Supplemental Material for 'Lexidate: Model Evaluation and Selection with Lexicase Selection'

Anonymous GECCO

2024-01-31

Contents


```r
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
library(scales) # to access break formatting functions


NAMES <- c("10-fold cv","90/10","70/30","50/50")
SHAPE <- c(21,24,22,25)
cb_palette <- c('#D81B60','#1E88E5','#FFC107','#004D40')
TSIZE <- 22
task_id_lists <- c(167104, 167184, 167168, 167161, 167185, 189905)

p_theme <- theme(
  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=20),
  legend.text=element_text(size=20),
  axis.title = element_text(size=20),
  axis.text = element_text(size=19),
  legend.position="bottom",
  panel.background = element_rect(fill = "#f1f2f5",
                                  colour = "white",
                                  size = 0.5, linetype = "solid")
)
```

Chapter 1

Introduction

This is not intended as a stand-alone document, but as a companion to our manuscript.

1.1 About our supplemental material

As you may have noticed (unless you're reading a pdf version of this), our supplemental material is hosted using GitHub pages. We compiled our data analyses and supplemental documentation into this nifty web-accessible book using bookdown.

The code used for this supplemental material can be found in this GitHub repository.

Our supplemental material includes the following:

- Accuracy results (Section 2)
- Complexity results (Section 3)

1.2 Contributing authors

- Jose Guadalupe Hernandez
- Anil Kumar Saini
- Jason H. Moore

Chapter 2

Accuracy results

Here we report the accuracy for **final pipeline returned from a TPOT2 run** on each OpenML task. 40 replicates were conducted for each evaluation strategy explored.

2.1 Analysis setup

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

scores <- read.csv("./scores.csv", header = TRUE, stringsAsFactors = FALSE)
scores$acro <- factor(scores$acro, levels = NAMES)
```

2.2 Accuracy per OpenML task

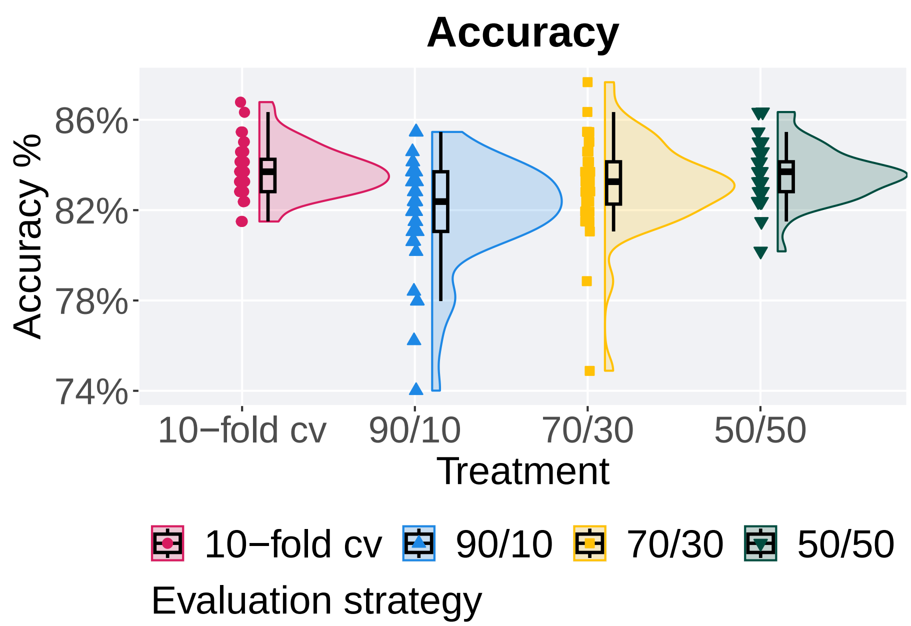
Accuracy of the returned pipeline from an evolutionary run.

2.2.1 Task 167104

```
filter(scores, taskid == task_id_lists[1]) %>%
  ggplot(., aes(x = acro, y = accuracy, color = acro,
                fill = acro, shape = acro)) +
```

```
geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                 scale = "width", alpha = 0.2, width = 1.5) +
geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
             size = 0.8, position = position_nudge(x = .15, y = 0)) +
geom_point(position = position_jitter(width = .015, height = .0001),
           size = 2.0, alpha = 1.0) +
scale_y_continuous(
  name = "Accuracy %",
  breaks = c(.74, .78, .82, .86),
  labels = scales::percent

) +
scale_x_discrete(
  name = "Treatment"
) +
scale_shape_manual(values = SHAPE,) +
scale_colour_manual(values = cb_palette,) +
scale_fill_manual(values = cb_palette,) +
ggtitle("Accuracy") +
p_theme +
guides(
  shape=guide_legend(nrow = 1, title.position = "bottom",
                     title = "Evaluation strategy"),
  color=guide_legend(nrow = 1, title.position = "bottom",
                     title = "Evaluation strategy"),
  fill=guide_legend(nrow = 1, title.position = "bottom",
                    title = "Evaluation strategy")
)
```



Summary statistics for the generation a satisfactory solution is found.

```
performance <- filter(scores, taskid == task_id_lists[1])
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(accuracy)),
    min = min(accuracy, na.rm = TRUE),
    median = median(accuracy, na.rm = TRUE),
    mean = mean(accuracy, na.rm = TRUE),
    max = max(accuracy, na.rm = TRUE),
    IQR = IQR(accuracy, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acro       count na_cnt   min median  mean   max    IQR
##   <fct>      <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 10-fold cv    40      0 0.815  0.837 0.838 0.868 0.0143
## 2 90/10         40      0 0.740  0.824 0.819 0.855 0.0264
## 3 70/30         40      0 0.749  0.833 0.831 0.877 0.0187
## 4 50/50         40      0 0.802  0.837 0.836 0.863 0.0132
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  accuracy by acro
## Kruskal-Wallis chi-squared = 21.695, df = 3, p-value = 7.55e-05
```

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

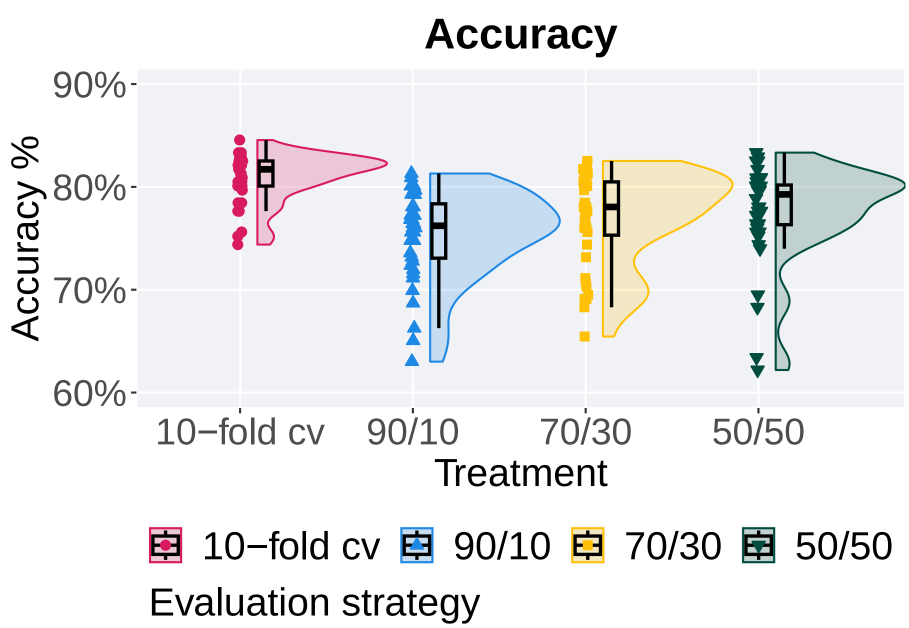
```
pairwise.wilcox.test(x = performance$accuracy, g = performance$acro,
                     p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = "t")
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$accuracy and performance$acro
##
##       10-fold cv 90/10   70/30
## 90/10 0.00016    -       -
## 70/30 0.41651    0.08840 -
## 50/50 1.00000    0.00111 1.00000
##
## P value adjustment method: bonferroni
```

2.2.2 Task 167184

```
filter(scores, taskid == task_id_lists[2]) %>%
  ggplot(., aes(x = acro, y = accuracy, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
               size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
             size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name = "Accuracy %",
    limits=c(.6, .9),
    labels = scales::percent
```

```
  ) +
  scale_x_discrete(
    name = "Treatment"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Accuracy") +
  p_theme +
  guides(
    shape=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    color=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    fill=guide_legend(nrow = 1, title.position = "bottom",
                      title = "Evaluation strategy")
  )
```



Summary statistics for the generation a satisfactory solution is found.

```
performance <- filter(scores, taskid == task_id_lists[2])
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
```

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

```
## # A tibble: 4 x 8
##   acro        count na_cnt   min median  mean   max    IQR
##   <fct>       <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 10-fold cv     40      0 0.744  0.817 0.810 0.846 0.0244
## 2 90/10          40      0 0.630  0.762 0.753 0.813 0.0528
## 3 70/30          40      0 0.654  0.780 0.770 0.825 0.0518
## 4 50/50          40      0 0.622  0.793 0.777 0.833 0.0386
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  accuracy by acro
## Kruskal-Wallis chi-squared = 47.311, df = 3, p-value = 2.984e-10
```

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

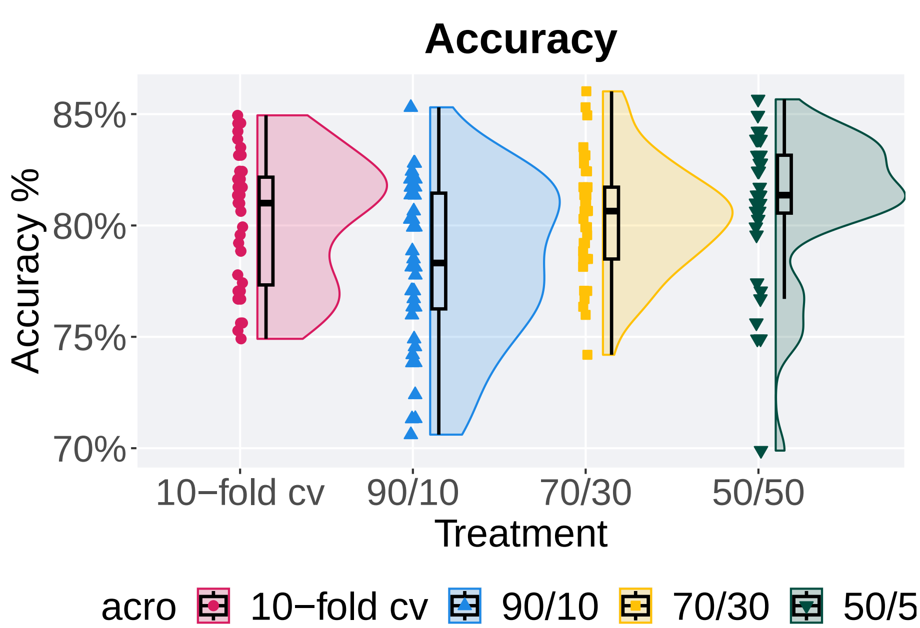
```
pairwise.wilcox.test(x = performance$accuracy, g = performance$acro,
                     p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = "t")
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$accuracy and performance$acro
##
##       10-fold cv 90/10   70/30
## 90/10 1.7e-09    -       -
## 70/30 1.3e-05    0.16648 -
## 50/50 0.00016    0.01281 1.00000
##
## P value adjustment method: bonferroni
```

2.2.3 Task 167168

```
filter(scores, taskid == task_id_lists[3]) %>%
  ggplot(., aes(x = acro, y = accuracy, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
               size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
             size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name = "Accuracy %",
    labels = scales::percent

  ) +
  scale_x_discrete(
    name = "Treatment"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Accuracy") +
  p_theme
```



Summary statistics for the generation a satisfactory solution is found.

```
performance <- filter(scores, taskid == task_id_lists[3])
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(accuracy)),
    min = min(accuracy, na.rm = TRUE),
    median = median(accuracy, na.rm = TRUE),
    mean = mean(accuracy, na.rm = TRUE),
    max = max(accuracy, na.rm = TRUE),
    IQR = IQR(accuracy, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acro       count na_cnt   min median  mean   max    IQR
##   <fct>      <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 10-fold cv    40      0 0.749  0.810 0.803 0.849 0.0484
## 2 90/10         40      0 0.706  0.783 0.783 0.853 0.0520
## 3 70/30         40      0 0.742  0.806 0.803 0.860 0.0323
## 4 50/50         40      0 0.699  0.814 0.811 0.857 0.0260
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  accuracy by acro
## Kruskal-Wallis chi-squared = 14.599, df = 3, p-value = 0.002193
```

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

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

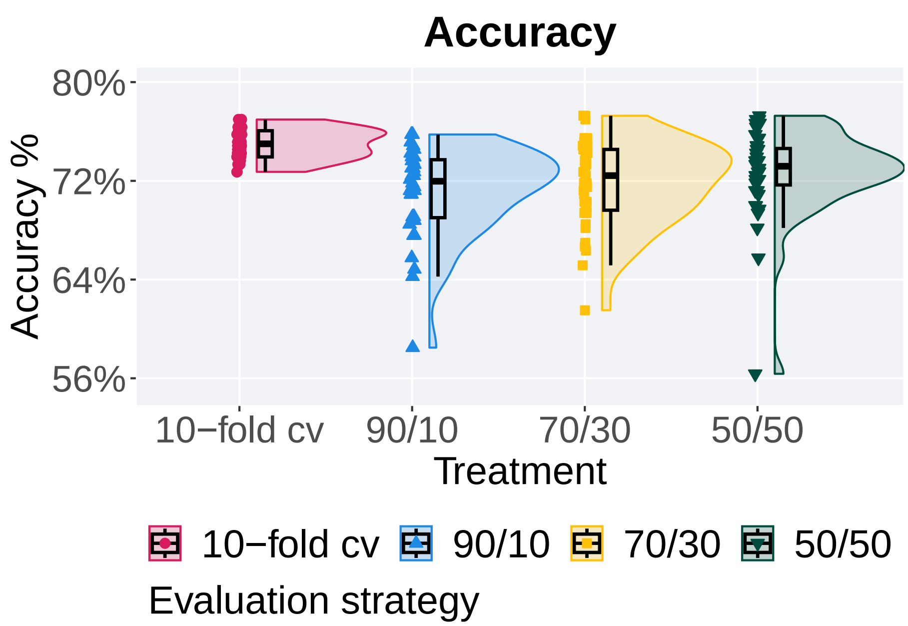
```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$accuracy and performance$acro
```

```
##
##       10-fold cv 90/10  70/30
## 90/10 0.1097     -      -
## 70/30 1.0000     0.1185 -
## 50/50 1.0000     0.0026 0.2697
##
## P value adjustment method: bonferroni
```

2.2.4 Task 167161

```
filter(scores, taskid == task_id_lists[4]) %>%
  ggplot(., aes(x = acro, y = accuracy, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
               size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
             size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name = "Accuracy %",
    limits=c(.55, .80),
    breaks=c(.56,.64,.72,.8),
    labels = scales::percent

  ) +
  scale_x_discrete(
    name = "Treatment"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Accuracy") +
  p_theme +
  guides(
    shape=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    color=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    fill=guide_legend(nrow = 1, title.position = "bottom",
                      title = "Evaluation strategy")
  )
```



Summary statistics for the generation a satisfactory solution is found.

```
performance <- filter(scores, taskid == task_id_lists[4])
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(accuracy)),
    min = min(accuracy, na.rm = TRUE),
    median = median(accuracy, na.rm = TRUE),
    mean = mean(accuracy, na.rm = TRUE),
    max = max(accuracy, na.rm = TRUE),
    IQR = IQR(accuracy, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acro        count na_cnt   min median  mean   max    IQR
##   <fct>       <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 10-fold cv     40      0 0.727  0.75  0.75  0.770 0.0212
## 2 90/10          40      0 0.585  0.720 0.712 0.758 0.0470
## 3 70/30          40      0 0.615  0.724 0.718 0.773 0.0492
## 4 50/50          40      0 0.564  0.732 0.727 0.773 0.0295
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  accuracy by acro
## Kruskal-Wallis chi-squared = 37.759, df = 3, p-value = 3.179e-08
```

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

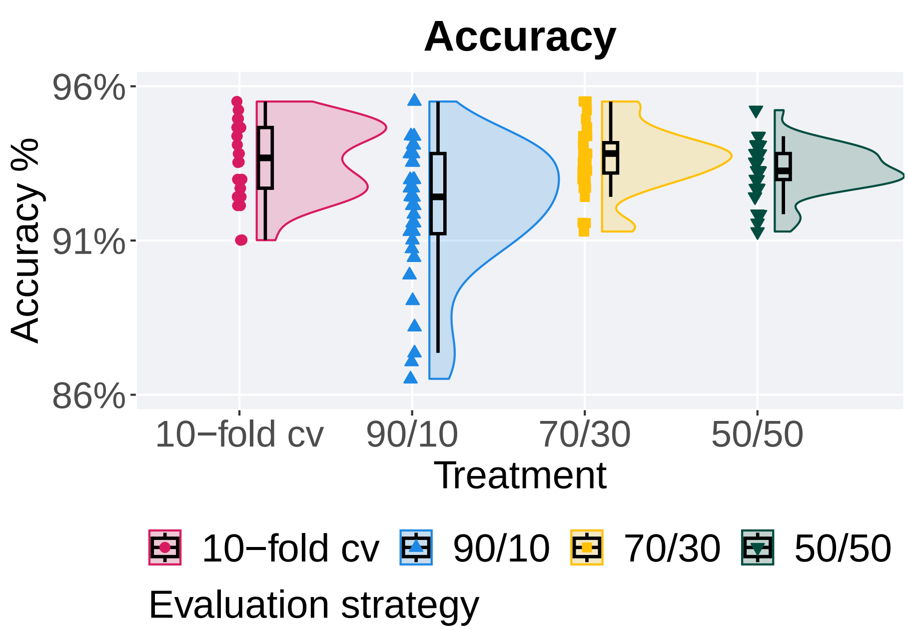
```r
pairwise.wilcox.test(x = performance$accuracy, g = performance$acro,
                     p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = "t")
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$accuracy and performance$acro
##
##       10-fold cv 90/10   70/30
## 90/10 5.7e-08    -       -
## 70/30 2.2e-05    1.00000 -
## 50/50 0.00077    0.27098 1.00000
##
## P value adjustment method: bonferroni
```

2.2.5 Task 167185

```r
filter(scores, taskid == task_id_lists[5]) %>%
  ggplot(., aes(x = acro, y = accuracy, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
               size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
             size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name = "Accuracy %",
    limits=c(.86, .96),
    breaks=c(.86,.91,.96),
    labels = scales::percent
```

```
) +
scale_x_discrete(
  name = "Treatment"
) +
scale_shape_manual(values = SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle("Accuracy") +
p_theme +
guides(
  shape=guide_legend(nrow = 1, title.position = "bottom",
                   title = "Evaluation strategy"),
  color=guide_legend(nrow = 1, title.position = "bottom",
                   title = "Evaluation strategy"),
  fill=guide_legend(nrow = 1, title.position = "bottom",
                  title = "Evaluation strategy")
)
```



Summary statistics for the generation a satisfactory solution is found.

```
performance <- filter(scores, taskid == task_id_lists[5])
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
```

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

```
## # A tibble: 4 x 8
##   acro       count na_cnt   min median  mean   max     IQR
##   <fct>      <int>  <int> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1 10-fold cv    40      0 0.910  0.937 0.936 0.955 0.0197
## 2 90/10         40      0 0.865  0.924 0.920 0.955 0.0260
## 3 70/30         40      0 0.913  0.938 0.936 0.955 0.00983
## 4 50/50         40      0 0.913  0.933 0.933 0.952 0.00843
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  accuracy by acro
## Kruskal-Wallis chi-squared = 20.962, df = 3, p-value = 0.0001072
```

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

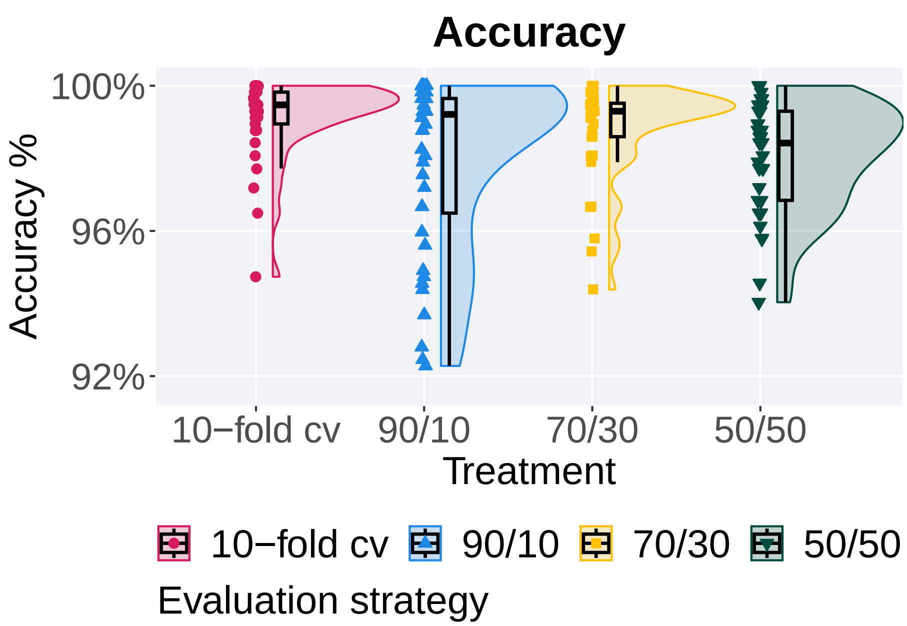
```
pairwise.wilcox.test(x = performance$accuracy, g = performance$acro,
                     p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = "t")
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$accuracy and performance$acro
##
##       10-fold cv 90/10   70/30
## 90/10 0.00095    -       -
## 70/30 1.00000    0.00110 -
## 50/50 0.89845    0.03149 0.30642
##
## P value adjustment method: bonferroni
```

2.2.6 Task 189905

```
filter(scores, taskid == task_id_lists[6]) %>%
  ggplot(., aes(x = acro, y = accuracy, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
               size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
             size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name = "Accuracy %",
    limits=c(.916,1.001),
    breaks=c(.92,.96,1),
    labels = scales::percent

  ) +
  scale_x_discrete(
    name = "Treatment"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Accuracy") +
  p_theme +
  guides(
    shape=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    color=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    fill=guide_legend(nrow = 1, title.position = "bottom",
                      title = "Evaluation strategy")
  )
```



Summary statistics for the generation a satisfactory solution is found.

```
performance <- filter(scores, taskid == task_id_lists[6])
performance %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(accuracy)),
    min = min(accuracy, na.rm = TRUE),
    median = median(accuracy, na.rm = TRUE),
    mean = mean(accuracy, na.rm = TRUE),
    max = max(accuracy, na.rm = TRUE),
    IQR = IQR(accuracy, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acro        count na_cnt   min median  mean   max     IQR
##   <fct>       <int>  <int> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1 10-fold cv     40      0 0.947  0.995 0.992     1 0.00877
## 2 90/10          40      0 0.923  0.992 0.979     1 0.0316
## 3 70/30          40      0 0.944  0.993 0.988     1 0.00921
## 4 50/50          40      0 0.940  0.984 0.980     1 0.0246
```

Kruskal–Wallis test illustrates evidence of statistical differences.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  accuracy by acro
## Kruskal-Wallis chi-squared = 15.64, df = 3, p-value = 0.001344
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$accuracy and performance$acro
##
##       10-fold cv 90/10   70/30
## 90/10 0.25039    -       -
## 70/30 0.72764    1.00000 -
## 50/50 0.00044    1.00000 0.03839
##
## P value adjustment method: bonferroni
```

Chapter 3

Complexity results

Here we report the complexity for **final pipeline returned from a TPOT2 run** on each OpenML task. 40 replicates were conducted for each evaluation strategy explored.

3.1 Analysis setup

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

comps <- read.csv("./complexity.csv", header = TRUE, stringsAsFactors = FALSE)
comps$acro <- factor(comps$acro, levels = NAMES)
```

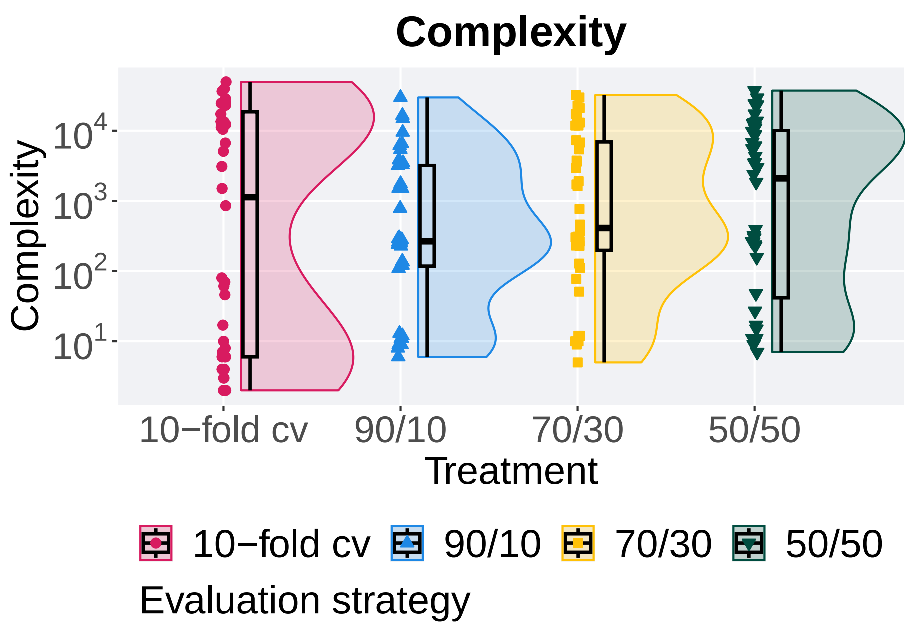
3.2 Accuracy per OpenML task

Accuracy of the returned pipeline from an evolutionary run.

3.2.1 Task 167104

```
filter(comps, taskid == task_id_lists[1]) %>%
  ggplot(., aes(x = acro, y = complexity, color = acro,
                fill = acro, shape = acro)) +
```

```
geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                 scale = "width", alpha = 0.2, width = 1.5) +
geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
             size = 0.8, position = position_nudge(x = .15, y = 0)) +
geom_point(position = position_jitter(width = .015, height = .0001),
           size = 2.0, alpha = 1.0) +
scale_y_log10(
  name = "Complexity",
  breaks = trans_breaks("log10", function(x) 10^x),
  labels = trans_format("log10", math_format(10^.x))
) +
scale_x_discrete(
  name = "Treatment"
) +
scale_shape_manual(values = SHAPE,) +
scale_colour_manual(values = cb_palette,) +
scale_fill_manual(values = cb_palette,) +
ggtitle("Complexity") +
p_theme +
guides(
  shape=guide_legend(nrow = 1, title.position = "bottom",
                     title = "Evaluation strategy"),
  color=guide_legend(nrow = 1, title.position = "bottom",
                     title = "Evaluation strategy"),
  fill=guide_legend(nrow = 1, title.position = "bottom",
                    title = "Evaluation strategy")
)
```



Summary statistics for the generation a satisfactory solution is found.

```
complexity <- filter(comps, taskid == task_id_lists[1])
complexity %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(complexity)),
    min = min(complexity, na.rm = TRUE),
    median = median(complexity, na.rm = TRUE),
    mean = mean(complexity, na.rm = TRUE),
    max = max(complexity, na.rm = TRUE),
    IQR = IQR(complexity, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acro        count na_cnt   min median  mean   max    IQR
##   <fct>       <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 10-fold cv     40      0     2  1178   9852. 49555 18698.
## 2 90/10          40      0     6   266.  2825. 29768  3084.
## 3 70/30          40      0     5   414.  5372. 32167  6711.
## 4 50/50          40      0     7  2119   6275. 37256 10039.
```

Kruskal–Wallis test illustrates evidence of no statistical differences.

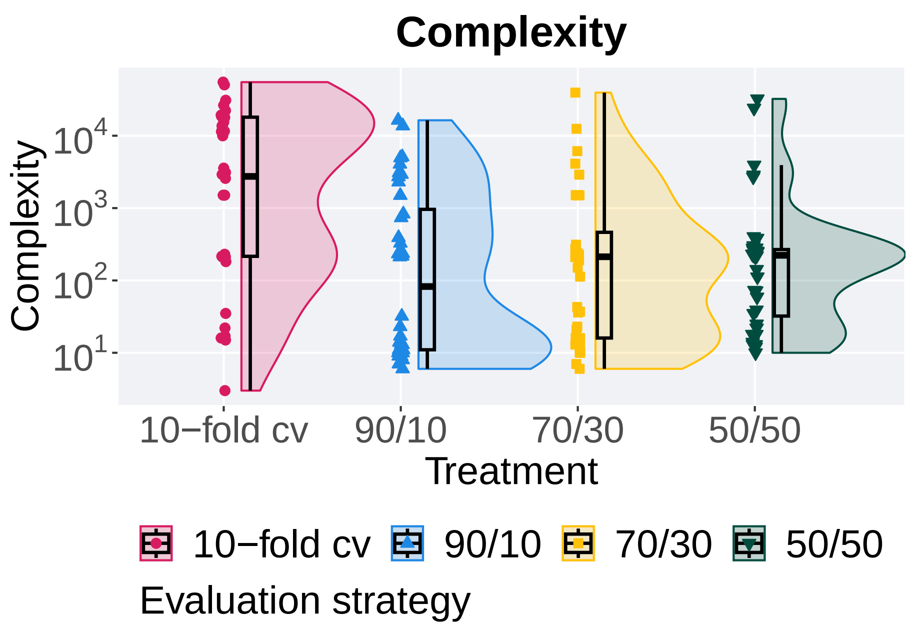
```
kruskal.test(complexity ~ acro, data = complexity)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  complexity by acro
## Kruskal-Wallis chi-squared = 2.4209, df = 3, p-value = 0.4898
```

3.2.2 Task 167184

```
filter(comps, taskid == task_id_lists[2]) %>%
  ggplot(., aes(x = acro, y = complexity, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
               size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
             size = 2.0, alpha = 1.0) +
  scale_y_log10(
    name = "Complexity",
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x)),

  ) +
  scale_x_discrete(
    name = "Treatment"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Complexity") +
  p_theme +
  guides(
    shape=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    color=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    fill=guide_legend(nrow = 1, title.position = "bottom",
                      title = "Evaluation strategy")
  )
```



Summary statistics for the generation a satisfactory solution is found.

```
complexity <- filter(comps, taskid == task_id_lists[2])
complexity %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(complexity)),
    min = min(complexity, na.rm = TRUE),
    median = median(complexity, na.rm = TRUE),
    mean = mean(complexity, na.rm = TRUE),
    max = max(complexity, na.rm = TRUE),
    IQR = IQR(complexity, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acro        count na_cnt   min median  mean   max    IQR
##   <fct>       <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 10-fold cv     40      0     3  2746. 9800. 55067 17839
## 2 90/10          40      0     6   122  1513. 16357   985.
## 3 70/30          40      0     6   212. 1898. 39455   594.
## 4 50/50          40      0    10   222. 1780. 32307   235.
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```r
kruskal.test(complexity ~ acro, data = complexity)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  complexity by acro
## Kruskal-Wallis chi-squared = 19.375, df = 3, p-value = 0.0002287
```

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

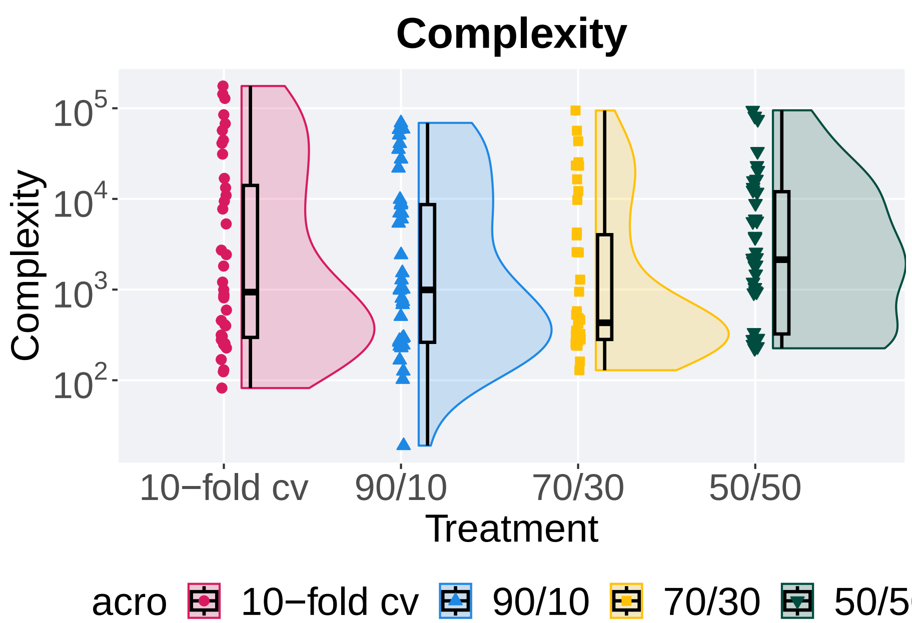
```r
pairwise.wilcox.test(x = complexity$complexity, g = complexity$acro,
                     p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = "l")
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  complexity$complexity and complexity$acro
##
##       10-fold cv 90/10   70/30
## 90/10 0.00036    -       -
## 70/30 0.00117    1.00000 -
## 50/50 0.00742    1.00000 1.00000
##
## P value adjustment method: bonferroni
```

3.2.3 Task 167168

```r
filter(comps, taskid == task_id_lists[3]) %>%
  ggplot(., aes(x = acro, y = complexity, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
               size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
             size = 2.0, alpha = 1.0) +
  scale_y_log10(
    name = "Complexity",
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x)),
  ) +
```

```
  scale_x_discrete(
    name = "Treatment"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Complexity") +
  p_theme
```



Summary statistics for the generation a satisfactory solution is found.

```
complexity <- filter(comps, taskid == task_id_lists[3])
complexity %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(complexity)),
    min = min(complexity, na.rm = TRUE),
    median = median(complexity, na.rm = TRUE),
    mean = mean(complexity, na.rm = TRUE),
    max = max(complexity, na.rm = TRUE),
    IQR = IQR(complexity, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
```

```
##   acro       count na_cnt   min median   mean    max    IQR
##   <fct>      <int>  <int> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 10-fold cv    40      0    82   940. 21344. 176015 13874.
## 2 90/10         40      0    19   992. 10726.  68977  8396.
## 3 70/30         40      0   129   430.  8209.  94505  3750.
## 4 50/50         40      0   225  2140. 11399.  94975 11698.
```

Kruskal–Wallis test illustrates evidence of no statistical differences.

```
kruskal.test(complexity ~ acro, data = complexity)
```

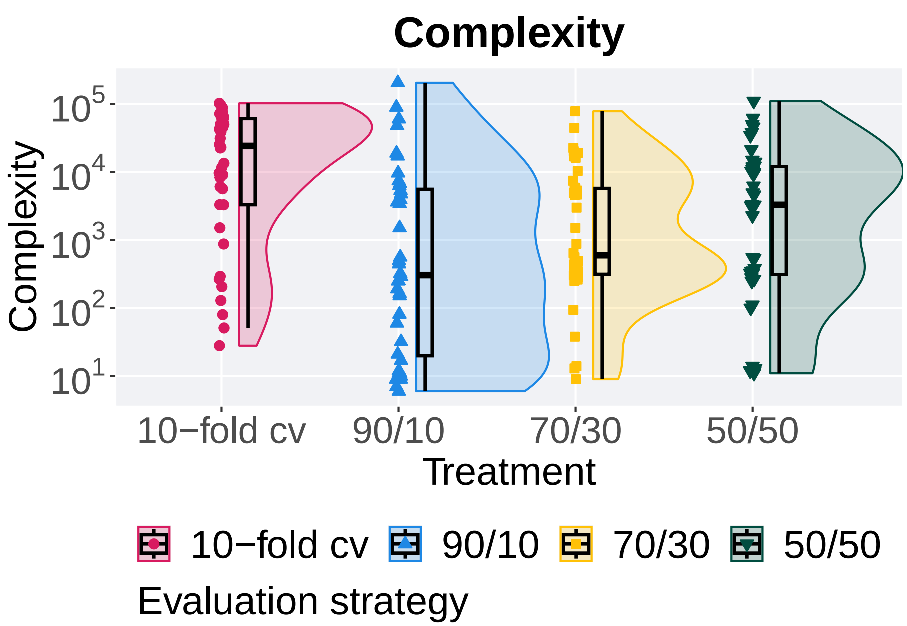
```
##
##  Kruskal-Wallis rank sum test
##
## data:  complexity by acro
## Kruskal-Wallis chi-squared = 3.0019, df = 3, p-value = 0.3913
```

3.2.4 Task 167161

```
filter(comps, taskid == task_id_lists[4]) %>%
  ggplot(., aes(x = acro, y = complexity, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
               size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
             size = 2.0, alpha = 1.0) +
  scale_y_log10(
    name = "Complexity",
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x)),

  ) +
  scale_x_discrete(
    name = "Treatment"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Complexity") +
  p_theme +
  guides(
```

```
    shape=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    color=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    fill=guide_legend(nrow = 1, title.position = "bottom",
                      title = "Evaluation strategy")
  )
```



Summary statistics for the generation a satisfactory solution is found.

```
complexity <- filter(comps, taskid == task_id_lists[4])
complexity %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(complexity)),
    min = min(complexity, na.rm = TRUE),
    median = median(complexity, na.rm = TRUE),
    mean = mean(complexity, na.rm = TRUE),
    max = max(complexity, na.rm = TRUE),
    IQR = IQR(complexity, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
```

```
##   acro       count na_cnt   min median   mean    max    IQR
##   <fct>      <int>  <int> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 10-fold cv    40      0    28 24090. 32974. 101704 57312.
## 2 90/10         40      0     6   306. 12308. 203928  5550.
## 3 70/30         40      0     9   600.  7058.  77688  5416
## 4 50/50         40      0    11  3278. 12379. 109417 11654.
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(complexity ~ acro, data = complexity)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  complexity by acro
## Kruskal-Wallis chi-squared = 25.893, df = 3, p-value = 1.004e-05
```

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

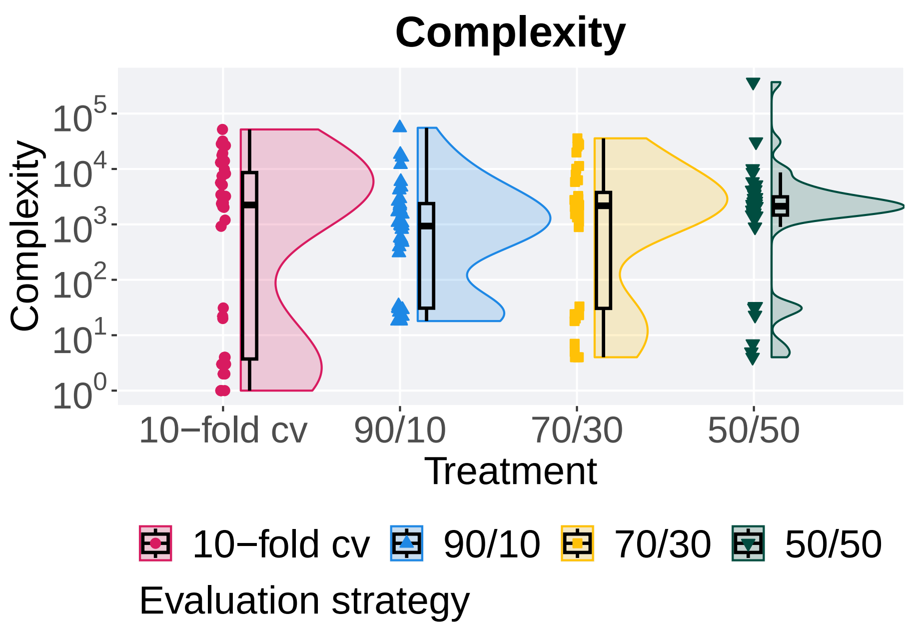
```
pairwise.wilcox.test(x = complexity$complexity, g = complexity$acro,
                     p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = "l")
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  complexity$complexity and complexity$acro
##
##       10-fold cv 90/10   70/30
## 90/10 4e-05      -       -
## 70/30 0.00029    1.00000 -
## 50/50 0.00574    1.00000 1.00000
##
## P value adjustment method: bonferroni
```

3.2.5 Task 167185

```
filter(comps, taskid == task_id_lists[5]) %>%
  ggplot(., aes(x = acro, y = complexity, color = acro,
                fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                   scale = "width", alpha = 0.2, width = 1.5) +
```

```
geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
             size = 0.8, position = position_nudge(x = .15, y = 0)) +
geom_point(position = position_jitter(width = .015, height = .0001),
           size = 2.0, alpha = 1.0) +
scale_y_log10(
  name = "Complexity",
  breaks = trans_breaks("log10", function(x) 10^x),
  labels = trans_format("log10", math_format(10^.x)),
) +
scale_x_discrete(
  name = "Treatment"
) +
scale_shape_manual(values = SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle("Complexity") +
p_theme +
guides(
  shape=guide_legend(nrow = 1, title.position = "bottom",
                     title = "Evaluation strategy"),
  color=guide_legend(nrow = 1, title.position = "bottom",
                     title = "Evaluation strategy"),
  fill=guide_legend(nrow = 1, title.position = "bottom",
                    title = "Evaluation strategy")
)
```



Summary statistics for the generation a satisfactory solution is found.

```
complexity <- filter(comps, taskid == task_id_lists[5])
complexity %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(complexity)),
    min = min(complexity, na.rm = TRUE),
    median = median(complexity, na.rm = TRUE),
    mean = mean(complexity, na.rm = TRUE),
    max = max(complexity, na.rm = TRUE),
    IQR = IQR(complexity, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acro        count na_cnt   min median    mean    max   IQR
##   <fct>       <int>  <int> <dbl>  <dbl>   <dbl>  <dbl> <dbl>
## 1 10-fold cv     40      0     1   2244   6931.  51694 8638.
## 2 90/10          40      0    18    934.  3667.  55371 2348.
## 3 70/30          40      0     4   2169   5487.  35738 3874.
## 4 50/50          40      0     4   2127  12314. 369366 1675
```

Kruskal–Wallis test illustrates evidence of no statistical differences.

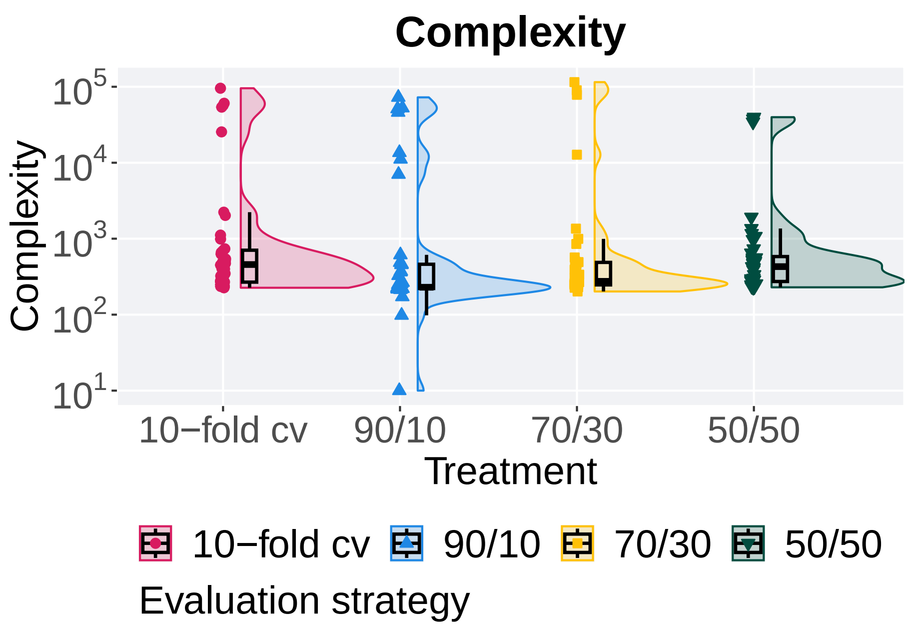
```
kruskal.test(complexity ~ acro, data = complexity)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  complexity by acro
## Kruskal-Wallis chi-squared = 3.3929, df = 3, p-value = 0.3349
```

3.2.6 Task 189905

```
filter(comps, taskid == task_id_lists[6]) %>%
  ggplot(., aes(x = acro, y = complexity, color = acro,
              fill = acro, shape = acro)) +
  geom_flat_violin(position = position_nudge(x = 0.1, y = 0),
                  scale = "width", alpha = 0.2, width = 1.5) +
  geom_boxplot(color = "black", width = .08, outlier.shape = NA, alpha = 0.0,
              size = 0.8, position = position_nudge(x = .15, y = 0)) +
  geom_point(position = position_jitter(width = .015, height = .0001),
            size = 2.0, alpha = 1.0) +
  scale_y_log10(
    name = "Complexity",
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x)),

  ) +
  scale_x_discrete(
    name = "Treatment"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Complexity") +
  p_theme +
  guides(
    shape=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    color=guide_legend(nrow = 1, title.position = "bottom",
                       title = "Evaluation strategy"),
    fill=guide_legend(nrow = 1, title.position = "bottom",
                      title = "Evaluation strategy")
  )
```



Summary statistics for the generation a satisfactory solution is found.

```
complexity <- filter(comps, taskid == task_id_lists[6])
complexity %>%
  group_by(acro) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(complexity)),
    min = min(complexity, na.rm = TRUE),
    median = median(complexity, na.rm = TRUE),
    mean = mean(complexity, na.rm = TRUE),
    max = max(complexity, na.rm = TRUE),
    IQR = IQR(complexity, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acro        count na_cnt   min median  mean    max   IQR
##   <fct>       <int>  <int> <dbl>  <dbl> <dbl>  <dbl> <dbl>
## 1 10-fold cv     40      0   226    457 7771.  95634  437.
## 2 90/10          40      0    10    230. 6579.  72557  240.
## 3 70/30          40      0   202    274. 7736. 115088  241
## 4 50/50          40      0   229    429 3262.  39776  310.
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```r
kruskal.test(complexity ~ acro, data = complexity)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  complexity by acro
## Kruskal-Wallis chi-squared = 17.387, df = 3, p-value = 0.0005884
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  complexity$complexity and complexity$acro
##
##       10-fold cv 90/10  70/30
## 90/10 0.0019     -      -
## 70/30 0.1592     1.0000 -
## 50/50 1.0000     1.0000 1.0000
##
## P value adjustment method: bonferroni
```