Supplemental Material for 'Runtime phylogenetic analysis enables extreme subsampling for test-based problems'

Alexander Lalejini, Marcos Sanson, Jack Garbus, Matthew Andres Moreno, and Emily Dolson

2024-01-27

Contents

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 source code/configuration files for this supplemental material can be found in this GitHub repository.

Our supplemental material includes the following:

- Data availability (Section 2)
- Local compilation (Section 4)
- GP instruction set (Section 3)

1.2 Contributing authors

- Alexander Lalejini
- Marcos Sanson
- Jack Garbus
- Emily Dolson

Chapter 2

Data Availability

2.1 Source code

The source code for this work is publicly accessible on GitHub: https://github.com/amlalejini/GECCO-2024-phylogeny-informed-subsampling.

2.1.1 Experiment software dependencies

- SignalGP: https://github.com/amlalejini/SignalGP
  - commit hash: 114e0f07cb31370ab5191516679889e387cda73b
- Empirical: https://github.com/devosoft/Empirical
  - commit hash: 5955a1cae2a5de36aa3a65df060a56b38f575bd0
- psb-cpp: https://github.com/amlalejini/psb-cpp
  - commit hash: e49896b957574ccd2f9e6e97e812971a0aa77f4b

2.2 Training and testing sets

The training and testing sets used for program synthesis problems can be found on GitHub: https://github.com/amlalejini/GECCO-2024-phylogeny-informed-subsampling/tree/main/experiments/2023-12-30-psynth/hpc/config.

2.3 Experimental results

All of our experimental data is available online from our OSF respository: https://osf.io/h3f52/

Chapter 3

SignalGP instruction set

Below, we document the instruction set used in our GP system for our 2023 GPTP experiments.

Abbreviations:

- EOP: End of program
- Reg: local register
  - Reg[0] indicates the value at the register specified by an instruction's first *argument*, Reg[1] indicates the value at the register specified by an instruction's second argument, and Reg[2] indicates the value at the register specified by the instruction's third argument.
  - Reg[0], Reg[1], *etc*: Register 0, Register 1, *etc.*
- Input: input buffer
  - Follows same scheme as Reg
- Output: output buffer
  - Follows same scheme as Reg
- Global: global memory buffer
  - Follows same scheme as Reg
- Arg: Instruction argument
  - Arg[i] indicates the i'th instruction argument (an integer encoded in the genome)
  - E.g., Arg[0] is an instruction's first argument

Instructions that would produce undefined behavior (e.g., division by zero) are treated as no operations.

3.1 Default Instructions

I.e., instructions used across all diagnostic tasks.

Instruction	Arguments Used	Description
Nop	0	No operation
Not	1	Reg[0] = !Reg[0]
Inc	1	Reg[0] = Reg[0] + 1
Dec	1	Reg[0] = Reg[0] - 1
Add	3	Reg[0] = Reg[1] + Reg[2]
Sub	3	Reg[0] = Reg[1] - Reg[2]
Mult	3	Reg[0] = Reg[1] * Reg[2]
Div	3	Reg[0] = Reg[1] / Reg[2]
Mod	3	Reg[0] = Reg[1] % Reg[2]
Nand	2	Reg[0] = !(R1g[0] & Reg[2])
TestEqu	3	Reg[0] = Reg[1] == Reg[2]
TestNEqu	3	Reg[0] = Reg[1] != Reg[2]
TestLess	3	Reg[0] = Reg[1] < Reg[2]
TestLessEqu	3	Reg[0] = Reg[1] <= Reg[2]
TestGreater	3	Reg[0] = Reg[1] > Reg[2]
TestGreaterEqu	3	Reg[0] = Reg[1] >= Reg[2]
SetMem	2	Reg[0] = Arg[1]
Terminal	1	Reg[0] = double value encoded by instruction tag
CopyMem	2	Reg[0] = Reg[1]
SwapMem	2	Swap(Reg[0], Reg[1])
InputToWorking	2	Reg[0] = Input[1]
WorkingToOutput	2	Output[1] = Reg[0]
If	1	If Reg[0] != 0, proceed. Otherwise skip to the next Close or EOP.

Instruction	Arguments Used	Description
While	1	While Reg[0] != 0, loop. Otherwise skip to next Close or EOP.
Close	0	Indicate the end of a control block of code (e.g., loop, if).
Break	0	Break out of current control flow (e.g., loop).
Call	0	Call a function, using this instruction's tag to determine which function is called.
Routine	0	Same as call, but local memory is shared. Sort of like a jump that will jump back when the routine ends.
Return	0	Return from the current function call.
WorkingToGlobal	2	Global[1] = Reg[0]
GlobalToWorking	2	Reg[1] = Global[0]
FullGlobalToWorking	0	Copy entire global memory buffer into working memory buffer
FullWorkingToGlobal	0	Copy entire working memory buffer into global memory buffer

Note that Nand performs a bitwise operation.

3.2 Problem-specific instructions

Each problem has problem-specific instructions for producing output.

3.2.1 Bouncing Balls

- SubmitOutput

3.2.2 Dice Game

- SubmitOutput

3.2.3 Fizz Buzz

- SubmitFizz
- SubmitBuzz
- SubmitFizzBuzz
- SubmitEcho

3.2.4 For loop index

- SubmitOutput

3.2.5 GCD

- SubmitOutput

3.2.6 Grade

- SubmitA
- SubmitB
- SubmitC
- SubmitD
- SubmitF

3.2.7 Median

- SubmitOutput

3.2.8 Small or large

- SubmitSmall
- SubmitLarge
- SubmitNeither

3.2.9 Smallest

- SubmitOutput

3.2.10 Snow Day

- SubmitOutput

Chapter 4

Local compilation

You will need a C++ compiler that supports at least C++17. We used g++13 for all local compilations.

First, clone GECCO−2024−phylogeny−informed−subsampling repository that contains the code needed to run our experiment software: https://github.com/amlalejini/GECCO-2024-phylogeny-informed-subsampling.

Once cloned, cd into your local GECCO−2024−phylogeny−informed−subsampling repository directory. Then, initialize and update all of the git submodules:

```
git submodule update --init --recursive
```

Once the submodules are updated, you should be able to compile either the diagnostics or program synthesis experiment code. To specify which experiment you would like to compile, adjust the PROJECT variable in the Makefile.

- PROJECT := prog_synth for compiling the program synthesis code
- PROJECT := diagnostics for compiling the selection scheme diagnostics code

To compile in debug mode, run make debug from repository directory, and to compile in release mode (all optimizations turned on), run make native.

If you get the following error:

```
third-party/Empirical/include/emp/matching/../../../third
    -party/robin-hood-hashing/src/include/robin_hood.h
    :54:14: fatal error: sys/auxv.h: No such file or
    directory
    54 | #     include <sys/auxv.h> // for getauxval
```

you can fix it by doing the following:

```
cd third-party/Empirical/third-party/robin-hood-hashing
git checkout master
```

Once you have an executable, you can generate a configuration file by running:

- ./prog_synth --gen prog_synth.cfg or
- ./diagnostics --gen diagnostics.cfg

Chapter 5

Exploitation rate diagnostic experiments

```
experiment_slug <- "2023-12-28-phylo-sampling-diag"

working_directory <- paste0(
  "experiments/",
  experiment_slug,
  "/analysis/"
)

if (exists("bookdown_wd_prefix")) {
  working_directory <- paste0(
    bookdown_wd_prefix,
    working_directory
  )
}
```

5.1 Dependencies

```
library(tidyverse)

## -- Attaching core tidyverse packages ------------------------------------------------
    tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
```

```
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts
    ------------------------------------------------------------------------
     tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib
    .org/>) to force all conflicts to become errors

library(cowplot)

##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate
    ':
##
##     stamp

library(RColorBrewer)
library(khroma)
library(rstatix)

##
## Attaching package: 'rstatix'
##
## The following object is masked from 'package:stats':
##
##     filter

library(knitr)
source("https://gist.githubusercontent.com/benmarwick/2
    a1bb0133ff568cbe28d/raw/
    fb53bd97121f7f9ce947837ef1a4c65a73bffb3f/geom_flat_
    violin.R")

print(version)

##                _
## platform       aarch64-apple-darwin20
## arch           aarch64
## os             darwin20
## system         aarch64, darwin20
## status
## major          4
```

```
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

5.2 Setup

```
# Configure our default graphing theme
theme_set(theme_cowplot())
# Create a directory to store plots
plot_directory <- paste0(working_directory, "plots/")
dir.create(plot_directory, showWarnings=FALSE)
# Constants
focal_diagnostic <- "exploitation-rate"
```

5.2.1 Load experiment summary data

```
summary_data_loc <- paste0(working_directory, "data/
    aggregate.csv")
summary_data <- read_csv(summary_data_loc)

## Rows: 1080 Columns: 58
## -- Column specification
    --------------------------------------------------------------------------------

## Delimiter: ","
## chr  (5): DIAGNOSTIC, EVAL_FIT_EST_MODE, EVAL_MODE,
    SELECTION, STOP_MODE
## dbl (53): ACCURACY, CREDIT, DIAGNOSTIC_DIMENSIONALITY,
     EVAL_MAX_PHYLO_SEARCH...
##
## i Use `spec()` to retrieve the full column
    specification for this data.
## i Specify the column types or set `show_col_types =
    FALSE` to quiet this message.

summary_data <- summary_data %>%
  mutate(
```

```
    evals_per_gen = case_when(
      EVAL_MODE == "cohort-full-compete" ~ 1.0 / NUM_
          COHORTS,
      EVAL_MODE == "cohort" ~ 1.0 / NUM_COHORTS,
      EVAL_MODE == "down-sample" ~ TEST_DOWNSAMPLE_RATE,
      EVAL_MODE == "full" ~ 1.0,
      EVAL_MODE == "indiv-rand-sample" ~ TEST_DOWNSAMPLE_
          RATE,
      EVAL_MODE == "phylo-informed-sample" ~ TEST_
          DOWNSAMPLE_RATE
    ),
    EVAL_FIT_EST_MODE = case_when(
      EVAL_FIT_EST_MODE == "ancestor-opt" ~ "ancestor",
      EVAL_FIT_EST_MODE == "relative-opt" ~ "relative",
      .default = EVAL_FIT_EST_MODE
    ),
    .keep = "all"
  ) %>%
  mutate(
    eval_label = case_when(
      # Clean up down-sample label
      EVAL_MODE == "down-sample" & EVAL_FIT_EST_MODE != "
          none" ~ paste("down-sample", EVAL_FIT_EST_MODE,
          sep="-"),
      .default = EVAL_MODE
    ),
  ) %>%
  mutate(
    evals_per_gen = as.factor(evals_per_gen),
    DIAGNOSTIC = as.factor(DIAGNOSTIC),
    SELECTION = as.factor(SELECTION),
    EVAL_MODE = as.factor(EVAL_MODE),
    NUM_COHORTS = as.factor(NUM_COHORTS),
    TEST_DOWNSAMPLE_RATE = as.factor(TEST_DOWNSAMPLE_RATE
        ),
    EVAL_FIT_EST_MODE = factor(
      EVAL_FIT_EST_MODE,
      levels = c(
        "none",
        "ancestor",
        "relative"
      ),
      labels = c(
        "None",
        "Ancestor",
        "Relative"
```

```
      )
    )
  )

# Grab just the exploitation rate data
exploit_summary_data <- filter(
  summary_data,
  DIAGNOSTIC == "exploitation-rate"
)
```

5.2.2 Load experiment time series data

```
ts_data_loc <- paste0(working_directory, "data/time_
    series.csv")
ts_data <- read_csv(ts_data_loc)

## Rows: 108000 Columns: 28
## -- Column specification
    --------------------------------------------------------

## Delimiter: ","
## chr  (4): DIAGNOSTIC, EVAL_FIT_EST_MODE, EVAL_MODE,
    SELECTION
## dbl (24): NUM_COHORTS, SEED, TEST_DOWNSAMPLE_RATE,
    ave_depth, deleterious_st...
##
## i Use `spec()` to retrieve the full column
    specification for this data.
## i Specify the column types or set `show_col_types =
    FALSE` to quiet this message.

ts_data <- ts_data %>%
  mutate(
    evals_per_gen = case_when(
      EVAL_MODE == "cohort-full-compete" ~ 1.0 / NUM_
          COHORTS,
      EVAL_MODE == "cohort" ~ 1.0 / NUM_COHORTS,
      EVAL_MODE == "down-sample" ~ TEST_DOWNSAMPLE_RATE,
      EVAL_MODE == "full" ~ 1.0,
      EVAL_MODE == "indiv-rand-sample" ~ TEST_DOWNSAMPLE_
          RATE,
      EVAL_MODE == "phylo-informed-sample" ~ TEST_
          DOWNSAMPLE_RATE
    ),
    EVAL_FIT_EST_MODE = case_when(
```

```
      EVAL_FIT_EST_MODE == "ancestor-opt" ~ "ancestor",
      EVAL_FIT_EST_MODE == "relative-opt" ~ "relative",
      .default = EVAL_FIT_EST_MODE
    ),
    .keep = "all"
  ) %>%
  mutate(
    eval_label = case_when(
      EVAL_MODE == "down-sample" & EVAL_FIT_EST_MODE != "
          none" ~ paste("down-sample", EVAL_FIT_EST_MODE,
          sep="-"),
      .default = EVAL_MODE
    )
  ) %>%
  mutate(
    evals_per_gen = as.factor(evals_per_gen),
    DIAGNOSTIC = as.factor(DIAGNOSTIC),
    SELECTION = as.factor(SELECTION),
    EVAL_MODE = as.factor(EVAL_MODE),
    NUM_COHORTS = as.factor(NUM_COHORTS),
    TEST_DOWNSAMPLE_RATE = as.factor(TEST_DOWNSAMPLE_RATE
        ),
    EVAL_FIT_EST_MODE = factor(
      EVAL_FIT_EST_MODE,
      levels = c(
        "none",
        "ancestor",
        "relative"
      ),
      labels = c(
        "None",
        "Ancestor",
        "Relative"
      )
    )
  )

# Grab just the exploitation rate data
exploit_ts_data <- ts_data %>%
  filter(DIAGNOSTIC == "exploitation-rate")
```

Summarize time series data:

```
ts_summary_data <- ts_data %>%
  group_by(SEED, DIAGNOSTIC, SELECTION, evals_per_gen,
      eval_label) %>%
```

```
  summarize(
    n = n(),
    avg_num_unique_selected = mean(num_unique_selected),
    total_optimal_trait_coverage_loss = sum(optimal_trait
        _coverage_loss)
  )

## `summarise()` has grouped output by 'SEED', '
    DIAGNOSTIC', 'SELECTION',
## 'evals_per_gen'. You can override using the `.groups`
    argument.
```

5.2.3 Plotting helper functions

The following function assist with exploratory plotting of different measurements from summary and time series data. Note that for these plots, standard lexicase reference is rendered at equivalent number of generations (instead of evaluations).

```
build_plot_summary_data <- function(data, diagnostic,
    selection, response) {
  diag_data <- data %>% filter(DIAGNOSTIC == diagnostic)

  full_median <- median(
    filter(
      diag_data,
      eval_label == "full" & SELECTION == selection
    )[[response]]
  )

  plot <- diag_data %>%
    filter(
      eval_label != "full" & SELECTION == selection
    ) %>%
    ggplot(
      aes_string(
        x = "eval_label",
        y = response,
        fill = "eval_label"
      )
    ) +
    geom_hline(
      yintercept = full_median,
      size = 1.0,
      alpha = 0.7,
```

```
      color = "black",
      linetype="dashed"
    ) +
    geom_flat_violin(
      position = position_nudge(x = .2, y = 0),
      alpha = .8,
      adjust = 1.5
    ) +
    geom_point(
      mapping = aes(color = eval_label),
      position = position_jitter(width = .15),
      size = .5,
      alpha = 0.8
    ) +
    geom_boxplot(
      width = .1,
      outlier.shape = NA,
      alpha = 0.5
    ) +
    scale_y_continuous(
      # limits = c(-0.5, 100)
    ) +
    scale_fill_bright() +
    scale_color_bright() +
    facet_grid(
      SELECTION ~ evals_per_gen,
      # nrow=2,
      labeller = label_both
    ) +
    theme(
      legend.position = "none",
      axis.text.x = element_text(
        angle = 30,
        hjust = 1
      ),
      panel.border = element_rect(color = "gray", size =
          2)
    )

  return(plot)
}

build_plot_time_series_single_sampling <- function(
  data,
  diagnostic,
  selection,
```

```
  sampling_level,
  response
) {

  diag_data <- data %>% filter(
    DIAGNOSTIC == diagnostic &
    SELECTION == selection &
    evals_per_gen == sampling_level
  ) %>%
  mutate(
    sampling_level_label = sampling_level
  )

  full_diag_data <- data %>% filter(
    DIAGNOSTIC == diagnostic & SELECTION == selection &
        eval_label == "full"
  ) %>%
  mutate(
    # Ensure that median line will sit in same facet
    sampling_level_label = sampling_level
  )

  plot <- diag_data %>%
    filter(
      eval_label != "full"
    ) %>%
    ggplot(
      aes_string(
        x = "ts_step",
        # x = "evaluations",
        y = {{ response }}
      )
    ) +
    stat_summary(
      geom = "line",
      fun = mean,
      aes(
        color = eval_label
      )
    ) +
    stat_summary(
      geom = "ribbon",
      fun.data = "mean_cl_boot",
      fun.args = list(conf.int = 0.95),
      alpha = 0.2,
      linetype = 0,
```

```
      aes(
        color = eval_label,
        fill = eval_label
      )
    ) +
    scale_fill_bright() +
    scale_color_bright() +
    # facet_wrap(
    #   ~ sampling_level_label,
    #   ncol = 1,
    #   labeller = label_both
    # ) +
    theme(
      legend.position = "right"
    ) +
    stat_summary(
      data = full_diag_data,
      geom = "line",
      fun = median,
      linetype = "dashed",
      color = "black"
    )

  return(plot)
}

build_plot_time_series <- function(
  data,
  diagnostic,
  selection,
  response
) {
  # Build 1% sampling plot and 10% sampling plot
  p_01 <- data  %>% build_plot_time_series_single_
      sampling(
    diagnostic,
    selection,
    "0.01",
    response
  )
  p_10 <- data %>% build_plot_time_series_single_sampling
      (
    diagnostic,
    selection,
    "0.1",
    response
```

```
  )

  title <- ggdraw() +
    draw_label(
      paste0(diagnostic, " - ", selection),
      fontface = 'bold',
      x = 0,
      hjust = 0
    ) +
    theme(
      # add margin on the left of the drawing canvas,
      # so title is aligned with left edge of first plot
      plot.margin = margin(0, 0, 0, 7)
    )

  plot <- plot_grid(
    title,
    p_01 + labs(title = "1% subsampling") + theme(legend.
        position = "none"),
    p_10 + labs(title = "10% subsampling") + theme(legend
        .position = "bottom"),
    nrow = 3,
    ncol = 1,
    rel_heights = c(0.075, 1, 1)
  )

  return(plot)
}
```

5.3 Aggregate score

5.3.1 Final - Lexicase selection

```
p <- summary_data %>% build_plot_summary_data(
  "exploitation-rate",
  "lexicase",
  "elite_true_agg_score"
)
```

```
## Warning: `aes_string()` was deprecated in ggplot2
    3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more
    information.
```

```
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see
    where this warning was generated.

## Warning: Using `size` aesthetic for lines was
    deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see
    where this warning was generated.

## Warning: The `size` argument of `element_rect()` is
    deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see
    where this warning was generated.

ggsave(
  filename = paste0(plot_directory, "exploit-score-final-
      lex.pdf"),
  plot = p + labs(title = "Exploitation rate - Lexicase
      selection"),
  width = 15,
  height = 10
)

## Warning: Using the `size` aesthetic with geom_polygon
    was deprecated in ggplot2 3.4.0.
## i Please use the `linewidth` aesthetic instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see
    where this warning was generated.
```

5.3.2 Final - Tournament selection

```
p <- summary_data %>% build_plot_summary_data(
  "exploitation-rate",
  "tournament",
  "elite_true_agg_score"
)
ggsave(
  filename = paste0(plot_directory, "exploit-score-final-
      tourn.pdf"),
```

```
  plot = p + labs(title = "Exploitation␣rate␣-␣Tournament
      ␣selection"),
  width = 15,
  height = 10
)
```

5.3.3 Statistical analysis

First, we'll create a table of median / mean values for easy reference.

```
exploit_summary_data %>%
  group_by(DIAGNOSTIC, SELECTION, evals_per_gen, eval_
      label) %>%
  summarize(
    score_median = median(elite_true_agg_score),
    score_mean = mean(elite_true_agg_score),
    replicates = n()
  ) %>%
  kable()
```

```
## `summarise()` has grouped output by 'DIAGNOSTIC', '
    SELECTION', 'evals_per_gen'.
## You can override using the `.groups` argument.
```

DIAGNOSTIC	SELECTION	evals_per_gen	eval_label	score_median	score_mean	repl
exploitation-rate	lexicase	0.01	down-sample	9933.1800	9933.2455	
exploitation-rate	lexicase	0.01	down-sample-ancestor	920.1625	913.6102	
exploitation-rate	lexicase	0.01	indiv-rand-sample	2117.1200	2137.2725	
exploitation-rate	lexicase	0.01	phylo-informed-sample	2157.9350	2162.8605	
exploitation-rate	lexicase	0.1	down-sample	9967.3500	9968.1275	
exploitation-rate	lexicase	0.1	down-sample-ancestor	6976.3600	6985.9325	
exploitation-rate	lexicase	0.1	indiv-rand-sample	9360.5800	9360.2230	
exploitation-rate	lexicase	0.1	phylo-informed-sample	9301.3500	9308.4105	
exploitation-rate	lexicase	1	full	9981.7200	9982.2910	
exploitation-rate	tournament	0.01	down-sample	9650.1650	9650.6660	
exploitation-rate	tournament	0.01	down-sample-ancestor	1023.4150	1011.8228	
exploitation-rate	tournament	0.01	indiv-rand-sample	9969.7650	9969.2945	
exploitation-rate	tournament	0.01	phylo-informed-sample	9970.8950	9970.1455	
exploitation-rate	tournament	0.1	down-sample	9972.3050	9972.0210	
exploitation-rate	tournament	0.1	down-sample-ancestor	9988.9200	9988.9365	
exploitation-rate	tournament	0.1	indiv-rand-sample	9999.8250	9999.8240	
exploitation-rate	tournament	0.1	phylo-informed-sample	9999.7700	9999.7800	
exploitation-rate	tournament	1	full	10000.0000	10000.0000	

Next, we run a Kruskal-Wallis test to check for differences. For these tests, we only compare within a single subsampling level (evals_per_gen) and within the same selection scheme.

```
kw_test <- exploit_summary_data %>%
  filter(eval_label != "full") %>%
  group_by(SELECTION, evals_per_gen) %>%
  kruskal_test(elite_true_agg_score ~ eval_label) %>%
  mutate(sig = (p < 0.05)) %>%
  unite(
    "comparison_group",
    SELECTION,
    evals_per_gen,
    sep = "_",
    remove = FALSE
  )
kable(kw_test)
```

comparison_group	SELECTION	evals_per_gen	.y.	n	statistic	df	p
lexicase_0.01	lexicase	0.01	elite_true_agg_score	80	67.04167	3	0
lexicase_0.1	lexicase	0.1	elite_true_agg_score	80	68.10074	3	0
tournament_0.01	tournament	0.01	elite_true_agg_score	80	66.76541	3	0
tournament_0.1	tournament	0.1	elite_true_agg_score	80	67.17274	3	0

Perform pairwise wilcoxon rank-sum tests for all significant comparison groups.

```
# Grab group names of significant comparisons
sig_kw_groups <- filter(kw_test, p < 0.05)$comparison_
    group

wrs_test <- exploit_summary_data %>%
  unite(
    "comparison_group",
    SELECTION,
    evals_per_gen,
    sep = "_",
    remove = FALSE
  ) %>%
  filter(
    eval_label != "full" & comparison_group %in% sig_kw_
        groups
  ) %>%
  group_by(SELECTION, evals_per_gen) %>%
  pairwise_wilcox_test(elite_true_agg_score ~ eval_label)
      %>%
  adjust_pvalue(method = "holm") %>%
  add_significance("p.adj")
```

```
kable(wrs_test)
```

SELECTION	evals_per_gen	.y.	group1	group2	n1	n
lexicase	0.01	elite_true_agg_score	down-sample	down-sample-ancestor	20	2
lexicase	0.01	elite_true_agg_score	down-sample	indiv-rand-sample	20	2
lexicase	0.01	elite_true_agg_score	down-sample	phylo-informed-sample	20	2
lexicase	0.01	elite_true_agg_score	down-sample-ancestor	indiv-rand-sample	20	2
lexicase	0.01	elite_true_agg_score	down-sample-ancestor	phylo-informed-sample	20	2
lexicase	0.01	elite_true_agg_score	indiv-rand-sample	phylo-informed-sample	20	2
lexicase	0.1	elite_true_agg_score	down-sample	down-sample-ancestor	20	2
lexicase	0.1	elite_true_agg_score	down-sample	indiv-rand-sample	20	2
lexicase	0.1	elite_true_agg_score	down-sample	phylo-informed-sample	20	2
lexicase	0.1	elite_true_agg_score	down-sample-ancestor	indiv-rand-sample	20	2
lexicase	0.1	elite_true_agg_score	down-sample-ancestor	phylo-informed-sample	20	2
lexicase	0.1	elite_true_agg_score	indiv-rand-sample	phylo-informed-sample	20	2
tournament	0.01	elite_true_agg_score	down-sample	down-sample-ancestor	20	2
tournament	0.01	elite_true_agg_score	down-sample	indiv-rand-sample	20	2
tournament	0.01	elite_true_agg_score	down-sample	phylo-informed-sample	20	2
tournament	0.01	elite_true_agg_score	down-sample-ancestor	indiv-rand-sample	20	2
tournament	0.01	elite_true_agg_score	down-sample-ancestor	phylo-informed-sample	20	2
tournament	0.01	elite_true_agg_score	indiv-rand-sample	phylo-informed-sample	20	2
tournament	0.1	elite_true_agg_score	down-sample	down-sample-ancestor	20	2
tournament	0.1	elite_true_agg_score	down-sample	indiv-rand-sample	20	2
tournament	0.1	elite_true_agg_score	down-sample	phylo-informed-sample	20	2
tournament	0.1	elite_true_agg_score	down-sample-ancestor	indiv-rand-sample	20	2
tournament	0.1	elite_true_agg_score	down-sample-ancestor	phylo-informed-sample	20	2
tournament	0.1	elite_true_agg_score	indiv-rand-sample	phylo-informed-sample	20	2

5.3.4 Over time - Lexicase

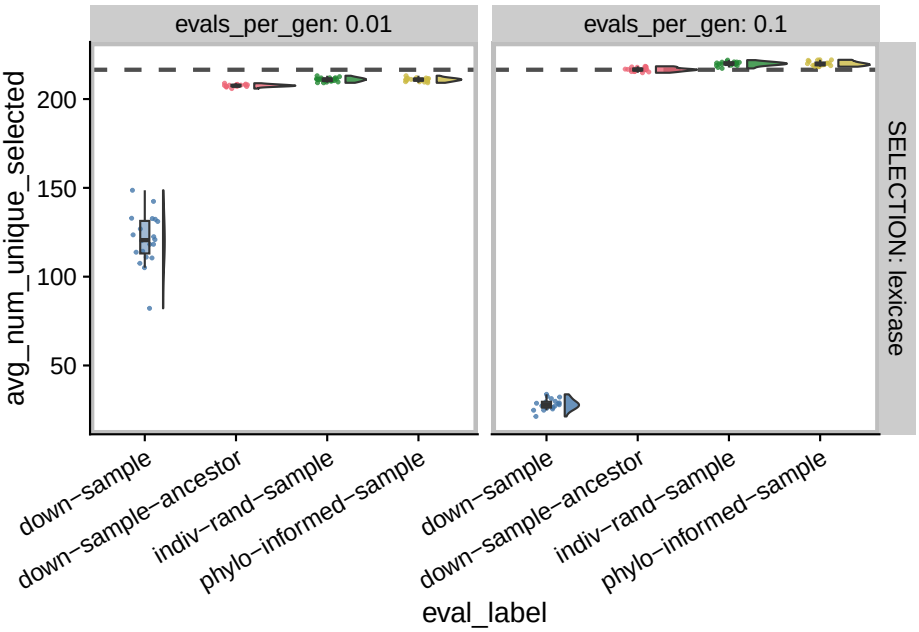
```
p <- ts_data %>% build_plot_time_series(
  "exploitation-rate",
  "lexicase",
  "max_agg_score"
)
ggsave(
  filename = paste0(plot_directory, "exploit-score-ts-lex
      .pdf"),
  plot = p,
  width = 15,
  height = 10
)
```

5.3.5 Over time - Tournament

```
p <- ts_data %>% build_plot_time_series(
  "exploitation-rate",
  "tournament",
  "max_agg_score"
)
ggsave(
  filename = paste0(plot_directory, "exploit-score-ts-
      tourn.pdf"),
  plot = p,
  width = 15,
  height = 10
)
```

5.4 Number unique individual selected

```
build_plot_summary_data(
  ts_summary_data,
  focal_diagnostic,
  "lexicase",
  "avg_num_unique_selected"
)
```



Average number selected by standard lexicase?

```
mean(filter(
  ts_summary_data,
  SELECTION == "lexicase" &
  DIAGNOSTIC == "exploitation-rate" &
  evals_per_gen == "1"
)$avg_num_unique_selected)
```

```
## [1] 216.6185
```

```
mean(filter(
  ts_summary_data,
  SELECTION == "lexicase" &
  DIAGNOSTIC == "exploitation-rate" &
  evals_per_gen == "0.1",
  eval_label == "down-sample"
)$avg_num_unique_selected)
```

```
## [1] 28.099
```

```
mean(filter(
  ts_summary_data,
  SELECTION == "lexicase" &
  DIAGNOSTIC == "exploitation-rate" &
  evals_per_gen == "0.01",
  eval_label == "down-sample"
)$avg_num_unique_selected)
```

```
## [1] 120.7455
```

5.5 Manuscript figures

Figures customized / cleaned up for the manuscript.

```
build_final_score_manuscript_plot <- function(
  selection,
  subsample_rate
) {
  # Extract median values for max aggregate score at same
      evaluation level
  # as sampling regimes
  max_eval <- max(
    filter(exploit_ts_data, evals_per_gen == subsample_
        rate)$evaluations
```

```
  )
  full_eval_steps <- as.numeric(
    levels(
      as.factor(
        filter(exploit_ts_data, eval_label == "full" &
            evaluations >= max_eval)$evaluations # nolint:
             line_length_linter.
      )
    )
  )
  full_eval <- full_eval_steps[which.min( full_eval_steps
       - max_eval )]
  full_median_score_evals <- median(
    filter(
      exploit_ts_data,
      SELECTION == selection & eval_label == "full" &
          evaluations == full_eval
    )$max_agg_score
  )

  plot <- exploit_summary_data %>%
    filter(
      eval_label != "full" &
      SELECTION == selection &
      evals_per_gen == subsample_rate
    ) %>%
    ggplot(
      aes(
        x = eval_label,
        y = elite_true_agg_score,
        fill = eval_label
      )
    ) +
    geom_hline(
      yintercept = full_median_score_evals,
      size = 1.0,
      alpha = 0.7,
      color = "black",
      linetype="dashed"
    ) +
    geom_flat_violin(
      position = position_nudge(x = .2, y = 0),
      alpha = .8,
      adjust = 1.5
    ) +
    geom_point(
```

```
      mapping = aes(color = eval_label),
      position = position_jitter(width = .15),
      size = .5,
      alpha = 0.8
    ) +
    geom_boxplot(
      width = .1,
      outlier.shape = NA,
      alpha = 0.5
    ) +
    scale_y_continuous(
      name = "Aggregate score",
      limits = c(0, 10010)
    ) +
    scale_x_discrete(
      name = "Subsampling regimes",
      breaks = c("down-sample", "down-sample-ancestor", "
          indiv-rand-sample", "phylo-informed-sample"),
      labels = c("DS", "DS+EST", "IRS", "ABS")
    ) +
    scale_fill_bright() +
    scale_color_bright() +
    theme(
      legend.position = "none",
      # axis.text.x = element_text(
      #   angle = 30,
      #   hjust = 1
      # ),
    )
  return(plot)
}
```

Build time series manuscript plot:

```
build_score_over_time_manuscript_plot <- function(
  selection,
  subsample_rate
) {

  max_eval <- max(
    filter(exploit_ts_data, evals_per_gen == subsample_
        rate)$evaluations
  )
  full_eval_steps <- as.numeric(
    levels(
      as.factor(
```

```
      filter(exploit_ts_data, eval_label == "full" &
          evaluations >= max_eval)$evaluations # nolint:
           line_length_linter.
    )
  )
)
full_eval <- full_eval_steps[which.min( full_eval_steps
    - max_eval )]

data <- exploit_ts_data %>%
  filter(
    SELECTION == selection &
    evals_per_gen == subsample_rate
  ) %>%
  mutate(
    sampling_level_label = subsample_rate
  )

full_diag_data <- exploit_ts_data %>%
  filter(
    SELECTION == selection & eval_label == "full" &
        evaluations <= full_eval
  ) %>%
  mutate(
    # Ensure that median line will sit in same facet
    sampling_level_label = subsample_rate
  )

plot <- data %>%
  filter(
    eval_label != "full"
  ) %>%
  ggplot(
    aes(
      x = evaluations,
      y = max_agg_score
    )
  ) +
  stat_summary(
    geom = "line",
    fun = mean,
    aes(
      color = eval_label
    )
  ) +
  stat_summary(
```

```
      geom = "ribbon",
      fun.data = "mean_cl_boot",
      fun.args = list(conf.int = 0.95),
      alpha = 0.2,
      linetype = 0,
      aes(
        color = eval_label,
        fill = eval_label
      )
    ) +
    scale_y_continuous(
      name = "Aggregate score",
      limits = c(0, 10010)
    ) +
    scale_x_continuous(
      name = "Evaluations"
    ) +
    scale_fill_bright(
      labels=c(
        "Down-sampling (DS), no estimation",
        "Down-sampling + Estimation (DS+EST)",
        "Individualized random sampling (IRS)",
        "Ancestor-based sampling (ABS)"
      )
    ) +
    scale_color_bright(
      labels=c(
        "Down-sampling (DS), no estimation",
        "Down-sampling + Estimation (DS+EST)",
        "Individualized random sampling (IRS)",
        "Ancestor-based sampling (ABS)"
      )
    ) +
    theme(
      legend.position = "none"
    ) +
    stat_summary(
      data = full_diag_data,
      geom = "line",
      fun = median,
      linetype = "dashed",
      color = "black"
    )

  return(plot)
}
```

Build plots of final scores (after fixed number of evaluations)

```
plot_final_lex_01 <- build_final_score_manuscript_plot(
  "lexicase",
  "0.01"
)
plot_final_lex_10 <- build_final_score_manuscript_plot(
  "lexicase",
  "0.1"
)
plot_final_tourn_01 <- build_final_score_manuscript_plot(
  "tournament",
  "0.01"
)
plot_final_tourn_10 <- build_final_score_manuscript_plot(
  "tournament",
  "0.1"
)
```

Build time series plots (with evaluations on x-axis)

```
plot_ts_lex_01 <- build_score_over_time_manuscript_plot(
  "lexicase",
  "0.01"
)

plot_ts_lex_10 <- build_score_over_time_manuscript_plot(
  "lexicase",
  "0.1"
)

plot_ts_tourn_01 <- build_score_over_time_manuscript_plot
    (
  "tournament",
  "0.01"
)

plot_ts_tourn_10 <- build_score_over_time_manuscript_plot
    (
  "tournament",
  "0.1"
)
```

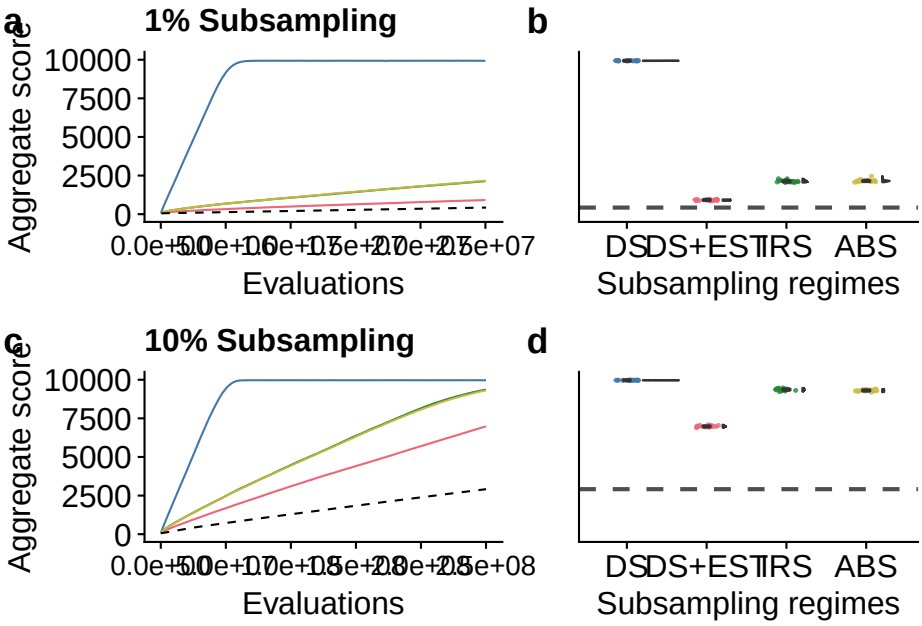
5.5.1 Lexicase selection manuscript figure

```
txt_size <- 16
legend <- get_legend(
  plot_ts_lex_01 +
    guides(
      color = guide_legend(nrow = 2, title = "Subsampling
          regime:"),
      fill = guide_legend(nrow = 2, title = "Subsampling
          regime:")
    ) +
    theme(
      legend.position = "bottom",
      legend.text = element_text(size = txt_size - 2),
      legend.title = element_text(size = txt_size)
    )
)

grid <- plot_grid(
  plot_ts_lex_01 +
    labs(title = "1% Subsampling") +
    theme(
      axis.text.x = element_text(size = txt_size-2),
      axis.text.y = element_text(size = txt_size),
      axis.title.x = element_text(size = txt_size),
      axis.title.y = element_text(size = txt_size)
    ),
  plot_final_lex_01 +
    theme(
      axis.text.y = element_blank(),
      axis.title.y = element_blank(),
      axis.ticks.y = element_blank(),
      plot.margin = margin(0, 0, 0, 1, "cm"),
      axis.text.x = element_text(size = txt_size),
      axis.title.x = element_text(size = txt_size)
    ),
  plot_ts_lex_10 +
    labs(title = "10% Subsampling") +
    theme(
      axis.text.x = element_text(size = txt_size-2),
      axis.text.y = element_text(size = txt_size),
      axis.title.x = element_text(size = txt_size),
      axis.title.y = element_text(size = txt_size)
    ),
  plot_final_lex_10 +
    theme(
      axis.text.y = element_blank(),
      axis.title.y = element_blank(),
```

```
      axis.ticks.y = element_blank(),
      plot.margin = margin(0, 0, 0, 1, "cm"),
      axis.text.x = element_text(size = txt_size),
      axis.title.x = element_text(size = txt_size)
    ),
  nrow = 2,
  ncol = 2,
  align = "h",
  labels = c("a", "b", "c", "d"),
  label_size = 18,
  rel_widths = c(1.3, 1, 1.3, 1)
)

grid
```




```
lex_fig <- plot_grid(
  grid,
  legend,
  nrow = 2,
  ncol = 1,
  rel_heights = c(1, 0.05)
)

# lex_fig
save_plot(
  filename = paste0(plot_directory, "2023-12-28-exploit-
```

```
      lex-fig.pdf"),
  plot = lex_fig,
  base_width = 10,
  base_height = 8,
  dpi = 600
)
```

5.5.2 Tournament selection manuscript figures

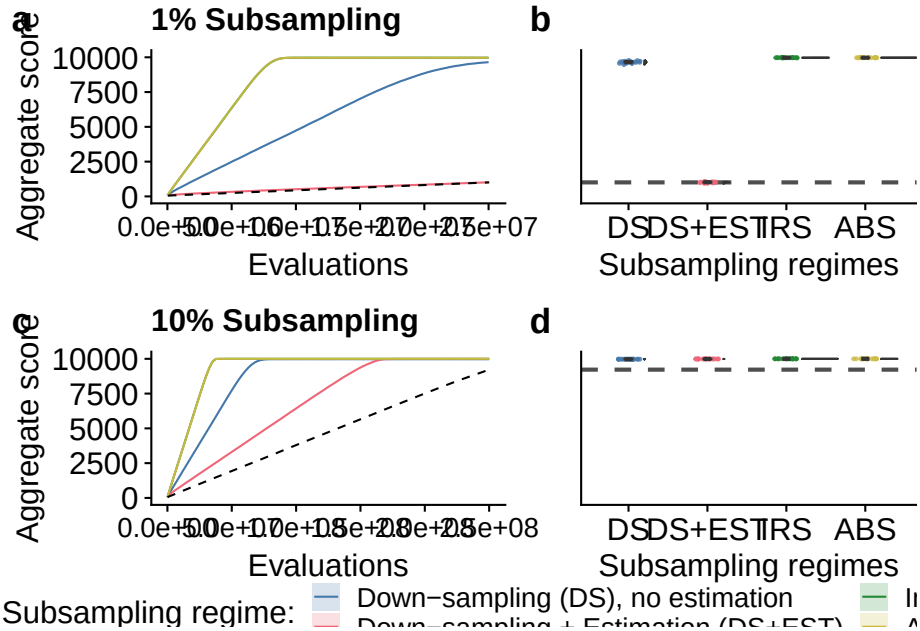
```
legend <- get_legend(
  plot_ts_tourn_01 +
    guides(
      color = guide_legend(nrow = 2, title = "Subsampling
          regime:"),
      fill = guide_legend(nrow = 2, title = "Subsampling
          regime:")
    ) +
    theme(
      legend.position = "bottom",
      legend.text = element_text(size = txt_size - 2),
      legend.title = element_text(size = txt_size)
    )
)

grid <- plot_grid(
  plot_ts_tourn_01 +
    labs(title = "1% Subsampling") +
    theme(
      axis.text.x = element_text(size = txt_size-2),
      axis.text.y = element_text(size = txt_size),
      axis.title.x = element_text(size = txt_size),
      axis.title.y = element_text(size = txt_size)
    ),
  plot_final_tourn_01 +
    theme(
      axis.text.y = element_blank(),
      axis.title.y = element_blank(),
      axis.ticks.y = element_blank(),
      plot.margin = margin(0, 0, 0, 1, "cm"),
      axis.text.x = element_text(size = txt_size),
      axis.title.x = element_text(size = txt_size)
    ),
  plot_ts_tourn_10 +
    labs(title = "10% Subsampling") +
```

```
    theme(
      axis.text.x = element_text(size = txt_size-2),
      axis.text.y = element_text(size = txt_size),
      axis.title.x = element_text(size = txt_size),
      axis.title.y = element_text(size = txt_size)
    ),
  plot_final_tourn_10 +
    theme(
      axis.text.y = element_blank(),
      axis.title.y = element_blank(),
      axis.ticks.y = element_blank(),
      plot.margin = margin(0, 0, 0, 1, "cm"),
      axis.text.x = element_text(size = txt_size),
      axis.title.x = element_text(size = txt_size)
    ),
  nrow = 2,
  ncol = 2,
  align = "h",
  labels = c("a", "b", "c", "d"),
  label_size = 18,
  rel_widths = c(1.3, 1, 1.3, 1)
)

tourn_fig <- plot_grid(
  grid,
  legend,
  nrow = 2,
  ncol = 1,
  rel_heights = c(1, 0.05)
)

tourn_fig
```




```
save_plot(
  filename = paste0(plot_directory, "2023-12-28-exploit-
      tourn-fig.pdf"),
  plot = tourn_fig,
  base_width = 10,
  base_height = 8,
  dpi = 600
)
```

Chapter 6

Contradictory objectives diagnostic

```
experiment_slug <- "2023-12-28-phylo-sampling-diag"

working_directory <- paste0(
  "experiments/",
  experiment_slug,
  "/analysis/"
)

if (exists("bookdown_wd_prefix")) {
  working_directory <- paste0(
    bookdown_wd_prefix,
    working_directory
  )
}
```

6.1 Dependencies

```
library(tidyverse)
library(cowplot)
library(RColorBrewer)
library(khroma)
library(rstatix)
library(knitr)
source("https://gist.githubusercontent.com/benmarwick/2
    a1bb0133ff568cbe28d/raw/
```

```
    fb53bd97121f7f9ce947837ef1a4c65a73bffb3f/geom_flat_
    violin.R")

print(version)

##                _
## platform       aarch64-apple-darwin20
## arch           aarch64
## os             darwin20
## system         aarch64, darwin20
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

6.2 Setup

```
# Configure our default graphing theme
theme_set(theme_cowplot())
# Create a directory to store plots
plot_directory <- paste0(working_directory, "plots/")
dir.create(plot_directory, showWarnings=FALSE)
# Constants
focal_diagnostic <- "contradictory-objectives"
```

6.2.1 Load experiment summary data

```
summary_data_loc <- paste0(working_directory, "data/
    aggregate.csv")
summary_data <- read_csv(summary_data_loc)

## Rows: 1080 Columns: 58
## -- Column specification
    --------------------------------------------------------

## Delimiter: ","
```

```
## chr  (5): DIAGNOSTIC, EVAL_FIT_EST_MODE, EVAL_MODE,
    SELECTION, STOP_MODE
## dbl (53): ACCURACY, CREDIT, DIAGNOSTIC_DIMENSIONALITY,
     EVAL_MAX_PHYLO_SEARCH...
##
## i Use `spec()` to retrieve the full column
    specification for this data.
## i Specify the column types or set `show_col_types =
    FALSE` to quiet this message.

summary_data <- summary_data %>%
  mutate(
    evals_per_gen = case_when(
      EVAL_MODE == "cohort-full-compete" ~ 1.0 / NUM_
          COHORTS,
      EVAL_MODE == "cohort" ~ 1.0 / NUM_COHORTS,
      EVAL_MODE == "down-sample" ~ TEST_DOWNSAMPLE_RATE,
      EVAL_MODE == "full" ~ 1.0,
      EVAL_MODE == "indiv-rand-sample" ~ TEST_DOWNSAMPLE_
          RATE,
      EVAL_MODE == "phylo-informed-sample" ~ TEST_
          DOWNSAMPLE_RATE
    ),
    EVAL_FIT_EST_MODE = case_when(
      EVAL_FIT_EST_MODE == "ancestor-opt" ~ "ancestor",
      EVAL_FIT_EST_MODE == "relative-opt" ~ "relative",
      .default = EVAL_FIT_EST_MODE
    ),
    .keep = "all"
  ) %>%
  mutate(
    eval_label = case_when(
      # Clean up down-sample label
      EVAL_MODE == "down-sample" & EVAL_FIT_EST_MODE != "
          none" ~ paste("down-sample", EVAL_FIT_EST_MODE,
          sep="-"),
      .default = EVAL_MODE
    ),
  ) %>%
  mutate(
    evals_per_gen = as.factor(evals_per_gen),
    DIAGNOSTIC = as.factor(DIAGNOSTIC),
    SELECTION = as.factor(SELECTION),
    EVAL_MODE = as.factor(EVAL_MODE),
    NUM_COHORTS = as.factor(NUM_COHORTS),
    TEST_DOWNSAMPLE_RATE = as.factor(TEST_DOWNSAMPLE_RATE
```

```
      ),
    EVAL_FIT_EST_MODE = factor(
      EVAL_FIT_EST_MODE,
      levels = c(
        "none",
        "ancestor",
        "relative"
      ),
      labels = c(
        "None",
        "Ancestor",
        "Relative"
      )
    )
  )

# Grab just the contradictory objectives data
con_obj_summary_data <- filter(
  summary_data,
  DIAGNOSTIC == "contradictory-objectives"
)
```

6.2.2 Load experiment time series data

```
ts_data_loc <- paste0(working_directory, "data/time_
    series.csv")
ts_data <- read_csv(ts_data_loc)

## Rows: 108000 Columns: 28
## -- Column specification
    --------------------------------------------------------

## Delimiter: ","
## chr  (4): DIAGNOSTIC, EVAL_FIT_EST_MODE, EVAL_MODE,
    SELECTION
## dbl (24): NUM_COHORTS, SEED, TEST_DOWNSAMPLE_RATE,
    ave_depth, deleterious_st...
##
## i Use `spec()` to retrieve the full column
    specification for this data.
## i Specify the column types or set `show_col_types =
    FALSE` to quiet this message.

ts_data <- ts_data %>%
  mutate(
```

```
    evals_per_gen = case_when(
      EVAL_MODE == "cohort-full-compete" ~ 1.0 / NUM_
          COHORTS,
      EVAL_MODE == "cohort" ~ 1.0 / NUM_COHORTS,
      EVAL_MODE == "down-sample" ~ TEST_DOWNSAMPLE_RATE,
      EVAL_MODE == "full" ~ 1.0,
      EVAL_MODE == "indiv-rand-sample" ~ TEST_DOWNSAMPLE_
          RATE,
      EVAL_MODE == "phylo-informed-sample" ~ TEST_
          DOWNSAMPLE_RATE
    ),
    EVAL_FIT_EST_MODE = case_when(
      EVAL_FIT_EST_MODE == "ancestor-opt" ~ "ancestor",
      EVAL_FIT_EST_MODE == "relative-opt" ~ "relative",
      .default = EVAL_FIT_EST_MODE
    ),
    .keep = "all"
  ) %>%
  mutate(
    eval_label = case_when(
      EVAL_MODE == "down-sample" & EVAL_FIT_EST_MODE != "
          none" ~ paste("down-sample", EVAL_FIT_EST_MODE,
          sep="-"),
      .default = EVAL_MODE
    )
  ) %>%
  mutate(
    evals_per_gen = as.factor(evals_per_gen),
    DIAGNOSTIC = as.factor(DIAGNOSTIC),
    SELECTION = as.factor(SELECTION),
    EVAL_MODE = as.factor(EVAL_MODE),
    NUM_COHORTS = as.factor(NUM_COHORTS),
    TEST_DOWNSAMPLE_RATE = as.factor(TEST_DOWNSAMPLE_RATE
        ),
    EVAL_FIT_EST_MODE = factor(
      EVAL_FIT_EST_MODE,
      levels = c(
        "none",
        "ancestor",
        "relative"
      ),
      labels = c(
        "None",
        "Ancestor",
        "Relative"
      )
```

```
    )
  )

# Grab just the contradictory objectives data
con_obj_ts_data <- ts_data %>%
  filter(DIAGNOSTIC == "contradictory-objectives")
```

Summarize time series data:

```
ts_summary_data <- ts_data %>%
  group_by(SEED, DIAGNOSTIC, SELECTION, evals_per_gen,
      eval_label) %>%
  summarize(
    n = n(),
    avg_num_unique_selected = mean(num_unique_selected),
    total_optimal_trait_coverage_loss = sum(optimal_trait
        _coverage_loss)
  )

## `summarise()` has grouped output by 'SEED', '
    DIAGNOSTIC', 'SELECTION',
## 'evals_per_gen'. You can override using the `.groups`
    argument.
```

6.2.3 Plotting helper functions

The following function assist with exploratory plotting of different measurements from summary and time series data. Note that for these plots, standard lexicase reference is rendered at equivalent number of generations (instead of evaluations).

```
build_plot_summary_data <- function(data, diagnostic,
    selection, response) {
  diag_data <- data %>% filter(DIAGNOSTIC == diagnostic)

  full_median <- median(
    filter(
      diag_data,
      eval_label == "full" & SELECTION == selection
    )[[response]]
  )

  plot <- diag_data %>%
    filter(
      eval_label != "full" & SELECTION == selection
    ) %>%
```

```
ggplot(
  aes_string(
    x = "eval_label",
    y = response,
    fill = "eval_label"
  )
) +
geom_hline(
  yintercept = full_median,
  size = 1.0,
  alpha = 0.7,
  color = "black",
  linetype="dashed"
) +
geom_flat_violin(
  position = position_nudge(x = .2, y = 0),
  alpha = .8,
  adjust = 1.5
) +
geom_point(
  mapping = aes(color = eval_label),
  position = position_jitter(width = .15),
  size = .5,
  alpha = 0.8
) +
geom_boxplot(
  width = .1,
  outlier.shape = NA,
  alpha = 0.5
) +
scale_y_continuous(
  # limits = c(-0.5, 100)
) +
scale_fill_bright() +
scale_color_bright() +
facet_grid(
  SELECTION ~ evals_per_gen,
  # nrow=2,
  labeller = label_both
) +
theme(
  legend.position = "none",
  axis.text.x = element_text(
    angle = 30,
    hjust = 1
  ),
```

```
      panel.border = element_rect(color = "gray", size =
          2)
    )

  return(plot)
}

build_plot_time_series_single_sampling <- function(
  data,
  diagnostic,
  selection,
  sampling_level,
  response
) {

  diag_data <- data %>% filter(
    DIAGNOSTIC == diagnostic &
    SELECTION == selection &
    evals_per_gen == sampling_level
  ) %>%
  mutate(
    sampling_level_label = sampling_level
  )

  full_diag_data <- data %>% filter(
    DIAGNOSTIC == diagnostic & SELECTION == selection &
        eval_label == "full"
  ) %>%
  mutate(
    # Ensure that median line will sit in same facet
    sampling_level_label = sampling_level
  )

  plot <- diag_data %>%
    filter(
      eval_label != "full"
    ) %>%
    ggplot(
      aes_string(
        x = "ts_step",
        # x = "evaluations",
        y = {{ response }}
      )
    ) +
    stat_summary(
      geom = "line",
```

```
      fun = mean,
      aes(
        color = eval_label
      )
    ) +
    stat_summary(
      geom = "ribbon",
      fun.data = "mean_cl_boot",
      fun.args = list(conf.int = 0.95),
      alpha = 0.2,
      linetype = 0,
      aes(
        color = eval_label,
        fill = eval_label
      )
    ) +
    scale_fill_bright() +
    scale_color_bright() +
    # facet_wrap(
    #   ~ sampling_level_label,
    #   ncol = 1,
    #   labeller = label_both
    # ) +
    theme(
      legend.position = "right"
    ) +
    stat_summary(
      data = full_diag_data,
      geom = "line",
      fun = median,
      linetype = "dashed",
      color = "black"
    )

  return(plot)
}

build_plot_time_series <- function(
  data,
  diagnostic,
  selection,
  response
) {
  # Build 1% sampling plot and 10% sampling plot
  p_01 <- data  %>% build_plot_time_series_single_
      sampling(
```

```
    diagnostic,
    selection,
    "0.01",
    response
  )
  p_10 <- data %>% build_plot_time_series_single_sampling
      (
    diagnostic,
    selection,
    "0.1",
    response
  )

  title <- ggdraw() +
    draw_label(
      paste0(diagnostic, " - ", selection),
      fontface = 'bold',
      x = 0,
      hjust = 0
    ) +
    theme(
      # add margin on the left of the drawing canvas,
      # so title is aligned with left edge of first plot
      plot.margin = margin(0, 0, 0, 7)
    )

  plot <- plot_grid(
    title,
    p_01 + labs(title = "1% subsampling") + theme(legend.
        position = "none"),
    p_10 + labs(title = "10% subsampling") + theme(legend
        .position = "bottom"),
    nrow = 3,
    ncol = 1,
    rel_heights = c(0.075, 1, 1)
  )

  return(plot)
}
```

6.3 Population-wide satisfactory trait coverage

6.3.1 Final - Lexicase selection

```
p <- summary_data %>% build_plot_summary_data(
  focal_diagnostic,
  "lexicase",
  "pop_optimal_trait_coverage"
)
ggsave(
  filename = paste0(plot_directory, "con-obj-sat-cov-
      final-lex.pdf"),
  plot = p + labs(title = "Contradictory objectives - 
      Lexicase selection"),
  width = 15,
  height = 10
)
```

6.3.1.1 Statistics

First, we'll create a table of median / mean values for easy reference.

```
con_obj_summary_data %>%
  group_by(DIAGNOSTIC, SELECTION, evals_per_gen, eval_
      label) %>%
  summarize(
    cov_median = median(pop_optimal_trait_coverage),
    cov_mean = mean(pop_optimal_trait_coverage),
    replicates = n()
  ) %>%
  kable()
```

```
## `summarise()` has grouped output by 'DIAGNOSTIC', '
    SELECTION', 'evals_per_gen'.
## You can override using the `.groups` argument.
```

DIAGNOSTIC	SELECTION	evals_per_gen	eval_label	cov_median	c
contradictory-objectives	lexicase	0.01	down-sample	1.0	
contradictory-objectives	lexicase	0.01	down-sample-ancestor	2.5	
contradictory-objectives	lexicase	0.01	indiv-rand-sample	8.0	
contradictory-objectives	lexicase	0.01	phylo-informed-sample	8.5	
contradictory-objectives	lexicase	0.1	down-sample	1.0	
contradictory-objectives	lexicase	0.1	down-sample-ancestor	17.5	
contradictory-objectives	lexicase	0.1	indiv-rand-sample	24.5	
contradictory-objectives	lexicase	0.1	phylo-informed-sample	24.0	
contradictory-objectives	lexicase	1	full	38.0	
contradictory-objectives	tournament	0.01	down-sample	1.0	
contradictory-objectives	tournament	0.01	down-sample-ancestor	1.0	
contradictory-objectives	tournament	0.01	indiv-rand-sample	1.0	
contradictory-objectives	tournament	0.01	phylo-informed-sample	1.0	
contradictory-objectives	tournament	0.1	down-sample	1.0	
contradictory-objectives	tournament	0.1	down-sample-ancestor	1.0	
contradictory-objectives	tournament	0.1	indiv-rand-sample	1.0	
contradictory-objectives	tournament	0.1	phylo-informed-sample	1.0	
contradictory-objectives	tournament	1	full	1.0	

Next, we run a Kruskal-Wallis test to check for differences. For these tests, we only compare within a single subsampling level (evals_per_gen) and within the same selection scheme.

```
kw_test <- con_obj_summary_data %>%
  filter(eval_label != "full") %>%
  group_by(SELECTION, evals_per_gen) %>%
  kruskal_test(pop_optimal_trait_coverage ~ eval_label)
    %>%
  mutate(sig = (p < 0.05)) %>%
  unite(
    "comparison_group",
    SELECTION,
    evals_per_gen,
    sep = "_",
    remove = FALSE
  )
kable(kw_test)
```

comparison_group	SELECTION	evals_per_gen	.y.	n	statistic
lexicase_0.01	lexicase	0.01	pop_optimal_trait_coverage	80	58.24682
lexicase_0.1	lexicase	0.1	pop_optimal_trait_coverage	80	62.11510
tournament_0.01	tournament	0.01	pop_optimal_trait_coverage	80	NaN
tournament_0.1	tournament	0.1	pop_optimal_trait_coverage	80	NaN

```
# Grab group names of significant comparisons
```

```
sig_kw_groups <- filter(kw_test, p < 0.05)$comparison_
    group

wrs_test <- con_obj_summary_data %>%
  unite(
    "comparison_group",
    SELECTION,
    evals_per_gen,
    sep = "_",
    remove = FALSE
  ) %>%
  filter(
    eval_label != "full" & comparison_group %in% sig_kw_
        groups
  ) %>%
  group_by(SELECTION, evals_per_gen) %>%
  pairwise_wilcox_test(pop_optimal_trait_coverage ~ eval_
      label) %>%
  adjust_pvalue(method = "holm") %>%
  add_significance("p.adj")

kable(wrs_test)
```

SELECTION	evals_per_gen	.y.	group1	group2
lexicase	0.01	pop_optimal_trait_coverage	down-sample	down-sample-ancestor
lexicase	0.01	pop_optimal_trait_coverage	down-sample	indiv-rand-sample
lexicase	0.01	pop_optimal_trait_coverage	down-sample	phylo-informed-sample
lexicase	0.01	pop_optimal_trait_coverage	down-sample-ancestor	indiv-rand-sample
lexicase	0.01	pop_optimal_trait_coverage	down-sample-ancestor	phylo-informed-sample
lexicase	0.01	pop_optimal_trait_coverage	indiv-rand-sample	phylo-informed-sample
lexicase	0.1	pop_optimal_trait_coverage	down-sample	down-sample-ancestor
lexicase	0.1	pop_optimal_trait_coverage	down-sample	indiv-rand-sample
lexicase	0.1	pop_optimal_trait_coverage	down-sample	phylo-informed-sample
lexicase	0.1	pop_optimal_trait_coverage	down-sample-ancestor	indiv-rand-sample
lexicase	0.1	pop_optimal_trait_coverage	down-sample-ancestor	phylo-informed-sample
lexicase	0.1	pop_optimal_trait_coverage	indiv-rand-sample	phylo-informed-sample

6.3.2 Over time - lexicase selection

```
p <- ts_data  %>% build_plot_time_series(
  focal_diagnostic,
  "lexicase",
  "pop_optimal_trait_coverage"
)
ggsave(
```

```
  filename = paste0(plot_directory, "con-obj-sat-cov-ts-
      lex.pdf"),
  plot = p,
  width = 15,
  height = 10
)
```

6.3.3 Final - Tournament selection

```
p <- summary_data %>% build_plot_summary_data(
  focal_diagnostic,
  "tournament",
  "pop_optimal_trait_coverage"
)
ggsave(
  filename = paste0(plot_directory, "con-obj-sat-cov-
      final-tourn.pdf"),
  plot = p + labs(title = "Contradictory objectives - 
      Tournament selection"),
  width = 15,
  height = 10
)
```

6.3.4 Over time - tournament selection

```
p <- ts_data  %>% build_plot_time_series(
  focal_diagnostic,
  "tournament",
  "pop_optimal_trait_coverage"
)
ggsave(
  filename = paste0(plot_directory, "con-obj-sat-cov-ts-
      tourn.pdf"),
  plot = p,
  width = 15,
  height = 10
)
```

6.4 MRCA changes

```
p <- summary_data %>% build_plot_summary_data(
  focal_diagnostic,
  "lexicase",
  "phylo_mrca_changes"
)
ggsave(
  filename = paste0(plot_directory, "con-obj-mrca-chgs-
      final-lex.pdf"),
  plot = p + labs(title = "Contradictory objectives - 
      Lexicase selection"),
  width = 15,
  height = 10
)
```

6.5 Mean genotype deleterious steps

```
p <- summary_data %>% build_plot_summary_data(
  focal_diagnostic,
  "lexicase",
  "phylo_mean_genoetype_deleterious_steps"
)
ggsave(
  filename = paste0(plot_directory, "con-obj-del-steps-
      final-lex.pdf"),
  plot = p + labs(title = "Contradictory objectives - 
      Lexicase selection"),
  width = 15,
  height = 10
)
```

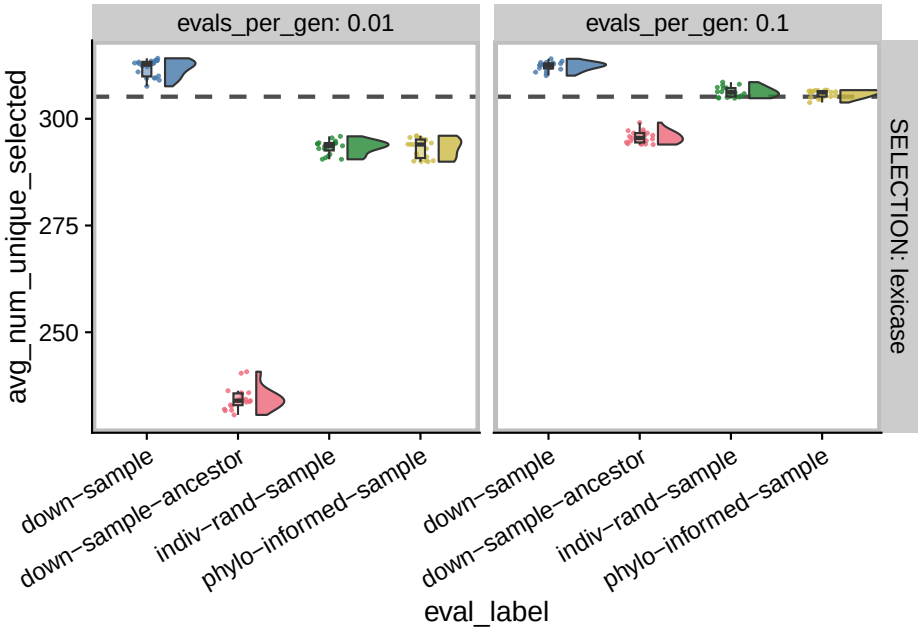
6.6 Mean genotype pairwise distance

```
p <- summary_data %>% build_plot_summary_data(
  focal_diagnostic,
  "lexicase",
  "phylo_mean_genotype_pairwise_distance"
)
ggsave(
  filename = paste0(plot_directory, "con-obj-pw-dist-
      final-lex.pdf"),
  plot = p + labs(title = "Contradictory objectives - 
      Lexicase selection"),
```

```
  width = 15,
  height = 10
)
```

6.7 Number unique individual selected

```
build_plot_summary_data(
  ts_summary_data,
  focal_diagnostic,
  "lexicase",
  "avg_num_unique_selected"
)
```



6.8 Manuscript figures

Time series graphs don't add a ton here, so just final graphs.

```
build_final_score_manuscript_plot <- function(
  selection,
  subsample_rate
) {
```

```
# Extract median values for max aggregate score at same
    evaluation level as sampling regimes
max_eval <- max(
  filter(con_obj_summary_data, evals_per_gen ==
      subsample_rate)$evaluations
)
full_eval_steps <- as.numeric(
  levels(
    as.factor(
      filter(con_obj_summary_data, eval_label == "full"
          & evaluations >= max_eval)$evaluations #
          nolint: line_length_linter.
    )
  )
)
full_eval <- full_eval_steps[which.min( full_eval_steps
    - max_eval )]
full_median_score_evals <- median(
  filter(
    con_obj_summary_data,
    SELECTION == selection & eval_label == "full" &
        evaluations == full_eval
  )$pop_optimal_trait_coverage
)

plot <- con_obj_summary_data %>%
  filter(
    eval_label != "full" &
    SELECTION == selection &
    evals_per_gen == subsample_rate
  ) %>%
  ggplot(
    aes(
      x = eval_label,
      y = pop_optimal_trait_coverage,
      fill = eval_label
    )
  ) +
  geom_hline(
    yintercept = full_median_score_evals,
    size = 1.0,
    alpha = 0.7,
    color = "black",
    linetype="dashed"
  ) +
  geom_flat_violin(
```

```
      position = position_nudge(x = .2, y = 0),
      alpha = .8,
      adjust = 1.5
    ) +
    geom_point(
      mapping = aes(color = eval_label),
      position = position_jitter(width = .15),
      size = .5,
      alpha = 0.8
    ) +
    geom_boxplot(
      width = .1,
      outlier.shape = NA,
      alpha = 0.5
    ) +
    scale_y_continuous(
      name = "Satisfactory trait coverage",
      limits = c(0, 100)
    ) +
    scale_x_discrete(
      name = "Subsampling regime",
      breaks = c("down-sample", "down-sample-ancestor", "
          indiv-rand-sample", "phylo-informed-sample"),
      labels = c("DS\n(no est.)", "DS+EST", "IRS", "ABS")
    ) +
    scale_fill_bright() +
    scale_color_bright() +
    theme(
      legend.position = "none",
      # axis.text.x = element_text(
      #   angle = 30,
      #   hjust = 1
      # ),
    )
  return(plot)
}
```

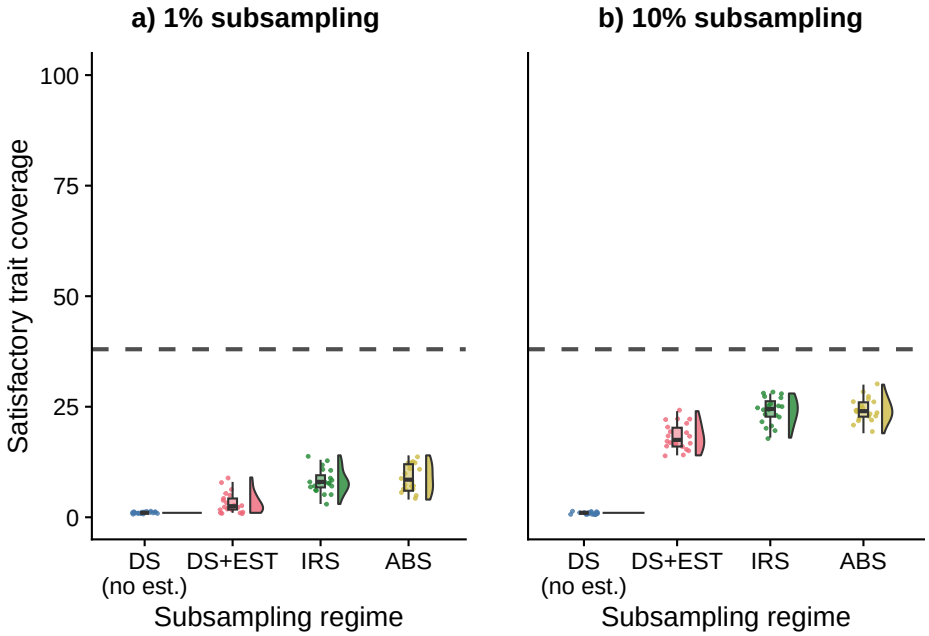
Build end-of-run plots (fixed number of evaluations)

```
plot_final_lex_01 <- build_final_score_manuscript_plot(
  "lexicase",
  "0.01"
)
plot_final_lex_10 <- build_final_score_manuscript_plot(
  "lexicase",
  "0.1"
```

)

Combine into single figure

```
lex_fig <- plot_grid(
  plot_final_lex_01 +
    # labs(
    #   title = "1% subsampling"
    # ) +
    theme(
      plot.margin = margin(1, 0, 0, 0, "cm")
    ),
  plot_final_lex_10 +
    # labs(
    #   title = "10% subsampling"
    # ) +
    theme(
      axis.text.y = element_blank(),
      axis.title.y = element_blank(),
      axis.ticks.y = element_blank(),
      plot.margin = margin(1, 0, 0, 1, "cm")
    ),
  nrow = 1,
  ncol = 2,
  align = "h",
  labels = c("a) 1% subsampling", "b) 10% subsampling"),
  rel_widths = c(1, 1)
)
lex_fig
```




```
save_plot(
  filename = paste0(plot_directory, "2023-12-28-con-obj-
      lex-fig.pdf"),
  plot = lex_fig,
  base_width = 7,
  base_height = 4,
  dpi = 600
)
```

Chapter 7

Multi-path exploration diagnostic

```
experiment_slug <- "2023-12-28-phylo-sampling-diag"

working_directory <- paste0(
  "experiments/",
  experiment_slug,
  "/analysis/"
)

if (exists("bookdown_wd_prefix")) {
  working_directory <- paste0(
    bookdown_wd_prefix,
    working_directory
  )
}
```

7.1 Dependencies

```
library(tidyverse)
library(cowplot)
library(RColorBrewer)
library(khroma)
library(rstatix)
library(knitr)
source("https://gist.githubusercontent.com/benmarwick/2
    a1bb0133ff568cbe28d/raw/
```

```
    fb53bd97121f7f9ce947837ef1a4c65a73bffb3f/geom_flat_
    violin.R")

print(version)

##                _
## platform       aarch64-apple-darwin20
## arch           aarch64
## os             darwin20
## system         aarch64, darwin20
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

7.2 Setup

```
# Configure our default graphing theme
theme_set(theme_cowplot())
# Create a directory to store plots
plot_directory <- paste0(working_directory, "plots/")
dir.create(plot_directory, showWarnings=FALSE)
# Constants
focal_diagnostic <- "multipath-exploration"
```

7.2.1 Load experiment summary data

```
summary_data_loc <- paste0(working_directory, "data/
    aggregate.csv")
summary_data <- read_csv(summary_data_loc)

## Rows: 1080 Columns: 58
## -- Column specification
    --------------------------------------------------------

## Delimiter: ","
```

```
## chr  (5): DIAGNOSTIC, EVAL_FIT_EST_MODE, EVAL_MODE,
    SELECTION, STOP_MODE
## dbl (53): ACCURACY, CREDIT, DIAGNOSTIC_DIMENSIONALITY,
     EVAL_MAX_PHYLO_SEARCH...
##
## i Use `spec()` to retrieve the full column
    specification for this data.
## i Specify the column types or set `show_col_types =
    FALSE` to quiet this message.

summary_data <- summary_data %>%
  mutate(
    evals_per_gen = case_when(
      EVAL_MODE == "cohort-full-compete" ~ 1.0 / NUM_
          COHORTS,
      EVAL_MODE == "cohort" ~ 1.0 / NUM_COHORTS,
      EVAL_MODE == "down-sample" ~ TEST_DOWNSAMPLE_RATE,
      EVAL_MODE == "full" ~ 1.0,
      EVAL_MODE == "indiv-rand-sample" ~ TEST_DOWNSAMPLE_
          RATE,
      EVAL_MODE == "phylo-informed-sample" ~ TEST_
          DOWNSAMPLE_RATE
    ),
    EVAL_FIT_EST_MODE = case_when(
      EVAL_FIT_EST_MODE == "ancestor-opt" ~ "ancestor",
      EVAL_FIT_EST_MODE == "relative-opt" ~ "relative",
      .default = EVAL_FIT_EST_MODE
    ),
    .keep = "all"
  ) %>%
  mutate(
    eval_label = case_when(
      # Clean up down-sample label
      EVAL_MODE == "down-sample" & EVAL_FIT_EST_MODE != "
          none" ~ paste("down-sample", EVAL_FIT_EST_MODE,
          sep="-"),
      .default = EVAL_MODE
    ),
  ) %>%
  mutate(
    evals_per_gen = as.factor(evals_per_gen),
    DIAGNOSTIC = as.factor(DIAGNOSTIC),
    SELECTION = as.factor(SELECTION),
    EVAL_MODE = as.factor(EVAL_MODE),
    NUM_COHORTS = as.factor(NUM_COHORTS),
    TEST_DOWNSAMPLE_RATE = as.factor(TEST_DOWNSAMPLE_RATE
```

```
      ),
    EVAL_FIT_EST_MODE = factor(
      EVAL_FIT_EST_MODE,
      levels = c(
        "none",
        "ancestor",
        "relative"
      ),
      labels = c(
        "None",
        "Ancestor",
        "Relative"
      )
    )
  )

explore_summary_data <- filter(
  summary_data,
  DIAGNOSTIC == "multipath-exploration"
)
```

7.2.2 Load experiment time series data

```
ts_data_loc <- paste0(working_directory, "data/time_
    series.csv")
ts_data <- read_csv(ts_data_loc)

## Rows: 108000 Columns: 28
## -- Column specification
    --------------------------------------------------------

## Delimiter: ","
## chr  (4): DIAGNOSTIC, EVAL_FIT_EST_MODE, EVAL_MODE,
    SELECTION
## dbl (24): NUM_COHORTS, SEED, TEST_DOWNSAMPLE_RATE,
    ave_depth, deleterious_st...
##
## i Use `spec()` to retrieve the full column
    specification for this data.
## i Specify the column types or set `show_col_types =
    FALSE` to quiet this message.

ts_data <- ts_data %>%
  mutate(
    evals_per_gen = case_when(
```

```
      EVAL_MODE == "cohort-full-compete" ~ 1.0 / NUM_
          COHORTS,
      EVAL_MODE == "cohort" ~ 1.0 / NUM_COHORTS,
      EVAL_MODE == "down-sample" ~ TEST_DOWNSAMPLE_RATE,
      EVAL_MODE == "full" ~ 1.0,
      EVAL_MODE == "indiv-rand-sample" ~ TEST_DOWNSAMPLE_
          RATE,
      EVAL_MODE == "phylo-informed-sample" ~ TEST_
          DOWNSAMPLE_RATE
    ),
    EVAL_FIT_EST_MODE = case_when(
      EVAL_FIT_EST_MODE == "ancestor-opt" ~ "ancestor",
      EVAL_FIT_EST_MODE == "relative-opt" ~ "relative",
      .default = EVAL_FIT_EST_MODE
    ),
    .keep = "all"
  ) %>%
  mutate(
    eval_label = case_when(
      EVAL_MODE == "down-sample" & EVAL_FIT_EST_MODE != "
          none" ~ paste("down-sample", EVAL_FIT_EST_MODE,
          sep="-"),
      .default = EVAL_MODE
    )
  ) %>%
  mutate(
    evals_per_gen = as.factor(evals_per_gen),
    DIAGNOSTIC = as.factor(DIAGNOSTIC),
    SELECTION = as.factor(SELECTION),
    EVAL_MODE = as.factor(EVAL_MODE),
    NUM_COHORTS = as.factor(NUM_COHORTS),
    TEST_DOWNSAMPLE_RATE = as.factor(TEST_DOWNSAMPLE_RATE
        ),
    EVAL_FIT_EST_MODE = factor(
      EVAL_FIT_EST_MODE,
      levels = c(
        "none",
        "ancestor",
        "relative"
      ),
      labels = c(
        "None",
        "Ancestor",
        "Relative"
      )
    )
```

```
  )

explore_ts_data <- ts_data %>%
  filter(DIAGNOSTIC == "multipath-exploration")
```

Summarize time series data

```
ts_summary_data <- ts_data %>%
  group_by(SEED, DIAGNOSTIC, SELECTION, evals_per_gen,
      eval_label) %>%
  summarize(
    n = n(),
    avg_num_unique_selected = mean(num_unique_selected),
    total_optimal_trait_coverage_loss = sum(optimal_trait
        _coverage_loss)
  )

## `summarise()` has grouped output by 'SEED', '
    DIAGNOSTIC', 'SELECTION',
## 'evals_per_gen'. You can override using the `.groups`
    argument.
```

7.2.3 Plotting helper functions

The following function assist with exploratory plotting of different measurements from summary and time series data. Note that for these plots, standard lexicase reference is rendered at equivalent number of generations (instead of evaluations).

```
build_plot_summary_data <- function(data, diagnostic,
    selection, response) {
  diag_data <- data %>% filter(DIAGNOSTIC == diagnostic)

  full_median <- median(
    filter(
      diag_data,
      eval_label == "full" & SELECTION == selection
    )[[response]]
  )

  plot <- diag_data %>%
    filter(
      eval_label != "full" & SELECTION == selection
    ) %>%
    ggplot(
      aes_string(
```

```
      x = "eval_label",
      y = response,
      fill = "eval_label"
    )
  ) +
  geom_hline(
    yintercept = full_median,
    size = 1.0,
    alpha = 0.7,
    color = "black",
    linetype="dashed"
  ) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8,
    adjust = 1.5
  ) +
  geom_point(
    mapping = aes(color = eval_label),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_y_continuous(
    # limits = c(-0.5, 100)
  ) +
  scale_fill_bright() +
  scale_color_bright() +
  facet_grid(
    SELECTION ~ evals_per_gen,
    # nrow=2,
    labeller = label_both
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(
      angle = 30,
      hjust = 1
    ),
    panel.border = element_rect(color = "gray", size =
      2)
```

```
    )

  return(plot)
}

build_plot_time_series_single_sampling <- function(
  data,
  diagnostic,
  selection,
  sampling_level,
  response
) {

  diag_data <- data %>% filter(
    DIAGNOSTIC == diagnostic &
    SELECTION == selection &
    evals_per_gen == sampling_level
  ) %>%
  mutate(
    sampling_level_label = sampling_level
  )

  full_diag_data <- data %>% filter(
    DIAGNOSTIC == diagnostic & SELECTION == selection &
        eval_label == "full"
  ) %>%
  mutate(
    # Ensure that median line will sit in same facet
    sampling_level_label = sampling_level
  )

  plot <- diag_data %>%
    filter(
      eval_label != "full"
    ) %>%
    ggplot(
      aes_string(
        x = "ts_step",
        # x = "evaluations",
        y = {{ response }}
      )
    ) +
    stat_summary(
      geom = "line",
      fun = mean,
      aes(
```

```
        color = eval_label
      )
    ) +
    stat_summary(
      geom = "ribbon",
      fun.data = "mean_cl_boot",
      fun.args = list(conf.int = 0.95),
      alpha = 0.2,
      linetype = 0,
      aes(
        color = eval_label,
        fill = eval_label
      )
    ) +
    scale_fill_bright() +
    scale_color_bright() +
    # facet_wrap(
    #   ~ sampling_level_label,
    #   ncol = 1,
    #   labeller = label_both
    # ) +
    theme(
      legend.position = "right"
    ) +
    stat_summary(
      data = full_diag_data,
      geom = "line",
      fun = median,
      linetype = "dashed",
      color = "black"
    )

  return(plot)
}

build_plot_time_series <- function(
  data,
  diagnostic,
  selection,
  response
) {
  # Build 1% sampling plot and 10% sampling plot
  p_01 <- data  %>% build_plot_time_series_single_
      sampling(
    diagnostic,
    selection,
```

```
    "0.01",
    response
  )
  p_10 <- data %>% build_plot_time_series_single_sampling
      (
    diagnostic,
    selection,
    "0.1",
    response
  )

  title <- ggdraw() +
    draw_label(
      paste0(diagnostic, " - ", selection),
      fontface = 'bold',
      x = 0,
      hjust = 0
    ) +
    theme(
      # add margin on the left of the drawing canvas,
      # so title is aligned with left edge of first plot
      plot.margin = margin(0, 0, 0, 7)
    )

  plot <- plot_grid(
    title,
    p_01 + labs(title = "1% subsampling") + theme(legend.
        position = "none"),
    p_10 + labs(title = "10% subsampling") + theme(legend
        .position = "bottom"),
    nrow = 3,
    ncol = 1,
    rel_heights = c(0.075, 1, 1)
  )

  return(plot)
}
```

7.3 Aggregate score

7.3.1 Final - Lexicase selection

Note that lexicase baseline is shown @ 50,000 generations (not same number of evaluations).

```
p <- summary_data %>% build_plot_summary_data(
  "multipath-exploration",
  "lexicase",
  "elite_true_agg_score"
)
ggsave(
  filename = paste0(plot_directory, "explore-score-final-
      lex.pdf"),
  plot = p + labs(title = "Exploration rate - Lexicase
      selection"),
  width = 15,
  height = 10
)
```

7.3.1.1 Statistics

First, we'll create a table of median / mean values for easy reference.

```
explore_summary_data %>%
  group_by(DIAGNOSTIC, SELECTION, evals_per_gen, eval_
      label) %>%
  summarize(
    score_median = median(elite_true_agg_score),
    score_mean = mean(elite_true_agg_score),
    replicates = n()
  ) %>%
  kable()
```

```
## `summarise()` has grouped output by 'DIAGNOSTIC', '
    SELECTION', 'evals_per_gen'.
## You can override using the `.groups` argument.
```

DIAGNOSTIC	SELECTION	evals_per_gen	eval_label	score_median	sc
multipath-exploration	lexicase	0.01	down-sample	481.7515	
multipath-exploration	lexicase	0.01	down-sample-ancestor	343.1445	
multipath-exploration	lexicase	0.01	indiv-rand-sample	1321.9050	
multipath-exploration	lexicase	0.01	phylo-informed-sample	1259.1250	
multipath-exploration	lexicase	0.1	down-sample	588.2735	
multipath-exploration	lexicase	0.1	down-sample-ancestor	2532.2150	
multipath-exploration	lexicase	0.1	indiv-rand-sample	2295.0250	
multipath-exploration	lexicase	0.1	phylo-informed-sample	2579.3150	
multipath-exploration	lexicase	1	full	9082.8750	
multipath-exploration	tournament	0.01	down-sample	656.5385	
multipath-exploration	tournament	0.01	down-sample-ancestor	547.7415	
multipath-exploration	tournament	0.01	indiv-rand-sample	3524.0450	
multipath-exploration	tournament	0.01	phylo-informed-sample	2894.6900	
multipath-exploration	tournament	0.1	down-sample	2349.5100	
multipath-exploration	tournament	0.1	down-sample-ancestor	3789.5600	
multipath-exploration	tournament	0.1	indiv-rand-sample	5149.1700	
multipath-exploration	tournament	0.1	phylo-informed-sample	5449.0750	
multipath-exploration	tournament	1	full	4649.8450	

Next, we run a Kruskal-Wallis test to check for differences. For these tests, we only compare within a single subsampling level (evals_per_gen) and within the same selection scheme.

```
kw_test <- explore_summary_data %>%
  filter(eval_label != "full") %>%
  group_by(SELECTION, evals_per_gen) %>%
  kruskal_test(elite_true_agg_score ~ eval_label) %>%
  mutate(sig = (p < 0.05)) %>%
  unite(
    "comparison_group",
    SELECTION,
    evals_per_gen,
    sep = "_",
    remove = FALSE
  )
kable(kw_test)
```

comparison_group	SELECTION	evals_per_gen	.y.	n	statistic	df	
lexicase_0.01	lexicase	0.01	elite_true_agg_score	80	63.64833	3	0
lexicase_0.1	lexicase	0.1	elite_true_agg_score	80	48.94519	3	0
tournament_0.01	tournament	0.01	elite_true_agg_score	80	30.85796	3	
tournament_0.1	tournament	0.1	elite_true_agg_score	80	10.82091	3	

```
# Grab group names of significant comparisons
sig_kw_groups <- filter(kw_test, p < 0.05)$comparison_
    group
```

```
wrs_test <- explore_summary_data %>%
  unite(
    "comparison_group",
    SELECTION,
    evals_per_gen,
    sep = "_",
    remove = FALSE
  ) %>%
  filter(
    eval_label != "full" & comparison_group %in% sig_kw_
        groups
  ) %>%
  group_by(SELECTION, evals_per_gen) %>%
  pairwise_wilcox_test(elite_true_agg_score ~ eval_label)
      %>%
  adjust_pvalue(method = "holm") %>%
  add_significance("p.adj")

kable(wrs_test)
```

SELECTION	evals_per_gen	.y.	group1	group2
lexicase	0.01	elite_true_agg_score	down-sample	down-sample-ances
lexicase	0.01	elite_true_agg_score	down-sample	indiv-rand-sample
lexicase	0.01	elite_true_agg_score	down-sample	phylo-informed-san
lexicase	0.01	elite_true_agg_score	down-sample-ancestor	indiv-rand-sample
lexicase	0.01	elite_true_agg_score	down-sample-ancestor	phylo-informed-san
lexicase	0.01	elite_true_agg_score	indiv-rand-sample	phylo-informed-san
lexicase	0.1	elite_true_agg_score	down-sample	down-sample-ances
lexicase	0.1	elite_true_agg_score	down-sample	indiv-rand-sample
lexicase	0.1	elite_true_agg_score	down-sample	phylo-informed-san
lexicase	0.1	elite_true_agg_score	down-sample-ancestor	indiv-rand-sample
lexicase	0.1	elite_true_agg_score	down-sample-ancestor	phylo-informed-san
lexicase	0.1	elite_true_agg_score	indiv-rand-sample	phylo-informed-san
tournament	0.01	elite_true_agg_score	down-sample	down-sample-ances
tournament	0.01	elite_true_agg_score	down-sample	indiv-rand-sample
tournament	0.01	elite_true_agg_score	down-sample	phylo-informed-san
tournament	0.01	elite_true_agg_score	down-sample-ancestor	indiv-rand-sample
tournament	0.01	elite_true_agg_score	down-sample-ancestor	phylo-informed-san
tournament	0.01	elite_true_agg_score	indiv-rand-sample	phylo-informed-san
tournament	0.1	elite_true_agg_score	down-sample	down-sample-ances
tournament	0.1	elite_true_agg_score	down-sample	indiv-rand-sample
tournament	0.1	elite_true_agg_score	down-sample	phylo-informed-san
tournament	0.1	elite_true_agg_score	down-sample-ancestor	indiv-rand-sample
tournament	0.1	elite_true_agg_score	down-sample-ancestor	phylo-informed-san
tournament	0.1	elite_true_agg_score	indiv-rand-sample	phylo-informed-san

7.3.2 Over time - Lexicase selection

```
p <- ts_data %>% build_plot_time_series(
  "multipath-exploration",
  "lexicase",
  "max_agg_score"
)
ggsave(
  filename = paste0(plot_directory, "explore-score-ts-lex
      .pdf"),
  plot = p,
  width = 15,
  height = 10
)
```

7.4 Manuscript figures

Figures customized / cleaned up for the manuscript.

```
build_final_score_manuscript_plot <- function(
  selection,
  subsample_rate
) {

  # Extract median values for max aggregate score at same
      evaluation level as sampling regimes
  max_eval <- max(
    filter(explore_ts_data, evals_per_gen == subsample_
        rate)$evaluations
  )
  full_eval_steps <- as.numeric(
    levels(
      as.factor(
        filter(explore_ts_data, eval_label == "full" &
            evaluations >= max_eval)$evaluations # nolint:
             line_length_linter.
      )
    )
  )
  full_eval <- full_eval_steps[which.min( full_eval_steps
      - max_eval )]
  full_median_score_evals <- median(
    filter(
      explore_ts_data,
      SELECTION == selection & eval_label == "full" &
          evaluations == full_eval
    )$max_agg_score
  )

  plot <- explore_summary_data %>%
    filter(
      eval_label != "full" &
      SELECTION == selection &
      evals_per_gen == subsample_rate
    ) %>%
    ggplot(
      aes(
        x = eval_label,
        y = elite_true_agg_score,
        fill = eval_label
      )
```

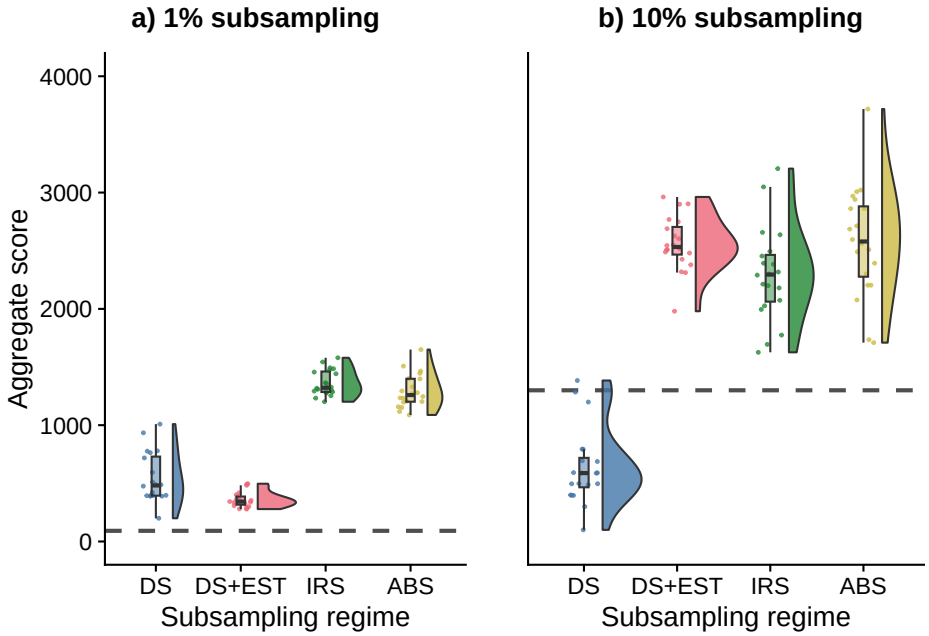
```
    ) +
    geom_hline(
      yintercept = full_median_score_evals,
      size = 1.0,
      alpha = 0.7,
      color = "black",
      linetype="dashed"
    ) +
    geom_flat_violin(
      position = position_nudge(x = .2, y = 0),
      alpha = .8,
      adjust = 1.5
    ) +
    geom_point(
      mapping = aes(color = eval_label),
      position = position_jitter(width = .15),
      size = .5,
      alpha = 0.8
    ) +
    geom_boxplot(
      width = .1,
      outlier.shape = NA,
      alpha = 0.5
    ) +
    scale_y_continuous(
      name = "Aggregate␣score",
      limits = c(0, 4000)
    ) +
    scale_x_discrete(
      name = "Subsampling␣regime",
      breaks = c("down-sample", "down-sample-ancestor", "
          indiv-rand-sample", "phylo-informed-sample"),
      labels = c("DS", "DS+EST", "IRS", "ABS")
    ) +
    scale_fill_bright() +
    scale_color_bright() +
    theme(
      legend.position = "none",
      # axis.text.x = element_text(
      #   angle = 30,
      #   hjust = 1
      # ),
    )
  return(plot)
}
```

Build end-of-run plots (fixed number of evaluations)

```
plot_final_lex_01 <- build_final_score_manuscript_plot(
  "lexicase",
  "0.01"
)
plot_final_lex_10 <- build_final_score_manuscript_plot(
  "lexicase",
  "0.1"
)
```

Combine into single figure

```
lex_fig <- plot_grid(
  plot_final_lex_01 +
    theme(
      plot.margin = margin(1, 0, 0, 0, "cm")
    ),
  plot_final_lex_10 +
    theme(
      axis.text.y = element_blank(),
      axis.title.y = element_blank(),
      axis.ticks.y = element_blank(),
      plot.margin = margin(1, 0, 0, 1, "cm")
    ),
  nrow = 1,
  ncol = 2,
  align = "h",
  labels = c("a) 1% subsampling", "b) 10% subsampling"),
  rel_widths = c(1, 1)
)
lex_fig
```




```
save_plot(
  filename = paste0(plot_directory, "2023-12-28-explore-
      lex-fig.pdf"),
  plot = lex_fig,
  base_width = 7,
  base_height = 3,
  dpi = 600
)
```

Chapter 8

Program synthesis experiments

```
experiment_slug <- "2023-12-30-psynth"

working_directory <- paste0(
  "experiments/",
  experiment_slug,
  "/analysis/"
)

if (exists("bookdown_wd_prefix")) {
  working_directory <- paste0(
    bookdown_wd_prefix,
    working_directory
  )
}
```

8.1 Dependencies

```
library(tidyverse)
library(ggplot2)
library(cowplot)
library(RColorBrewer)
library(khroma)
library(rstatix)
library(knitr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows

source("https://gist.githubusercontent.com/benmarwick/2
    a1bb0133ff568cbe28d/raw/
    fb53bd97121f7f9ce947837ef1a4c65a73bffb3f/geom_flat_
    violin.R")

print(version)

##                _
## platform       aarch64-apple-darwin20
## arch           aarch64
## os             darwin20
## system         aarch64, darwin20
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

8.2 Setup

```
# Configure our default graphing theme
theme_set(theme_cowplot())
# Create a directory to store plots
plot_directory <- paste0(working_directory, "plots/")
dir.create(plot_directory, showWarnings=FALSE)
```

8.2.1 Load summary data

```
summary_data_loc <- paste0(working_directory, "data/
    aggregate.csv")
summary_data <- read_csv(summary_data_loc)
```

```
## Rows: 5000 Columns: 73
## -- Column specification
--------------------------------------------------------------------------------

## Delimiter: ","
## chr (11): ANCESTOR_FILE_PATH, EVAL_FIT_EST_MODE,
    EVAL_MODE, POP_INIT_MODE, P...
## dbl (62): EVAL_CPU_CYCLES_PER_TEST,
    EVAL_MAX_PHYLO_SEARCH_DEPTH, MAX_ACTIVE_...
##
## i Use `spec()` to retrieve the full column
    specification for this data.
## i Specify the column types or set `show_col_types =
    FALSE` to quiet this message.

summary_data <- summary_data %>%
  mutate(
    eval_mode_row = case_when(
      EVAL_MODE == "full" & TEST_DOWNSAMPLE_RATE == "1" ~
          "down-sample",
      EVAL_MODE == "full" & NUM_COHORTS == "1" ~ "cohort"
          ,
      .default = EVAL_MODE
    ),
    evals_per_gen = case_when(
      EVAL_MODE == "cohort" ~ 1.0 / NUM_COHORTS,
      EVAL_MODE == "down-sample" ~ TEST_DOWNSAMPLE_RATE,
      EVAL_MODE == "indiv-rand-sample" ~ TEST_DOWNSAMPLE_
          RATE,
      EVAL_MODE == "phylo-informed-sample" ~ TEST_
          DOWNSAMPLE_RATE,
      EVAL_MODE == "full" ~ 1.0
    ),
    EVAL_FIT_EST_MODE = case_when(
      EVAL_FIT_EST_MODE == "ancestor-opt" ~ "ancestor",
      EVAL_FIT_EST_MODE == "relative-opt" ~ "relative",
      .default = EVAL_FIT_EST_MODE
    ),
    est_mode_with_depth = paste(
      EVAL_FIT_EST_MODE,
      EVAL_MAX_PHYLO_SEARCH_DEPTH,
      sep = "-"
    ),
    eval_mode_est_mode_depth = paste(
      EVAL_MODE,
      EVAL_FIT_EST_MODE,
```

```
      EVAL_MAX_PHYLO_SEARCH_DEPTH,
      sep = "-"
    ),
    .keep = "all"
  ) %>%
  mutate(
    eval_label = case_when(
      # Clean up down-sample label
      EVAL_MODE == "down-sample" & EVAL_FIT_EST_MODE != "
          none" ~ paste("down-sample", EVAL_FIT_EST_MODE,
          sep="-"),
      .default = EVAL_MODE
    ),
  ) %>%
  mutate(
    evals_per_gen = as.factor(evals_per_gen),
    est_mode_with_depth = as.factor(est_mode_with_depth),
    eval_mode_est_mode_depth = as.factor(eval_mode_est_
        mode_depth),
    EVAL_MAX_PHYLO_SEARCH_DEPTH = as.factor(EVAL_MAX_
        PHYLO_SEARCH_DEPTH),
    PROBLEM = as.factor(PROBLEM),
    SELECTION = as.factor(SELECTION),
    EVAL_MODE = as.factor(EVAL_MODE),
    NUM_COHORTS = as.factor(NUM_COHORTS),
    TEST_DOWNSAMPLE_RATE = as.factor(TEST_DOWNSAMPLE_RATE
        ),
    EVAL_FIT_EST_MODE = factor(
      EVAL_FIT_EST_MODE,
      levels = c(
        "none",
        "ancestor",
        "relative"
      ),
      labels = c(
        "None",
        "Ancestor",
        "Relative"
      )
    ),
    .keep = "all"
  )

solution_counts <- summary_data %>%
  group_by(
    PROBLEM,
```

```
    evals_per_gen,
    eval_mode_row,
    EVAL_FIT_EST_MODE,
    est_mode_with_depth,
    eval_mode_est_mode_depth,
    EVAL_MODE,
    eval_label,
    EVAL_MAX_PHYLO_SEARCH_DEPTH
  ) %>%
  summarize(
    solution_count = sum(found_solution == "1"),
    replicates = n(),
    no_solution_count = n() - sum(found_solution == "1")
  )

## `summarise()` has grouped output by 'PROBLEM', '
    evals_per_gen', 'eval_mode_row', 'EVAL_FIT_EST_MODE',
    'est_mode_with_depth', 'eval_mode_est_mode_depth', '
    EVAL_MODE', 'eval_label'. You can override using the
## `.groups` argument.

# print(solution_counts, n=208)
solution_table <- kable(solution_counts) %>%
  kable_styling(latex_options = "striped", font_size =
      25)
save_kable(solution_table, paste0(plot_directory, "
    solution_counts_table.pdf"))
solution_table

# Summarize avg num selected
# -- Not totally great because weird stuff happens when a
     solution is found (population collapses, etc)
ts_data_loc <- paste0(working_directory, "data/time_
    series.csv")
ts_data <- read_csv(ts_data_loc)

## Rows: 99773 Columns: 24
## -- Column specification
    --------------------------------------------------------

## Delimiter: ","
## chr  (6): EVAL_FIT_EST_MODE, EVAL_MODE, PROBLEM,
    SELECTION, TESTING_SET_PATH...
## dbl (18): EVAL_MAX_PHYLO_SEARCH_DEPTH, NUM_COHORTS,
    SEED, TEST_DOWNSAMPLE_RA...
##
```

PROBLEM	evals_per_gen	eval_mode_
bouncing-balls	0.01	down-sample
bouncing-balls	0.01	down-sample
bouncing-balls	0.01	indiv-rand-sa
bouncing-balls	0.01	phylo-informe
bouncing-balls	0.1	down-sample
bouncing-balls	0.1	down-sample
bouncing-balls	0.1	indiv-rand-sa
bouncing-balls	0.1	phylo-informe
bouncing-balls	1	full
dice-game	0.01	down-sample
dice-game	0.01	down-sample
dice-game	0.01	indiv-rand-sa
dice-game	0.01	phylo-informe
dice-game	0.1	down-sample
dice-game	0.1	down-sample
dice-game	0.1	indiv-rand-sa
dice-game	0.1	phylo-informe
dice-game	1	full
fizz-buzz	0.01	down-sample
fizz-buzz	0.01	down-sample
fizz-buzz	0.01	indiv-rand-sa
fizz-buzz	0.01	phylo-informe
fizz-buzz	0.1	down-sample
fizz-buzz	0.1	down-sample

```
## i Use 'spec()' to retrieve the full column
    specification for this data.
## i Specify the column types or set 'show_col_types =
    FALSE' to quiet this message.

ts_data <- ts_data %>%
  mutate(
    eval_mode_row = case_when(
      EVAL_MODE == "full" & TEST_DOWNSAMPLE_RATE == "1" ~
          "down-sample",
      EVAL_MODE == "full" & NUM_COHORTS == "1" ~ "cohort"
          ,
      .default = EVAL_MODE
    ),
    evals_per_gen = case_when(
      EVAL_MODE == "cohort" ~ 1.0 / NUM_COHORTS,
      EVAL_MODE == "down-sample" ~ TEST_DOWNSAMPLE_RATE,
      EVAL_MODE == "indiv-rand-sample" ~ TEST_DOWNSAMPLE_
          RATE,
      EVAL_MODE == "phylo-informed-sample" ~ TEST_
          DOWNSAMPLE_RATE,
      EVAL_MODE == "full" ~ 1.0
    ),
    EVAL_FIT_EST_MODE = case_when(
      EVAL_FIT_EST_MODE == "ancestor-opt" ~ "ancestor",
      EVAL_FIT_EST_MODE == "relative-opt" ~ "relative",
      .default = EVAL_FIT_EST_MODE
    ),
    est_mode_with_depth = paste(
      EVAL_FIT_EST_MODE,
      EVAL_MAX_PHYLO_SEARCH_DEPTH,
      sep = "-"
    ),
    eval_mode_est_mode_depth = paste(
      EVAL_MODE,
      EVAL_FIT_EST_MODE,
      EVAL_MAX_PHYLO_SEARCH_DEPTH,
      sep = "-"
    ),
    .keep = "all"
  ) %>%
  mutate(
    eval_label = case_when(
      # Clean up down-sample label
      EVAL_MODE == "down-sample" & EVAL_FIT_EST_MODE != "
          none" ~ paste("down-sample", EVAL_FIT_EST_MODE,
```

```
          sep="-"),
        .default = EVAL_MODE
      ),
    ) %>%
    mutate(
      evals_per_gen = as.factor(evals_per_gen),
      est_mode_with_depth = as.factor(est_mode_with_depth),
      eval_mode_est_mode_depth = as.factor(eval_mode_est_
          mode_depth),
      EVAL_MAX_PHYLO_SEARCH_DEPTH = as.factor(EVAL_MAX_
          PHYLO_SEARCH_DEPTH),
      PROBLEM = as.factor(PROBLEM),
      SELECTION = as.factor(SELECTION),
      EVAL_MODE = as.factor(EVAL_MODE),
      NUM_COHORTS = as.factor(NUM_COHORTS),
      TEST_DOWNSAMPLE_RATE = as.factor(TEST_DOWNSAMPLE_RATE
          ),
      EVAL_FIT_EST_MODE = factor(
        EVAL_FIT_EST_MODE,
        levels = c(
          "none",
          "ancestor",
          "relative"
        ),
        labels = c(
          "None",
          "Ancestor",
          "Relative"
        )
      ),
      .keep = "all"
    )

ts_avgs <- ts_data %>%
    group_by(
      SEED,
      eval_label,
      evals_per_gen,
      PROBLEM
    ) %>%
    summarize(
      n = n(),
      avg_num_unique_selected = mean(num_unique_selected),
      avg_entropy_selected_ids = mean(entropy_selected_ids)
    ) %>%
    mutate(
```

```
    eval_label = as.factor(eval_label),
    evals_per_gen = as.factor(evals_per_gen),
    PROBLEM = as.factor(PROBLEM)
  )

## `summarise()` has grouped output by 'SEED', '
    eval_label', 'evals_per_gen'. You
## can override using the `.groups` argument.
```

8.3 Problem-solving success statistics

```
sol_stats_data <- solution_counts %>%
  filter(EVAL_MODE != "full") %>%
  ungroup() %>%
  unite(
    "grouping",
    PROBLEM,
    evals_per_gen,
    sep="_"
  ) %>%
  select(
    grouping, eval_label, solution_count, no_solution_
        count
  ) %>%
  mutate(
    grouping = as.factor(grouping)
  )

fisher_results <- data.frame(
  comparison = character(),
  group1 = character(),
  group2 = character(),
  n = integer(),
  p = double(),
  p.adj = double(),
  p.adj.signif = character()
)

groupings <- levels(sol_stats_data$grouping)
for (g in groupings) {

  ft_results <- sol_stats_data %>%
    filter(grouping == g) %>%
    select(!grouping) %>%
```

```
    column_to_rownames(var = "eval_label") %>%
    pairwise_fisher_test(
      p.adjust.method = "holm"
    ) %>%
    add_significance("p.adj")

  ft_results <- ft_results %>%
    mutate(
      comparison = rep(g, nrow(ft_results)),
      .keep = "all"
    ) %>%
    relocate(comparison)

  fisher_results <- rbind(
    fisher_results,
    ft_results
  )
}
fisher_results <- as.tibble(fisher_results)


## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?
    as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see
    where this warning was generated.


fisher_results <- fisher_results %>%
  mutate(
    comparison = as.factor(comparison),
    group1 = as.factor(group1),
    group2 = as.factor(group2),
  ) %>%
  group_by(
    comparison
  )

fisher_table <- kbl(fisher_results) %>% kable_styling()
save_kable(fisher_table, paste0(plot_directory, "stats_
    table.pdf"))
fisher_table
```

comparison	group1	group2	n	p	p.adj	p.adj.signif
bouncing-balls_0.01	down-sample	down-sample-ancestor	100	3.09e-02	1.85e-01	ns
bouncing-balls_0.01	down-sample	indiv-rand-sample	100	5.94e-02	2.97e-01	ns
bouncing-balls_0.01	down-sample	phylo-informed-sample	100	3.62e-01	1.00e+00	ns
bouncing-balls_0.01	down-sample-ancestor	indiv-rand-sample	100	1.00e+00	1.00e+00	ns
bouncing-balls_0.01	down-sample-ancestor	phylo-informed-sample	100	3.57e-01	1.00e+00	ns
bouncing-balls_0.01	indiv-rand-sample	phylo-informed-sample	100	5.25e-01	1.00e+00	ns
bouncing-balls_0.1	down-sample	down-sample-ancestor	100	1.17e-01	7.02e-01	ns
bouncing-balls_0.1	down-sample	indiv-rand-sample	100	4.95e-01	1.00e+00	ns
bouncing-balls_0.1	down-sample	phylo-informed-sample	100	2.42e-01	1.00e+00	ns
bouncing-balls_0.1	down-sample-ancestor	indiv-rand-sample	100	6.78e-01	1.00e+00	ns
bouncing-balls_0.1	down-sample-ancestor	phylo-informed-sample	100	1.00e+00	1.00e+00	ns
bouncing-balls_0.1	indiv-rand-sample	phylo-informed-sample	100	1.00e+00	1.00e+00	ns
dice-game_0.01	down-sample	down-sample-ancestor	100	0.00e+00	0.00e+00	****
dice-game_0.01	down-sample	indiv-rand-sample	100	0.00e+00	0.00e+00	****
dice-game_0.01	down-sample	phylo-informed-sample	100	0.00e+00	0.00e+00	****
dice-game_0.01	down-sample-ancestor	indiv-rand-sample	100	1.00e+00	1.00e+00	ns
dice-game_0.01	down-sample-ancestor	phylo-informed-sample	100	4.19e-01	9.42e-01	ns
dice-game_0.01	indiv-rand-sample	phylo-informed-sample	100	3.14e-01	9.42e-01	ns
dice-game_0.1	down-sample	down-sample-ancestor	100	1.00e+00	1.00e+00	ns
dice-game_0.1	down-sample	indiv-rand-sample	100	5.21e-01	1.00e+00	ns
dice-game_0.1	down-sample	phylo-informed-sample	100	1.00e+00	1.00e+00	ns
dice-game_0.1	down-sample-ancestor	indiv-rand-sample	100	3.95e-01	1.00e+00	ns
dice-game_0.1	down-sample-ancestor	phylo-informed-sample	100	1.00e+00	1.00e+00	ns
dice-game_0.1	indiv-rand-sample	phylo-informed-sample	100	5.21e-01	1.00e+00	ns
fizz-buzz_0.01	down-sample	down-sample-ancestor	100	2.62e-01	5.24e-01	ns
fizz-buzz_0.01	down-sample	indiv-rand-sample	100	8.60e-06	4.28e-05	****
fizz-buzz_0.01	down-sample	phylo-informed-sample	100	2.00e-07	1.20e-06	****
fizz-buzz_0.01	down-sample-ancestor	indiv-rand-sample	100	1.59e-03	4.77e-03	**
fizz-buzz_0.01	down-sample-ancestor	phylo-informed-sample	100	8.31e-05	3.32e-04	***
fizz-buzz_0.01	indiv-rand-sample	phylo-informed-sample	100	5.46e-01	5.46e-01	ns
fizz-buzz_0.1	down-sample	down-sample-ancestor	100	8.03e-01	8.38e-01	ns
fizz-buzz_0.1	down-sample	indiv-rand-sample	100	9.55e-05	4.78e-04	***
fizz-buzz_0.1	down-sample	phylo-informed-sample	100	3.46e-03	1.04e-02	*
fizz-buzz_0.1	down-sample-ancestor	indiv-rand-sample	100	1.26e-05	7.56e-05	****
fizz-buzz_0.1	down-sample-ancestor	phylo-informed-sample	100	6.80e-04	2.72e-03	**
fizz-buzz_0.1	indiv-rand-sample	phylo-informed-sample	100	4.19e-01	8.38e-01	ns
for-loop-index_0.01	down-sample	down-sample-ancestor	100	0.00e+00	0.00e+00	****
for-loop-index_0.01	down-sample	indiv-rand-sample	100	0.00e+00	0.00e+00	****
for-loop-index_0.01	down-sample	phylo-informed-sample	100	0.00e+00	0.00e+00	****
for-loop-index_0.01	down-sample-ancestor	indiv-rand-sample	100	1.12e-01	2.24e-01	ns
for-loop-index_0.01	down-sample-ancestor	phylo-informed-sample	100	2.67e-02	8.01e-02	ns
for-loop-index_0.01	indiv-rand-sample	phylo-informed-sample	100	1.00e+00	1.00e+00	ns
for-loop-index_0.1	down-sample	down-sample-ancestor	100	2.98e-01	1.00e+00	ns
for-loop-index_0.1	down-sample	indiv-rand-sample	100	6.82e-01	1.00e+00	ns
for-loop-index_0.1	down-sample	phylo-informed-sample	100	8.40e-01	1.00e+00	ns
for-loop-index_0.1	down-sample-ancestor	indiv-rand-sample	100	6.71e-01	1.00e+00	ns
for-loop-index_0.1	down-sample-ancestor	phylo-informed-sample	100	1.49e-01	8.94e-01	ns
for-loop-index_0.1	indiv-rand-sample	phylo-informed-sample	100	4.16e-01	1.00e+00	ns
gcd_0.01	down-sample	down-sample-ancestor	100	2.31e-04	9.24e-04	***
gcd_0.01	down-sample	indiv-rand-sample	100	1.20e-06	5.90e-06	****
gcd_0.01	down-sample	phylo-informed-sample	100	1.00e-07	4.00e-07	****
gcd_0.01	down-sample-ancestor	indiv-rand-sample	100	2.75e-01	5.50e-01	ns
gcd_0.01	down-sample-ancestor	phylo-informed-sample	100	8.81e-02	2.64e-01	ns

8.4 Average number of unique candidates selected

```
full_avgs <- ts_data %>%
  filter(eval_label == "full") %>%
  group_by(PROBLEM) %>%
  summarize(
    n = n(),
    median_num_unique_selected = median(num_unique_
        selected),
    median_entropy_selected_ids = median(entropy_selected
        _ids),
    avg_num_unique_selected = mean(num_unique_selected),
    avg_entropy_selected_ids = mean(entropy_selected_ids)
  )


build_plot_summary_data <- function(
  data,
  response
) {
  plot <- data %>%
    filter(
      eval_label != "full"
    ) %>%
    ggplot(
      aes_string(
        x = "eval_label",
        y = response,
        fill = "eval_label"
      )
    ) +
    geom_flat_violin(
      position = position_nudge(x = .2, y = 0),
      alpha = .8,
      adjust = 1.5
    ) +
    geom_point(
      mapping = aes(color = eval_label),
      position = position_jitter(width = .15),
      size = .5,
      alpha = 0.8
    ) +
    geom_boxplot(
      width = .1,
```

```
      outlier.shape = NA,
      alpha = 0.5
    ) +
    scale_y_continuous(
      # limits = c(-0.5, 100)
    ) +
    scale_fill_bright() +
    scale_color_bright() +
    facet_grid(
      PROBLEM ~ evals_per_gen,
      # nrow=2,
      labeller = label_both
    ) +
    theme(
      legend.position = "none",
      axis.text.x = element_text(
        angle = 30,
        hjust = 1
      ),
      panel.border = element_rect(color = "gray", size =
          2)
    )

  return(plot)
}

plt <- build_plot_summary_data(
  ts_avgs,
  "avg_num_unique_selected"
)
ggsave(
  filename = paste0(plot_directory, "avg_num_unique_
      selected.pdf"),
  plot = plt
)

## Saving 6.5 x 4.5 in image

plt <- build_plot_summary_data(
  ts_avgs,
  "avg_entropy_selected_ids"
)
ggsave(
  filename = paste0(plot_directory, "avg_entropy_selected
      _ids.pdf"),
  plot = plt
```

```
)

## Saving 6.5 x 4.5 in image
```