

# Supplemental Material

Alexander Lalejini, Austin J. Ferguson, and Charles Ofria

2021-01-19



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Validation experiment</b>	<b>7</b>
2.1	Overview . . . . .	7
2.2	Analysis dependencies . . . . .	8
2.3	Setup . . . . .	8
2.4	Evolution of phenotypic plasticity . . . . .	10
<b>3</b>	<b>Effect of phenotypic plasticity on subsequent evolutionary dynamics</b>	<b>13</b>
3.1	Overview . . . . .	13
3.2	Analysis dependencies . . . . .	13
3.3	Setup . . . . .	14
3.4	Evolution of phenotypic plasticity . . . . .	16
3.5	Phenotypic volatility along dominant lineage . . . . .	18
3.6	Mutation accumulation along dominant lineage . . . . .	20
3.7	Selective sweeps . . . . .	24
3.8	Genome length . . . . .	26
3.9	Diversity over time . . . . .	27



# Chapter 1

## Introduction

TODO



## Chapter 2

# Validation experiment

In this experiment, we validate that (1) we observe the evolution of phenotypic plasticity in a changing environment when digital organisms have access to sensory instructions (capable of differentiating environmental states) and (2) that adaptive phenotypic plasticity does not evolve when populations lack access to sensory instructions.

### 2.1 Overview

```
total_updates <- 200000
replicates <- 100

all_traits <- c("not","nand","and","ornot","or","andnot")
traits_set_a <- c("not", "and", "or")
traits_set_b <- c("nand", "ornot", "andnot")

# Relative location of data.
working_directory <- "experiments/2021-01-07-validation/analysis/" # << For bookdown
# working_directory <- "./" # << For local analysis
```

We evolved populations of digital organisms under four conditions:

1. A fluctuating environment with access to sensory instructions
2. A fluctuating environment without access to sensory instructions (i.e., sensory instructions are no-operations)
3. A constant environment with access to sensory instructions
4. A constant environment without access to sensory instructions

In fluctuating environments, we alternate between rewarding and punishing different sets of computational tasks. In one environment, we reward tasks not,

and, or and punish tasks nand, ornot, andnot. In the alternative environment, we reward tasks nand, ornot, andnot and punish tasks not, and, or. In constant environments, we reward all tasks (not, nand, and, ornot, or, andnot).

For each replicate of each condition, we extract the dominant (i.e., most numerous) genotype at the end of the run to analyze further. We expect to observe the evolution of adaptive phenotypic plasticity in only the first experimental condition. In conditions without sensors, plasticity in any form should be unable to evolve.

## 2.2 Analysis dependencies

Load all required R libraries.

```
library(ggplot2)
library(tidyverse)
library(cowplot)
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd9")
```

These analyses were conducted/knitted with the following computing environment:

```
print(version)

##
## platform      x86_64-pc-linux-gnu
## arch          x86_64
## os            linux-gnu
## system        x86_64, linux-gnu
## status
## major         4
## minor         0.3
## year          2020
## month         10
## day           10
## svn rev       79318
## language      R
## version.string R version 4.0.3 (2020-10-10)
## nickname      Bunny-Wunnies Freak Out
```

## 2.3 Setup

```
data_loc <- paste0(working_directory, "data/aggregate.csv")
data <- read.csv(data_loc, na.strings="NONE")

data$DISABLE_REACTION_SENSORS <- as.factor(data$DISABLE_REACTION_SENSORS)
```



```

data$chg_env <- as.factor(data$chg_env)
data$dom_plastic_odd_even <- as.factor(data$dom_plastic_odd_even)
data$sensors <- data$DISABLE_REACTION_SENSORS == "0"
data$is_plastic <- data$dom_plastic_odd_even == "True"

env_label_fun <- function(chg_env) {
  if (chg_env) {
    return("Fluctuating")
  } else {
    return("Constant")
  }
}

sensors_label_fun <- function(has_sensors) {
  if (has_sensors) {
    return("Sensors")
  } else {
    return("No sensors")
  }
}

# Count observed plasticity for each condition (I'm sure there's a 'tidier' way to do this..)
observed_plasticity <- data.frame(
  environment=character(),
  sensors=character(),
  plastic=integer(),
  nonplastic=integer(),
  plastic_adaptive=integer(),
  plastic_optimal=integer(),
  plastic_nonadaptive=integer()
)

for (env_chg in levels(data$chg_env)) {
  for (disabled_sensors in levels(data$DISABLE_REACTION_SENSORS)) {
    cond_data <- filter(data, chg_env == env_chg & data$DISABLE_REACTION_SENSORS == disabled_sensors)
    environment_label <- env_label_fun(env_chg)
    sensors_label <- sensors_label_fun(disabled_sensors == "0")

    observed_plasticity <- observed_plasticity %>% add_row(
      environment=environment_label,
      sensors=sensors_label,
      plastic=nrow(filter(cond_data, is_plastic==TRUE)),
      nonplastic=nrow(filter(cond_data, is_plastic==FALSE)),
      plastic_adaptive=nrow(filter(cond_data, dom_adaptive_plasticity=="True")),
      plastic_optimal=nrow(filter(cond_data, dom_optimal_plastic=="True")),
      plastic_nonadaptive=nrow(filter(cond_data, is_plastic==TRUE & dom_adaptive_plasticity=="False"))
    )
  }
}

```

```

    )
  }
}

observed_plasticity <- pivot_longer(
  observed_plasticity,
  cols=c("plastic", "plastic_adaptive", "plastic_optimal", "plastic_nonadaptive", "nonplastic"),
  names_to="phenotype",
  values_to="phenotype_cnt"
)

##### misc #####
# Configure our default graphing theme
theme_set(theme_cowplot())

```

## 2.4 Evolution of phenotypic plasticity

For each experimental condition, do we observe the evolution of phenotypic plasticity? To test for phenotypic plasticity, we culture digital organisms in both environments from the fluctuating condition (including organisms evolved in a constant environment). Any plasticity that we observe from digital organisms evolved under constant conditions is cryptic variation (as these organisms were never exposed to these culturing environments).

```

ggplot(filter(observed_plasticity, phenotype %in% c("plastic", "nonplastic")), aes(x=phenotype, y=phenotype_cnt)) +
  geom_bar(
    stat="identity",
    position=position_dodge(0.9)
  ) +
  geom_text(
    stat="identity",
    mapping=aes(label=phenotype_cnt),
    vjust=0.05
  ) +
  scale_fill_brewer(palette="Accent") +
  scale_x_discrete(
    name="Phenotype",
    limits=c("plastic", "nonplastic"),
    labels=c("Plastic", "Non-plastic")
  ) +
  facet_grid(sensors~environment) +
  theme(
    legend.position="none"
  )

```



Indeed, we do not observe the evolution of phenotypic plasticity in any replicates in which digital organisms do not have access to sensory instructions. We do observe the evolution of plasticity (not necessarily adaptive plasticity) in both constant and fluctuating environments where sensors are enabled.

To what extent is the observed phenotypic plasticity adaptive?

```
ggplot(filter(observed_plasticity, environment=="Fluctuating" & sensors == "Sensors" & phenotype
  geom_bar(
    stat="identity",
    position=position_dodge(0.9)
  ) +
  geom_text(
    stat="identity",
    mapping=aes(label=phenotype_cnt),
    vjust=0.05
  ) +
  scale_fill_brewer(palette="Accent") +
  scale_x_discrete(
    name="Phenotype",
    limits=c("plastic", "plastic_adaptive", "plastic_optimal", "plastic_nonadaptive"),
    labels=c("Total plastic", "Adaptive plasticity", "Optimal plasticity", "Non-adaptive plasticity")
  ) +
  facet_grid(sensors~environment) +
  theme(
    legend.position="none"
```



## Chapter 3

# Effect of phenotypic plasticity on subsequent evolutionary dynamics

### 3.1 Overview

```
total_updates <- 200000
replicates <- 200

all_traits <- c("not","nand","and","ornot","or","andnot")
traits_set_a <- c("not", "and", "or")
traits_set_b <- c("nand", "ornot", "andnot")

# Relative location of data.
working_directory <- "experiments/2021-01-12-evo-dynamics/analysis/" # << For bookdown
# working_directory <- "." # << For local analysis
```

### 3.2 Analysis dependencies

Load all required R libraries.

```
library(ggplot2)
library(tidyverse)
library(cowplot)
library(Hmisc)
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce9")
```

These analyses were conducted/knitted with the following computing environment:

```
print(version)

##
## platform      _
## arch          x86_64-pc-linux-gnu
## os            linux-gnu
## system        x86_64, linux-gnu
## status
## major         4
## minor         0.3
## year          2020
## month         10
## day           10
## svn rev       79318
## language      R
## version.string R version 4.0.3 (2020-10-10)
## nickname      Bunny-Wunnies Freak Out
```

### 3.3 Setup

```
summary_data_loc <- paste0(working_directory, "data/aggregate.csv")
summary_data <- read.csv(summary_data_loc, na.strings="NONE")

summary_data$DISABLE_REACTION_SENSORS <- as.factor(summary_data$DISABLE_REACTION_SENSORS)
summary_data$chg_env <- summary_data$chg_env == "True"
summary_data$dominant_plastic_odd_even <- as.factor(summary_data$dominant_plastic_odd_even)
summary_data$sensors <- summary_data$DISABLE_REACTION_SENSORS == "0"
summary_data$is_plastic <- summary_data$dominant_plastic_odd_even == "True"

env_label_fun <- function(chg_env) {
  if (chg_env) {
    return("Fluctuating")
  } else {
    return("Constant")
  }
}

sensors_label_fun <- function(has_sensors) {
  if (has_sensors) {
    return("Sensors")
  } else {
    return("No sensors")
  }
}
```

```

    }
  }

  # note that this labeler makes assumptions about how we set up our experiment
  condition_label_fun <- function(has_sensors, env_chg) {
    if (has_sensors && env_chg) {
      return("Plastic (fluctuating)")
    } else if (env_chg) {
      return("Non-plastic (fluctuating)")
    } else {
      return("Non-plastic (constant)")
    }
  }
}

summary_data$env_label <- mapply(
  env_label_fun,
  summary_data$chg_env
)
summary_data$sensors_label <- mapply(
  sensors_label_fun,
  summary_data$sensors
)
summary_data$condition <- mapply(
  condition_label_fun,
  summary_data$sensors,
  summary_data$chg_env
)

condition_order = c(
  "Non-plastic (constant)",
  "Non-plastic (fluctuating)",
  "Plastic (fluctuating)"
)

##### time series #####
# time_series_data_loc <- paste0(working_directory, "data/time_series_u0-u200000.csv")
time_series_data_loc <- paste0(working_directory, "data/time_series_u10000-u20000.csv")

time_series_data <- read.csv(time_series_data_loc)
time_series_data$DISABLE_REACTION_SENSORS <- as.factor(time_series_data$DISABLE_REACTION_SENSORS)
time_series_data$chg_env <- time_series_data$chg_env == "True"
time_series_data$sensors <- time_series_data$DISABLE_REACTION_SENSORS == "0"

time_series_data$env_label <- mapply(
  env_label_fun,

```

```

    time_series_data$chg_env
  )
time_series_data$sensors_label <- mapapply(
  sensors_label_fun,
  time_series_data$sensors
)
time_series_data$condition <- mapapply(
  condition_label_fun,
  time_series_data$sensors,
  time_series_data$chg_env
)

##### misc #####
# Configure our default graphing theme
theme_set(theme_cowplot())

```

### 3.4 Evolution of phenotypic plasticity

For sensor-enabled populations in fluctuating environments, we only transferred populations containing an optimally plastic genotype to phase-two.

```

summary_data_grouped = dplyr::group_by(summary_data, sensors, env_label, condition)
summary_data_group_counts = dplyr::summarize(summary_data_grouped, n=dplyr::n())

```

## `summarise()` has grouped output by 'sensors', 'env\_label'. You can override using `ungroup()`

```

ggplot(summary_data_group_counts, aes(x=condition, y=n, fill=condition)) +
  geom_col(position=position_dodge(0.9)) +
  geom_text(aes(label=n, y=n+2)) +
  scale_x_discrete(
    name="Condition",
    limits=condition_order
  ) +
  ylab("Number of replicates in phase two") +
  theme(
    legend.position="none"
  )

```





We can confirm our expectation that the dominant genotypes in non-plastic conditions are not phenotypically plastic.

```
summary_data_grouped = dplyr::group_by(summary_data, condition, is_plastic)
summary_data_group_counts = dplyr::summarize(summary_data_grouped, n=dplyr::n())
```

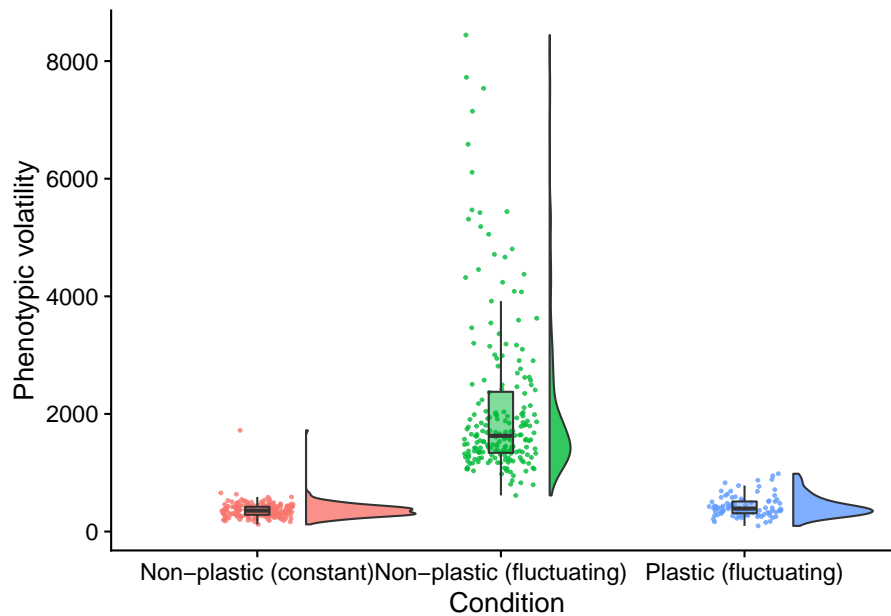
```
## `summarise()` has grouped output by 'condition'. You can override using the `.groups` argument
ggplot(filter(summary_data_group_counts, is_plastic), aes(x=condition, y=n, fill=condition)) +
  geom_col(position=position_dodge(0.9)) +
  scale_x_discrete(
    name="Condition",
    limits=condition_order
  ) +
  geom_text(aes(label=n, y=n+1)) +
  ylab("Number of replicates with a plastic dominant genotype") +
  theme(
    legend.position="none"
  )
```



### 3.5 Phenotypic volatility along dominant lineage

```
ggplot(summary_data, aes(x=condition, y=dominant_lineage_phenotypic_volatility, fill=condition)) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color=condition),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_x_discrete(
    name="Condition",
    limits=condition_order
  ) +
```

```
ylab("Phenotypic volatility") +
theme(
  legend.position="none"
)
```



```
kruskal.test(
  formula=dominant_lineage_phenotypic_volatility~condition,
  data=summary_data
)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: dominant_lineage_phenotypic_volatility by condition
## Kruskal-Wallis chi-squared = 351.93, df = 2, p-value < 2.2e-16
```

```
pairwise.wilcox.test(
  x=summary_data$dominant_lineage_phenotypic_volatility,
  g=summary_data$condition,
  p.adjust.method="bonferroni",
)
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: summary_data$dominant_lineage_phenotypic_volatility and summary_data$condition
```

```
##
##                               Non-plastic (constant) Non-plastic (fluctuating)
## Non-plastic (fluctuating) <2e-16 -
## Plastic (fluctuating)      0.014 <2e-16
##
## P value adjustment method: bonferroni
median(filter(summary_data, condition=="Plastic (fluctuating)")$dominant_lineage_phenon

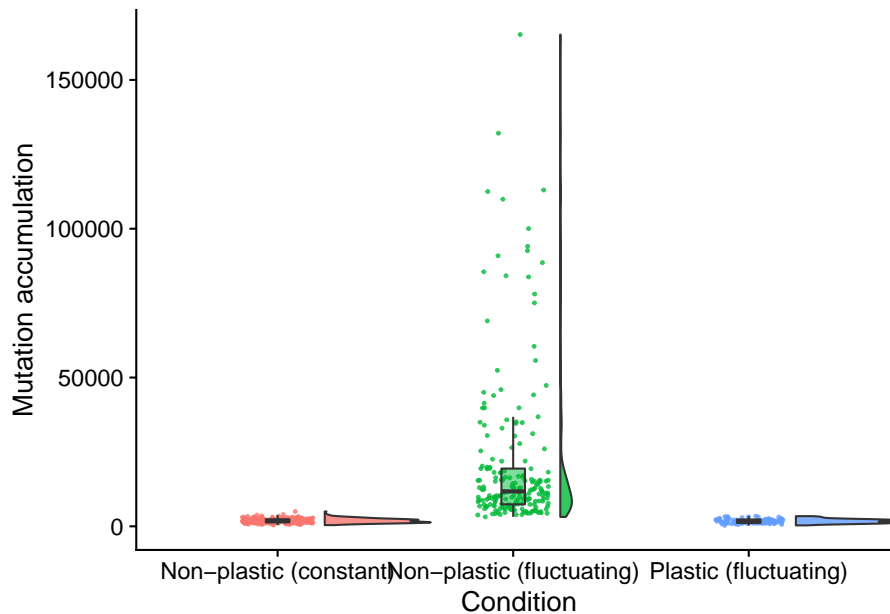
## [1] 391
median(filter(summary_data, condition=="Non-plastic (constant)")$dominant_lineage_phenon

## [1] 356
median(filter(summary_data, condition=="Non-plastic (fluctuating)")$dominant_lineage_phenon

## [1] 1628
```

### 3.6 Mutation accumulation along dominant lineage

```
ggplot(summary_data, aes(x=condition, y=dominant_lineage_total_mut_cnt, fill=condition)) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color=condition),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_x_discrete(
    name="Condition",
    limits=condition_order
  ) +
  ylab("Mutation accumulation") +
  theme(
    legend.position="none"
  )
```



```
kruskal.test(
  formula=dominant_lineage_total_mut_cnt~condition,
  data=summary_data
)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: dominant_lineage_total_mut_cnt by condition
## Kruskal-Wallis chi-squared = 352.67, df = 2, p-value < 2.2e-16
```

```
pairwise.wilcox.test(
  x=summary_data$dominant_lineage_total_mut_cnt,
  g=summary_data$condition,
  p.adjust.method="bonferroni",
)
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: summary_data$dominant_lineage_total_mut_cnt and summary_data$condition
##
##               Non-plastic (constant) Non-plastic (fluctuating)
## Non-plastic (fluctuating) <2e-16 -
## Plastic (fluctuating)    0.65 <2e-16
##
```

```
## P value adjustment method: bonferroni
median(filter(summary_data, condition=="Plastic (fluctuating)")$dominant_lineage_total.

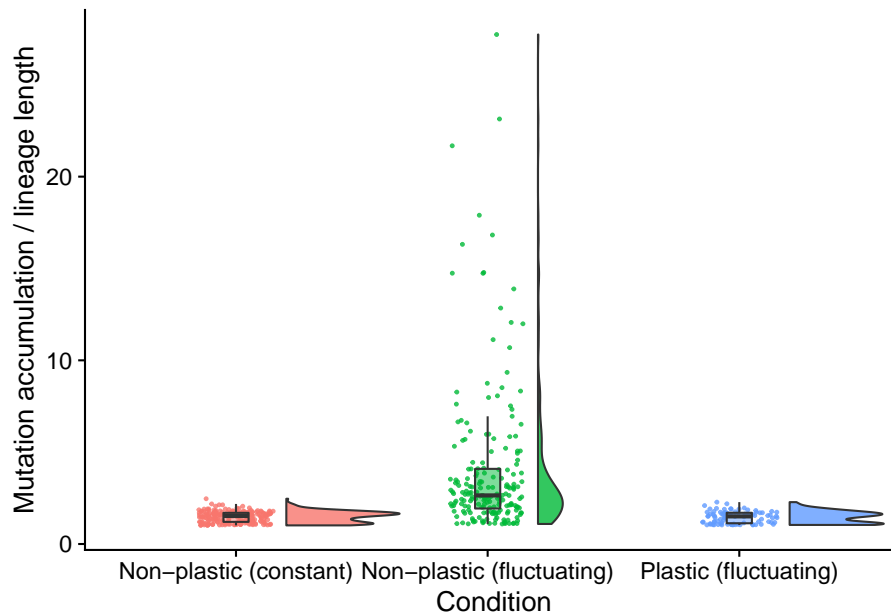
## [1] 1833
median(filter(summary_data, condition=="Non-plastic (constant)")$dominant_lineage_total.

## [1] 1823.5
median(filter(summary_data, condition=="Non-plastic (fluctuating)")$dominant_lineage_t

## [1] 11721
```

### 3.6.1 Mutation accumulation normalized by lineage length

```
summary_data$mutations_per_lineage_step <- summary_data$dominant_lineage_total_mut_cnt
ggplot(summary_data, aes(x=condition, y=mutations_per_lineage_step, fill=condition)) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color=condition),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_x_discrete(
    name="Condition",
    limits=condition_order
  ) +
  ylab("Mutation accumulation / lineage length") +
  theme(
    legend.position="none"
  )
)
```



```
kruskal.test(
  formula=mutations_per_lineage_step~condition,
  data=summary_data
)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: mutations_per_lineage_step by condition
## Kruskal-Wallis chi-squared = 194.32, df = 2, p-value < 2.2e-16
```

```
pairwise.wilcox.test(
  x=summary_data$mutations_per_lineage_step,
  g=summary_data$condition,
  p.adjust.method="bonferroni",
)
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: summary_data$mutations_per_lineage_step and summary_data$condition
##
##               Non-plastic (constant) Non-plastic (fluctuating)
## Non-plastic (fluctuating) <2e-16 -
## Plastic (fluctuating)      1 <2e-16
##
```

```
## P value adjustment method: bonferroni
median(filter(summary_data, condition=="Plastic (fluctuating)")$mutations_per_lineage,
method="bonferroni")

## [1] 1.504491
median(filter(summary_data, condition=="Non-plastic (constant)")$mutations_per_lineage,
method="bonferroni")

## [1] 1.526001
median(filter(summary_data, condition=="Non-plastic (fluctuating)")$mutations_per_lineage,
method="bonferroni")

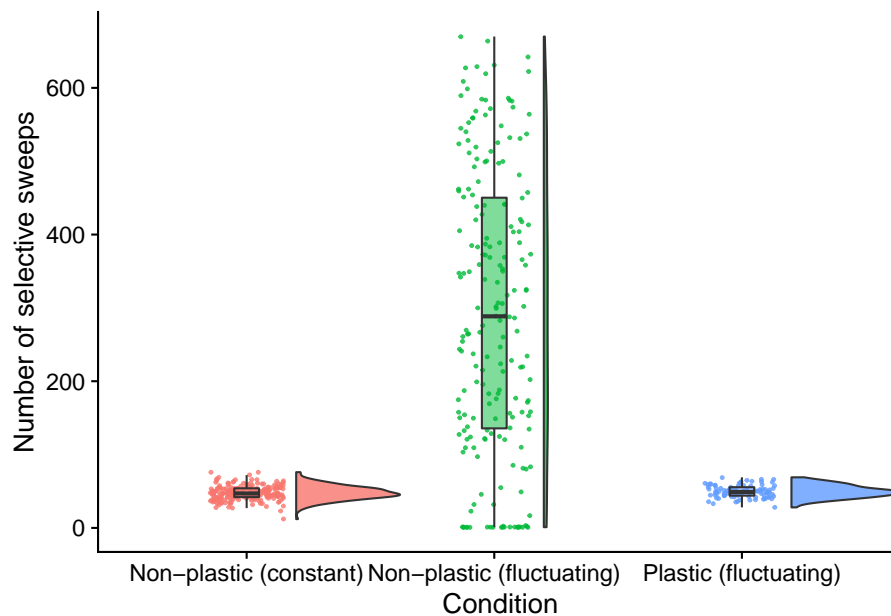
## [1] 2.645598
```

### 3.7 Selective sweeps

The number of times the most recent common ancestor changes gives us the number of selective sweeps that occur during the experiment.

```
ggplot(summary_data, aes(x=condition, y=phylo_mrca_changes, fill=condition)) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color=condition),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_x_discrete(
    name="Condition",
    limits=condition_order
  ) +
  ylab("Number of selective sweeps") +
  theme(
    legend.position="none"
  )
```





```
kruskal.test(
  formula=phylo_mrca_changes~condition,
  data=summary_data
)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: phylo_mrca_changes by condition
## Kruskal-Wallis chi-squared = 182.92, df = 2, p-value < 2.2e-16
```

```
pairwise.wilcox.test(
  x=summary_data$phylo_mrca_changes,
  g=summary_data$condition,
  p.adjust.method="bonferroni",
)
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: summary_data$phylo_mrca_changes and summary_data$condition
##
##               Non-plastic (constant) Non-plastic (fluctuating)
## Non-plastic (fluctuating) <2e-16 -
## Plastic (fluctuating)    0.35 <2e-16
##
```

```
## P value adjustment method: bonferroni
median(filter(summary_data, condition=="Plastic (fluctuating)")$phylo_mrca_changes)

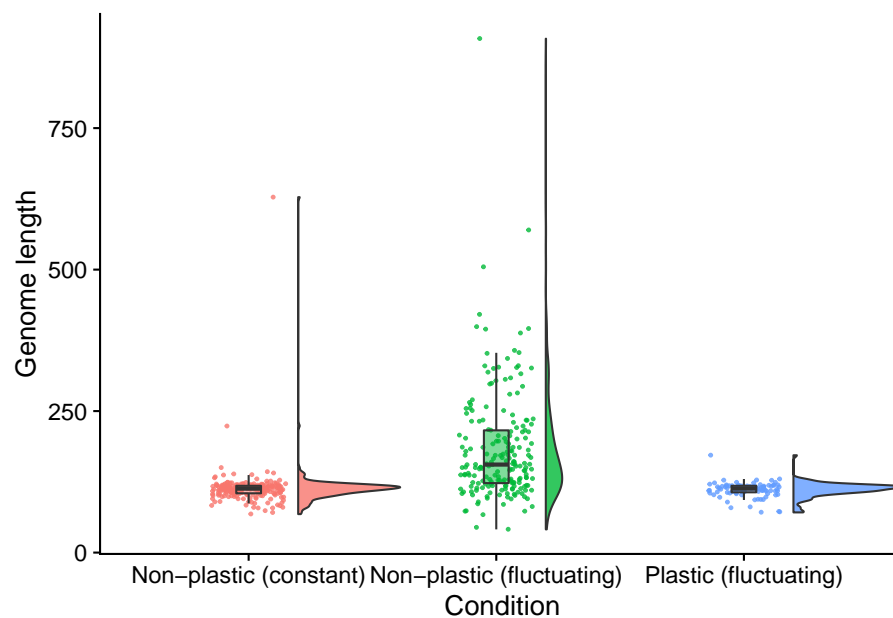
## [1] 49
median(filter(summary_data, condition=="Non-plastic (constant)")$phylo_mrca_changes)

## [1] 47
median(filter(summary_data, condition=="Non-plastic (fluctuating)")$phylo_mrca_changes)

## [1] 288.5
```

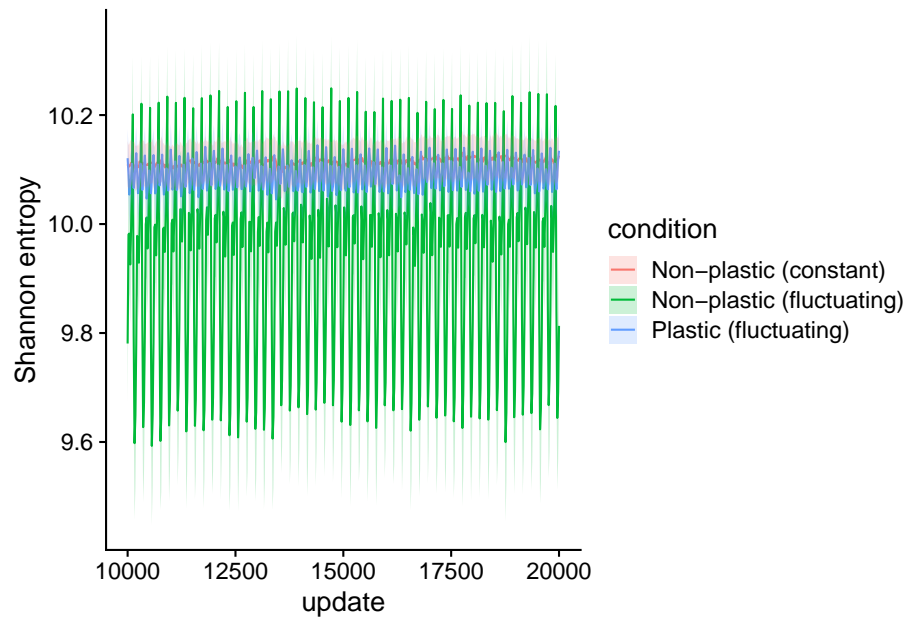
### 3.8 Genome length

```
ggplot(summary_data, aes(x=condition, y=dominant_genome_length, fill=condition)) +
  geom_flat_violin(
    position = position_nudge(x = .2, y = 0),
    alpha = .8
  ) +
  geom_point(
    mapping=aes(color=condition),
    position = position_jitter(width = .15),
    size = .5,
    alpha = 0.8
  ) +
  geom_boxplot(
    width = .1,
    outlier.shape = NA,
    alpha = 0.5
  ) +
  scale_x_discrete(
    name="Condition",
    limits=condition_order
  ) +
  ylab("Genome length") +
  theme(
    legend.position="none"
  )
```



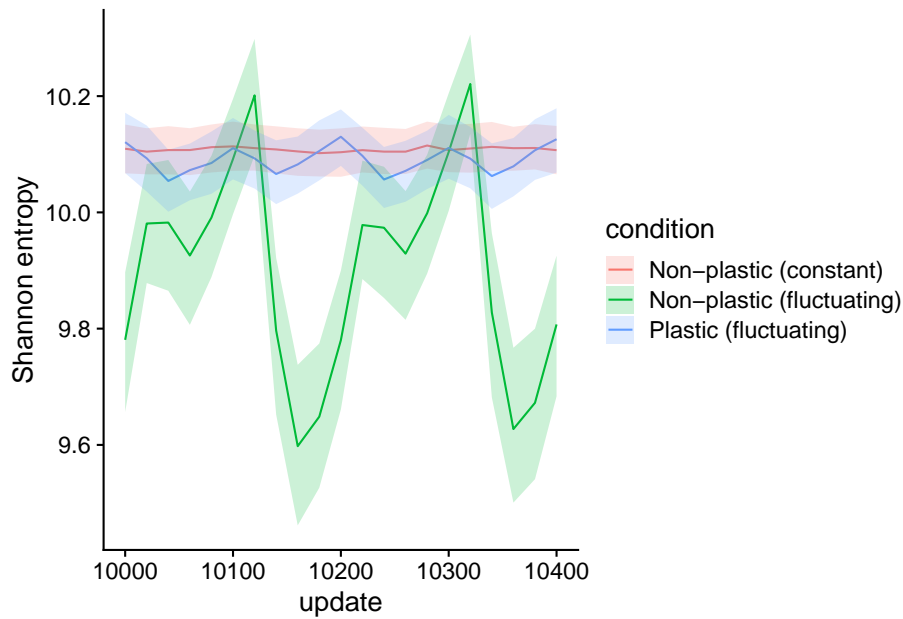
### 3.9 Diversity over time

```
# shannon entropy
ggplot(time_series_data, aes(x=update, y=phylo_diversity, fill=condition, color=condition)) +
  stat_summary(fun="mean", geom="line") +
  stat_summary(
    fun.data="mean_cl_boot",
    fun.args=list(conf.int=0.95),
    geom="ribbon",
    alpha=0.2,
    linetype=0
  ) +
  ylab("Shannon entropy")
```



Zoom in to just 400 update span.

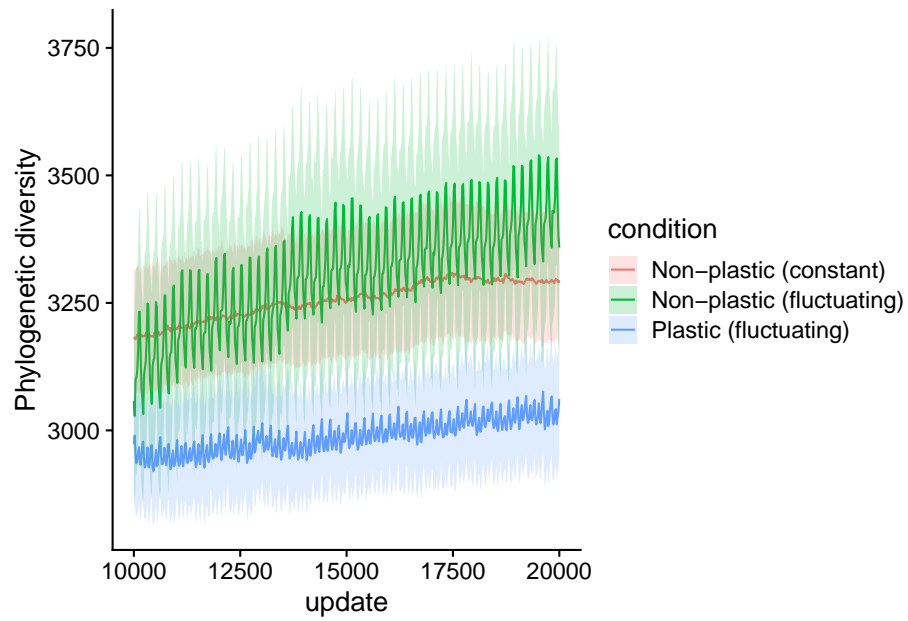
```
# shannon entropy
ggplot(filter(time_series_data, update <= 10400), aes(x=update, y=phylo_diversity, fil
  stat_summary(fun="mean", geom="line") +
  stat_summary(
    fun.data="mean_cl_boot",
    fun.args=list(conf.int=0.95),
    geom="ribbon",
    alpha=0.2,
    linetype=0
  ) +
  ylab("Shannon entropy")
```



### 3.9.2 Phylogenetic diversity over time

// From (Faith 1992, reviewed in Winters et al., 2013), phylogenetic diversity is the sum of edge  
 // This calculates phylogenetic diversity for all extant taxa in the tree.

```
ggplot(time_series_data, aes(x=update, y=phylo_current_phylogenetic_diversity, fill=condition, co
  stat_summary(fun="mean", geom="line") +
  stat_summary(
    fun.data="mean_cl_boot",
    fun.args=list(conf.int=0.95),
    geom="ribbon",
    alpha=0.2,
    linetype=0
  ) +
  ylab("Phylogenetic diversity")
```



```
ggplot(time_series_data, aes(x=update, y=phylo_num_taxa_extant, fill=condition, color=condition)) +
  stat_summary(fun="mean", geom="line") +
  stat_summary(
    fun.data="mean_cl_boot",
    fun.args=list(conf.int=0.95),
    geom="ribbon",
    alpha=0.2,
    linetype=0
  )
```

