9

# Genotypic fitness sharing

We present the results from our parameter sweep on genotypic fitness sharing. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, no similarity penalty is used, and only stochastic remainder selection is used to identify parent solutions.

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(Pupillometry)
```

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       Patched
## major         4
## minor        2.2
## year        2022
## month        11
## day          10
## svn rev     83330
## language     R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname    Innocent and Trusting
```

## 9.1 Exploitation rate results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the exploitation rate diagnostic. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, no similarity penalty is used, and only stochastic remainder selection is used to identify parent solutions.

### 9.1.1 Performance over time

Performance over time.

```
problem <- filter(gfs_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
```



### 9.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(gfs_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')

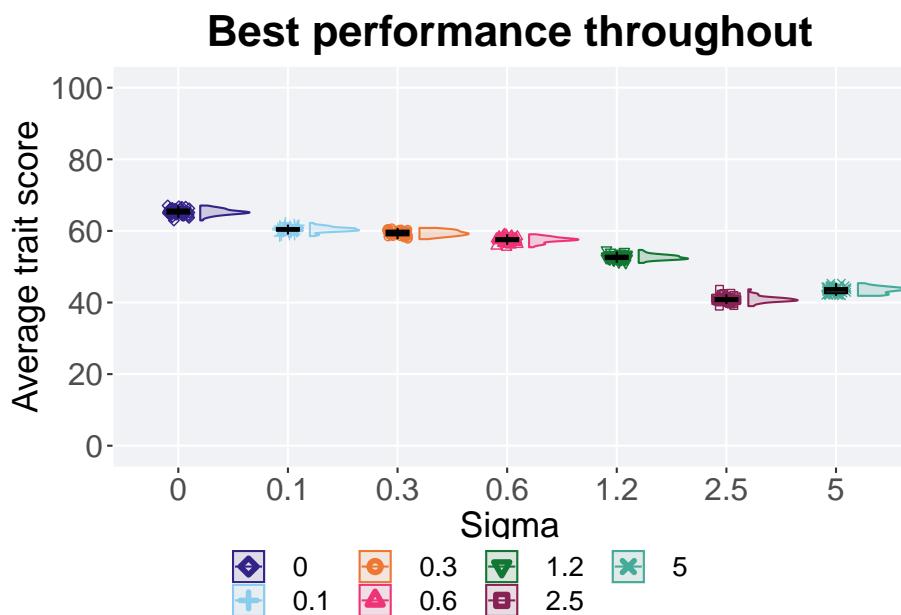
plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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) +
  p_theme

  plot_grid(
    plot +
      ggtitle("Best performance throughout") +
      theme(legend.position="none"),
    legend,
    nrow=2,
    rel_heights = c(2,.3),
    label_size = TSIZE
  )
)

```



### 9.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```

group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    sd = sd(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE)
  )

```

```

    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
)

## # A tibble: 7 x 8
##   Sigma count na_cnt  min median  mean  max   IQR
##   <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0  62.9  65.3  65.3  67.1 1.11
## 2 0.1    50     0  58.5  60.4  60.4  62.2 0.717
## 3 0.3    50     0  57.7  59.3  59.4  60.8 1.31
## 4 0.6    50     0  55.4  57.6  57.5  59.1 0.843
## 5 1.2    50     0  51.1  52.5  52.6  54.7 0.907
## 6 2.5    50     0  39.0  40.8  40.9  43.7 0.813
## 7 5      50     0  41.9  43.6  43.4  45.4 1.43

```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

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

```

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 336.49, df = 6, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

```

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##      0      0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -      -      -      -      -
## 0.3 < 2e-16 1.5e-07 -      -      -      -
## 0.6 < 2e-16 < 2e-16 3.0e-14 -      -      -
## 1.2 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -      -
## 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

```

## 9.2 Ordered exploitation results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

### 9.2.1 Performance over time

Performance over time.

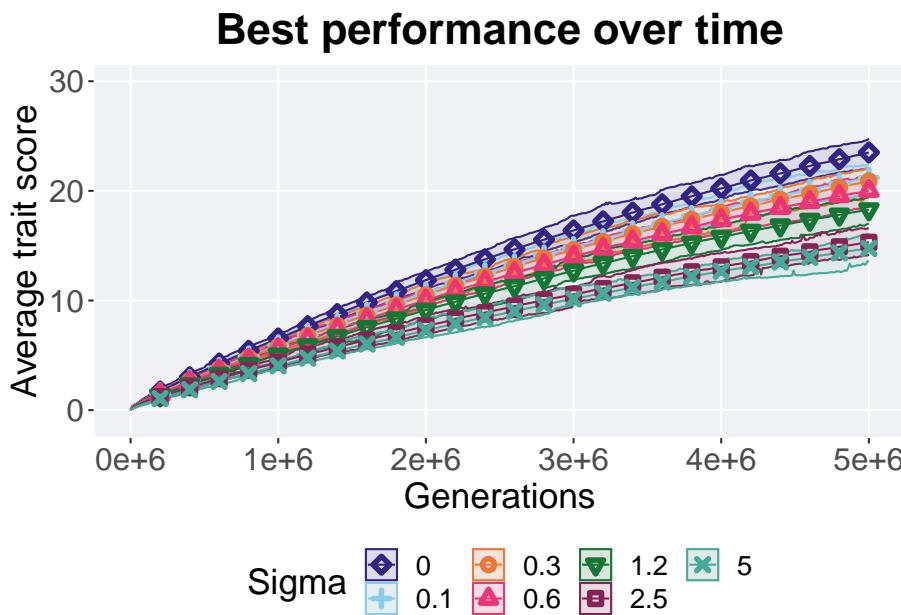
```
problem <- filter(gfs_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

## `summarise()` has grouped output by 'Sigma'. You can override using the
## ` `.groups` argument.
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30),
    breaks=seq(0, 30, 10),
    labels=c("0", "10", "20", "30"))
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6"))

  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme
```

ot



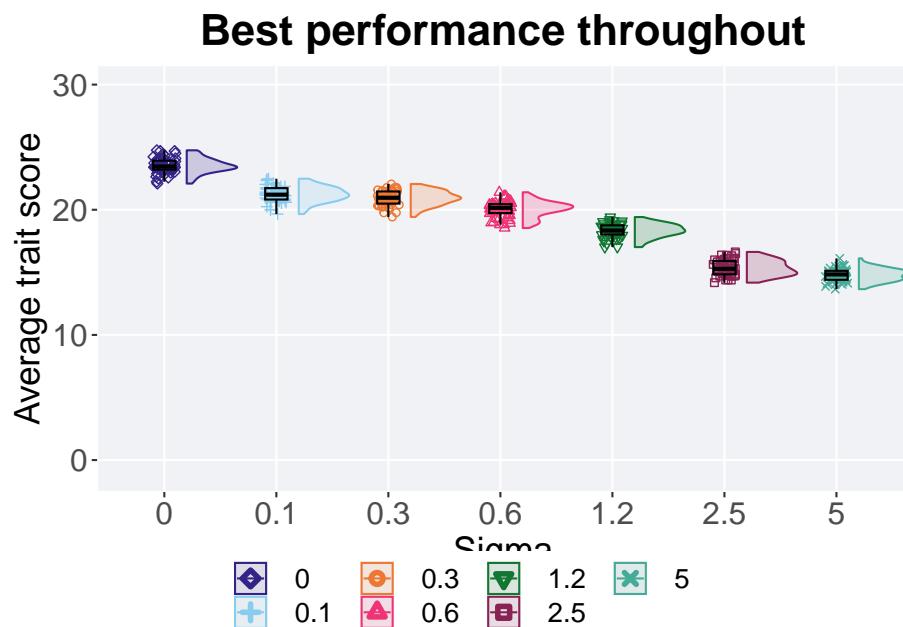
### 9.2.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(gfs_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30),
    breaks=seq(0, 30, 10),
    labels=c("0", "10", "20", "30"))
  ) +
  scale_x_discrete(
    name="Sigma")
) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```
plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```



### 9.2.2.1 Stats

Summary statistics about the best performance found.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt  min median  mean  max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0  22.1  23.5  23.5  24.8 0.713
## 2 0.1    50     0  19.7  21.2  21.2  22.5 0.911
## 3 0.3    50     0  19.4  21.0  20.9  22.1 0.970
## 4 0.6    50     0  18.5  20.2  20.0  21.4 0.716
## 5 1.2    50     0  17.0  18.3  18.3  19.4 0.729
## 6 2.5    50     0  14.2  15.3  15.4  16.6 1.06
## 7 5      50     0  13.7  14.8  14.8  16.1 0.717
```

Kruskal–Wallis test provides evidence of statistical differences for the best performance found in the population throughout 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 322.96, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found in the population throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##   0      0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -      -      -      -      -
## 0.3 < 2e-16 0.15850 -      -      -      -
## 0.6 < 2e-16 1.5e-11 5.6e-08 -      -      -
## 1.2 < 2e-16 < 2e-16 < 2e-16 3.4e-15 -      -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5    < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 0.00038
##
## P value adjustment method: bonferroni
```

### 9.3 Contradictory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate

on the ordered exploitation diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

### 9.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

#### 9.3.1.1 Coverage over time

Satisfactory trait coverage over time.

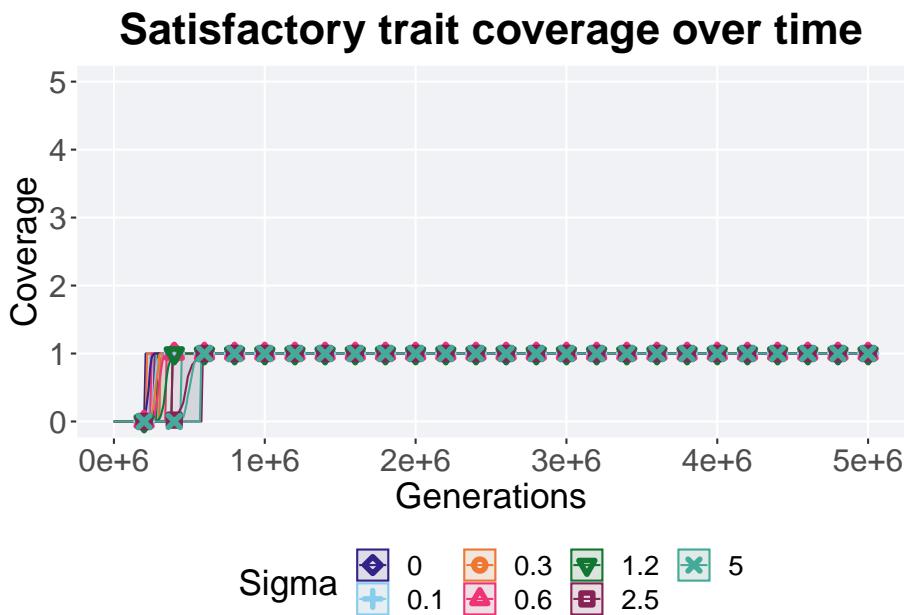
```
problem <- filter(gfs_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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
```

ot



### 9.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

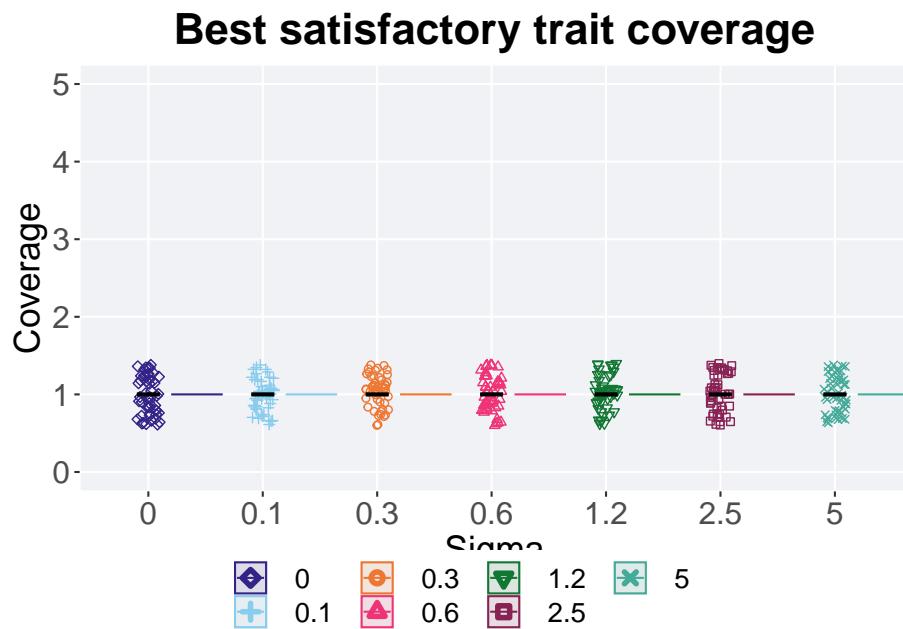
```
best = filter(gfs_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives')
plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    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) +
  p_theme

plot_grid(
  plot +
```

```

ggtile("Best satisfactory trait coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



### 9.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```

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

Kruskal–Wallis test provides evidence of no statistical difference among satisfactory trait coverage throughout 50,000 generations.

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

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

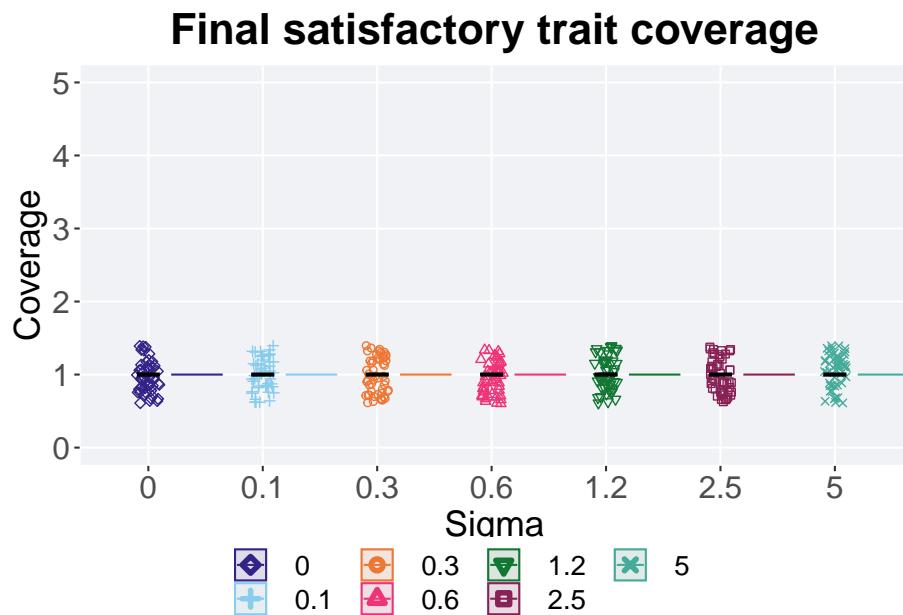
### 9.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the final population (50,000 generations).

```
end = filter(gfs_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    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) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
```

```
    label_size = TSIZE
)
```



### 9.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the final population (50,000 generations).

```
group_by(end, 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> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0     1     1     1     1     0
## 2 0.1    50     0     1     1     1     1     0
## 3 0.3    50     0     1     1     1     1     0
## 4 0.6    50     0     1     1     1     1     0
```

```
## 5 1.2      50     0     1     1     1     1     0
## 6 2.5      50     0     1     1     1     1     0
## 7 5        50     0     1     1     1     1     0
```

Kruskal–Wallis test provides evidence of no statistical difference among satisfactory trait coverage throughout 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = NaN, df = 6, p-value = NA
```

### 9.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

#### 9.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(gfs_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100"))
  ) +
  scale_x_continuous(
```

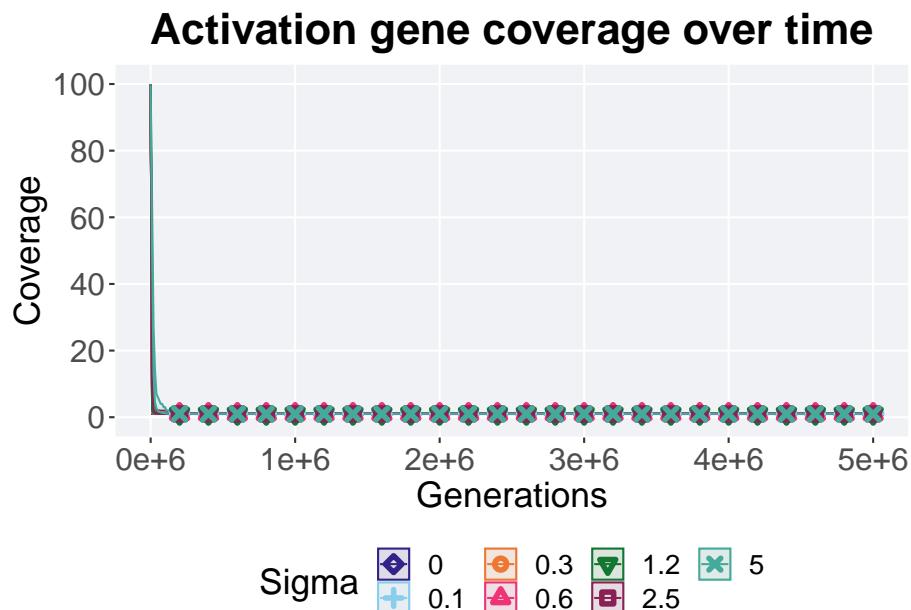
```

name="Generations",
limits=c(0, 50000),
breaks=c(0, 10000, 20000, 30000, 40000, 50000),
labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
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

ot

```



### 9.3.2.2 End of 50,000 generations

Activation gene coverage in the final population (50,000 generations).

```

end = filter(gfs_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha =
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",

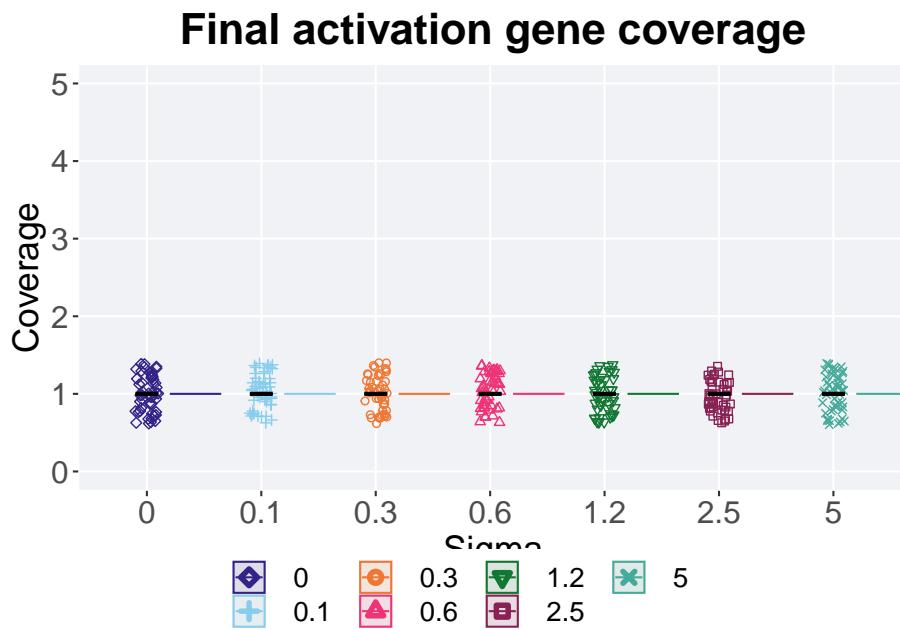
```

```

    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) +
p_theme

plot_grid(
  plot +
  ggtitle("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



### 9.3.2.2.1 Stats

Summary statistics for activation gene coverage in the final population (50,000 generations).

```

group_by(end, 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> <dbl> <dbl> <int> <dbl>
## 1 0      50     0     1     1     1     1     0
## 2 0.1    50     0     1     1     1     1     0
## 3 0.3    50     0     1     1     1     1     0
## 4 0.6    50     0     1     1     1     1     0
## 5 1.2    50     0     1     1     1     1     0
## 6 2.5    50     0     1     1     1     1     0
## 7 5      50     0     1     1     1     1     0

```

Kruskal–Wallis test provides evidence of no statistical difference for activation gene coverage in the final population (50,000 generations).

```

kruskal.test(uni_str_pos ~ Sigma, data = end)

##
##  Kruskal-Wallis rank sum test
##
## data: uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = NaN, df = 6, p-value = NA

```

## 9.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

### 9.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

#### 9.4.1.1 Performance over time

Performance over time.

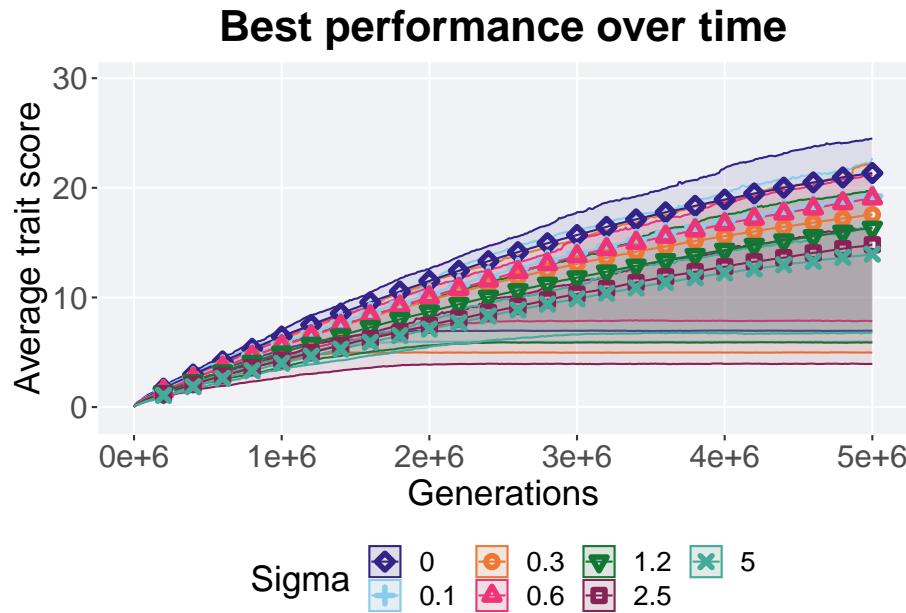
```
problem <- filter(gfs_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme

ot
```



#### 9.4.1.2 Best performance throughout

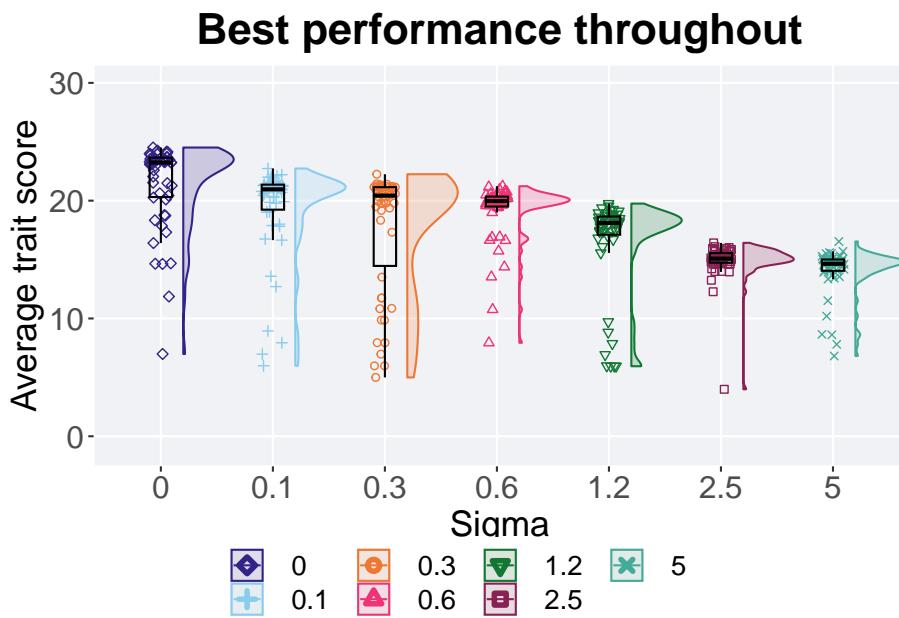
Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(gfs_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  plot + geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name = "Average trait score",
      limits = c(-1, 30)
    ) +
    scale_x_discrete(
      name = "Sigma"
    ) +
    scale_shape_manual(values = SHAPE) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    p_theme

plot_grid(
  plot +
```

```
ggtile("Best performance throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```



#### 9.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
  )

## # A tibble: 7 x 8
##   Sigma count na_cnt   min median   mean   max    IQR
##   <dbl> <int>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0  6.99  23.2  21.4  24.5 3.34
## 2 0.1    50     0  5.98  21.0  19.3  22.7 2.12
## 3 0.3    50     0  4.99  20.4  17.6  22.2 6.69
## 4 0.6    50     0  7.93  20.0  19.1  21.2 0.841
## 5 1.2    50     0  5.95  18.1  16.4  19.8 1.55
## 6 2.5    50     0  3.99  15.1  14.9  16.4 0.809
## 7 5      50     0  6.81  14.6  14.0  16.5 0.962
```

Kruskal–Wallis test provides evidence of statistical differences among the best performing solution found throughout 50,000 generations.

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

```
##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 161.29, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performing solution found throughout 50,000 generations.

```
pairwise.wilcox.test(x = best$val, g = best$Sigma, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##   0     0.1    0.3    0.6    1.2    2.5
## 0.1 0.00026 -     -     -     -     -
## 0.3 4.8e-06 0.53186 -     -     -     -
## 0.6 8.5e-06 0.00531 1.00000 -     -     -
## 1.2 8.7e-09 2.1e-07 0.00023 2.7e-08 -     -
## 2.5 1.0e-11 4.1e-10 0.00024 4.2e-12 6.6e-08 -
## 5   9.4e-13 2.2e-10 9.9e-05 8.9e-13 3.7e-08 0.00268
##
## P value adjustment method: bonferroni
```

#### 9.4.1.3 End of 50,000 generations

Best performance in the final population (50,000 generations).

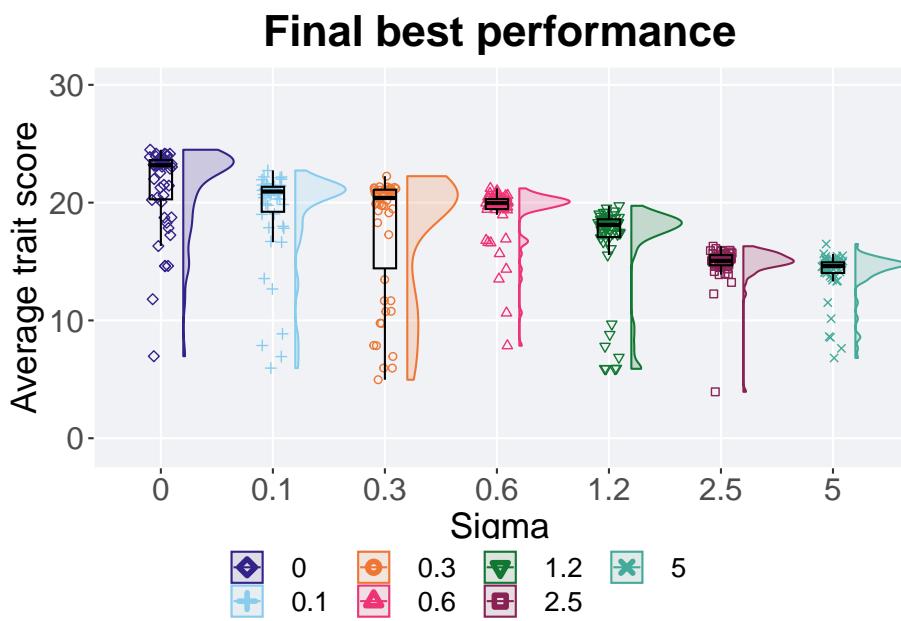
```
end = filter(gfs_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, color = Sigma, fill = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_text(x = 0.5, y = 0.05, label = 'Multipath exploration', color = 'red')
```

```

geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final best performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 9.4.1.3.1 Stats

Summary statistics for the best performance in the final population (50,000 generations).

```
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, 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  6.96  23.2  21.3  24.5 3.34
## 2 0.1       50      0  5.95  20.9  19.3  22.7 2.12
## 3 0.3       50      0  4.96  20.4  17.6  22.2 6.68
## 4 0.6       50      0  7.85  20.0  19.0  21.2 0.855
## 5 1.2       50      0  5.89  18.1  16.3  19.7 1.51
## 6 2.5       50      0  3.94  15.1  14.8  16.3 0.864
## 7 5         50      0  6.81  14.6  14.0  16.5 0.904
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the final population (50,000 generations).

```
kruskal.test(pop_fit_max ~ Sigma, data = end)

##
##  Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 161.62, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance in the final population (50,000 generations).

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

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$Sigma
##
##   0     0.1    0.3    0.6    1.2    2.5
## 0.1 0.00024 -      -      -      -      -
```

```

## 0.3 5.0e-06 0.45941 - - - -
## 0.6 8.2e-06 0.00431 1.00000 - - - -
## 1.2 6.7e-09 1.9e-07 0.00022 2.9e-08 - - -
## 2.5 9.4e-12 4.1e-10 0.00024 4.4e-12 6.6e-08 -
## 5 9.9e-13 2.2e-10 9.6e-05 8.9e-13 3.6e-08 0.00283
##
## P value adjustment method: bonferroni

```

## 9.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

### 9.4.2.1 Coverage over time

Activation gene coverage over time.

```

problem <- filter(gfs_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

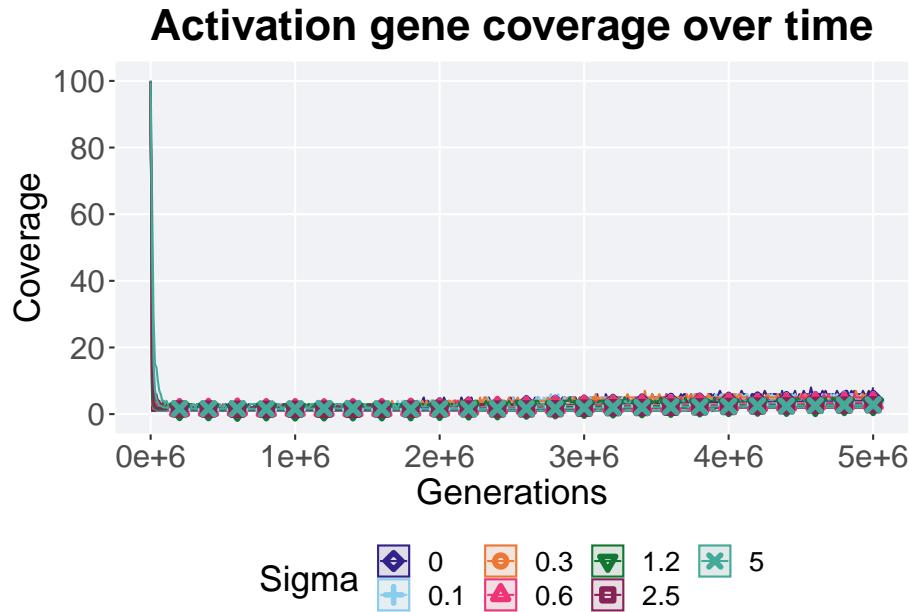
```

```

) +
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

ot

```



#### 9.4.2.2 End of 50,000 generations

Activation gene coverage in the final population (50,000 generations).

```

end = filter(gfs_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 10),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10"))
) +
  scale_x_discrete()

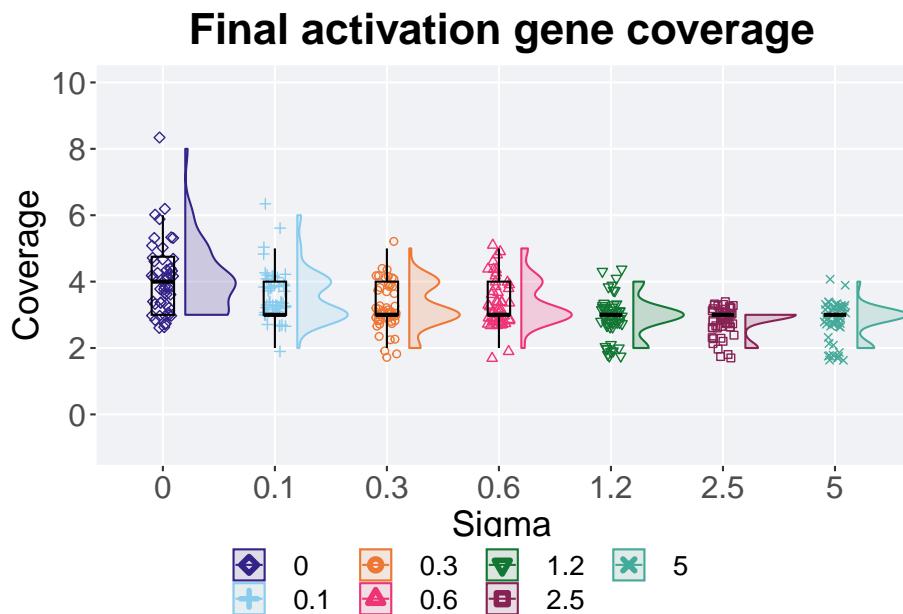
```

```

    name="Sigma"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 9.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the final population (50,000 generations).

```

group_by(end, 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
##   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0     3     4    4.04    8    1.75
## 2 0.1    50     0     2     3    3.54    6    1
## 3 0.3    50     0     2     3    3.24    5    1
## 4 0.6    50     0     2     3    3.38    5    1
## 5 1.2    50     0     2     3    2.94    4    0
## 6 2.5    50     0     2     3    2.8     3    0
## 7 5      50     0     2     3    2.8     4    0

```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the final population (50,000 generations).

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

```

##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 93.885, df = 6, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the activation gene coverage in the final population (50,000 generations).

```

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

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
##  data: end$uni_str_pos and end$Sigma
##
##          0      0.1     0.3     0.6     1.2     2.5
## 0.1 0.16972 -      -      -      -      -
## 0.3 0.00109 1.00000 -      -      -      -
## 0.6 0.00962 1.00000 1.00000 -      -      -
## 1.2 1.2e-07 0.00138 0.48268 0.03870 -      -
## 2.5 5.9e-11 7.7e-07 0.00563 6.8e-05 1.00000 -
## 5   4.4e-10 4.4e-06 0.01196 0.00025 1.00000 1.00000

```

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

## 9.5 Multi-valley crossing results

Here we present the results for the **best performances** and **best gene value** generated by each genotypic fitness sharing replicate on the multi-valley crossing diagnostic. Best performance found refers to the largest average trait score found in a given population. Best gene value refers to maximum gene value found in the population; this gives us a range of values in [0.0, 100.0]. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, no similarity penalty is used, and only stochastic remainder selection is used to identify parent solutions.

### 9.5.1 Performance

Here we analyze the performances for each parameter replicate on the multi-valley crossing diagnostic.

#### 9.5.1.1 Performance over time

Performance over time.

```
lines = filter(gfs_ot, diagnostic == 'multivalley_crossing') %>%  
  group_by(Sigma, gen) %>%  
  dplyr::summarise(  
    min = min(pop_fit_max),  
    mean = mean(pop_fit_max),  
    max = max(pop_fit_max)  
  )  
  
## `summarise()` has grouped output by 'Sigma'. You can override using the  
## `.` argument.  
points = filter(lines, gen %% 2000 == 0 & gen != 0)  
  
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma,  
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +  
  geom_line(size = 0.5) +  
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +  
  scale_y_continuous(  
    name="Average trait score",  
    limits=c(-1, 51),  
    breaks=seq(0,50, 10),  
    labels=c("0", "10", "20", "30", "40", "50")  
  ) +  
  scale_x_continuous(
```

```

name="Generations",
limits=c(0, 50000),
breaks=c(0, 10000, 20000, 30000, 40000, 50000),
labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
ggtitle("Best performance over time") +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

ot

```



### 9.5.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```

best = filter(gfs_best, col == 'pop_fit_max' & diagnostic == 'multivalley_crossing')
plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  guides(fill = "none", color = 'none', shape = 'none') +
  scale_y_continuous()

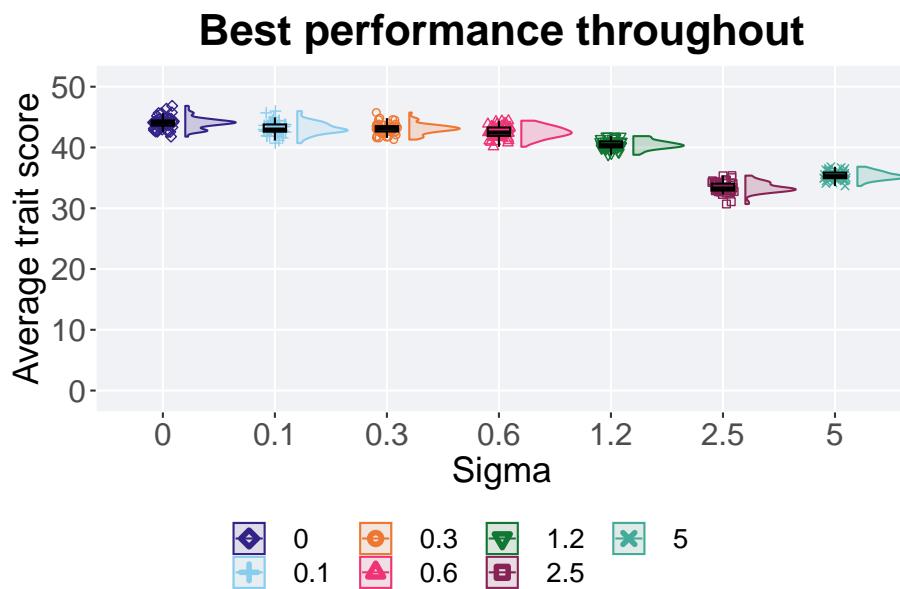
```

```

      name="Average trait score",
      limits=c(-1, 51),
      breaks=seq(0,50, 10),
      labels=c("0", "10", "20", "30", "40", "50")
    ) +
    scale_x_discrete(
      name="Sigma"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    p_theme

plot_grid(
  plot +
  ggtitle("Best performance throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 9.5.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
best$Sigma = factor(best$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 5.0, 2.5))
group_by(best, 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 4179. 4410. 4403. 4685. 85.0
## 2 0.1       50     0 4072. 4294. 4318. 4599. 119.
## 3 0.3       50     0 4130. 4311. 4310. 4575. 87.6
## 4 0.6       50     0 4012. 4246. 4254. 4441. 139.
## 5 1.2       50     0 3880. 4037. 4050. 4187. 98.0
## 6 5          50     0 3367. 3527. 3535. 3684. 91.0
## 7 2.5       50     0 3070. 3324. 3342. 3537. 114.
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

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

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 290.89, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##      0      0.1     0.3     0.6     1.2      5
```

```

## 0.1 0.00086 - - - -
## 0.3 4.0e-05 1.00000 - - - -
## 0.6 5.6e-09 0.03681 0.06862 - - - -
## 1.2 < 2e-16 6.3e-16 4.7e-16 1.0e-13 - - -
## 5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 5.5e-14
##
## P value adjustment method: bonferroni

```

### 9.5.1.3 End of 50,000 generations

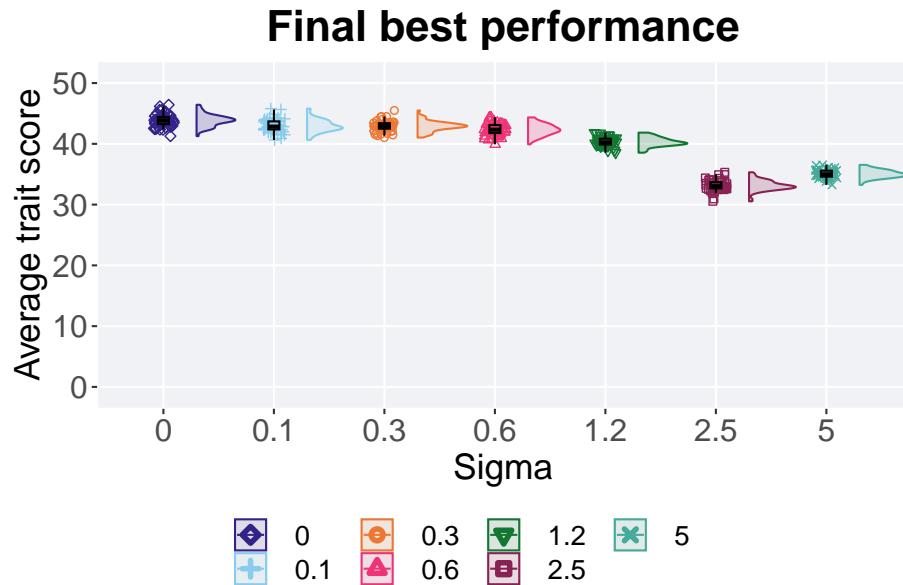
Best performance in the final population (50,000 generations).

```

end = filter(gfs_end, diagnostic == 'multivalley_crossing')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_flat_violin(position = position_nudge(x = .3), alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 10.8) +
  geom_boxplot(width = .1, outlier.shape = NA, alpha = 0.1, col = "black") +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final best performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 9.5.1.3.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
end$Sigma = factor(end$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 5.0, 2.5))
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max, na.rm = TRUE),
    median = median(pop_fit_max, na.rm = TRUE),
    mean = mean(pop_fit_max, na.rm = TRUE),
    max = max(pop_fit_max, na.rm = TRUE),
    IQR = IQR(pop_fit_max, 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 4127. 4385. 4386. 4642. 118.
## 2 0.1       50      0 4064. 4283. 4298. 4580. 132.
## 3 0.3       50      0 4104. 4291. 4290. 4549. 90.7
## 4 0.6       50      0 3992. 4233. 4235. 4435. 132.
## 5 1.2       50      0 3854. 4023. 4032. 4184. 96.2
## 6 5         50      0 3324. 3499. 3505. 3654. 97.7
```

```
## 7 2.5      50      0 3055. 3303. 3315. 3532. 101.
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

```
kruskal.test(pop_fit_max ~ Sigma, data = end)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 290.89, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$Sigma
##
##          0       0.1      0.3      0.6      1.2      5
## 0.1 0.0004 -       -       -       -       -
## 0.3 3.7e-05 1.0000 -       -       -       -
## 0.6 4.7e-09 0.0352 0.0568 -       -       -
## 1.2 < 2e-16 7.5e-16 3.9e-16 1.9e-13 -       -
## 5    < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 7.6e-14
##
## P value adjustment method: bonferroni
```

## 9.5.2 Best gene value

Here we analyze the gene values for each parameter replicate on the multi-valley crossing diagnostic.

### 9.5.2.1 Gene value over time

Best gene value over time.

```
lines = filter(gfs_ot, diagnostic == 'multivalley_crossing') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_max_gene),
    mean = mean(pop_max_gene),
```

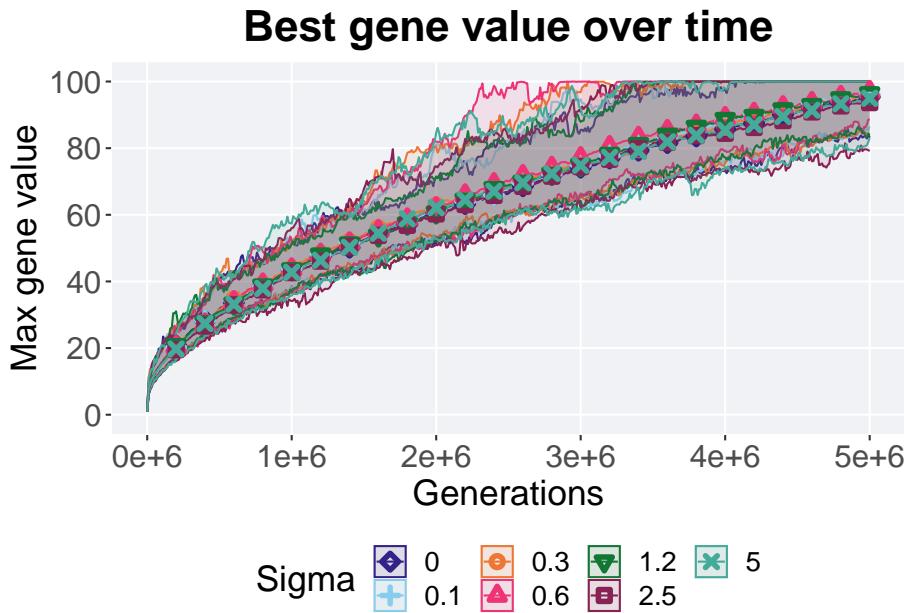
```
    max = max(pop_max_gene)
)
```

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Max gene value",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  ggtitle("Best gene value over time") +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

ot
```



#### 9.5.2.2 Best gene value throughout

The best gene value found throughout 50,000 generations.

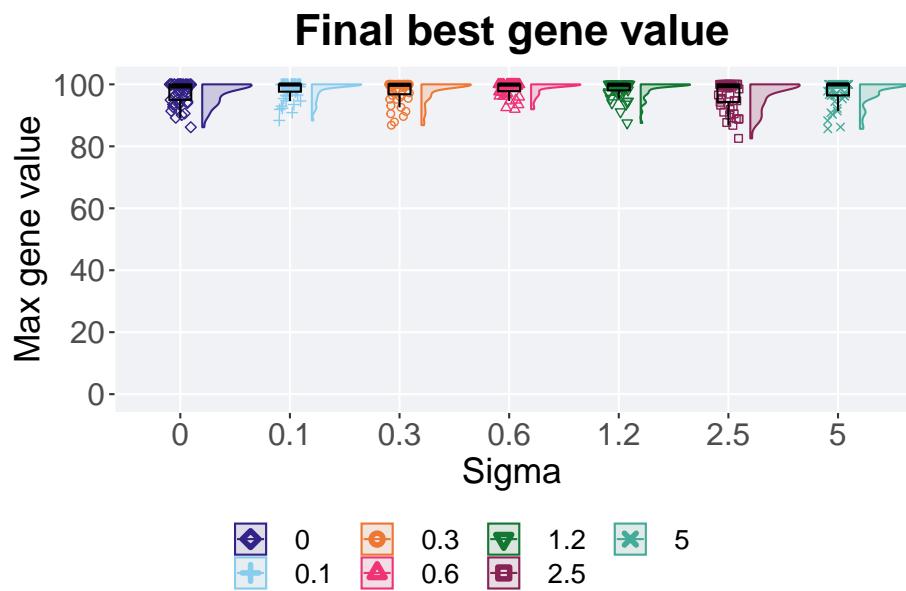
```
best = filter(gfs_best, col == 'pop_max_gene' & diagnostic == 'multivalley_crossing')
plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  guides(fill = "none", color = 'none', shape = 'none') +
  scale_y_continuous(
    name = "Max gene value",
    limits = c(-1, 101),
    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) +
  p_theme

plot_grid(
```

```

plot +
  ggtitle("Final best gene value") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 9.5.2.2.1 Stats

Summary statistics for the best gene value found throughout 50,000 generations.

```

best$Sigma = factor(best$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0))
group_by(best, 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  86.2  99.3  97.3  100   5.00
## 2 0.1    50     0  88.3 100.   98.2  100   2.35
## 3 0.3    50     0  86.8  99.9  97.9  100   3.20
## 4 0.6    50     0  92.0  99.9  98.7  100   2.19
## 5 1.2    50     0  87.5  99.9  98.5  100   1.94
## 6 2.5    50     0  82.6  99.5  96.6  100   5.67
## 7 5      50     0  85.6 100.   97.5  100   3.56
```

Kruskal–Wallis test provides evidence of no significant differences among sigma values on the best gene value found throughout 50,000 generations

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

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 5.9808, df = 6, p-value = 0.4253
```

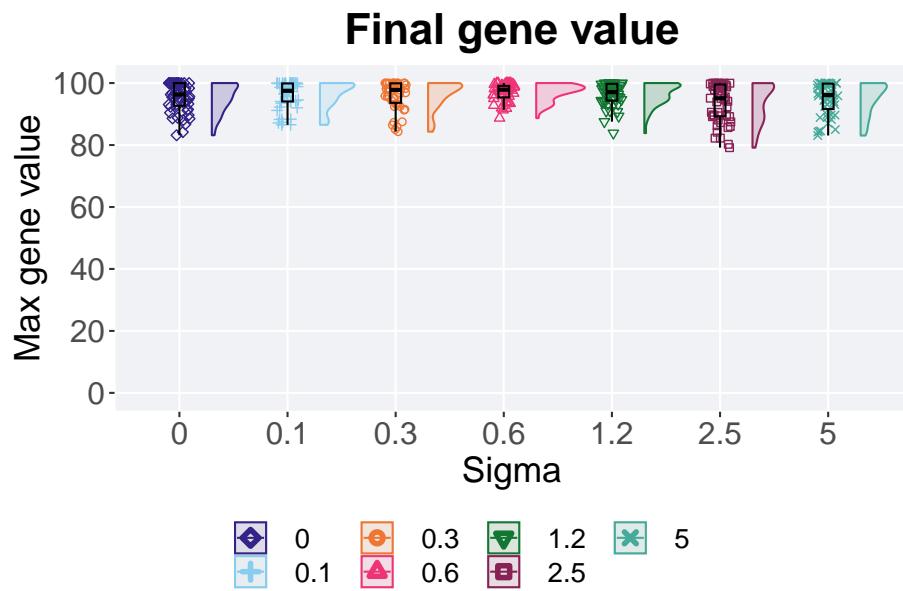
### 9.5.2.3 End of 50,000 generations

Best gene value in the final population (50,000 generations).

```
end = filter(gfs_end, diagnostic == 'multivalley_crossing')
plot = ggplot(end, aes(x = Sigma, y = pop_max_gene, group = Sigma, fill = Sigma, color = Sigma,
                       geom_flat_violin(position = position_nudge(x = .3), alpha = 0.2) +
                       geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 10.8) +
                       geom_boxplot(width = .1, outlier.shape = NA, alpha = 0.1, col = "black") +
                       scale_y_continuous(
                         name="Max gene value",
                         limits=c(-1, 101),
                         breaks=seq(0,100, 20),
                         labels=c("0", "20", "40", "60", "80", "100"))
                     ) +
                     scale_shape_manual(values=SHAPE) +
                     scale_colour_manual(values = cb_palette) +
                     scale_fill_manual(values = cb_palette) +
                     p_theme

plot_grid(
  plot +
    ggtitle("Final gene value") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
```

```
    label_size = TSIZE
)
```



#### 9.5.2.3.1 Stats

Summary statistics for the best gene value found at the end of 50,000 generations.

```
end$Sigma = factor(end$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0))
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_max_gene)),
    min = min(pop_max_gene, na.rm = TRUE),
    median = median(pop_max_gene, na.rm = TRUE),
    mean = mean(pop_max_gene, na.rm = TRUE),
    max = max(pop_max_gene, na.rm = TRUE),
    IQR = IQR(pop_max_gene, 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  83.2    96.3   95.3  100.  7.32
## 2 0.1     50      0  86.5    97.5   96.1  100.  5.82
## 3 0.3     50      0  84.3    97.8   96.1  100.  6.29
## 4 0.6     50      0  88.7    97.9   96.9  100.  3.71
```

```
## 5 1.2      50      0  83.8   97.1  96.4 100.  5.40
## 6 2.5      50      0  79.2   95.1  93.7 100. 10.3
## 7 5       50      0  83.1   96.1  95.0 100.  8.23
```

Kruskal–Wallis test provides evidence of no significant differences among sigma values on the best gene values found at the end of 50,000 generations.

```
kruskal.test(pop_max_gene ~ Sigma, data = end)

##
## Kruskal-Wallis rank sum test
##
## data: pop_max_gene by Sigma
## Kruskal-Wallis chi-squared = 8.3006, df = 6, p-value = 0.2169
```



## Chapter 10

# Phenotypic fitness sharing

We present the results from our parameter sweep on phenotypic fitness sharing. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, performance with no similarity penalty and stochastic remainder selection is used to identify parent solutions.

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(Pupillometry)
```

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         Patched
## major          4
## minor          2.2
## year           2022
## month          11
## day            10
## svn rev        83330
## language       R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname       Innocent and Trusting
```

## 10.1 Exploitation rate results

Here we present the results for **best performances** found by each phenotypic fitness sharing sigma value replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, no similarity penalty is used, and only stochastic remainder selection is used to identify parent solutions.

### 10.1.1 Performance over time

Performance over time.

```
problem <- filter(pfs_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
```

```
ggttitle("Best performance over time") +
p_theme
```

```
ot
```



### 10.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(pfs_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')

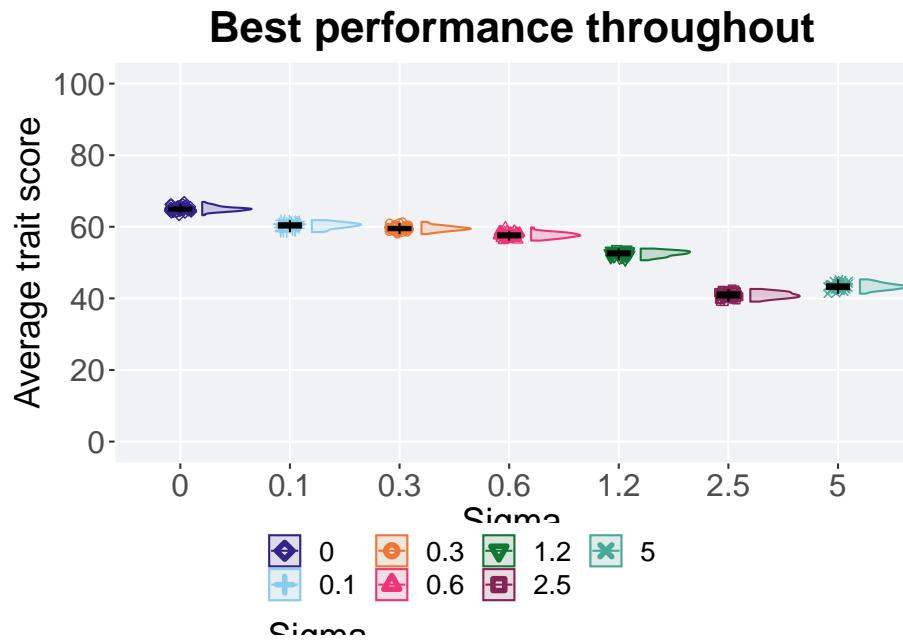
plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    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) +
p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 10.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```

group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE)
  )

```

```

mean = mean(val / TRAITS, na.rm = TRUE),
max = max(val / TRAITS, na.rm = TRUE),
IQR = IQR(val / TRAITS, na.rm = TRUE)
)

## # A tibble: 7 x 8
##   Sigma count na_cnt  min median  mean  max  IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0  63.2  65.0  65.0  66.9 0.816
## 2 0.1    50     0  58.5  60.4  60.4  61.9 1.19
## 3 0.3    50     0  58.0  59.5  59.5  61.4 0.908
## 4 0.6    50     0  56.2  57.6  57.6  59.8 1.11
## 5 1.2    50     0  50.7  52.6  52.5  53.9 1.11
## 6 2.5    50     0  39.1  40.9  40.9  42.6 1.40
## 7 5      50     0  41.3  43.3  43.3  45.3 1.29

```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

```

kruskal.test(val ~ Sigma, data = best)

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 335.72, df = 6, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

```

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##      0      0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -      -      -      -      -
## 0.3 < 2e-16 2.5e-05 -      -      -      -
## 0.6 < 2e-16 3.3e-16 7.4e-15 -      -      -
## 1.2 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -      -
## 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

```

## 10.2 Ordered exploitation results

Here we present the results for **best performances** found by each phenotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, no similarity penalty is used, and only stochastic remainder selection is used to identify parent solutions.

### 10.2.1 Performance over time

Performance over time.

```
problem <- filter(pfs_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

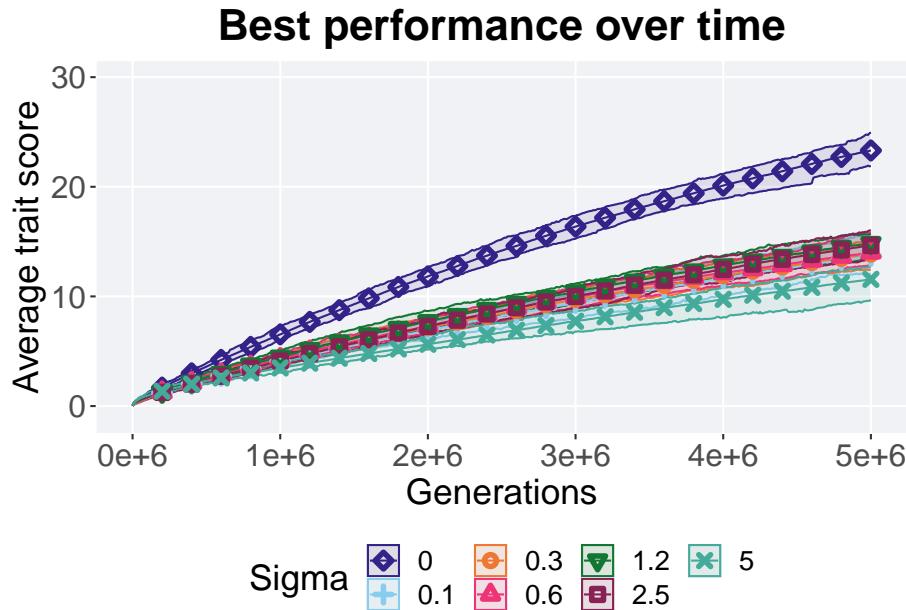
## `summarise()` has grouped output by 'Sigma'. You can override using the
## ` `.groups` argument.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30),
    breaks=seq(0, 30, 10),
    labels=c("0", "10", "20", "30")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
```

```
ggtile("Best performance over time") +
p_theme
```

```
ot
```



### 10.2.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(pfs_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30),
    breaks=seq(0, 30, 10),
    labels=c("0", "10", "20", "30"))
  ) +
  scale_x_discrete(
    name="Sigma")
) +
  scale_shape_manual(values = SHAPE) +
```

```

scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



### 10.2.2.1 Stats

Summary statistics about the best performance found.

```

group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    sd = sd(val / TRAITS, na.rm = TRUE)
  )

```

```

    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
)

## # A tibble: 7 x 8
##   Sigma count na_cnt  min median  mean  max   IQR
##   <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0 21.9   23.4  23.3  25.0 0.785
## 2 0.1    50     0 12.1   13.7  13.7  15.7 0.858
## 3 0.3    50     0 12.5   14.1  13.9  15.1 0.871
## 4 0.6    50     0 12.8   14.1  14.1  15.9 0.801
## 5 1.2    50     0 13.6   14.9  14.9  15.8 0.801
## 6 2.5    50     0 13.4   14.8  14.7  16.1 0.713
## 7 5      50     0  9.66   11.6  11.6  12.7 0.854

```

Kruskal–Wallis test provides evidence of statistical differences for the best performance found in the population throughout 50,000 generations.

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

```

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 265.26, df = 6, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found in the population throughout 50,000 generations.

```

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##      0     0.1    0.3    0.6    1.2    2.5
## 0.1 <2e-16 -     -     -     -     -
## 0.3 <2e-16 1.00 -     -     -     -
## 0.6 <2e-16 1.00 1.00 -     -     -
## 1.2 <2e-16 1.00 1.00 1.00 -     -
## 2.5 <2e-16 1.00 1.00 1.00 0.76 -
## 5   <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 10.3 Contradictory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each phenotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

### 10.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

#### 10.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
problem <- filter(pfs_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

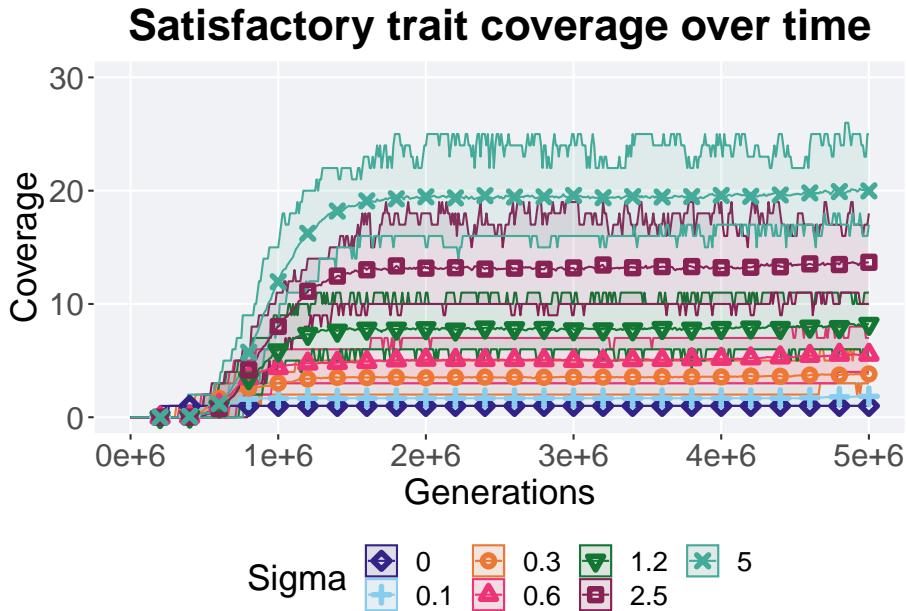
ot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 30)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
```

```

scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggttitle("Satisfactory trait coverage over time") +
p_theme

ot

```



### 10.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```

best = filter(pfs_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives')

plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  ) +
  scale_shape_manual(values=SHAPE) +

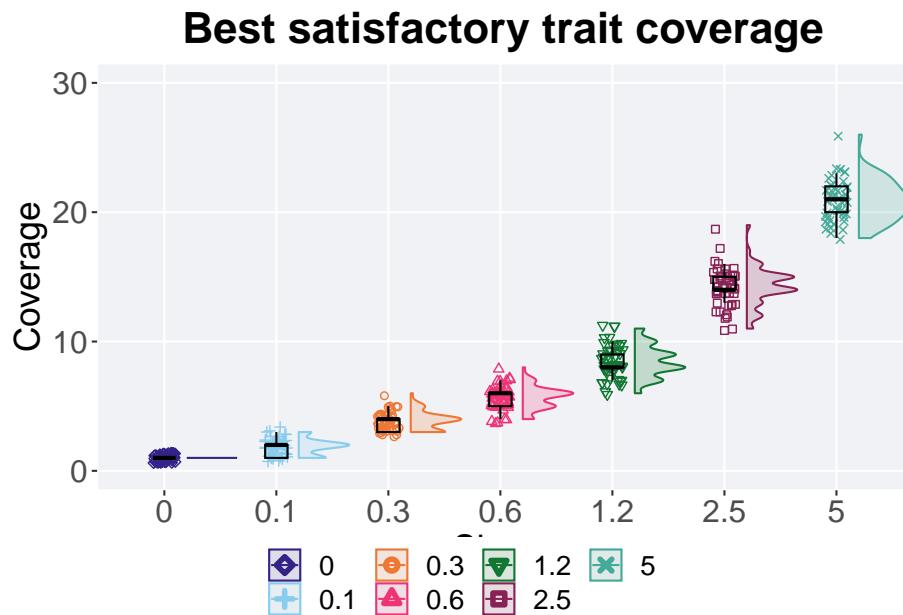
```

```

scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



#### 10.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```

group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    sd = sd(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>
## 1 0      50     0     1     1     1     1     0
## 2 0.1    50     0     1     2     1.84   3     1
## 3 0.3    50     0     3     4     3.84   6     1
## 4 0.6    50     0     4     6     5.66   8     1
## 5 1.2    50     0     6     8     8.46  11     1
## 6 2.5    50     0    11    14    14.2   19     1
## 7 5      50     0    18    21    20.9   26     2

```

Kruskal–Wallis test provides evidence of statistical difference among the best satisfactory trait coverage throughout 50,000 generations.

```

kruskal.test(val ~ Sigma, data = best)

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 338.57, df = 6, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best satisfactory trait coverage throughout 50,000 generations.

```

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##      0      0.1     0.3     0.6     1.2     2.5
## 0.1 6.8e-12 -      -      -      -      -
## 0.3 < 2e-16 3.4e-16 -      -      -      -
## 0.6 < 2e-16 < 2e-16 3.6e-13 -      -      -
## 1.2 < 2e-16 < 2e-16 < 2e-16 3.0e-15 -      -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
##
## P value adjustment method: bonferroni

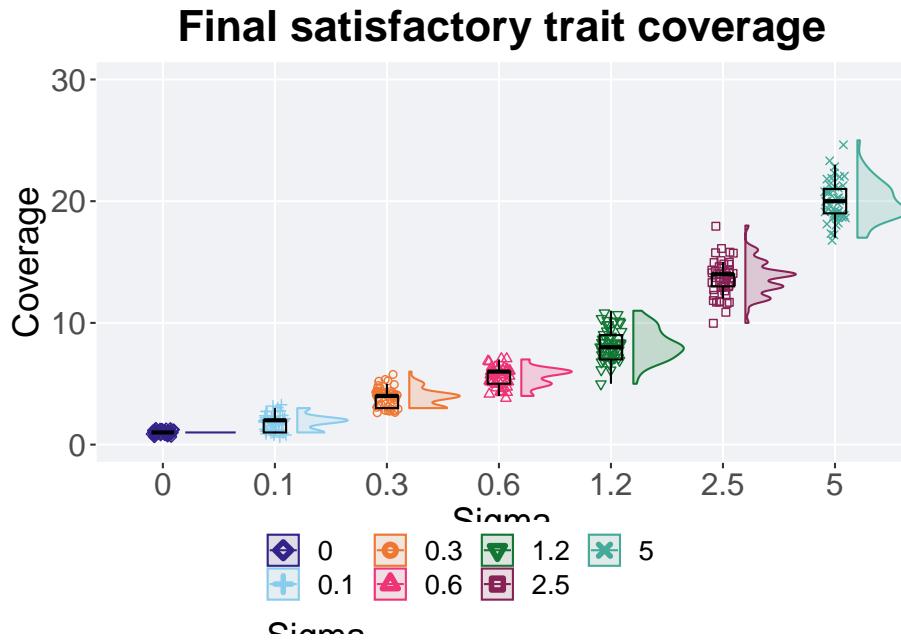
```

### 10.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
end = filter(pfs_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```



#### 10.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
group_by(end, 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> <dbl> <dbl> <dbl> <dbl>
## 1 0      50       0     1     1     1     1     0
## 2 0.1    50       0     1     2     1.84   3     1
## 3 0.3    50       0     3     4     3.82   6     1
## 4 0.6    50       0     4     6     5.54   7     1
## 5 1.2    50       0     5     8     8.22   11    2
## 6 2.5    50       0    10    14    13.7   18    1
## 7 5      50       0    17    20    20.0   25    2
```

Kruskal–Wallis test provides evidence of statistical difference among satisfactory trait coverage in the population at the end of 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = 338.19, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best satisfactory trait coverage in the population at the end of 50,000 generations.

```
pairwise.wilcox.test(x = end$pop_uni_obj, g = end$Sigma, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_uni_obj and end$Sigma
##
##      0      0.1     0.3     0.6     1.2     2.5
## 0.1 6.8e-12 -      -      -      -      -
## 0.3 < 2e-16 4.2e-16 -      -      -      -
## 0.6 < 2e-16 < 2e-16 8.6e-13 -      -      -
## 1.2 < 2e-16 < 2e-16 < 2e-16 5.4e-15 -      -
## 2.5 < 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
```

### 10.3.2 Activation gene coverage

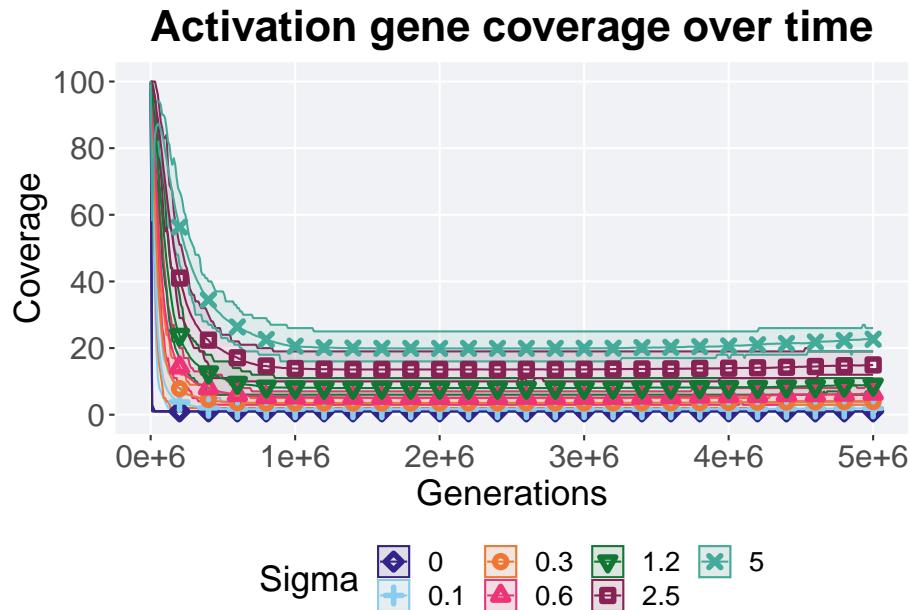
Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

#### 10.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(pfs_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.  
points = filter(lines, gen %% 2000 == 0 & gen != 0)  
  
ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +  
  scale_y_continuous(  
    name="Coverage",  
    limits=c(-1, 101),  
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")  
) +  
  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  
ot
```



#### 10.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

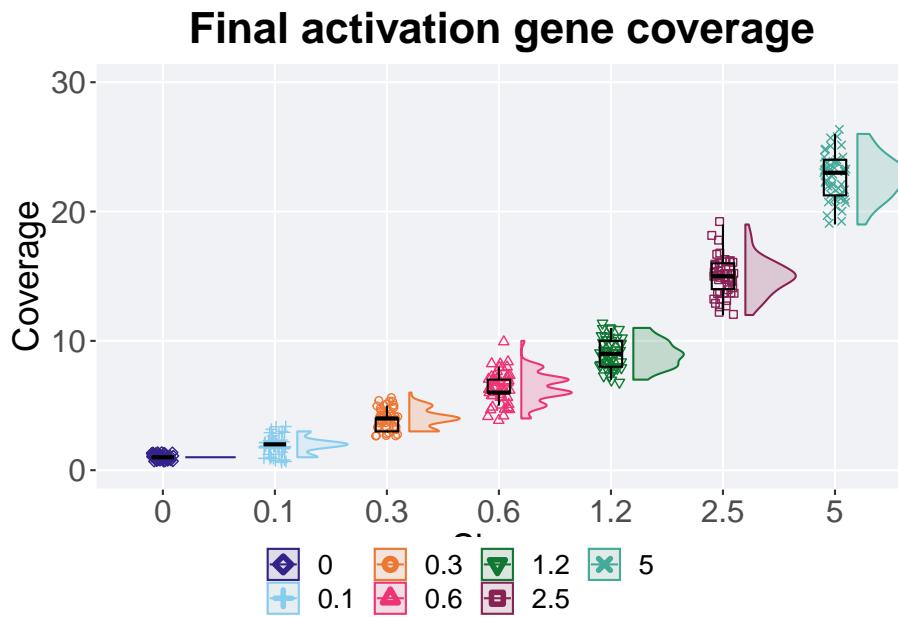
```
end = filter(pfs_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma))
  plot + geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="Coverage",
      limits=c(0, 30)
    ) +
    scale_x_discrete(
      name="Sigma"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  nrow = 1, ncol = 1)
```

```

    legend,
    nrow=2,
    rel_heights = c(2,.2),
    label_size = TSIZE
)

```



#### 10.3.2.2.1 Stats

Summary statistics for the best activation gene coverage in the population at the end of 50,000 generations.

```

group_by(end, 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> <dbl> <dbl> <dbl> <int> <dbl>
## 1 0         50       0     1      1     1     1     0

```

```
## 2 0.1      50     0     1      2   1.92     3   0
## 3 0.3      50     0     3      4   3.98     6   1
## 4 0.6      50     0     4      6   6.38    10   1
## 5 1.2      50     0     7      9   8.98    11   2
## 6 2.5      50     0    12     15  14.9    19   2
## 7 5        50     0    19     23 22.7    26  2.75
```

Kruskal–Wallis test provides evidence of no statistical difference among activation gene coverage in the population at the end of 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 338.71, df = 6, p-value < 2.2e-16
```

## 10.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each phenotypic fitness sharing sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

### 10.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

#### 10.4.1.1 Performance over time

Performance over time.

```
problem <- filter(pfs_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

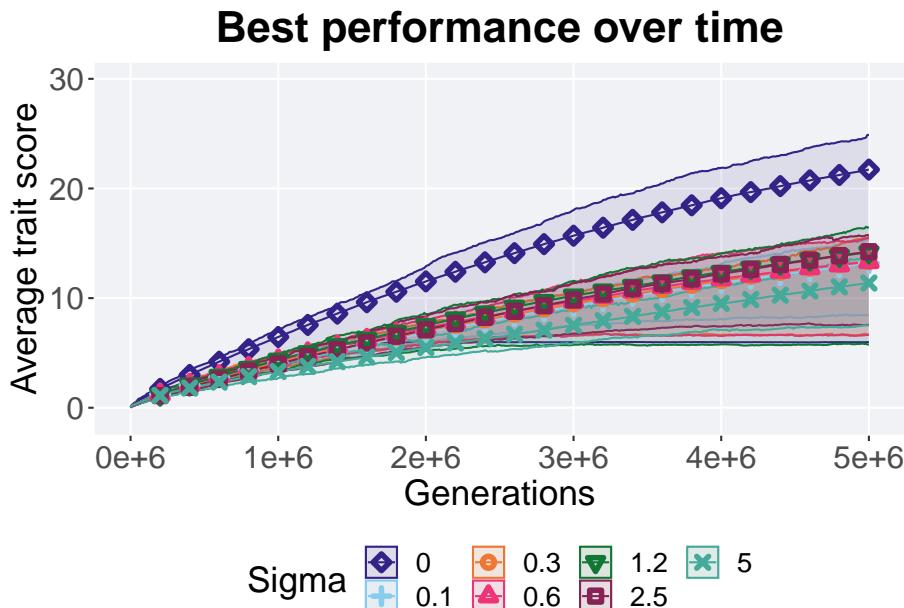
```

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme

ot

```



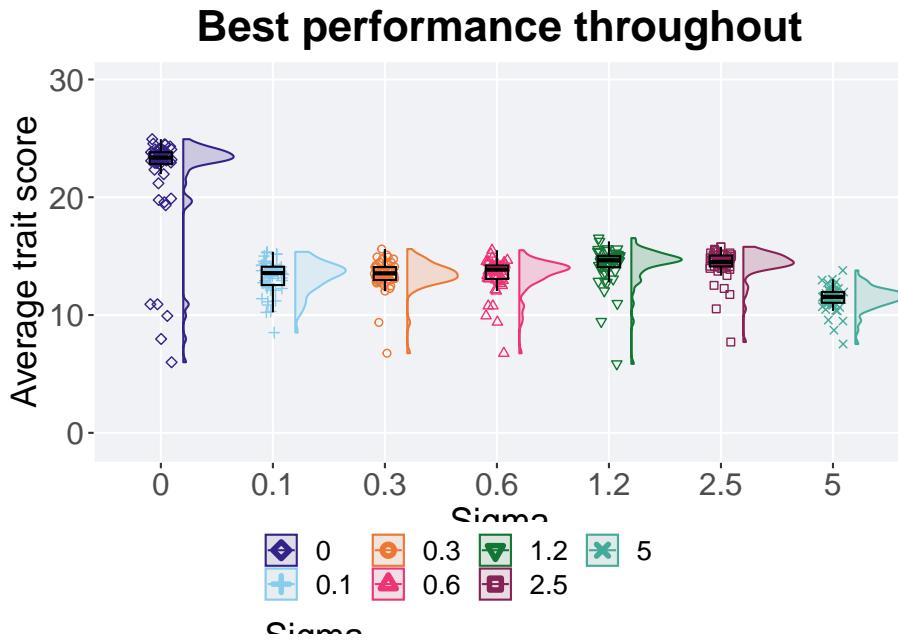
#### 10.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(pfs_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 30)
  ) +
  scale_x_discrete(
    name = "Sigma"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .3),
  label_size = TSIZE
)
```



#### 10.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  5.99  23.4  21.8  24.9  0.986
## 2 0.1      50      0  8.51  13.6  13.2  15.4  1.55
## 3 0.3      50      0  6.76  13.5  13.4  15.6  1.10
## 4 0.6      50      0  6.75  13.9  13.4  15.5  1.14
## 5 1.2      50      0  5.86  14.6  14.2  16.5  0.949
## 6 2.5      50      0  7.71  14.5  14.3  15.8  0.925
## 7 5        50      0  7.54  11.5  11.4  13.8  0.914
```

Kruskal–Wallis test provides evidence of statistical differences among the best performing solution found throughout 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 175.44, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performing solution found throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##      0     0.1    0.3    0.6    1.2    2.5
## 0.1 7.5e-11 -     -     -     -     -
## 0.3 8.3e-11 1.00000 -     -     -     -
## 0.6 6.2e-11 1.00000 1.00000 -     -     -
## 1.2 7.9e-11 0.00014 0.00020 0.00019 -     -
## 2.5 8.7e-11 3.9e-05 2.8e-05 7.0e-05 1.00000 -
## 5   3.1e-11 1.1e-08 2.4e-12 3.9e-10 1.3e-12 1.9e-13
##
## P value adjustment method: bonferroni
```

#### 10.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

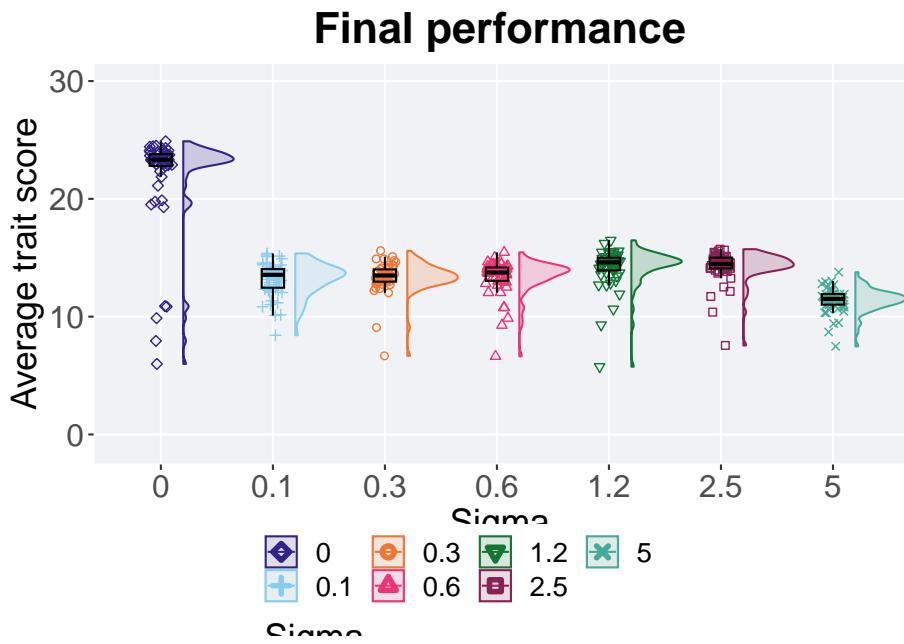
```
end = filter(pfs_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, color = Sigma, fill = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 30)
  ) +
  scale_x_discrete(
    name = "Sigma"
  ) +
```

```

scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Final performance") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 10.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```

group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS),
    mean = mean(pop_fit_max / TRAITS),
    median = median(pop_fit_max / TRAITS)
  )

```

```

median = median(pop_fit_max / TRAITS, na.rm = TRUE),
mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
max = max(pop_fit_max / TRAITS, na.rm = TRUE),
IQR = IQR(pop_fit_max / TRAITS, 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  5.99  23.3  21.7  24.9  0.997
## 2 0.1    50     0  8.42  13.6  13.2  15.4  1.58 
## 3 0.3    50     0  6.67  13.5  13.3  15.6  1.04 
## 4 0.6    50     0  6.63  13.7  13.3  15.5  1.14 
## 5 1.2    50     0  5.76  14.6  14.2  16.5  1.08 
## 6 2.5    50     0  7.56  14.5  14.2  15.7  0.915
## 7 5      50     0  7.48  11.5  11.4  13.8  0.898

```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ Sigma, data = end)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 175.09, df = 6, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance in the population at the end of 50,000 generations.

```

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$Sigma
##
##          0       0.1      0.3      0.6      1.2      2.5
## 0.1 7.5e-11 -       -       -       -       -
## 0.3 8.3e-11 1e+00 -       -       -       -
## 0.6 6.2e-11 1e+00 1e+00 -       -       -
## 1.2 7.2e-11 2e-04 2e-04 2e-04 -       -
## 2.5 8.7e-11 3.4e-05 2.1e-05 3.9e-05 1e+00 -
## 5    3.1e-11 1.3e-08 2.4e-12 4.9e-10 2.2e-12 2.1e-13
##
## P value adjustment method: bonferroni

```

### 10.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

#### 10.4.2.1 Coverage over time

Activation gene coverage over time.

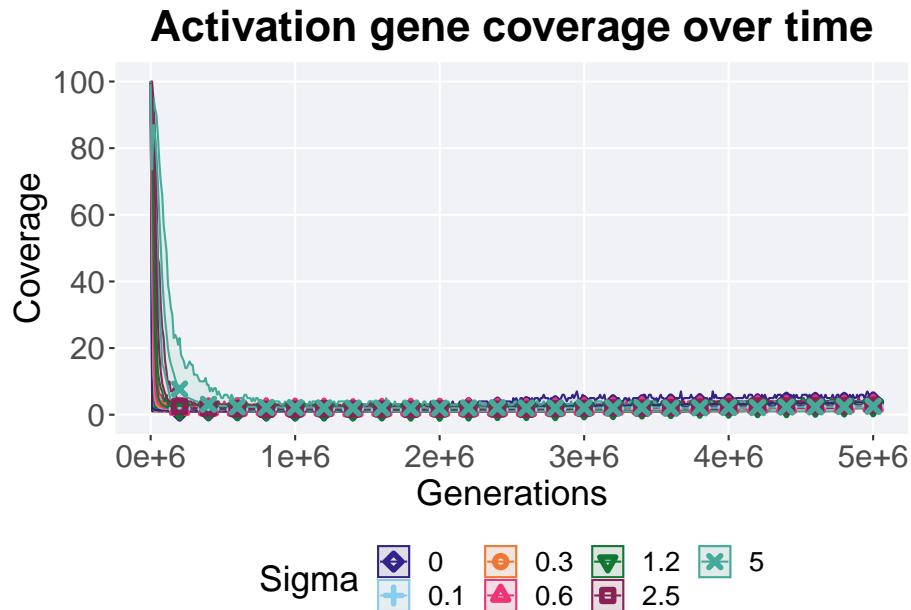
```
problem <- filter(pfs_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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

ot
```



#### 10.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

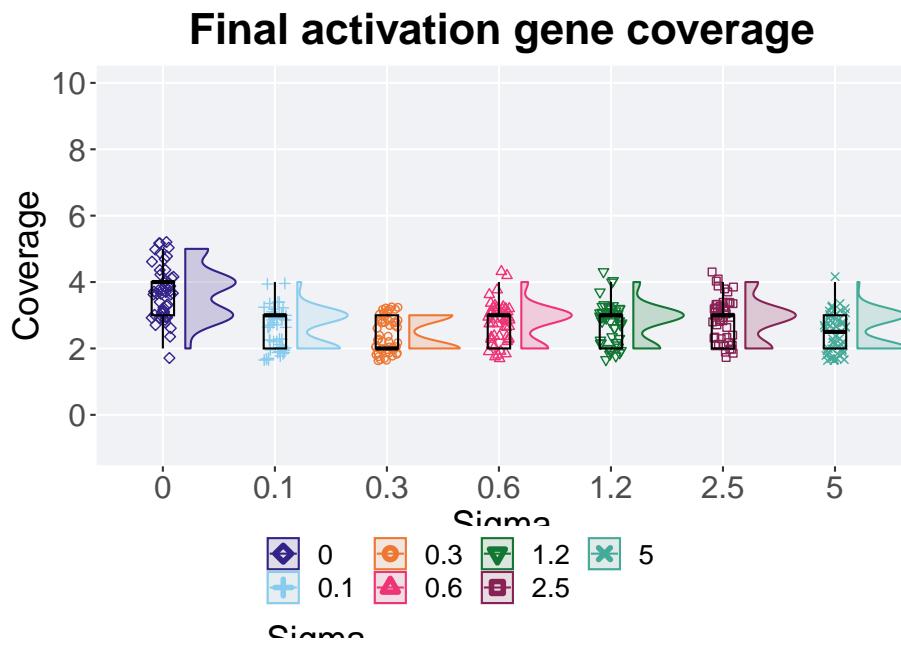
```
end = filter(pfs_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 10),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10"))
  ) +
  scale_x_discrete(
    name="Sigma")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```

ggtile("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 10.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```

group_by(end, 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
##   <dbl> <int>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     0     7     0.0  0.00  0.00  0.00  0.00  0.00
## 2     0.1    7     0.0  0.00  0.00  0.00  0.00  0.00
## 3     0.3    7     0.0  0.00  0.00  0.00  0.00  0.00
## 4     0.6    7     0.0  0.00  0.00  0.00  0.00  0.00
## 5     1.2    7     0.0  0.00  0.00  0.00  0.00  0.00
## 6     2.5    7     0.0  0.00  0.00  0.00  0.00  0.00
## 7      5     7     0.0  0.00  0.00  0.00  0.00  0.00

```

```
##   <fct> <int> <int> <int> <dbl> <dbl> <int> <dbl>
## 1 0      50     0     2     4    3.72     5     1
## 2 0.1    50     0     2     3    2.6      4     1
## 3 0.3    50     0     2     2    2.46     3     1
## 4 0.6    50     0     2     3    2.76     4     1
## 5 1.2    50     0     2     3    2.76     4     1
## 6 2.5    50     0     2     3    2.84     4     1
## 7 5      50     0     2     2.5   2.52     4     1
```

Kruskal-Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 81.793, df = 6, p-value = 1.522e-15
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the activation gene coverage in the population at the end of 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$uni_str_pos and end$Sigma
##
##   0      0.1    0.3    0.6    1.2    2.5
## 0.1 4.4e-09 -      -      -      -      -
## 0.3 5.8e-11 1.000 -      -      -      -
## 0.6 2.9e-07 1.000 0.250 -      -      -
## 1.2 2.9e-07 1.000 0.250 1.000 -      -
## 2.5 5.8e-06 1.000 0.069 1.000 1.000 -
## 5   4.3e-10 1.000 1.000 0.915 0.915 0.283
##
## P value adjustment method: bonferroni
```

## 10.5 Multi-valley crossing results

Here we present the results for the **best performances** and **best gene value** generated by each phenotypic fitness sharing replicate on the multi-valley crossing diagnostic. Best performance found refers to the largest average trait score found in a given population. Best gene value refers to maximum gene value found in

the population; this gives us a range of values in [0.0, 100.0]. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, no similarity penalty is used, and only stochastic remainder selection is used to identify parent solutions.

### 10.5.1 Performance

Here we analyze the performances for each parameter replicate on the multi-valley crossing diagnostic.

#### 10.5.1.1 Performance over time

Performance over time.

```
lines = filter(pfs_ot, diagnostic == 'multivalley_crossing') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

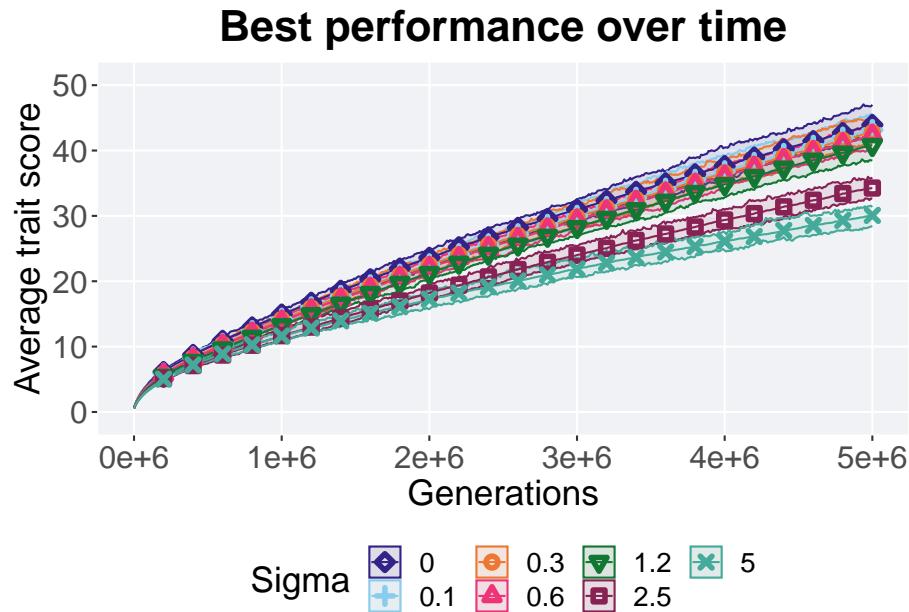
## `summarise()` has grouped output by 'Sigma'. You can override using the
## `.`groups` argument.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma,
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  ggttitle("Best performance over time") +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
```

```
scale_fill_manual(values = cb_palette) +
p_theme
```

```
ot
```



### 10.5.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

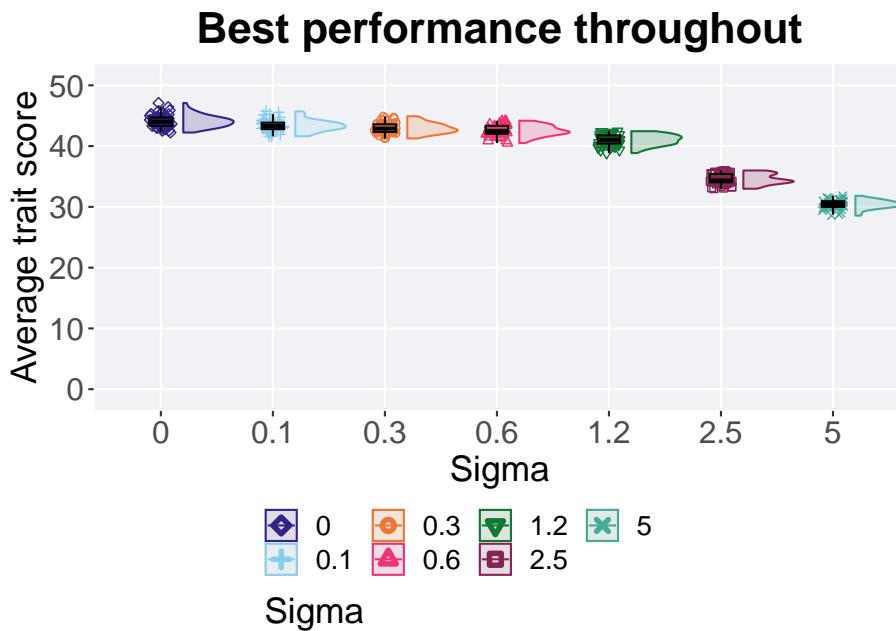
```
best = filter(pfs_best, col == 'pop_fit_max' & diagnostic == 'multivalley_crossing')
plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0)
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  guides(fill = "none", color = 'none', shape = 'none') +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 51),
    breaks = seq(0, 50, 10),
    labels = c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_discrete(
    name = "Sigma"
  ) +
  scale_shape_manual(values = SHAPE) +
```

```

    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    p_theme

plot_grid(
  plot +
  ggtitle("Best performance throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(.2,.55),
  label_size = TSIZE
)

```



#### 10.5.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```

best$Sigma = factor(best$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0))
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    sd = sd(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 4224. 4401. 4411. 4709. 136.
## 2 0.1    50     0 4163. 4332. 4336. 4572. 106.
## 3 0.3    50     0 4127. 4288. 4296. 4492. 126.
## 4 0.6    50     0 4050. 4250. 4254. 4417. 131.
## 5 1.2    50     0 3885. 4104. 4101. 4245. 132.
## 6 2.5    50     0 3299. 3434. 3456. 3599. 136.
## 7 5      50     0 2856. 3045. 3041. 3182. 94.0

```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

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

```
##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 289.29, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

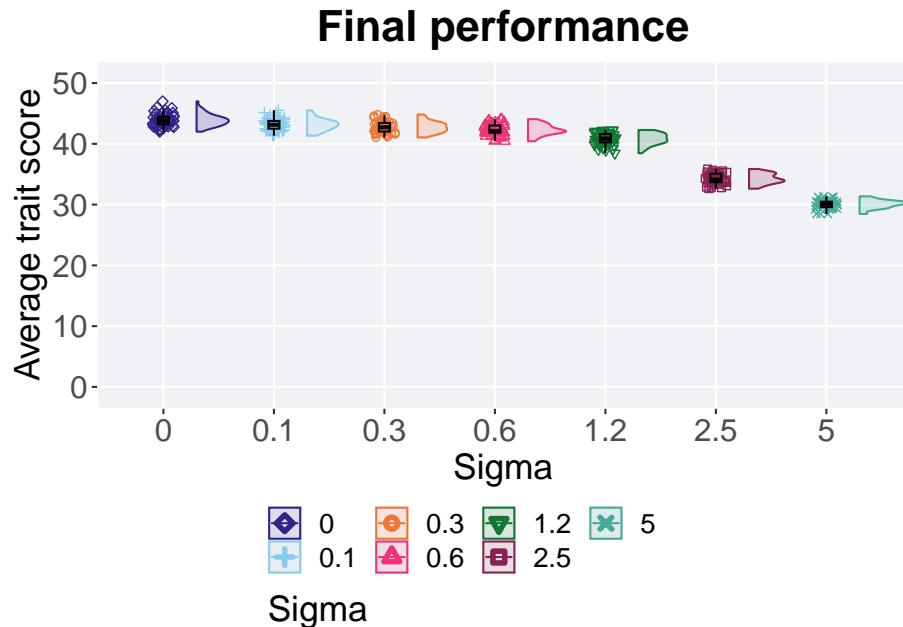
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##          0       0.1      0.3      0.6      1.2      2.5
## 0.1 0.00466 -       -       -       -       -
## 0.3 2.3e-06 0.38235 -       -       -       -
## 0.6 6.8e-10 0.00072 0.36959 -       -       -
## 1.2 < 2e-16 3.6e-15 1.7e-13 3.1e-10 -       -
## 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
```

### 10.5.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```
end = filter(pfs_end, diagnostic == 'multivalley_crossing')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_flat_violin(position = position_nudge(x = .3), alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 10.8) +
  geom_boxplot(width = .1, outlier.shape = NA, alpha = 0.1, col = "black") +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 51),
    breaks = seq(0, 50, 10),
    labels = c("0", "10", "20", "30", "40", "50")
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .55),
  label_size = TSIZE
)
```



#### 10.5.1.3.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
end$Sigma = factor(end$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 5.0, 2.5))
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max, na.rm = TRUE),
    median = median(pop_fit_max, na.rm = TRUE),
    mean = mean(pop_fit_max, na.rm = TRUE),
    max = max(pop_fit_max, na.rm = TRUE),
    IQR = IQR(pop_fit_max, 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 4200. 4383. 4392. 4699. 124.
## 2 0.1       50      0 4134. 4315. 4315. 4552. 128.
## 3 0.3       50      0 4105. 4272. 4275. 4482. 140.
## 4 0.6       50      0 4043. 4211. 4232. 4407. 117.
## 5 1.2       50      0 3845. 4087. 4081. 4228. 135.
## 6 5         50      0 2847. 3014. 3007. 3136. 88.1
```

```
## 7 2.5      50      0 3262. 3411. 3426. 3586. 139.
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

```
kruskal.test(pop_fit_max ~ Sigma, data = end)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 287.58, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$Sigma
##
##          0       0.1      0.3      0.6      1.2      5
## 0.1 0.0040 -       -       -       -       -
## 0.3 2.9e-06 0.4905 -       -       -       -
## 0.6 5.2e-10 0.0011 0.4300 -       -       -
## 1.2 < 2e-16 1.3e-14 9.9e-13 7.8e-10 -       -
## 5    < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.0000
##
## P value adjustment method: bonferroni
```

## 10.5.2 Best gene value

Here we analyze the gene values for each parameter replicate on the multi-valley crossing diagnostic.

### 10.5.2.1 Gene value over time

Best gene value over time.

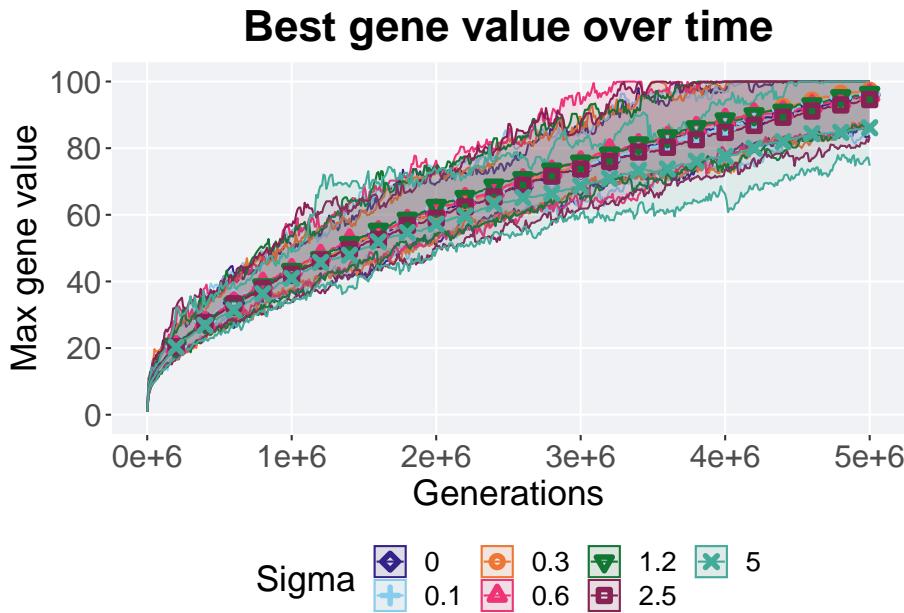
```
lines = filter(pfs_ot, diagnostic == 'multivalley_crossing') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_max_gene),
    mean = mean(pop_max_gene),
```

```
    max = max(pop_max_gene)
)
```

```
## `summarise()` has grouped output by 'Sigma'. You can override using the
## `.groups` argument.
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Max gene value",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  ggtitle("Best gene value over time") +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

ot
```



### 10.5.2.2 Best gene value throughout

The best gene value found throughout 50,000 generations.

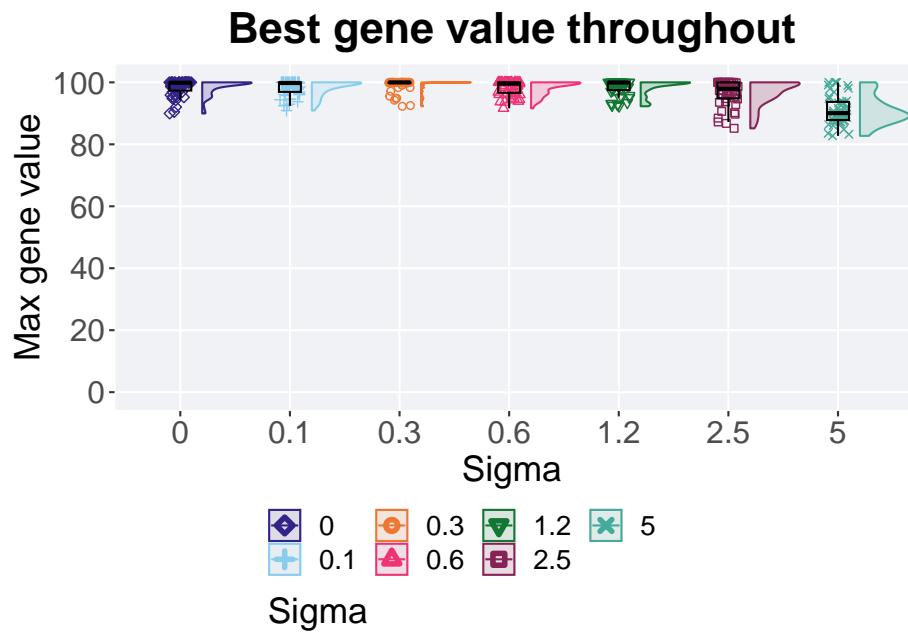
```
best = filter(pfs_best, col == 'pop_max_gene' & diagnostic == 'multivalley_crossing')
plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  guides(fill = "none", color = 'none', shape = 'none') +
  scale_y_continuous(
    name = "Max gene value",
    limits = c(-1, 101),
    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) +
  p_theme

plot_grid(
```

```

plot +
  ggtitle("Best gene value throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 10.5.2.2.1 Stats

Summary statistics for the best gene value found throughout 50,000 generations.

```

best$Sigma = factor(best$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0))
group_by(best, 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
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0  90.0  99.9  98.3  100  2.64
## 2 0.1    50     0  90.9  99.8  98.1  100  3.07
## 3 0.3    50     0  92.3  100.   99.1  100  0.195
## 4 0.6    50     0  91.6  99.6  98.2  100  3.36
## 5 1.2    50     0  92.3  100.   98.3  100  2.49
## 6 2.5    50     0  85.2  97.9  96.6  100  5.17
## 7 5      50     0  82.7  90.1  91.1  100  5.86
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best gene value found throughout 50,000 generations

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

```
##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 86.457, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best gene value found throughout 50,000 generations.

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

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$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 0.2233 -     -     -
## 1.2 1.0000 1.0000 1.0000 1.0000 -     -
## 2.5 0.4593 0.9100 0.0038 1.0000 0.4593 -
## 5   2.6e-10 4.9e-10 3.1e-12 2.4e-10 2.3e-10 2.0e-06
##
## P value adjustment method: bonferroni
```

### 10.5.2.3 End of 50,000 generations

Best gene value in the population at the end of 50,000 generations.

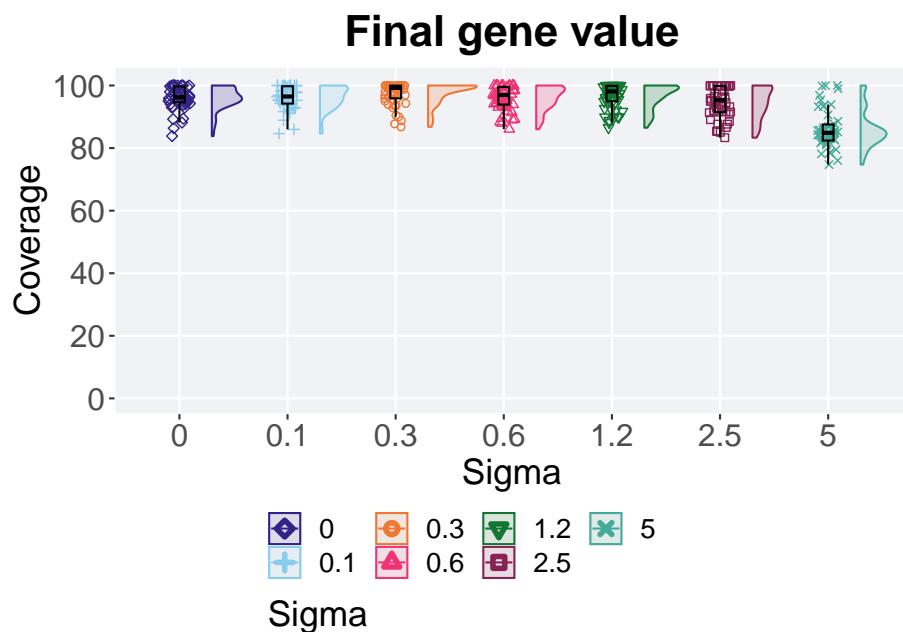
```
end = filter(pfs_end, diagnostic == 'multivalley_crossing')
plot = ggplot(end, aes(x = Sigma, y = pop_max_gene, group = Sigma, fill = Sigma, color = Sigma, sh
```

```

geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 10.8) +
  geom_boxplot(width = .1, outlier.shape = NA, alpha = 0.1, col = "black") +
  scale_y_continuous(
    name = "Coverage",
    limits = c(0, 101),
    breaks = seq(0, 100, 20),
    labels = c("0", "20", "40", "60", "80", "100")
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
  ggtitle("Final gene value") +
  theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .55),
  label_size = TSIZE
)

```



#### 10.5.2.3.1 Stats

Summary statistics for the best gene value found at the end of 50,000 generations.

```
end$Sigma = factor(end$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 5.0, 2.5))
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_max_gene)),
    min = min(pop_max_gene, na.rm = TRUE),
    median = median(pop_max_gene, na.rm = TRUE),
    mean = mean(pop_max_gene, na.rm = TRUE),
    max = max(pop_max_gene, na.rm = TRUE),
    IQR = IQR(pop_max_gene, 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  83.8   96.3   96.0  100   5.07
## 2 0.1       50     0  84.6   96.5   96.1  100   5.55
## 3 0.3       50     0  86.8   99.2   97.4  100   4.06
## 4 0.6       50     0  86.1   97.0   96.0  100   5.61
## 5 1.2       50     0  86.5   98.1   96.5  100   4.91
## 6 5         50     0  74.7   84.9   86.1  100   5.13
## 7 2.5       50     0  83.3   95.3   94.6  100   8.37
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best gene value found at the end of 50,000 generations.

```
kruskal.test(pop_max_gene ~ Sigma, data = end)

##
## Kruskal-Wallis rank sum test
##
## data: pop_max_gene by Sigma
## Kruskal-Wallis chi-squared = 87.784, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best gene value found at the end of 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_max_gene and end$Sigma
##
##      0      0.1      0.3      0.6      1.2      5
## 0.1 1.000  -      -      -      -      -
```

```
## 0.3 1.000 1.000 - - - -  
## 0.6 1.000 1.000 0.280 - - -  
## 1.2 1.000 1.000 1.000 1.000 - -  
## 5 1.9e-10 1.8e-10 4.4e-12 1.5e-10 4.4e-11 -  
## 2.5 1.000 1.000 0.035 1.000 0.663 1.000  
##  
## P value adjustment method: bonferroni
```

## Chapter 11

# Nondominated sorting

We present the results from our parameter sweep on genotypic fitness sharing. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, performance with no similarity penalty and stochastic remainder selection is used to identify parent solutions.

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

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         Patched
## major          4
## minor          2.2
## year           2022
## month          11
## day            10
## svn rev        83330
## language       R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname       Innocent and Trusting
```

## 11.1 Exploitation rate results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0. Note that when `sigma = 0.0`, no similarity penalty is used within nondominated front, and then stochastic remainder selection is used to identify parent solutions.

### 11.1.1 Performance over time

Performance over time.

```
problem <- filter(nds_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

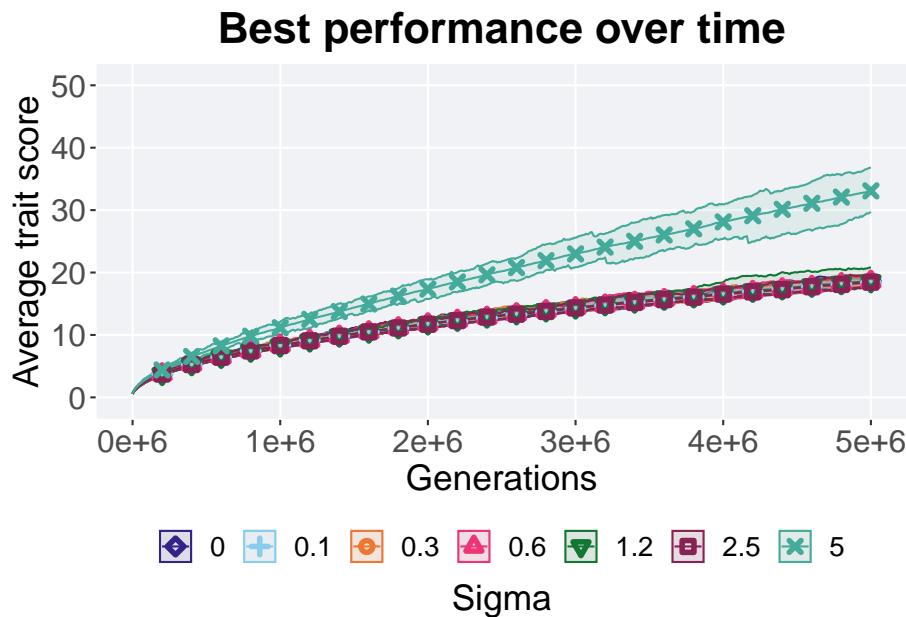
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
```

```

ggttitle("Best performance over time") +
p_theme+
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.justification="center",
    legend.title.align=0.5
  )
)

```

ot



### 11.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```

best = filter(nds_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +

```

```

geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_discrete(
    name="Sigma"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



### 11.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
  )

## # A tibble: 7 x 8
##   Sigma count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0  17.9  18.6  18.7  19.6 0.568
## 2 0.1    50     0  17.9  18.6  18.6  19.3 0.482
## 3 0.3    50     0  17.9  18.4  18.5  19.5 0.516
## 4 0.6    50     0  18.0  18.7  18.7  19.6 0.542
## 5 1.2    50     0  17.8  18.7  18.8  20.9 0.481
## 6 2.5    50     0  17.8  18.5  18.6  19.7 0.572
## 7 5      50     0  29.7  33.0  33.2  36.9 2.17
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

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

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 135.65, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

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

```

##      0     0.1    0.3    0.6    1.2    2.5
## 0.1 1.00  -     -     -     -     -
## 0.3 0.93 1.00  -     -     -     -
## 0.6 1.00 1.00  0.37  -     -     -
## 1.2 1.00 1.00  0.10  1.00  -     -
## 2.5 1.00 1.00  1.00  1.00  1.00  -
## 5   <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 11.2 Ordered exploitation results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0. Note that when `sigma = 0.0`, no similarity penalty is used within nondominated front, and then stochastic remainder selection is used to identify parent solutions.

### 11.2.1 Performance over time

Performance over time.

```

problem <- filter(nds_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

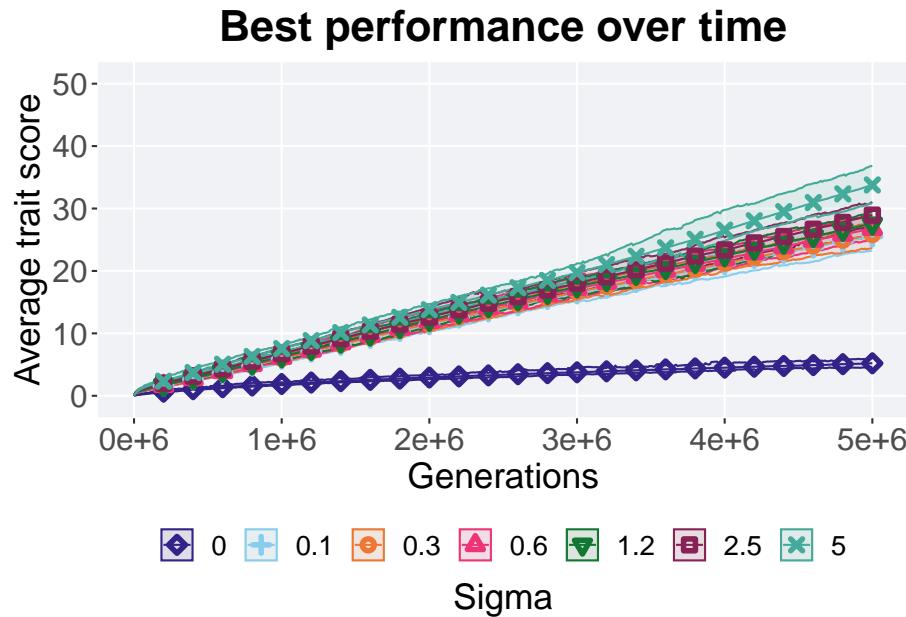
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")

```

```
) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6"))

) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.justification="center",
    legend.title.align=0.5
  )
)
ot
```



#### 11.2.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(nds_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')

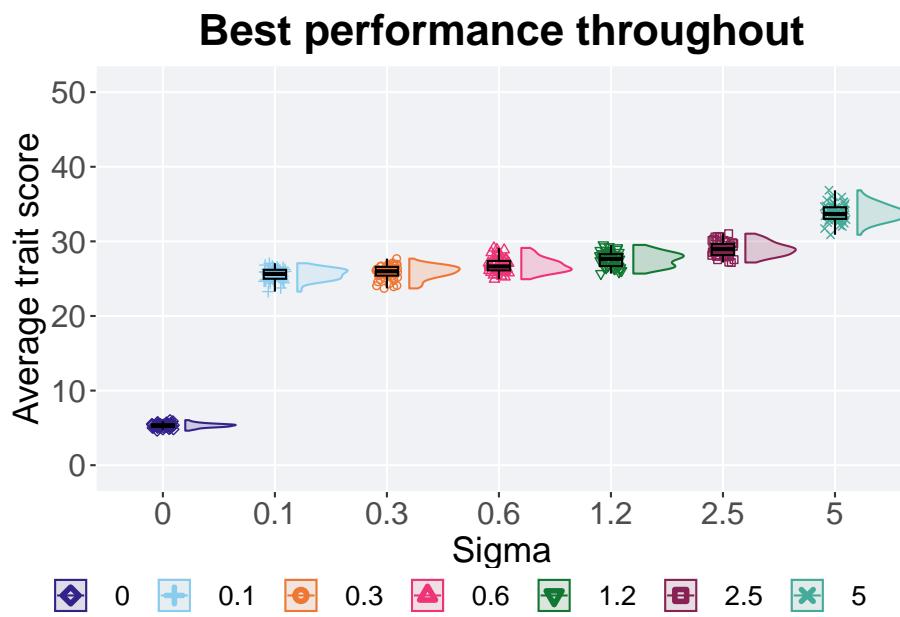
plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 51),
    breaks = seq(0, 50, 10),
    labels = c("0", "10", "20", "30", "40", "50"))
  ) +
  scale_x_discrete(
    name = "Sigma")
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
```

```

plot +
  ggtitle("Best performance throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



### 11.2.2.1 Stats

Summary statistics about the best performance found.

```

group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
  )

## # A tibble: 7 x 8
##   Sigma count na_cnt   min median   mean   max   IQR
#>   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1     0     10      0  5.00  5.00  5.00  5.00  5.00
#> 2     0.1    10      0 23.0  25.0  25.0  28.0  23.0
#> 3     0.3    10      0 25.0  26.0  26.0  28.0  25.0
#> 4     0.6    10      0 27.0  28.0  28.0  29.0  27.0
#> 5     1.2    10      0 26.0  27.0  27.0  28.0  26.0
#> 6     2.5    10      0 28.0  29.0  29.0  30.0  28.0
#> 7     5      10      0 32.0  33.0  33.0  36.0  32.0

```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0 4.63   5.32  5.32  6.04 0.313
## 2 0.1    50     0 23.3   25.6  25.6  27.1 1.20
## 3 0.3    50     0 23.7   26.0  25.9  27.7 1.17
## 4 0.6    50     0 24.9   26.6  26.8  29.1 1.26
## 5 1.2    50     0 25.7   27.6  27.6  29.5 1.59
## 6 2.5    50     0 27.2   29.0  29.0  31.0 1.47
## 7 5      50     0 30.9   33.7  33.8  36.8 1.56
```

Kruskal–Wallis test provides evidence of statistical differences for the best performance found in the population throughout 50,000 generations.

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

```
##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 301.08, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found in the population throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$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 3.1e-07 0.00059 -      -      -
## 1.2 < 2e-16 2.1e-13 1.4e-10 0.00536 -      -
## 2.5 < 2e-16 < 2e-16 < 2e-16 9.0e-13 1.7e-07 -
## 5    < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

### 11.3 Contradictory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the contradictory objectives diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage

refers to the count of unique activation genes in the population. Note that both coverage values fall between 0.0 and 100.0. Note that when `sigma = 0.0`, no similarity penalty is used within nondominated front, and then stochastic remainder selection is used to identify parent solutions.

### 11.3.1 Satisfactory trait coverage

Here we analyze the satisfactory trait coverage for each parameter replicate on the contradictory objectives diagnostic.

#### 11.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
problem <- filter(nds_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

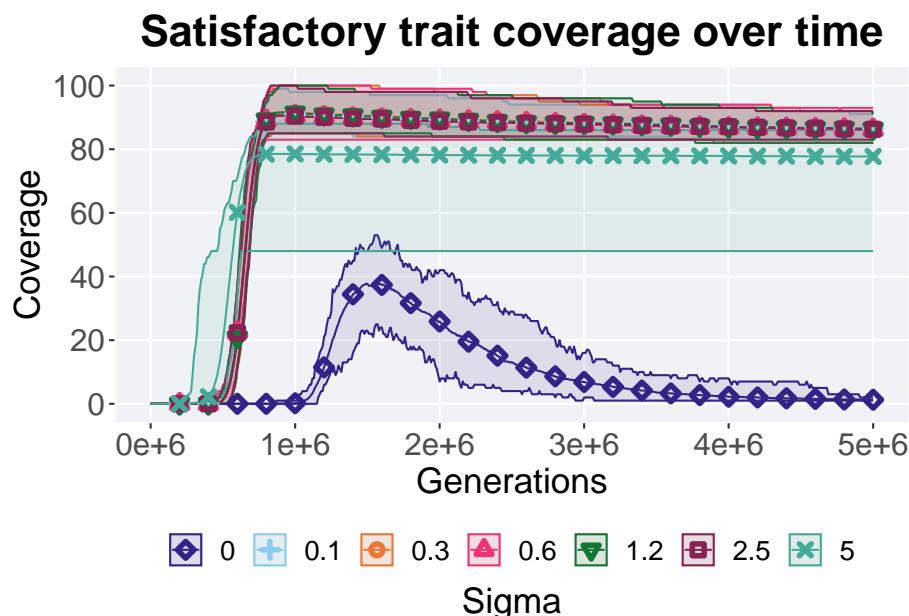
ot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma))
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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 +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.justification="center",
    legend.title.align=0.5
  )
)

```

ot



### 11.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```

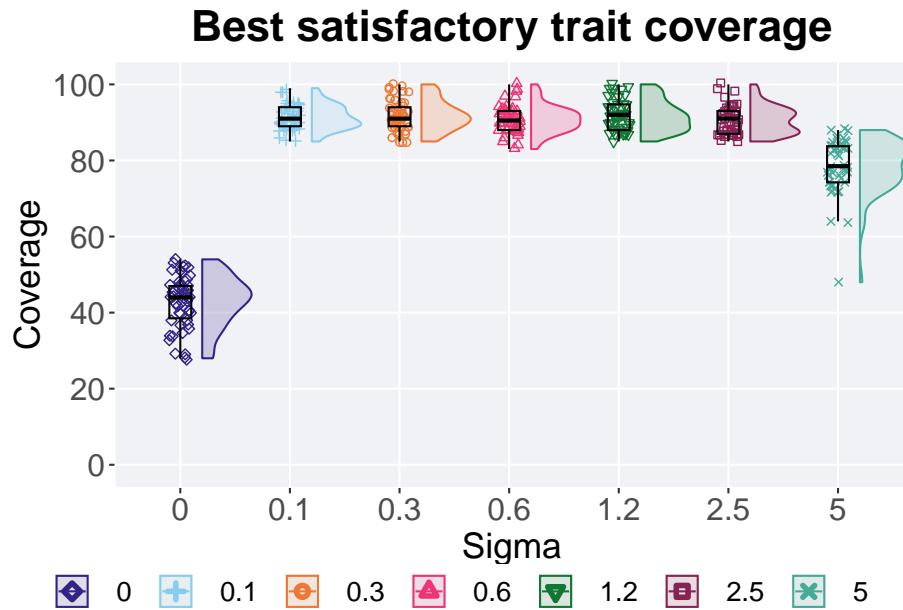
best = filter(nds_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives')

plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5)

```

```
geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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) +
  p_theme

plot_grid(
  plot +
  ggtitle("Best satisfactory trait coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```



#### 11.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```
group_by(best, 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
##   <dbl> <int>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0      50     0     28    44    42.8    54    8.5
## 2 0.1    50     0     85    91    91.4    99    5
## 3 0.3    50     0     85    91    91.8    100   5
## 4 0.6    50     0     83    90.5   90.9    100   5
## 5 1.2    50     0     85    92    91.8    100   6.75
## 6 2.5    50     0     85    91    90.7    100   6
## 7 5      50     0     48    78.5   78.6    88    9.5
```

Kruskal–Wallis test provides evidence of statistical difference for the best satisfactory trait coverage throughout 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 214.97, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction for the best satisfactory trait coverage throughout 50,000 generations..

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$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    2.6e-16 7.5e-16 1.7e-15 4.9e-15 1.7e-15 6.9e-15
##
## P value adjustment method: bonferroni
```

### 11.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

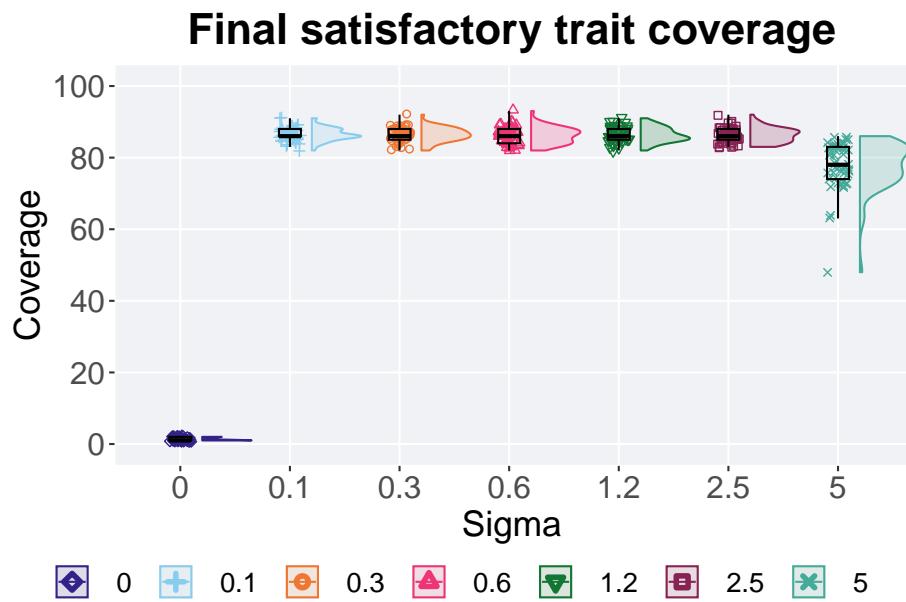
```
end = filter(nds_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Coverage",
    limits = c(-1, 101),
    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) +
p_theme

plot_grid(
  plot +
  ggtitle("Final satisfactory trait coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 11.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```

group_by(end, 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> <dbl> <dbl> <int> <dbl>
## 1 0      50     0     1     1  1.28     2     1
## 2 0.1    50     0    82    86  86.5     91     2
## 3 0.3    50     0    82    86  86.4     92     3
## 4 0.6    50     0    82    86  86.2     93     4
## 5 1.2    50     0    82    86  86.4     91     3
## 6 2.5    50     0    83    86  86.3     92     3
## 7 5      50     0    48    78  77.7     86     9

```

Kruskal–Wallis test provides evidence of statistical difference for satisfactory trait coverage in the population at the end of 50,000 generations.

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

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = 205.41, df = 6, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction for satisfactory trait coverage in the population at the end of 50,000 generations.

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

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_uni_obj and end$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 1.8e-14 8.5e-14 9.2e-13 1.1e-13 2.2e-13

```

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

### 11.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

#### 11.3.2.1 Coverage over time

Activation gene coverage over time.

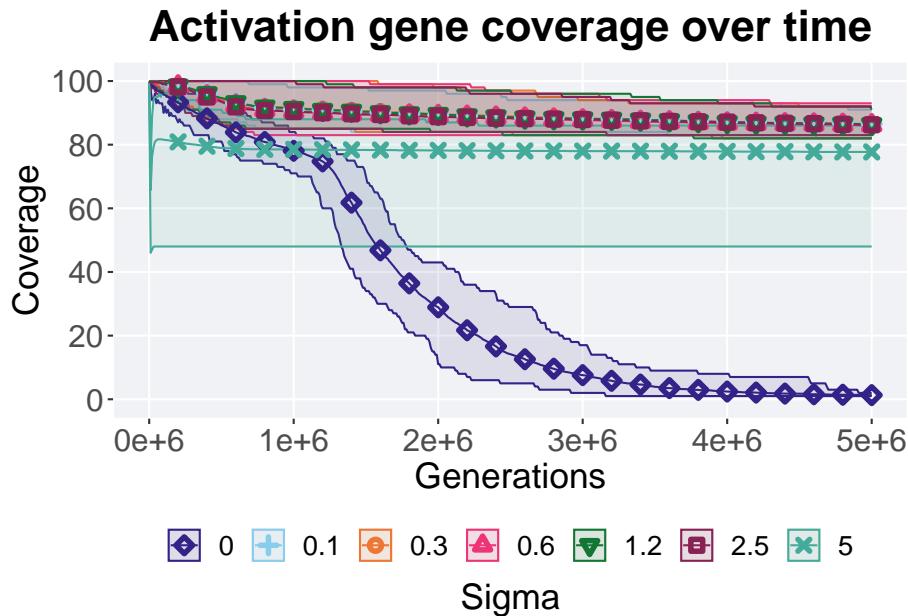
```
problem <- filter(nds_ot, diagnostic == 'contradictory_objectives')  
lines = problem %>%  
  group_by(Sigma, gen) %>%  
  dplyr::summarise(  
    min = min(uni_str_pos),  
    mean = mean(uni_str_pos),  
    max = max(uni_str_pos)  
  )  
  
## `summarise()` has grouped output by 'Sigma'. You can override using the  
## ` `.groups` argument.  
points = filter(lines, gen %% 2000 == 0 & gen != 0)  
  
ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +  
  scale_y_continuous(  
    name="Coverage",  
    limits=c(-1, 101),  
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")  
  ) +  
  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 +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.justification="center",
    legend.title.align=0.5
  )
)

ot

```



### 11.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```

end = filter(nds_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous()

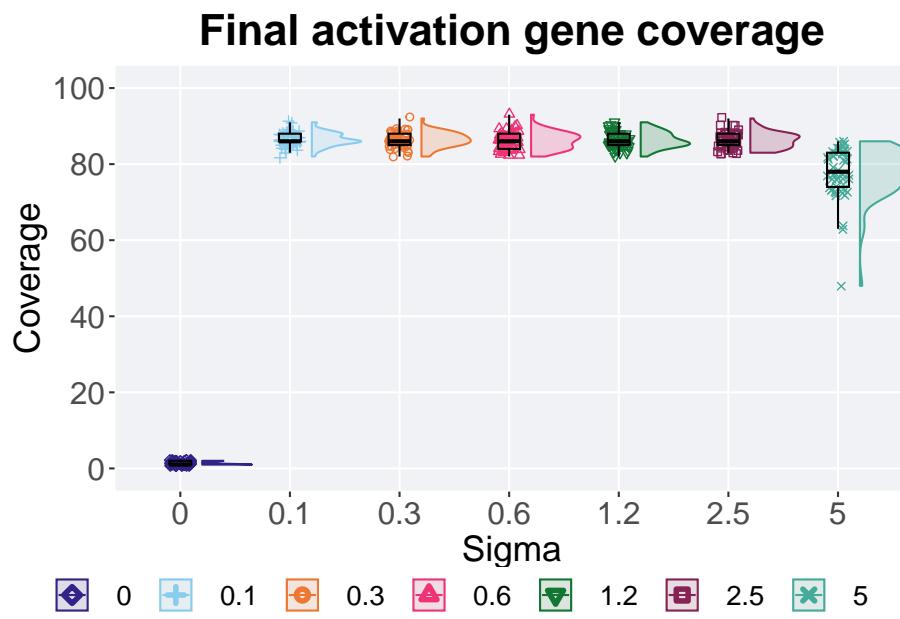
```

```

name="Coverage",
limits=c(-1, 101),
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) +
p_theme

plot_grid(
  plot +
  ggtitle("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

```



#### 11.3.2.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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> <dbl> <dbl> <dbl> <dbl>
## 1 0         50     0      1     1     1.3     2     1
## 2 0.1       50     0     82     86    86.5    91     2
## 3 0.3       50     0     82     86    86.4    92     3
## 4 0.6       50     0     82     86    86.2    93     4
## 5 1.2       50     0     82     86    86.4    91     3
## 6 2.5       50     0     83     86    86.3    92     3
## 7 5         50     0     48     78    77.7    86     9
```

Kruskal–Wallis test provides evidence of statistical difference for activation gene coverage in the population at the end of 50,000 generations.

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

##
##  Kruskal-Wallis rank sum test
##
## data: uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 205.39, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction for activation gene coverage in the population at the end of 50,000 generations.

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

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$uni_str_pos and end$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 1.8e-14 8.5e-14 9.2e-13 1.1e-13 2.2e-13
##
## P value adjustment method: bonferroni

```

## 11.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each nondominated sorting sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0.0 and 100.0. Note that when `sigma = 0.0`, no similarity penalty is used within nondominated front, and then stochastic remainder selection is used to identify parent solutions.

### 11.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

#### 11.4.1.1 Performance over time

Performance over time.

```

problem <- filter(nds_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

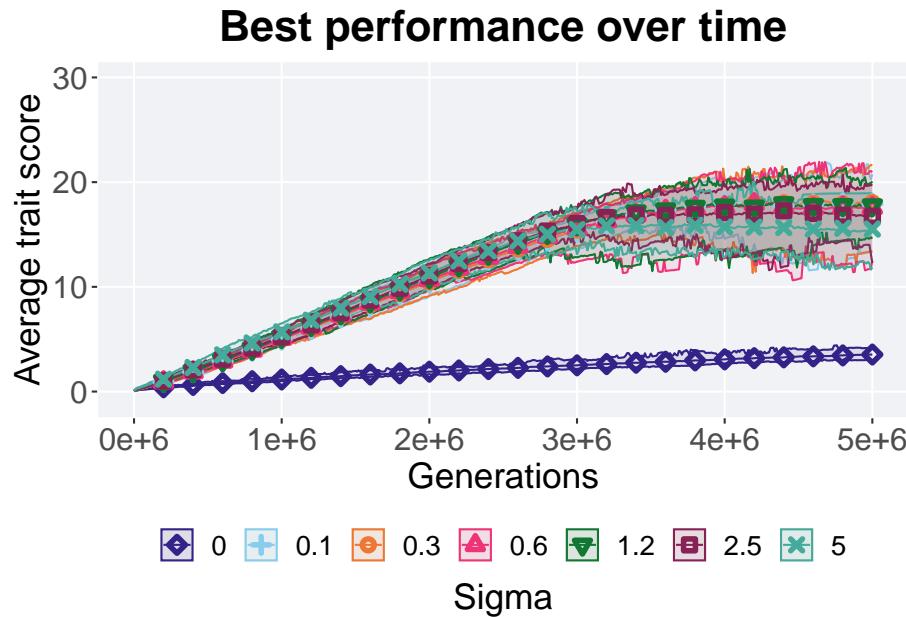
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma))
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    trans = log10_trans(),
    labels = scales::label_number_si()
  )

```

```
  name="Average trait score",
  limits=c(-1, 30)
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme +
guides(
  shape=guide_legend(nrow=1, title.position = "bottom"),
  color=guide_legend(nrow=1, title.position = "bottom"),
  fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)

ot
```



#### 11.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(nds_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

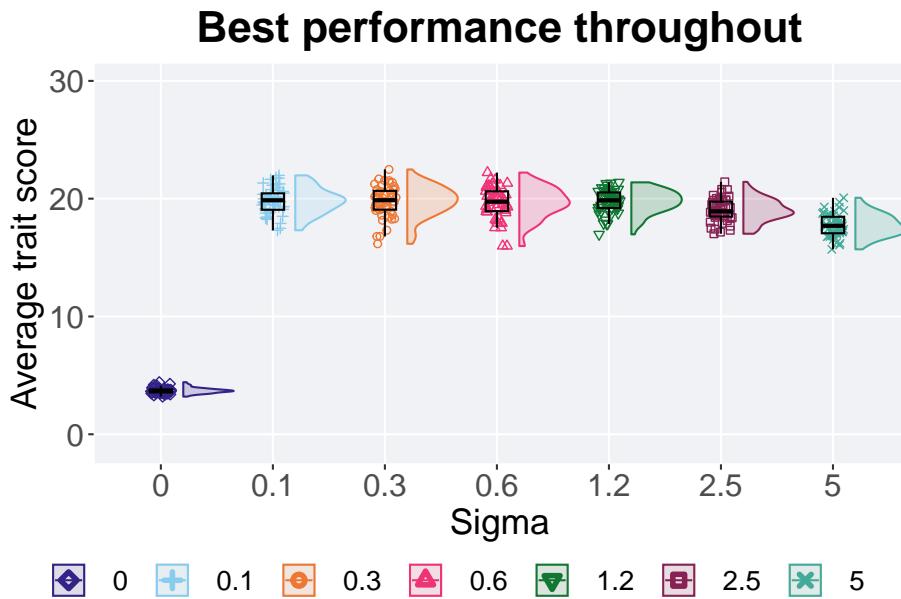
plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```

ggtile("Best performance throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 11.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```

group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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 3.20  3.68  3.71  4.42 0.248
## 2 0.1    50     0 17.3  19.9  19.8  22.0 1.39
## 3 0.3    50     0 16.2  19.9  19.8  22.5 1.59
## 4 0.6    50     0 16.0  19.7  19.6  22.2 1.69
## 5 1.2    50     0 17.0  19.9  19.8  21.4 1.33
## 6 2.5    50     0 17.0  18.9  19.0  21.4 1.27
## 7 5      50     0 15.7  17.7  17.7  20.1 1.40
```

Kruskal–Wallis test provides evidence of statistical differences among the best performing solution found throughout 50,000 generations.

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

```
##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 192.67, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performing solution found throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$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 1.0000 1.0000 1.0000 -     -
## 2.5 < 2e-16 0.0194 0.0199 0.1236 0.0058 -
## 5   < 2e-16 1.3e-11 7.8e-10 2.6e-09 8.8e-12 4.7e-07
##
## P value adjustment method: bonferroni
```

#### 11.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

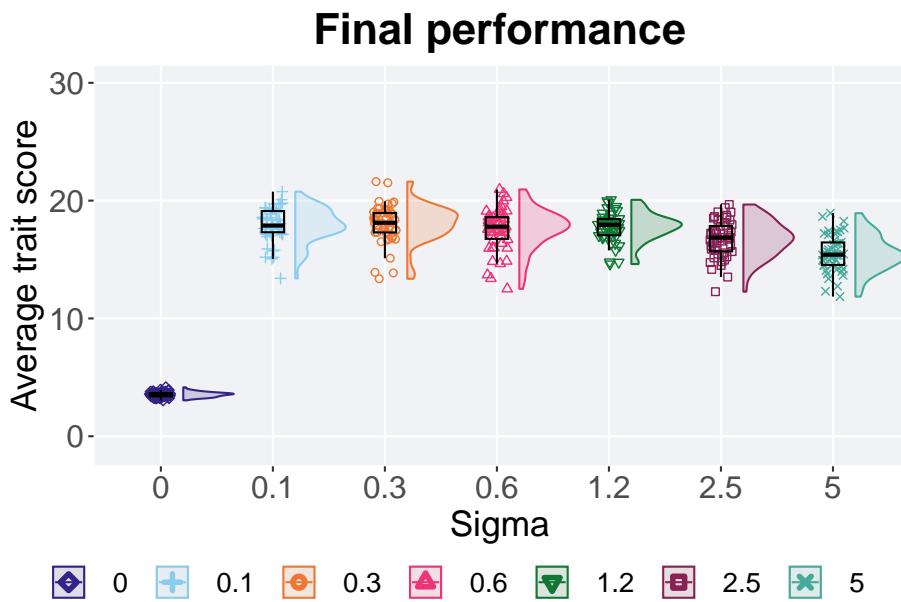
```
end = filter(nds_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, color = Sigma, fill = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = .7)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
```

```

geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
  ggtitle("Final performance") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 11.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, na.rm = TRUE)
  )

## # A tibble: 7 x 8
##   Sigma count na_cnt   min median   mean   max   IQR
##   <fct> <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 0         50      0  3.05  3.57  3.54  4.15 0.275
## 2 0.1       50      0 13.4   17.9   17.9  20.8  1.80
## 3 0.3       50      0 13.4   18.1   18.0  21.6  1.65
## 4 0.6       50      0 12.5   17.8   17.6  21.0  1.84
## 5 1.2       50      0 14.6   18.0   17.8  20.1  1.37
## 6 2.5       50      0 12.3   16.9   16.7  19.7  2.15
## 7 5         50      0 11.9   15.4   15.5  18.9  1.91
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ Sigma, data = end)

##
##  Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 182.39, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance in the population at the end of 50,000 generations.

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

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$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 0.00221 0.00091 0.12360 0.01237 -
## 5 < 2e-16 4.1e-09 4.2e-09 8.7e-07 6.9e-09 0.00443
##
## P value adjustment method: bonferroni

```

### 11.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

#### 11.4.2.1 Coverage over time

Activation gene coverage over time.

```

problem <- filter(nds_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

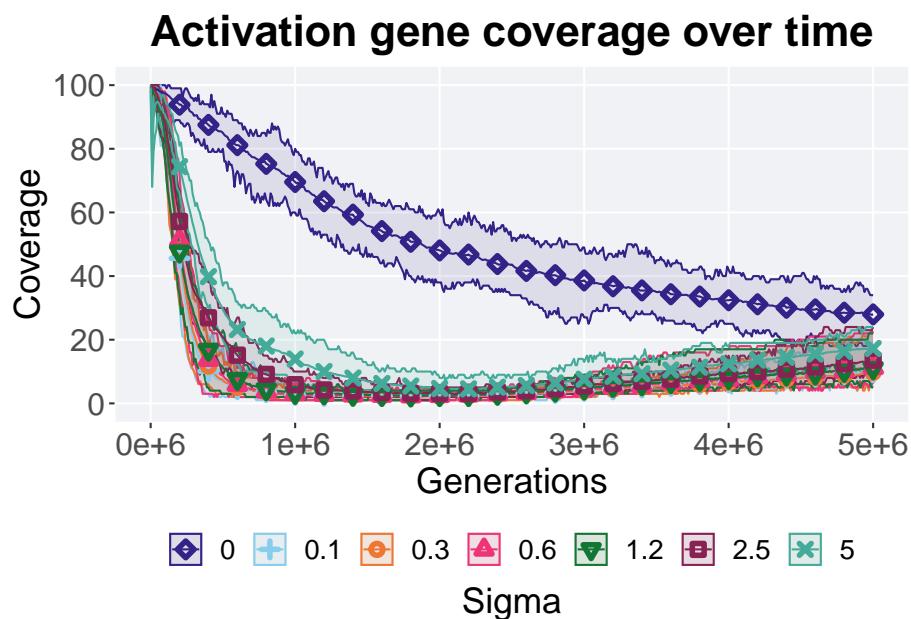
```

```

) +
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 +
guides(
shape=guide_legend(nrow=1, title.position = "bottom"),
color=guide_legend(nrow=1, title.position = "bottom"),
fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
legend.position = "bottom",
legend.box="vertical",
legend.justification="center",
legend.title.align=0.5
)
)

```

ot

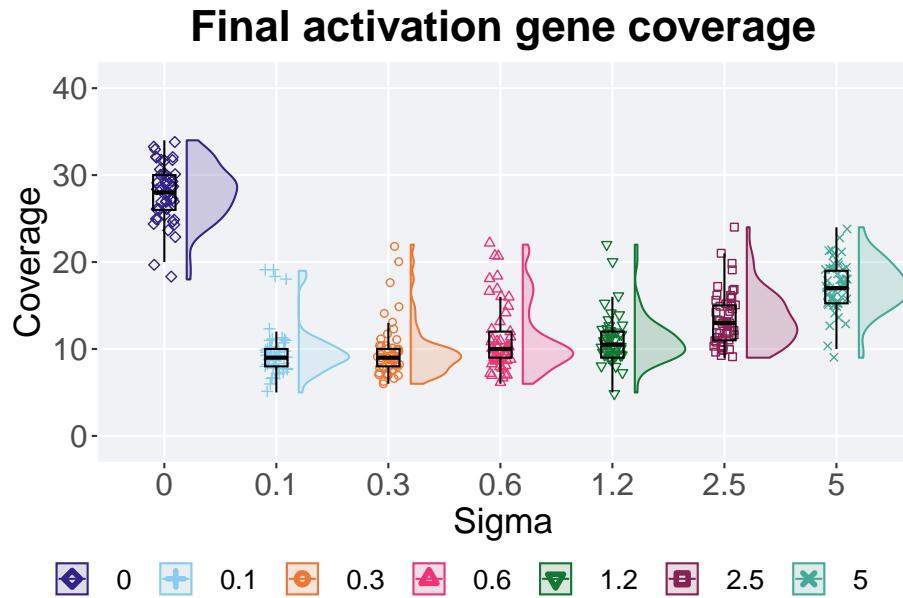


#### 11.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
end = filter(nds_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 41),
    breaks=seq(0,40, 10),
    labels=c("0", "10", "20", "30", "40")
  ) +
  scale_x_discrete(
    name="Sigma"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
  ggtitle("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```



#### 11.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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
##   <dbl> <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 0      50      0     18    28.0  28.0   34    4
## 2 0.1    50      0      5     9     9.76   19    2
## 3 0.3    50      0      6     9     9.76   22    2
## 4 0.6    50      0      6    10    11.2   22    3
## 5 1.2    50      0      5    10.5  11.0   22    3
## 6 2.5    50      0      9    13    13.6   24    4
## 7 5      50      0      9    17    17.2   24   3.75
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 218.65, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the activation gene coverage in the population at the end of 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$uni_str_pos and end$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 0.65720 -      -      -
## 1.2 < 2e-16 0.01207 0.01118 1.00000 -      -
## 2.5 2.3e-16 6.0e-09 1.3e-08 0.00084 0.00019 -
## 5   8.2e-16 1.7e-12 1.4e-12 4.4e-09 4.3e-12 8.0e-07
##
## P value adjustment method: bonferroni
```

## 11.5 Multi-valley crossing results

Here we present the results for the **best performances** and **best gene value** generated by each nondominated sorting replicate on the multi-valley crossing diagnostic. Best performance found refers to the largest average trait score found in a given population. Best gene value refers to maximum gene value found in the population; this gives us a range of values in [0.0, 100.0]. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, no similarity penalty is used within nondominated front, and then stochastic remainder selection is used to identify parent solutions.

### 11.5.1 Performance

Here we analyze the performances for each parameter replicate on the multi-valley crossing diagnostic.

### 11.5.1.1 Performance over time

Performance over time.

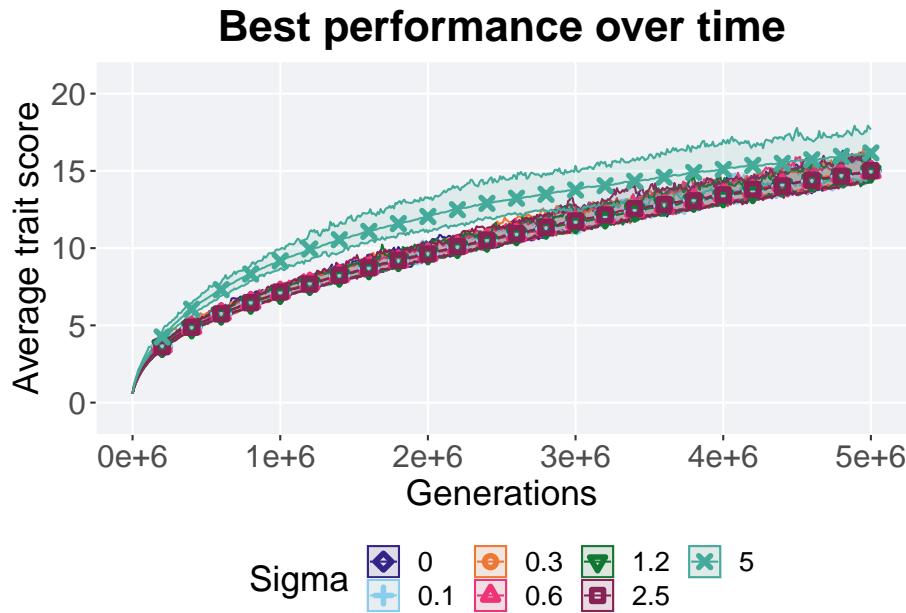
```
lines = filter(nds_ot, diagnostic == 'multivalley_crossing') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma,
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 21),
    breaks=seq(0, 20, 5),
    labels=c("0", "5", "10", "15", "20")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  ggtitle("Best performance over time") +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

ot
```



#### 11.5.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

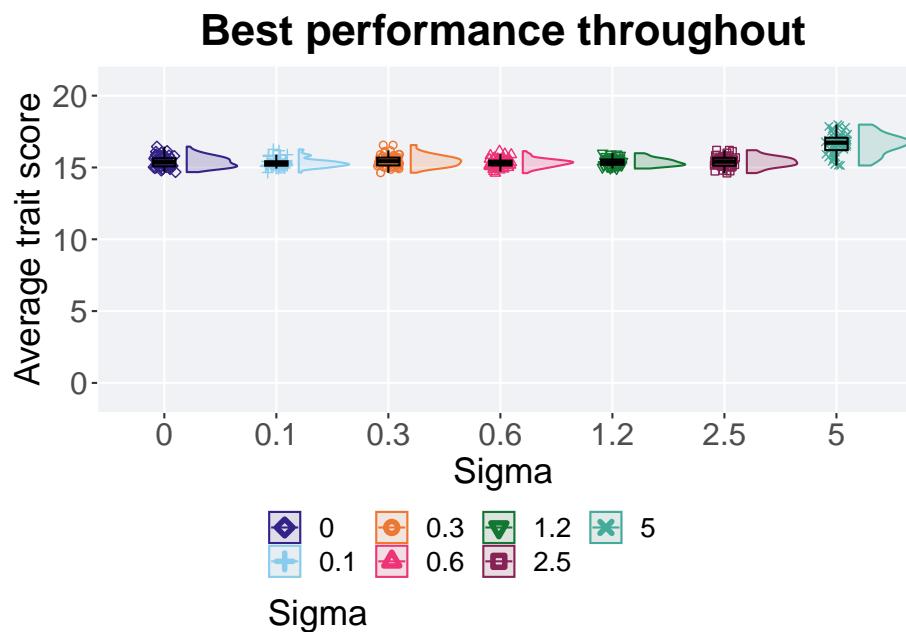
```
best = filter(nds_best, col == 'pop_fit_max' & diagnostic == 'multivalley_crossing')
plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  guides(fill = "none", color = 'none', shape = 'none') +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 21),
    breaks=seq(0, 20, 5),
    labels=c("0", "5", "10", "15", "20"))
  ) +
  scale_x_discrete(
    name="Sigma"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
```

```

plot +
  ggtitle("Best performance throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 11.5.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```

best$Sigma = factor(best$Sigma, levels = c(0.0,0.1,0.3,0.6,1.2,2.5,5.0))
group_by(best, 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 1468. 1537. 1539. 1645. 56.6
## 2 0.1    50     0 1465. 1525. 1531. 1626. 30.6
## 3 0.3    50     0 1463. 1544. 1547. 1656. 54.5
## 4 0.6    50     0 1460. 1532. 1531. 1614. 33.7
## 5 1.2    50     0 1493. 1536. 1538. 1599. 36.6
## 6 2.5    50     0 1461. 1541. 1540. 1620. 55.4
## 7 5      50     0 1514. 1673. 1662. 1798. 86.5
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

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

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 97.33, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$Sigma
##
##   0    0.1   0.3   0.6   1.2   2.5
## 0.1 1.0   -     -     -     -     -
## 0.3 1.0   0.4   -     -     -     -
## 0.6 1.0   1.0   1.0   -     -     -
## 1.2 1.0   1.0   1.0   1.0   -     -
## 2.5 1.0   1.0   1.0   1.0   1.0   -
## 5   5.1e-12 5.3e-13 4.4e-11 4.0e-13 1.6e-12 6.6e-12
##
## P value adjustment method: bonferroni
```

### 11.5.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

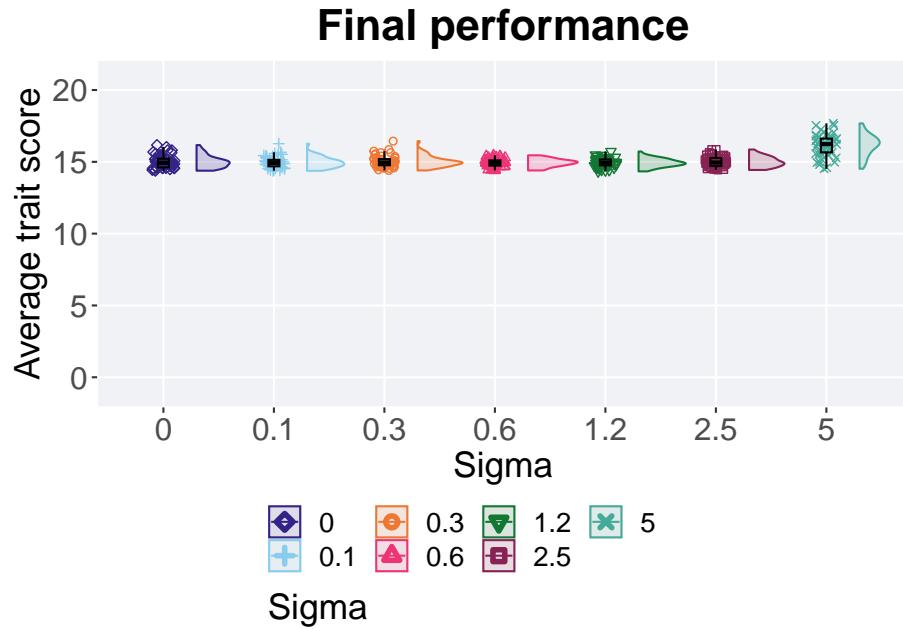
```
end = filter(nds_end, diagnostic == 'multivalley_crossing')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, group = Sigma, fill = Sigma, color = S
```

```

geom_flat_violin(position = position_nudge(x = .3), alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 10.8) +
  geom_boxplot(width = .1, outlier.shape = NA, alpha = 0.1, col = "black") +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 21),
    breaks = seq(0, 20, 5),
    labels = c("0", "5", "10", "15", "20")
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .55),
  label_size = TSIZE
)

```



### 11.5.1.3.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
end$Sigma = factor(end$Sigma, levels = c(0.0,0.1,0.3,0.6,1.2,2.5,5.0))
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max, na.rm = TRUE),
    median = median(pop_fit_max, na.rm = TRUE),
    mean = mean(pop_fit_max, na.rm = TRUE),
    max = max(pop_fit_max, na.rm = TRUE),
    IQR = IQR(pop_fit_max, na.rm = TRUE)
  )

## # A tibble: 7 x 8
##   Sigma count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl>
## 1 0         50     0 1438. 1493. 1500. 1617. 60.2
## 2 0.1       50     0 1437. 1491. 1496. 1626. 44.8
## 3 0.3       50     0 1440. 1496. 1502. 1644. 42.9
## 4 0.6       50     0 1440. 1494. 1493. 1545. 36.3
## 5 1.2       50     0 1434. 1493. 1494. 1572. 41.3
## 6 2.5       50     0 1444. 1495. 1501. 1586. 56.0
## 7 5         50     0 1452. 1623. 1616. 1768. 95.4
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

```
kruskal.test(pop_fit_max ~ Sigma, data = end)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 84.486, df = 6, p-value = 4.219e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

```
pairwise.wilcox.test(x = end$pop_fit_max, g = end$Sigma, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$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 1      1      1      1      1      -  
## 5  1.3e-10 2.7e-11 1.9e-10 8.1e-12 9.9e-12 6.4e-11  
##  
## P value adjustment method: bonferroni
```

### 11.5.2 Best gene value

Here we analyze the gene values for each parameter replicate on the multi-valley crossing diagnostic.

#### 11.5.2.1 Gene value over time

Best gene value over time.

```
lines = filter(nds_ot, diagnostic == 'multivalley_crossing') %>%  
  group_by(Sigma, gen) %>%  
  dplyr::summarise(  
    min = min(pop_max_gene),  
    mean = mean(pop_max_gene),  
    max = max(pop_max_gene)  
  )  
  
## `summarise()` has grouped output by 'Sigma'. You can override using the  
## ` `.groups` argument.  
points = filter(lines, gen %% 2000 == 0 & gen != 0)  
  
ot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape =  
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +  
  geom_line(size = 0.5) +  
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +  
  scale_y_continuous(  
    name="Max gene value",  
    limits=c(-1, 101),  
    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),
```

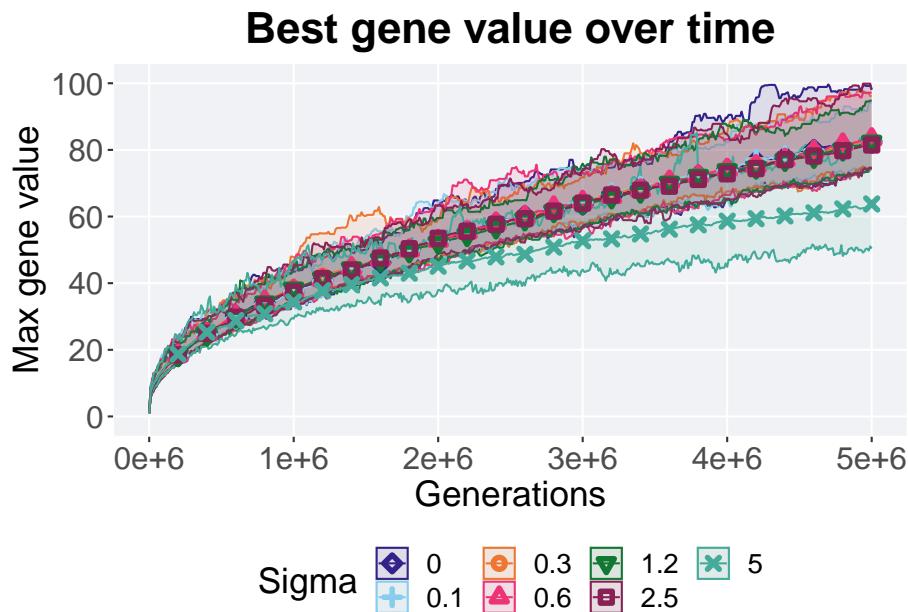
```

    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
ggtitle("Best gene value over time") +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

ot

```



### 11.5.2.2 Best gene value throughout

The best gene value found throughout 50,000 generations.

```

best = filter(nds_best, col == 'pop_max_gene' & diagnostic == 'multivalley_crossing')
plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  guides(fill = "none", color = 'none', shape = 'none') +
  scale_y_continuous(
    name="Max gene value",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),

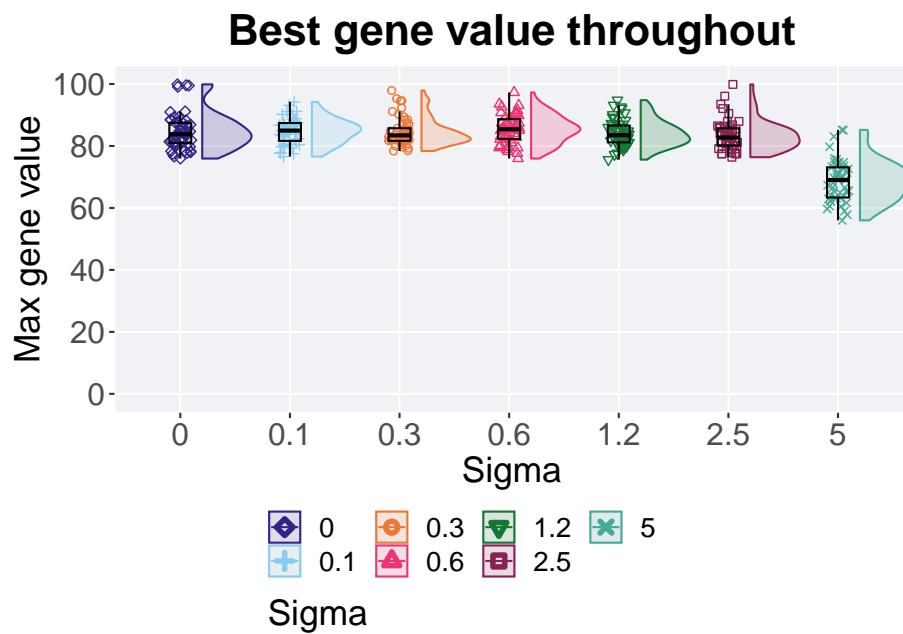
```

```

    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) +
  p_theme

plot_grid(
  plot +
  ggtitle("Best gene value throughout") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



#### 11.5.2.2.1 Stats

Summary statistics for the best gene value found throughout 50,000 generations.

```

best$Sigma = factor(best$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0))
group_by(best, 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  76.0  83.8  84.7  99.9  6.47
## 2 0.1       50      0  76.6  85.0  84.7  94.2  5.75
## 3 0.3       50      0  78.4  83.4  84.6  97.9  4.07
## 4 0.6       50      0  76.0  85.4  85.4  97.3  6.38
## 5 1.2       50      0  75.6  83.5  84.1  94.8  5.41
## 6 2.5       50      0  76.4  82.7  83.4  99.9  5.56
## 7 5         50      0  56.0  69.0  68.8  85.2  9.74

```

Kruskal-Wallis test provides evidence of significant differences among sigma values on the best gene value found throughout 50,000 generations

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

```

##
## Kruskal-Wallis rank sum test
##
## data: val by Sigma
## Kruskal-Wallis chi-squared = 116.45, df = 6, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best gene value found throughout 50,000 generations.

```

pairwise.wilcox.test(x = best$val, g = best$Sigma, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$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 1.00 1.00 - - -
## 1.2 1.00 1.00 1.00 - - -
## 2.5 1.00 0.36 1.00 0.16 1.00 - 
## 5 1.5e-14 6.6e-15 1.8e-14 3.8e-15 2.4e-14 4.0e-14
##
## P value adjustment method: bonferroni

```

### 11.5.2.3 End of 50,000 generations

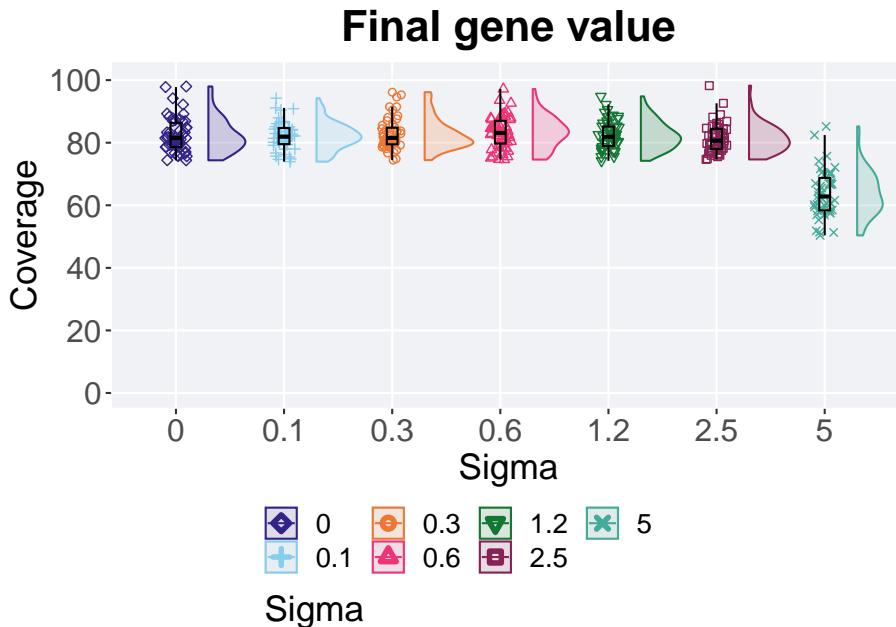
Best gene value in the population at the end of 50,000 generations.

```

end = filter(nds_end, diagnostic == 'multivalley_crossing')
plot = ggplot(end, aes(x = Sigma, y = pop_max_gene, group = Sigma, fill = Sigma, color =
  geom_flat_violin(position = position_nudge(x = .3), alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 10.8) +
  geom_boxplot(width = .1, outlier.shape = NA, alpha = 0.1, col = "black") +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
  ggtitle("Final gene value") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)

```



### 11.5.2.3.1 Stats

Summary statistics for the best gene value found at the end of 50,000 generations.

```
end$Sigma = factor(end$Sigma, levels = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0))
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_max_gene)),
    min = min(pop_max_gene, na.rm = TRUE),
    median = median(pop_max_gene, na.rm = TRUE),
    mean = mean(pop_max_gene, na.rm = TRUE),
    max = max(pop_max_gene, na.rm = TRUE),
    IQR = IQR(pop_max_gene, 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  74.4  81.5  82.5  98.0  7.67
## 2 0.1     50      0  73.9  81.9  82.2  94.2  5.14
## 3 0.3     50      0  74.4  81.5  82.8  96.2  5.30
## 4 0.6     50      0  74.6  83.1  83.3  97.2  7.26
## 5 1.2     50      0  74.2  81.9  82.4  94.8  6.21
## 6 2.5     50      0  74.7  80.6  81.5  98.2  6.45
## 7 5       50      0  50.4  62.8  63.8  85.2 10.4
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best gene value found at the end of 50,000 generations.

```
kruskal.test(pop_max_gene ~ Sigma, data = end)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: pop_max_gene by Sigma  
## Kruskal-Wallis chi-squared = 117.76, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best gene value found at the end of 50,000 generations.

```
pairwise.wilcox.test(x = end$pop_max_gene, g = end$Sigma, p.adjust.method = "bonferroni"  
                      paired = FALSE, conf.int = FALSE, alternative = '1')  
  
##  
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction  
##  
## data: end$pop_max_gene and end$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   1.00   1.00   -      -      -  
## 1.2 1.00   1.00   1.00   1.00   -      -  
## 2.5 1.00   1.00   1.00   0.52   1.00   -  
## 5    4.5e-15 5.3e-15 4.0e-15 2.7e-15 3.8e-15 6.3e-15  
##  
## P value adjustment method: bonferroni
```

# Chapter 12

## Novelty Search

We present the results from our parameter sweep on novelty search. 50 replicates are conducted for each K-nearest neighbors parameter value explored.

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

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         Patched
## major          4
## minor          2.2
## year          2022
## month         11
## day           10
## svn rev       83330
## language      R
## version.string R version 4.2.2 Patched (2022-11-10 r83330)
## nickname      Innocent and Trusting
```

## 12.1 Exploitation rate results

Here we present the results for **best performances** found by each novelty search K value replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

### 12.1.1 Performance over time

Performance over time.

```
problem <- filter(nov_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

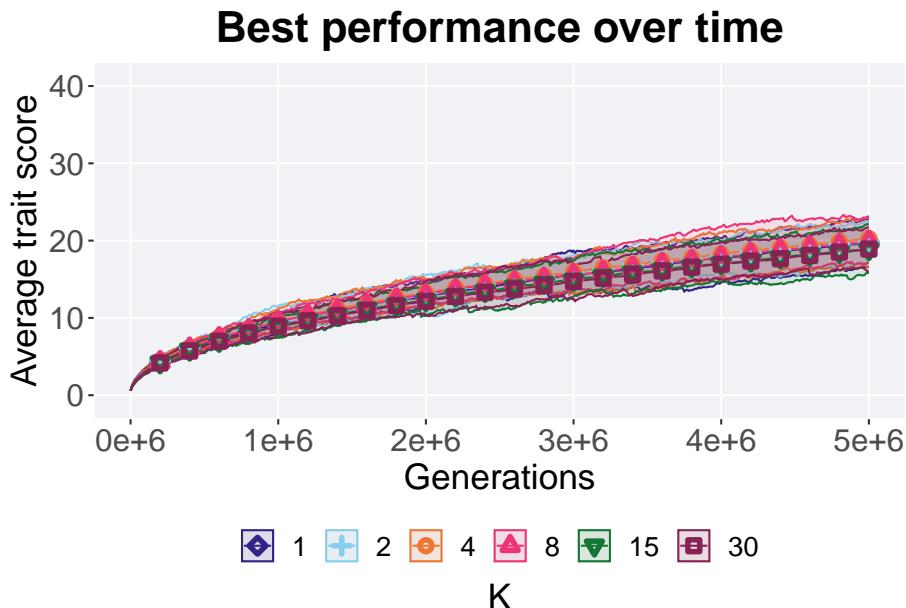
ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = K, fill = K, color = K, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 41),
    breaks=seq(0, 40, 10),
    labels=c("0", "10", "20", "30", "40")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
```

```

guides(
  shape=guide_legend(nrow=1, title.position = "bottom"),
  color=guide_legend(nrow=1, title.position = "bottom"),
  fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)

```

ot



#### 12.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```

best = filter(nov_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')

plot = ggplot(best, aes(x = K, y = val / TRAITS, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous()

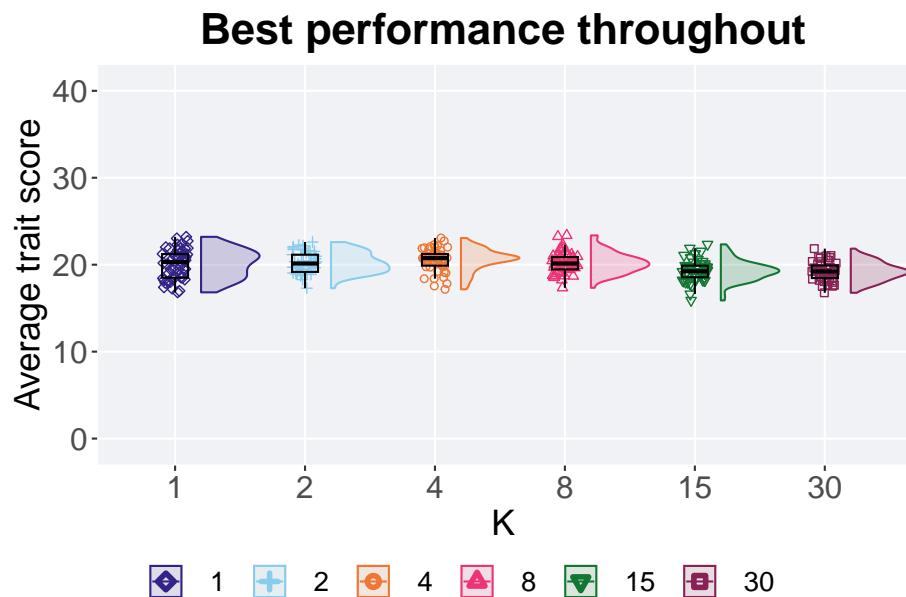
```

```

name = "Average trait score",
limits = c(-1, 41),
breaks = seq(0, 40, 10),
labels = c("0", "10", "20", "30", "40")
) +
scale_x_discrete(
  name = "K"
) +
scale_shape_manual(values = SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Best performance throughout") +
  theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .3),
  label_size = TSIZE
)

```



### 12.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
group_by(best, K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
  )

## # A tibble: 6 x 8
##   K     count na_cnt   min median   mean   max   IQR
##   <fct> <int> <int> <dbl> <dbl> <dbl> <dbl>
## 1 1      50     0  16.8  20.3  20.1  23.2  2.72
## 2 2      50     0  17.3  20.1  20.2  22.6  1.98
## 3 4      50     0  17.2  20.8  20.6  23.1  1.34
## 4 8      50     0  17.3  20.1  20.2  23.4  1.42
## 5 15     50     0  15.9  19.2  19.3  22.3  1.34
## 6 30     50     0  16.8  19.2  19.2  21.8  1.44
```

Kruskal-Wallis test provides evidence of significant differences among K values on the best performance found throughout 50,000 generations

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

##
## Kruskal-Wallis rank sum test
##
## data: val by K
## Kruskal-Wallis chi-squared = 41.239, df = 5, p-value = 8.394e-08
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$K
##
##   1     2     4     8     15
```

```

## 2 1.0000 - - - -
## 4 1.0000 1.0000 - - - -
## 8 1.0000 1.0000 0.5941 - - - -
## 15 0.1905 0.0068 3.9e-05 0.0072 - -
## 30 0.0777 0.0023 6.2e-06 0.0025 1.0000
##
## P value adjustment method: bonferroni

```

## 12.2 Ordered exploitation results

Here we present the results for **best performances** found by each novelty search K value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

### 12.2.1 Performance over time

Performance over time.

```

problem <- filter(nov_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = K, fill = K, color = K, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),

```

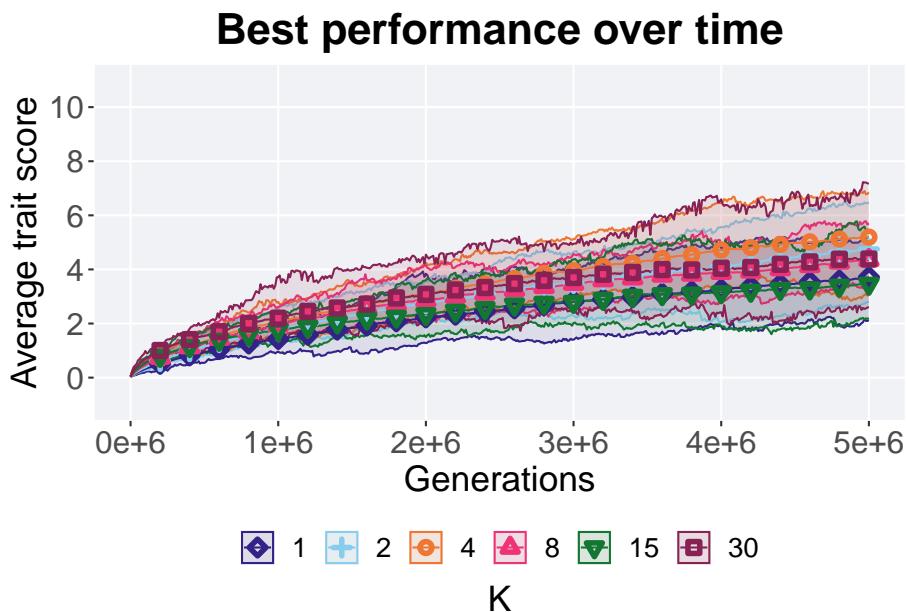
```

breaks=c(0, 10000, 20000, 30000, 40000, 50000),
labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme+
guides(
shape=guide_legend(nrow=1, title.position = "bottom"),
color=guide_legend(nrow=1, title.position = "bottom"),
fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
legend.position = "bottom",
legend.box="vertical",
legend.justification="center",
legend.title.align=0.5
)
)

ot

```



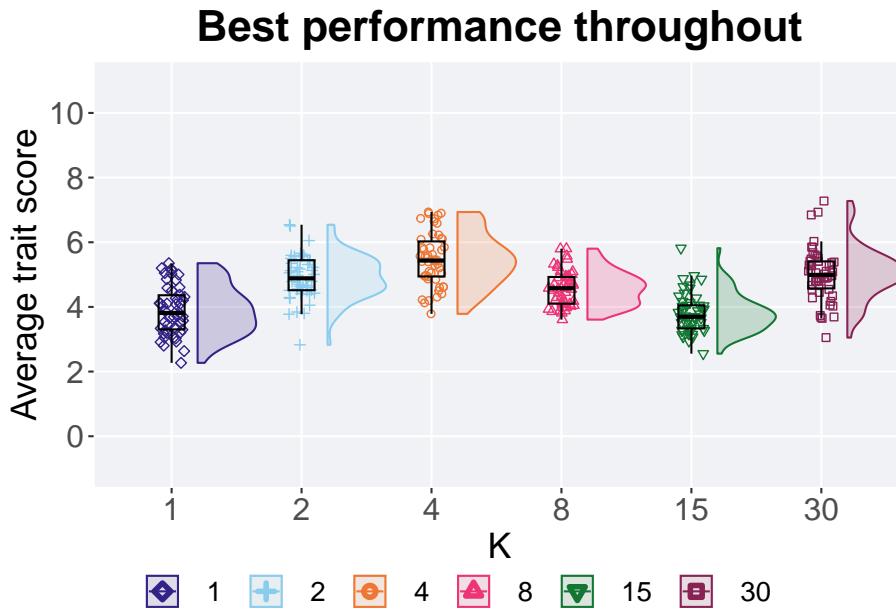
### 12.2.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(nov_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')

plot = ggplot(best, aes(x = K, y = val / TRAITS, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score",
    limits = c(-1, 11),
    breaks = seq(0, 10, 2),
    labels = c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_discrete(
    name = "K"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .2),
  label_size = TSIZE
)
```



#### 12.2.2.1 Stats

Summary statistics about the best performance found.

```
group_by(best, K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  2.27   3.81   3.89   5.35  1.06
## 2 2       50     0  2.83   4.88   4.92   6.54  0.928
## 3 4       50     0  3.79   5.43   5.46   6.94  1.09
## 4 8       50     0  3.61   4.58   4.58   5.80  0.826
## 5 15      50     0  2.55   3.70   3.80   5.82  0.718
## 6 30      50     0  3.05   4.99   5.00   7.28  0.835
```

Kruskal–Wallis test provides evidence of statistical differences for the best

performance found in the population throughout 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: val by K
## Kruskal-Wallis chi-squared = 127.68, df = 5, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
the best performance found in the population throughout 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$K
##
##    1      2      4      8     15
## 2 5.1e-08 -      -      -      -
## 4 6.3e-12 0.02655 -      -      -
## 8 0.00017 0.11567 1.1e-06 -      -
## 15 1.00000 2.9e-10 1.5e-13 5.5e-08 -
## 30 8.4e-08 1.00000 0.10010 0.03273 4.9e-10
##
## P value adjustment method: bonferroni
```

## 12.3 Contradictory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each novelty search K value replicate on the contradictory objectives diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

### 12.3.1 Satisfactory trait coverage

Here we analyze the satisfactory trait coverage for each parameter replicate on the contradictory objectives diagnostic.

#### 12.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```

problem <- filter(nov_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )

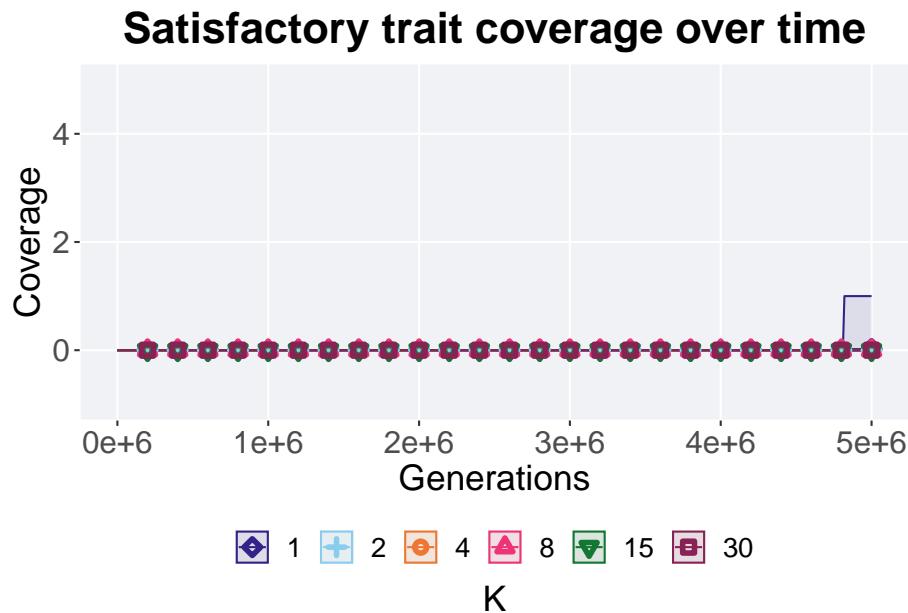
## `summarise()` has grouped output by 'K'. You can override using the `groups` argument.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 5)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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 +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.justification="center",
    legend.title.align=0.5
  )

```

```
ot
```



#### 12.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(nov_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives')

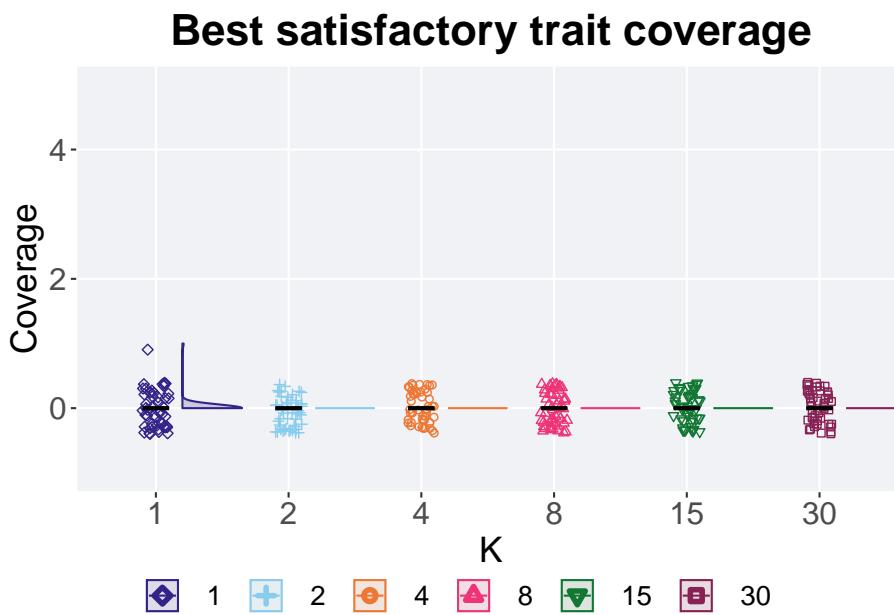
plot = ggplot(best, aes(x = K, y = val, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Coverage",
    limits = c(-1, 5)
  ) +
  scale_x_discrete(
    name = "K"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
```

```

plot +
  ggtitle("Best satisfactory trait coverage") +
  theme(legend.position="none"),
legend,
nrow=2,
rel_heights = c(2,.2),
label_size = TSIZE
)

```



#### 12.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```

group_by(best, 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     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

```

### 12.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

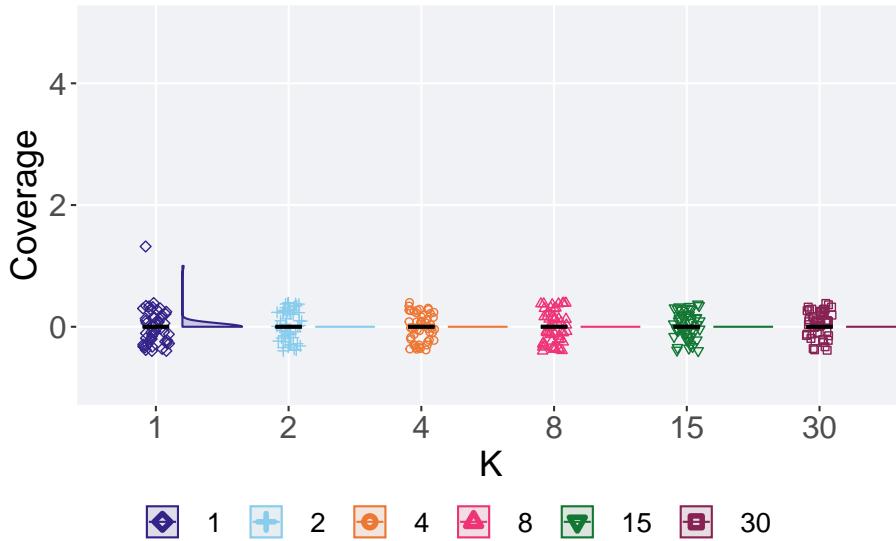
```

end = filter(nov_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = K, y = pop_uni_obj, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Coverage",
    limits = c(-1, 5)
  ) +
  scale_x_discrete(
    name = "K"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position = "none"),
  legend,
  nrow = 2,
  rel_heights = c(2, .3),
  label_size = TSIZE
)

```

## Final satisfactory trait coverage



### 12.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
group_by(end, 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> <dbl> <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
```

### 12.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

#### 12.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(nov_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
  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.

points = filter(lines, gen %% 2000 == 0 & gen != 0)

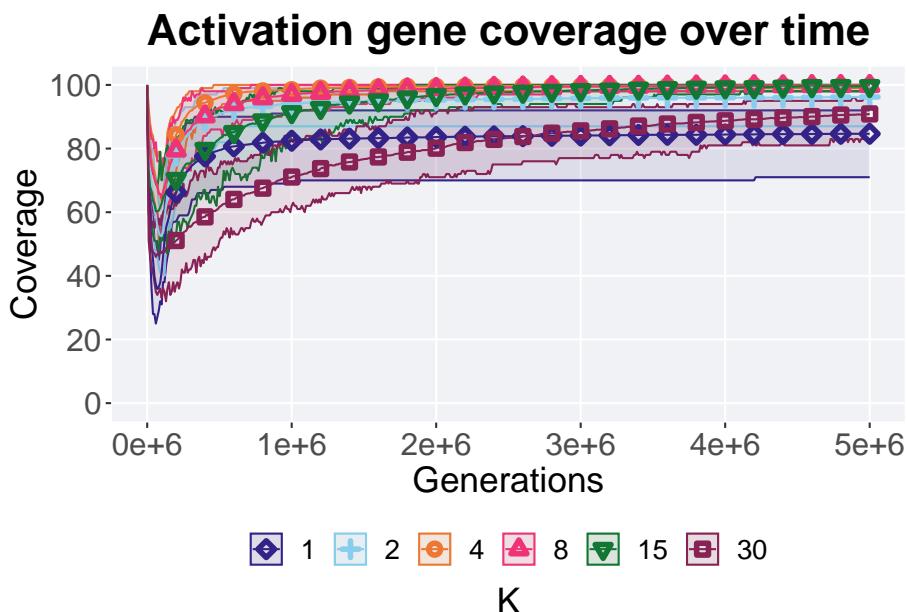
ot = ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  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 +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )
```

```

    fill=guide_legend(nrow=1, title.position = "bottom")
) +
theme(
  legend.position = "bottom",
  legend.box="vertical",
  legend.justification="center",
  legend.title.align=0.5
)
)

ot

```



### 12.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```

end = filter(nov_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = K, y = uni_str_pos, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 41),
    breaks=seq(0, 40, 10),
    labels=c("0", "10", "20", "30", "40")

```

```

) +
scale_x_discrete(
  name="K"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
  ggtitle("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)

## Warning: Removed 300 rows containing non-finite values (`stat_ydensity()`).

## Warning: Computation failed in `stat_ydensity()`
## Caused by error in `$<- .data.frame`:
## ! replacement has 1 row, data has 0

## Warning: Removed 300 rows containing non-finite values (`stat_boxplot()`).

## Warning in min(x): no non-missing arguments to min; returning Inf

## Warning in max(x): no non-missing arguments to max; returning -Inf

## Warning in min(diff(sort(x))): no non-missing arguments to min; returning Inf

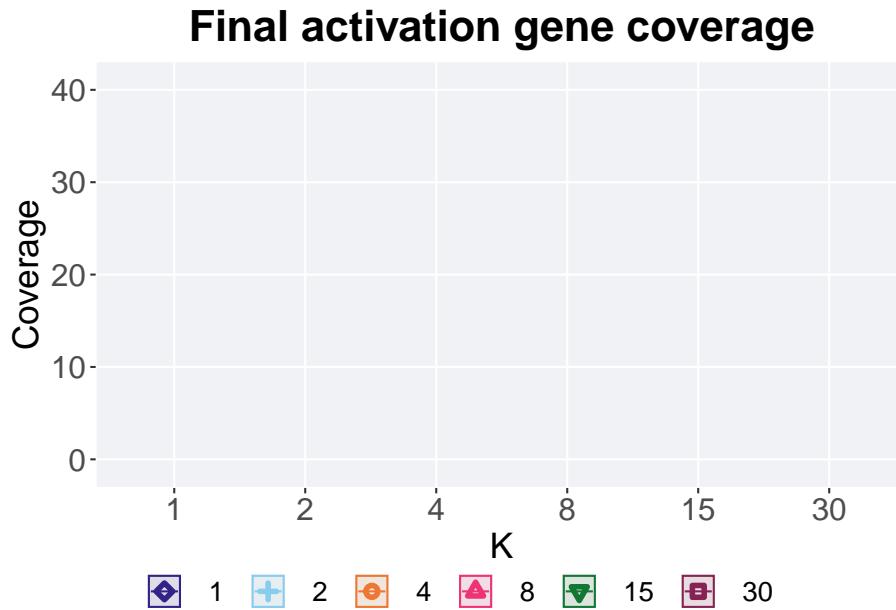
## Warning in min(x): no non-missing arguments to min; returning Inf

## Warning in max(x): no non-missing arguments to max; returning -Inf

## Warning in stats::runif(length(x), -amount, amount): NAs produced

## Warning: Removed 300 rows containing missing values (`geom_point()`).

```



#### 12.3.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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    71    85  84.6    92     5
## 2 2       50     0    87    97  96.2   100     3
## 3 4       50     0    99   100  99.8   100     0
## 4 8       50     0    98   100  99.7   100  0.75
## 5 15      50     0    98   100  99.6   100     1
## 6 30      50     0    83    91  90.9    96     3
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
## 
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by K
## Kruskal-Wallis chi-squared = 260.74, df = 5, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the activation gene coverage in the population at the end of 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$uni_str_pos and end$K
##
##    1      2      4      8      15
## 2 4.4e-16 -      -      -      -
## 4 < 2e-16 < 2e-16 -      -      -
## 8 < 2e-16 4.5e-16 1.00 -      -
## 15 < 2e-16 6.4e-15 0.38 1.00 - 
## 30 7.2e-10 1.0e-12 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

## 12.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each novelty search K value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

### 12.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

#### 12.4.1.1 Performance over time

Performance over time.

```

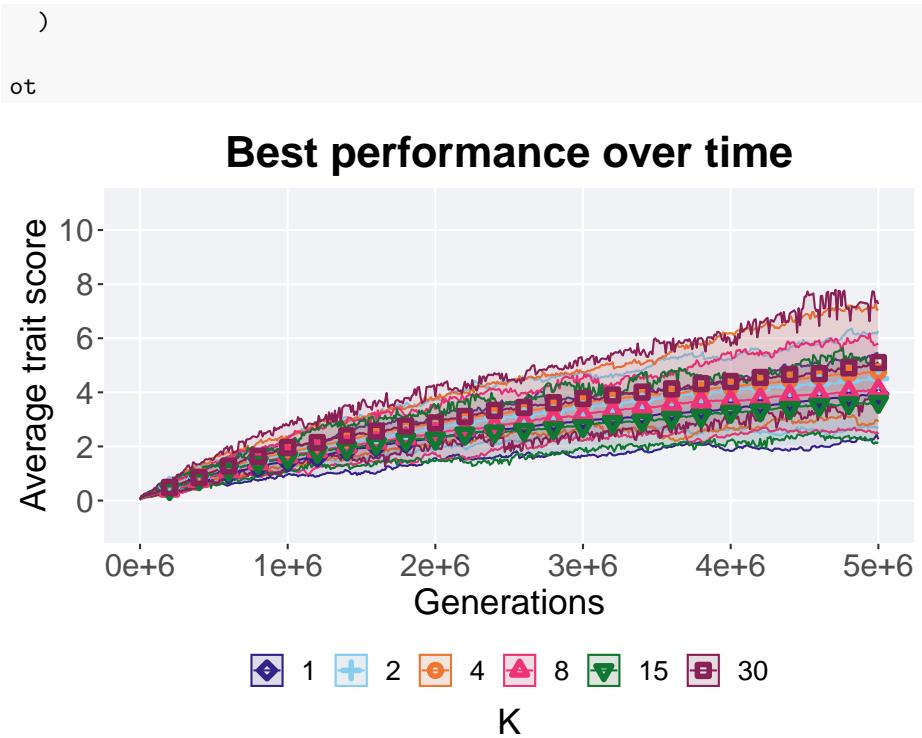
problem <- filter(nov_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max),
    mean = mean(pop_fit_max),
    max = max(pop_fit_max)
  )

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

points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0, 10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="vertical",
    legend.justification="center",
    legend.title.align=0.5
  )

```



#### 12.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(nov_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

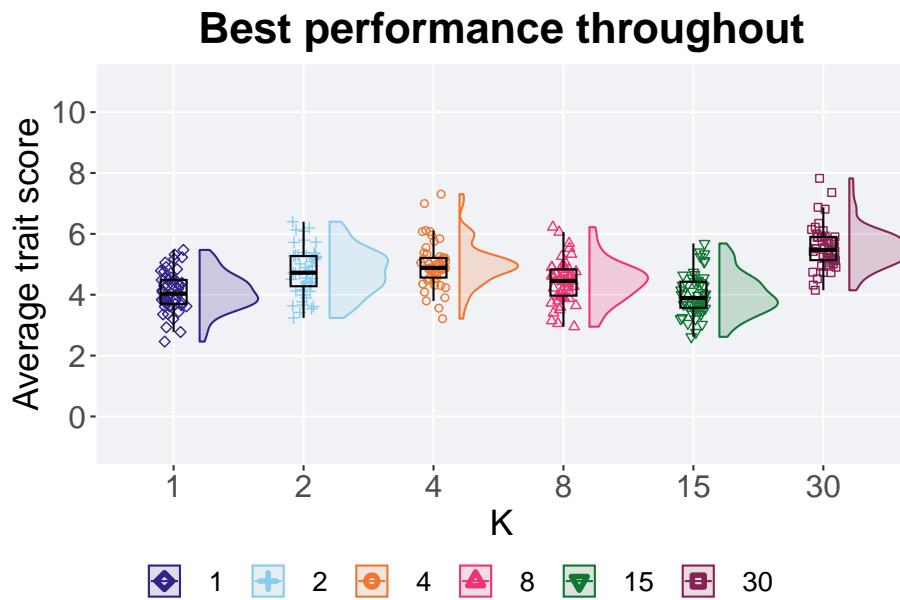
plot = ggplot(best, aes(x = K, y = val / TRAITS, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0, 10, 2),
    labels=c("0", "2", "4", "6", "8", "10"))
  ) +
  scale_x_discrete(
    name="K")
  )+
  scale_shape_manual(values=SHAPE)+
```

```

scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

```



#### 12.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```

group_by(best, K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    sd = sd(val / TRAITS, na.rm = TRUE)
  )

```

```

mean = mean(val / TRAITS, na.rm = TRUE),
max = max(val / TRAITS, na.rm = TRUE),
IQR = IQR(val / TRAITS, 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  2.46   4.03   4.10   5.47  0.786
## 2 2       50     0  3.24   4.72   4.76   6.40  0.992
## 3 4       50     0  3.21   4.88   4.99   7.30  0.645
## 4 8       50     0  2.95   4.45   4.43   6.22  0.864
## 5 15      50     0  2.62   3.89   4.01   5.68  0.860
## 6 30      50     0  4.15   5.47   5.55   7.82  0.765

```

Kruskal–Wallis test provides evidence of statistical differences among the best performing solution found throughout 50,000 generations.

```

kruskal.test(val ~ K, data = best)

##
## Kruskal-Wallis rank sum test
##
## data: val by K
## Kruskal-Wallis chi-squared = 112.24, df = 5, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performing solution found throughout 50,000 generations.

```

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$val and best$K
##
##   1     2     4     8     15
## 2 0.00070 -    -    -
## 4 3.9e-07 1.00000 -    -    -
## 8 0.21797 0.45226 0.00187 -    -
## 15 1.00000 0.00013 1.3e-07 0.03587 -
## 30 2.5e-13 2.8e-05 0.00061 4.2e-10 3.4e-13
##
## P value adjustment method: bonferroni

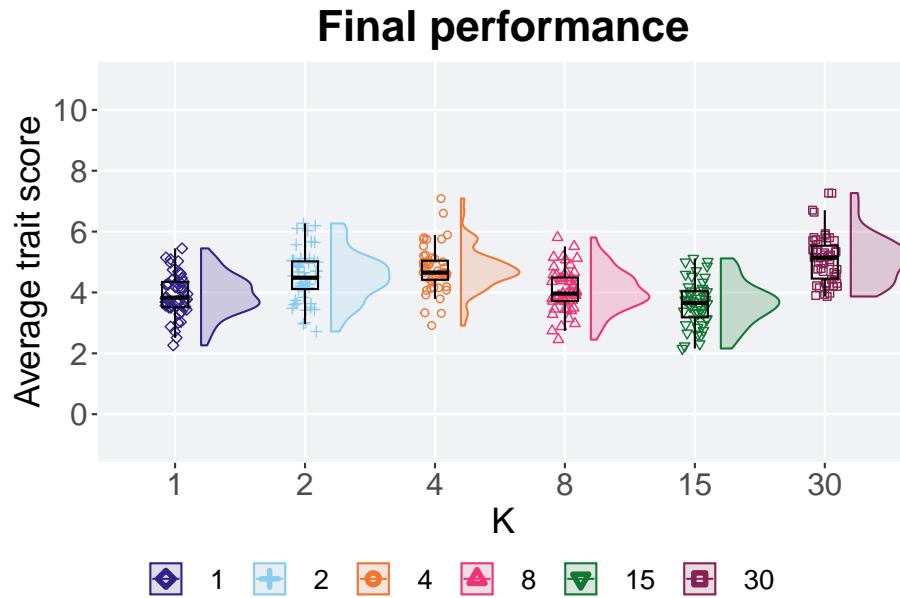
```

### 12.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```
end = filter(nov_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = K, y = pop_fit_max / TRAITS, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0, 10, 2),
    labels=c("0", "2", "4", "6", "8", "10"))
  ) +
  scale_x_discrete(
    name="K")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```



#### 12.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, 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  2.26  3.83  3.93  5.45  0.832
## 2 2      50       0  2.72  4.48  4.51  6.27  0.910
## 3 4      50       0  2.91  4.65  4.76  7.09  0.627
## 4 8      50       0  2.45  3.96  4.08  5.81  0.777
## 5 15     50       0  2.16  3.66  3.64  5.12  0.859
## 6 30     50       0  3.87  5.14  5.10  7.27  1.08
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ K, data = end)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by K
## Kruskal-Wallis chi-squared = 95.737, df = 5, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance in the population at the end of 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$pop_fit_max and end$K
##
##   1     2     4     8    15
## 2 0.01055 -     -     -
## 4 4.1e-06 1.00000 -     -     -
## 8 1.00000 0.16610 0.00013 -     -
## 15 0.63500 1.5e-05 5.2e-09 0.05030 -
## 30 2.3e-09 0.01195 0.38606 1.5e-07 4.0e-12
##
## P value adjustment method: bonferroni
```

## 12.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

### 12.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(nov_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
  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.  

points = filter(lines, gen %% 2000 == 0 & gen != 0)  

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +  

  scale_y_continuous(  

    name="Coverage",  

    limits=c(-1, 101),  

    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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")  

  ) +  

  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+  

  guides(  

    shape=guide_legend(nrow=1, title.position = "bottom"),  

    color=guide_legend(nrow=1, title.position = "bottom"),  

    fill=guide_legend(nrow=1, title.position = "bottom")  

  ) +  

  theme(  

    legend.position = "bottom",  

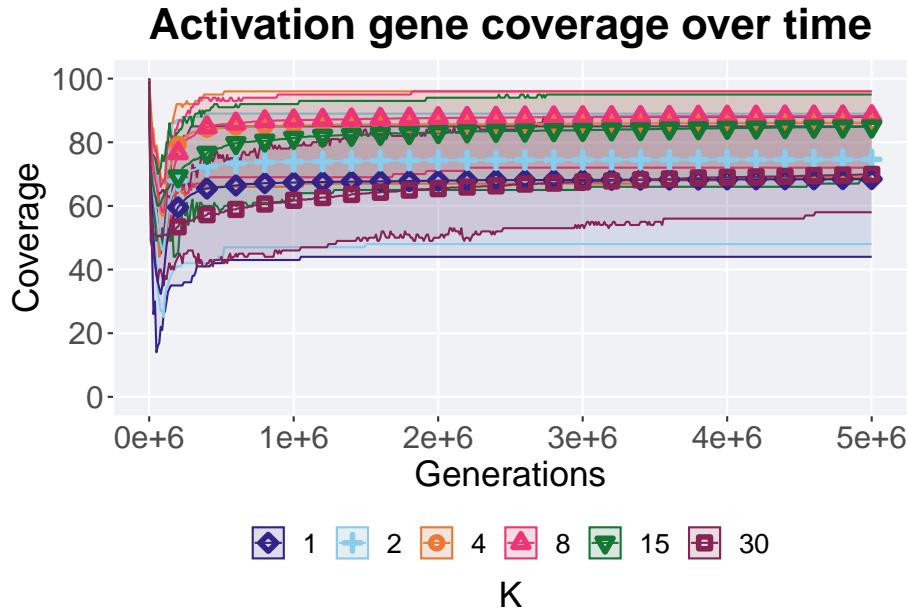
    legend.box="vertical",  

    legend.justification="center",  

    legend.title.align=0.5
  )

```

ot



#### 12.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
end = filter(nov_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = K, y = uni_str_pos, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 21),
    breaks=seq(0,20, 5),
    labels=c("0", "5", "10", "15", "20"))
  ) +
  scale_x_discrete(
    name="K"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```

ggtile("Final activation gene coverage") +
  theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)

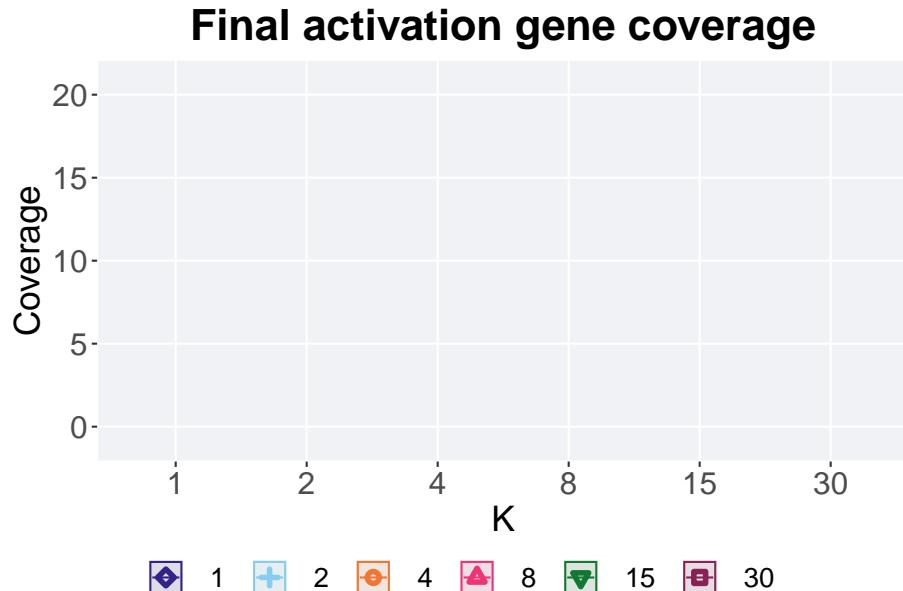
## Warning: Removed 300 rows containing non-finite values (`stat_ydensity()`).

## Warning: Computation failed in `stat_ydensity()`
## Caused by error in `$<-.data.frame`:
## ! replacement has 1 row, data has 0

## Warning: Removed 300 rows containing non-finite values (`stat_boxplot()`).

## Warning in min(x): no non-missing arguments to min; returning Inf
## Warning in max(x): no non-missing arguments to max; returning -Inf
## Warning in min(diff(sort(x))): no non-missing arguments to min; returning Inf
## Warning in min(x): no non-missing arguments to min; returning Inf
## Warning in max(x): no non-missing arguments to max; returning -Inf
## Warning in stats::runif(length(x), -amount, amount): NAs produced
## Warning: Removed 300 rows containing missing values (`geom_point()`).

```



### 12.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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> <dbl> <dbl> <int> <dbl>
## 1 1      50     0     44   69.5  68.4   85  11.8
## 2 2      50     0     48   74.5  74.6   89  13.8
## 3 4      50     0     69   87    86.1   96  7.75
## 4 8      50     0     72   89    88.2   96   5
## 5 15     50     0     68   85.5  84.9   95   6
## 6 30     50     0     58   71    69.9   89   8
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

##
## Kruskal-Wallis rank sum test
##
## data: uni_str_pos by K
## Kruskal-Wallis chi-squared = 176.13, df = 5, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the activation gene coverage in the population at the end of 50,000 generations.

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

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: end$uni_str_pos and end$K
##
##   1     2     4     8     15
```

```
## 2 0.025 - - - -  
## 4 9.0e-14 5.3e-08 - - -  
## 8 1.3e-15 2.4e-11 0.880 - -  
## 15 2.7e-13 8.4e-07 1.000 0.017 -  
## 30 1.000 0.035 2.5e-14 1.5e-15 1.3e-13  
##  
## P value adjustment method: bonferroni
```