Supplement for: "Potentiating Mutations Facilitate the Evolution of Associative Learning in Digital Organisms"

Austin J. Ferguson and Charles Ofria

2023-05-20

Contents

Chapter 1

Introduction

Here we show all additional graphs that would not fit in the paper. Please use the navigation on the left to view the different sections.

For data, source code, and analyses, please refer to the GitHub repo: https://github.com/FergusonAJ/replaying_evolution_of_learning

This bookdown supplement is based on Alex Lalejini's example: https://github.com/amlalejini/auto-deploying-bookdown-example.

Chapter 2

Initial 200 parallel replicates

Here we plot data associated with the initial 200 replicates. All data, analysis scripts, etc. are found in the experiment directory: `/experiments/alife_2023/2023_02_21_01__initial_10_cooldown/`.

2.1 Dependencies

```
# External
library(ggplot2)
library(dplyr)
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce9

base_repo_dir = '../..'
exp_dir = paste0(base_repo_dir, '/experiments/alife_2023/2023_02_21_01__initial_10_cooldown')

# Internal
source(paste0(base_repo_dir, '/global_shared_files/constant_vars__three_cues_one_set.R'))
source(paste0(base_repo_dir, '/global_shared_files/shared_funcs__three_cues_one_set.R'))
```

2.2 Loading data

To save time, we load the pre-processed data.

To see how the data gets processed, please see `./analysis/final_dominant_analysis.R` from the experiment directory.

```
# Data for all trials from all seeds
df = read.csv(paste0(exp_dir, '/data/processed_data/processed_full.csv'))
# One line per seed that summarizes all 100 trials
```

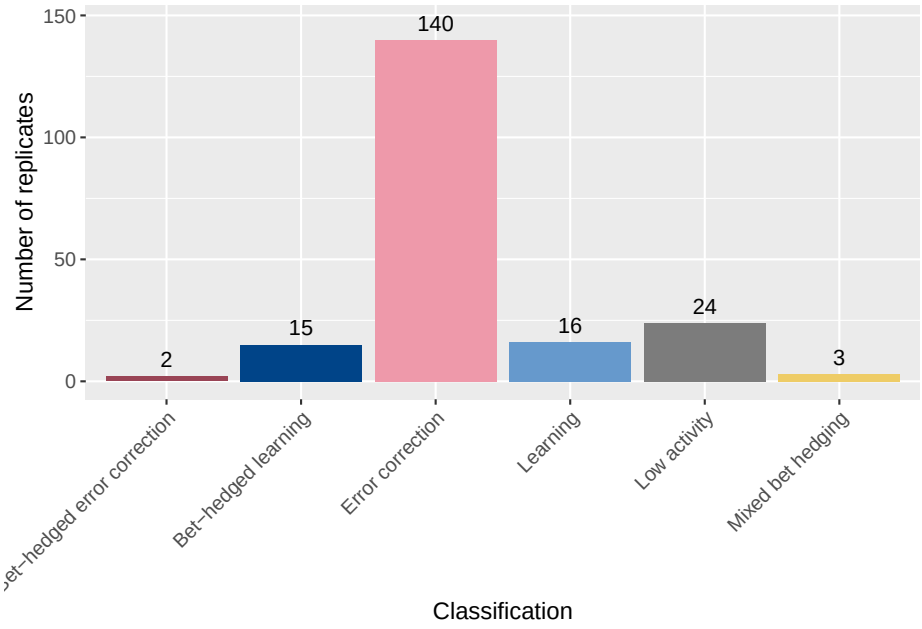
```
df_summary = read.csv(paste0(exp_dir, '/data/processed_data/processed_summary.csv'))
# Sumamrize each category, e.g., how many replicates evolved error correction?
classification_summary = read.csv(paste0(exp_dir, '/data/processed_data/processed_class
```

2.3 Behavior classification summary

Of the 200 initial replicates, how many evolved learning? Error correction?

Here we plot the number of replicates that evolved each behavior. To determine the behavior of the replicate, we analyzed the most abundant genotype at the end of evolution for each replicate.

```
ggplot(classification_summary, aes(x = seed_classification_factor, y = count, fill = se
  geom_col() +
  geom_text(aes(y = count + 7, label = count), size = 4) +
  scale_fill_manual(values = color_map) +
  xlab('Classification') +
  ylab('Number of replicates') +
  theme(legend.position = 'none') +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust = 1)) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12))
```



2.4 Accuracy plots

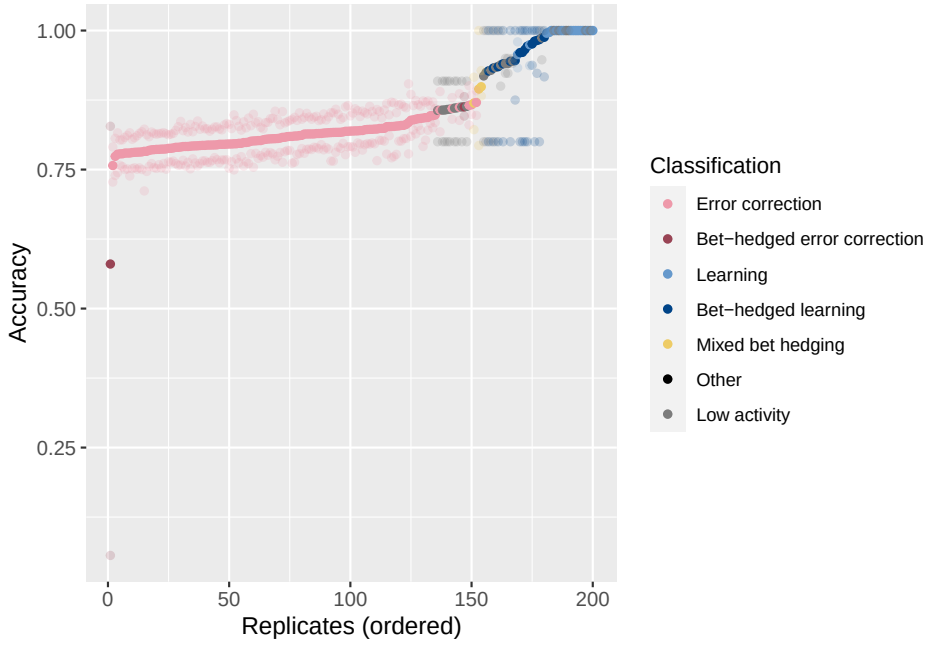
Next, we'll look at the accuracy (number of non-backwards movements correct / non-backwards movements taken) for each replicate.

Order the points by mean accuracy.

```
df_summary = df_summary[order(df_summary$accuracy_mean),]
df_summary$seed_order = 1:nrow(df_summary)
df$seed_order = NA
for(seed in unique(df$seed)){
  df[df$seed == seed,]$seed_order = df_summary[df_summary$seed == seed,]$seed_order
}
```

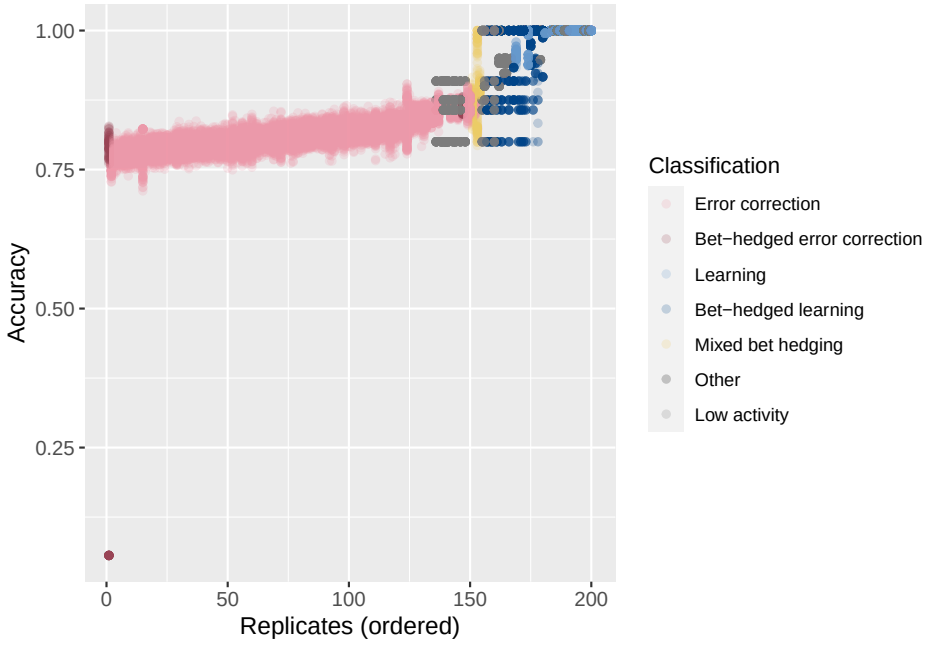
For each replicate, plot mean accuracy as a solid point and min/max as partially transparent points.

```
ggplot(df_summary, aes(x = seed_order, color = seed_classification)) +
  geom_point(aes(y = accuracy_mean)) +
  geom_point(aes(y = accuracy_min), alpha = 0.2) +
  geom_point(aes(y = accuracy_max), alpha = 0.2) +
  scale_color_manual(values = color_map) +
  xlab('Replicates (ordered)') +
  ylab('Accuracy') +
  #theme(axis.text.x = element_text(angle=45, vjust = 1, hjust = 1)) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification')
```



Now plot the accuracy of each sample, maintaining the same order on the x-axis.

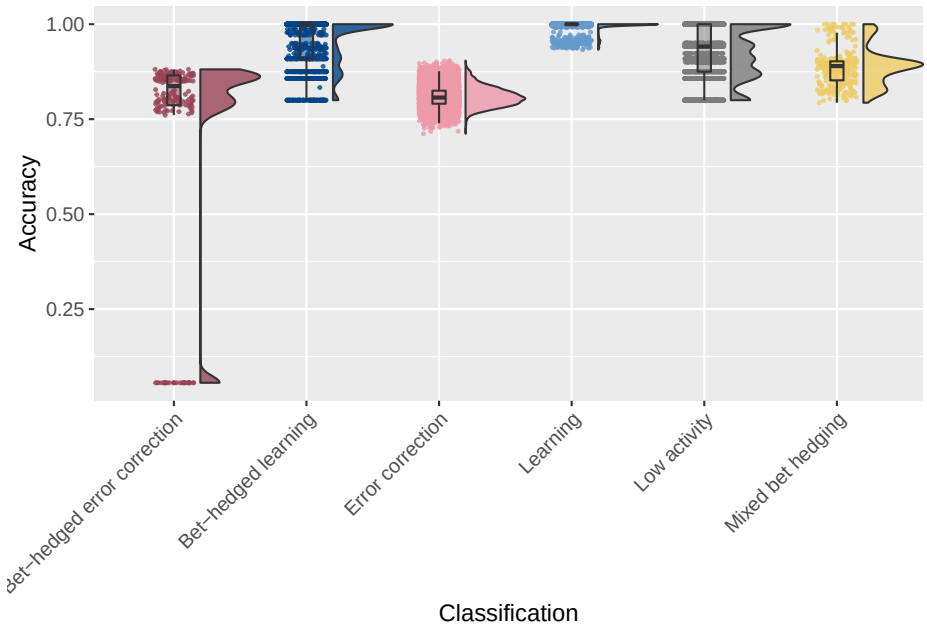
```
ggplot(df, aes(x = seed_order, y = accuracy, color = seed_classification)) +
  geom_point(alpha = 0.2) +
  scale_color_manual(values = color_map) +
  xlab('Replicates (ordered)') +
  ylab('Accuracy') +
  #theme(axis.text.x = element_text(angle=45, vjust = 1, hjust = 1)) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification')
```



2.5 Raincloud plots (individual trials)

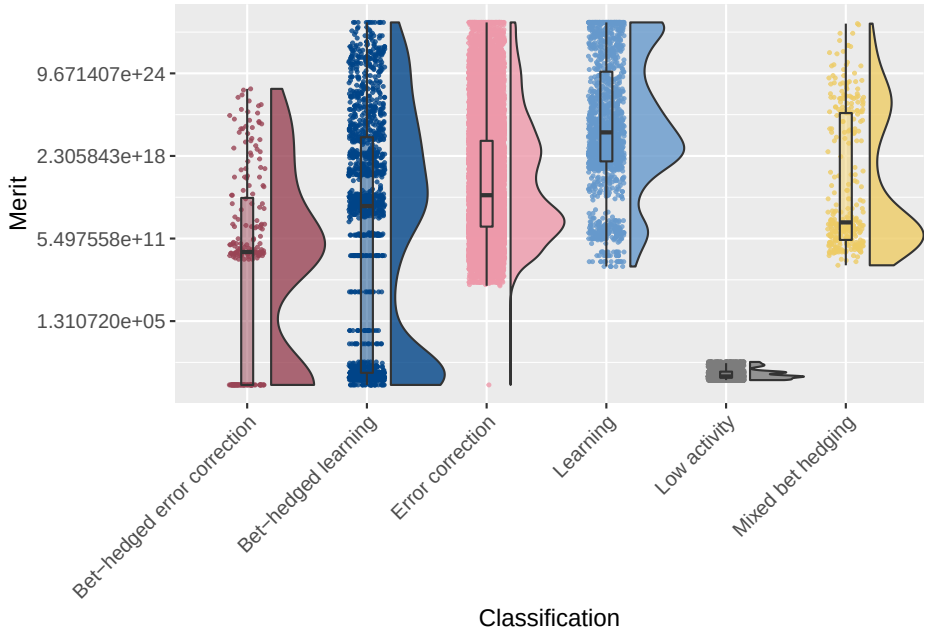
Now, we look at accuracy and merit of individual trials, grouped only by behavior. Thus, all points are individual trials.

```
# Raincloud plot of accuracy
ggplot(df, aes(x = seed_classification_factor, y = accuracy, fill = seed_classification_factor))
  geom_flat_violin(scale="width", position = position_nudge(x = .2, y = 0), alpha = .8 ) +
  geom_point(mapping=aes(color=seed_classification_factor), position = position_jitter(width = .1
  geom_boxplot( width = .1, outlier.shape = NA, alpha = 0.5 ) +
  scale_fill_manual(values = color_map) +
  scale_color_manual(values = color_map) +
  xlab('Classification') +
  ylab('Accuracy') +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust = 1)) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  theme(legend.position = 'none')
```



Note that merit is plotted on a logarithmic (base 2) scale:

```
# Raincloud plot of merit
ggplot(df, aes(x = seed_classification_factor, y = merit, fill = seed_classification_fa
  geom_flat_violin(scale="width", position = position_nudge(x = .2, y = 0), alpha = .8
  geom_point(mapping=aes(color=seed_classification_factor), position = position_jitter
  geom_boxplot( width = .1, outlier.shape = NA, alpha = 0.5 ) +
  scale_fill_manual(values = color_map) +
  scale_color_manual(values = color_map) +
  scale_y_continuous(trans = 'log2') +
  xlab('Classification') +
  ylab('Merit') +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust = 1)) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  theme(legend.position = 'none')
```

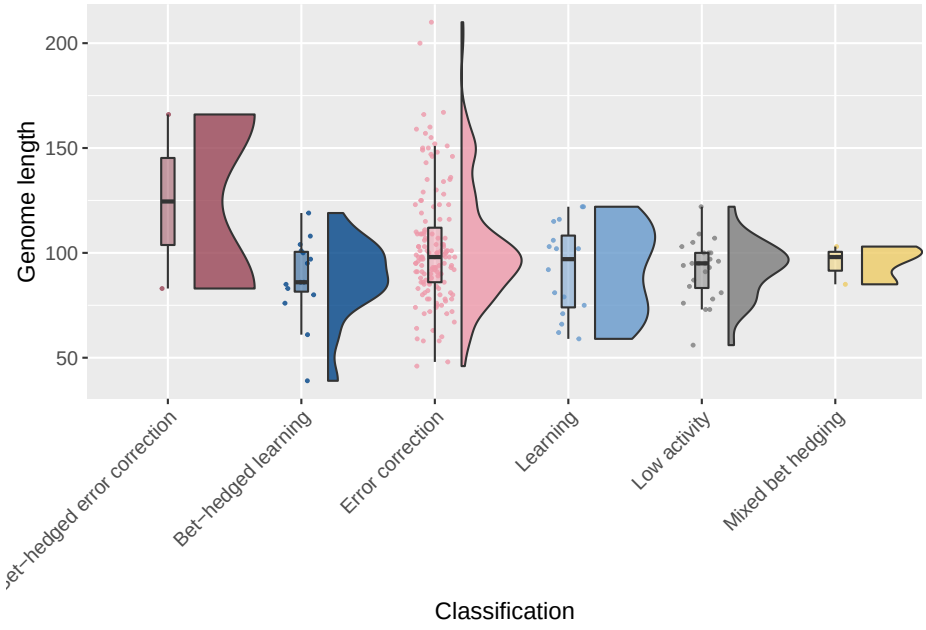


2.6 Raincloud plots (per replicate)

Finally, we again use raincloud plots, but here we look at *replicates* instead of individual trials. Each point represents one of the initial 200 replicates.

First we look at genome length (the ancestor had 100 instructions):

```
# Raincloud plot of genome length
ggplot(df_summary, aes(x = seed_classification_factor, y = genome_length, fill = seed_classificat
  geom_flat_violin(scale="width", position = position_nudge(x = .2, y = 0), alpha = .8 ) +
  geom_point(mapping=aes(color=seed_classification_factor), position = position_jitter(width = .1
  geom_boxplot( width = .1, outlier.shape = NA, alpha = 0.5 ) +
  scale_fill_manual(values = color_map) +
  scale_color_manual(values = color_map) +
  xlab('Classification') +
  ylab('Genome length') +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust = 1)) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  theme(legend.position = 'none')
```

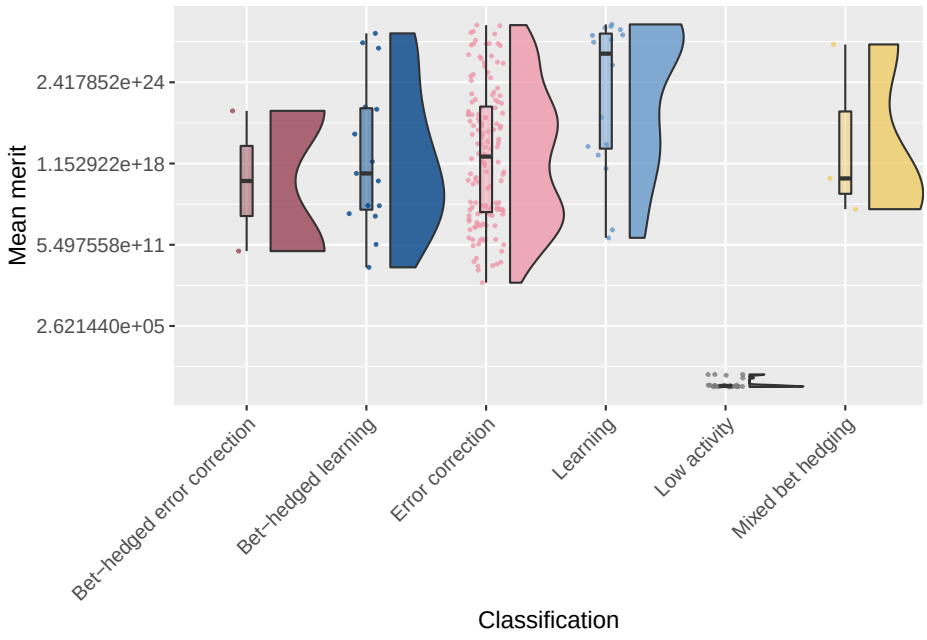


Next we look at the mean accuracy for each replicate:

```
# Raincloud plot of mean accuracy of replicates
ggplot(df_summary, aes(x = seed_classification_factor, y = accuracy_mean, fill = seed_
  geom_flat_violin(scale="width", position = position_nudge(x = .2, y = 0), alpha = .8
  geom_point(mapping=aes(color=seed_classification_factor), position = position_jitter
  geom_boxplot( width = .1, outlier.shape = NA, alpha = 0.5 ) +
  scale_fill_manual(values = color_map) +
  scale_color_manual(values = color_map) +
  scale_y_continuous(limits = c(0,1)) +
  xlab('Classification') +
  ylab('Mean accuracy') +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust = 1)) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  theme(legend.position = 'none')
```

Finally, we look at the average merit for each replicate. Note that the y-axis is on a log2 scale.

```
# Raincloud plot of mean merit of replicates
ggplot(df_summary, aes(x = seed_classification_factor, y = merit_mean, fill = seed_classificatior
  geom_flat_violin(scale="width", position = position_nudge(x = .2, y = 0), alpha = .8 ) +
  geom_point(mapping=aes(color=seed_classification_factor), position = position_jitter(width = .1
  geom_boxplot( width = .1, outlier.shape = NA, alpha = 0.5 ) +
  scale_fill_manual(values = color_map) +
  scale_color_manual(values = color_map) +
  scale_y_continuous(trans = 'log2') +
  xlab('Classification') +
  ylab('Mean merit') +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust = 1)) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  theme(legend.position = 'none')
```


Mean merit
2.417852e+24
1.152922e+18
5.497558e+11
2.621440e+05
Bet−hedged error correction
Bet−hedged learning
Error correction
Learning
Low activity
Mixed bet hedging
Classification

Chapter 3

Lineage A (Seed 86)

Here we plot more details of Lineage A than would fit in the paper. If you are looking through the experiment files, Lineage A was seed 86. All data for Lineage A can be found in the `data/reps/86/` directory.

3.1 Dependencies

```
rm(list = ls())
# External
library(ggplot2)
library(dplyr)
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce9

# Configuration variables
base_repo_dir = '../..'
exp_dir = paste0(base_repo_dir, '/experiments/alife_2023/2023_02_21_01__initial_10_cooldown')

seed = 86
potentiation_window_start = 400
potentiation_window_stop = 500
potentiation_target = 484

# Internal
source(paste0(base_repo_dir, '/global_shared_files/constant_vars__three_cues_one_set.R'))
source(paste0(base_repo_dir, '/global_shared_files/shared_funcs__three_cues_one_set.R'))
```

3.2 Loading data

To save time, we load the pre-processed data.

To see how the data gets processed, please see `./analysis/replay_analysis.R` from the experiment directory.

```
# Data for all trials from all replay replicates
df = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/processed_data/processed_
# One line per replay replicate that summarizes all 100 trials
df_summary = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/processed_data/pro
# Sumamrize each category, e.g., how many replicates evolved error correction?
classification_summary = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/proces
# Also grab lineage data
df_lineage = read.csv(paste0(exp_dir, '/data/reps/', seed, '/dominant_lineage_summary.c
# Because lineage data was collected on the cluster before we renamed a few classifica
df_lineage[df_lineage$seed_classification == 'Bet-hedged imprinting',]$seed_classificat
df_lineage[df_lineage$seed_classification == 'Bet-hedged imprinting',]$seed_classificat
df_lineage[df_lineage$seed_classification == 'Small',]$seed_classification_factor = see
df_lineage[df_lineage$seed_classification == 'Small',]$seed_classification = seed_class
```

3.3 Lineage data

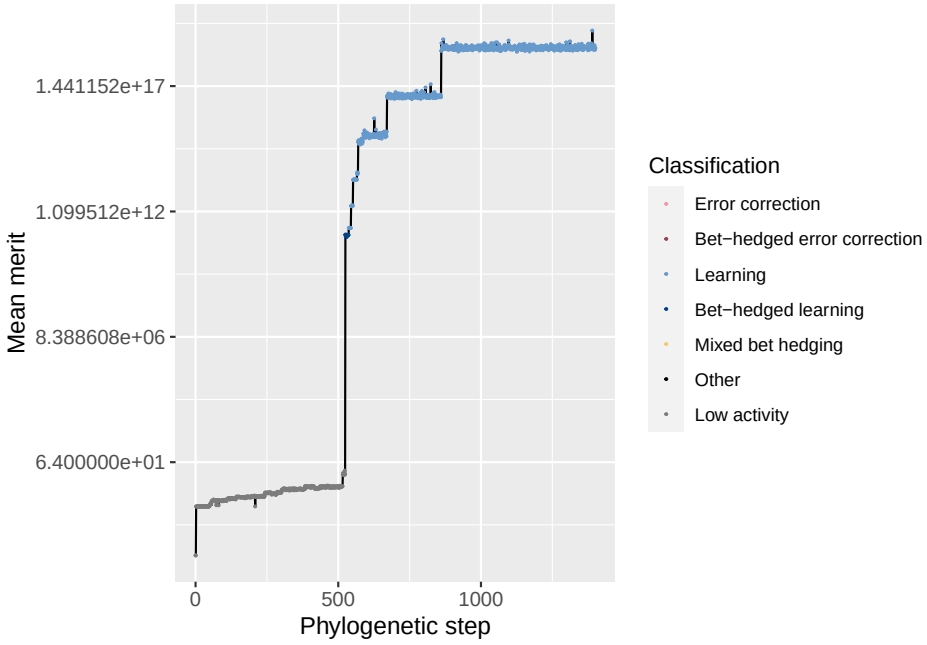
While not shown in the paper due to space limitations, it is important to look at the initial replicate's lineage and how it changes over time.

The code for these plots is copied from the `single_seed_lineage_analysis.R` script.

3.3.1 Merit

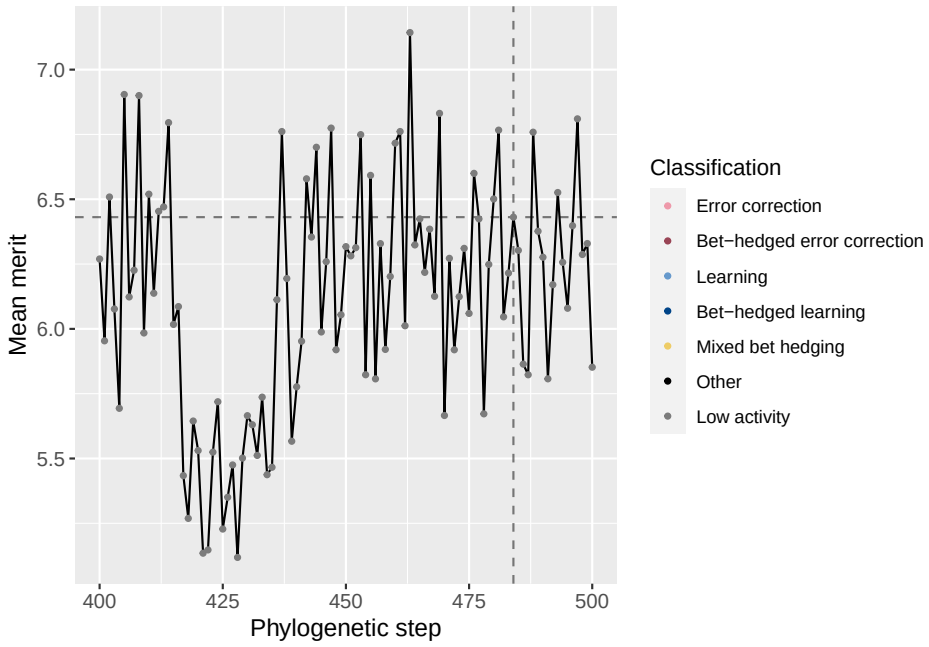
First, we look at the lineage as a whole and how average merit changes over time (note the y-axis is logarithmic)

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = merit_mean)) +
  geom_point(aes(y = merit_mean, color = seed_classification), size = 0.25) +
  scale_y_continuous(trans = 'log2') +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```



Additionally, we can zoom in to view just the potentiation window. The vertical dashed line shows the main potentiating step, step 484, while the horizontal dashed line shows average merit at that step for easier comparisons. Note that this plot is *not* logarithmic.

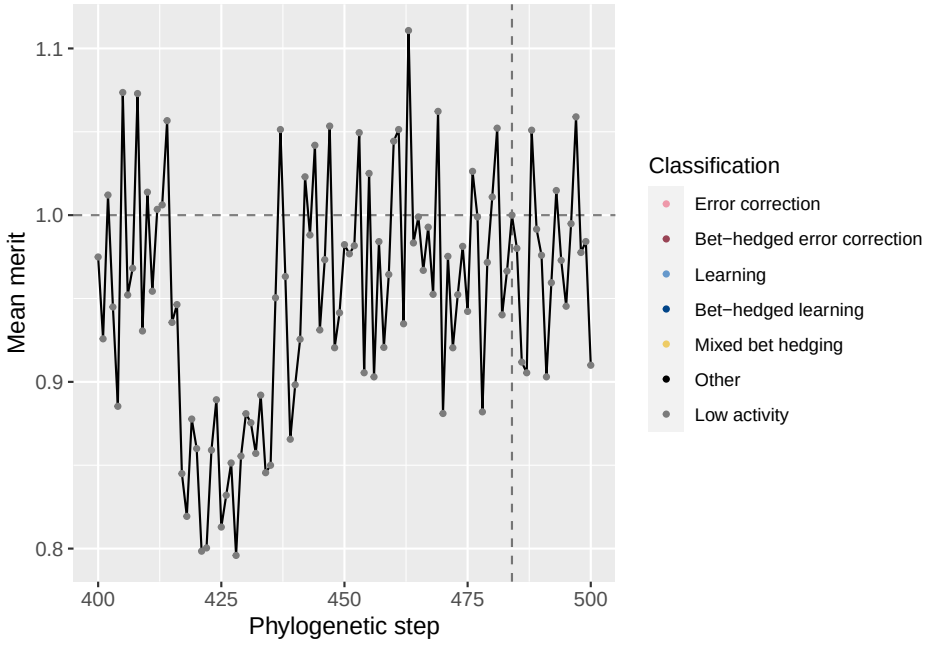
```
potentiation_mask = df_lineage$depth >= potentiation_window_start & df_lineage$depth <= potentiat
target_step_merit_mean = df_lineage[df_lineage$depth == potentiation_target,]$merit_mean
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_merit_mean), linetype = 'dashed', alpha = 0.5) +
  geom_line(aes(y = merit_mean)) +
  geom_point(aes(y = merit_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```



We do not see substantial differences in fitness around step 484.

While it may not be needed for this lineage, we can also normalize the merit based on the potentiating step. Remember that one additional correct state confers a merit bonus of 10%. Here, we can see that the fluctiations around the potentiating mutation generally fall in that +/-10% range.

```
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = 1), linetype = 'dashed', alpha = 0.5) +
  geom_line(aes(y = merit_mean / target_step_merit_mean)) +
  geom_point(aes(y = merit_mean / target_step_merit_mean, color = seed_classification)) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```

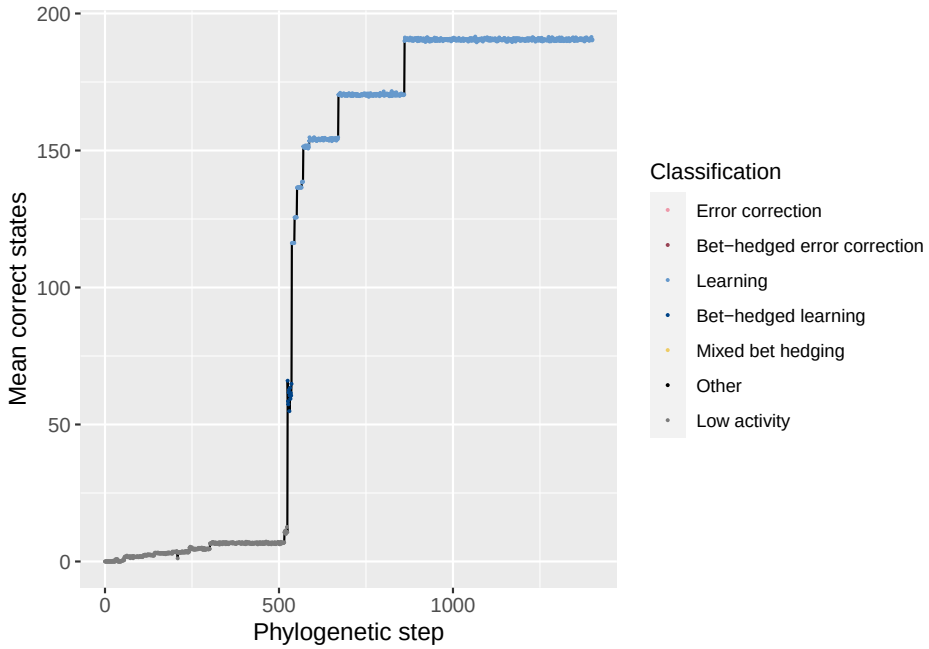


3.3.2 Correct states

Similarly, we can look at the number of correct states both overall and in the potentiation window:

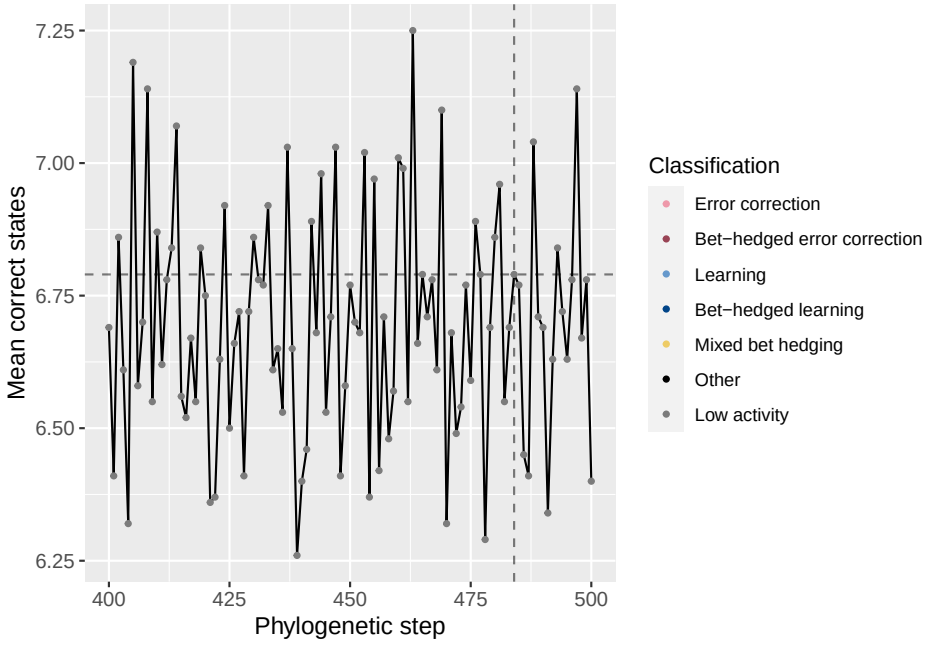
Overall:

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = correct_doors_mean)) +
  geom_point(aes(y = correct_doors_mean, color = seed_classification), size = 0.25) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean correct states') +
  xlab('Phylogenetic step')
```



Potentiation window (dashed lines show the main potentiating mutation):

```
target_step_correct_doors_mean = df_lineage[df_lineage$depth == potentiation_target,]$c
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_correct_doors_mean), linetype = 'dashed', alp
  geom_line(aes(y = correct_doors_mean)) +
  geom_point(aes(y = correct_doors_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean correct states') +
  xlab('Phylogenetic step')
```



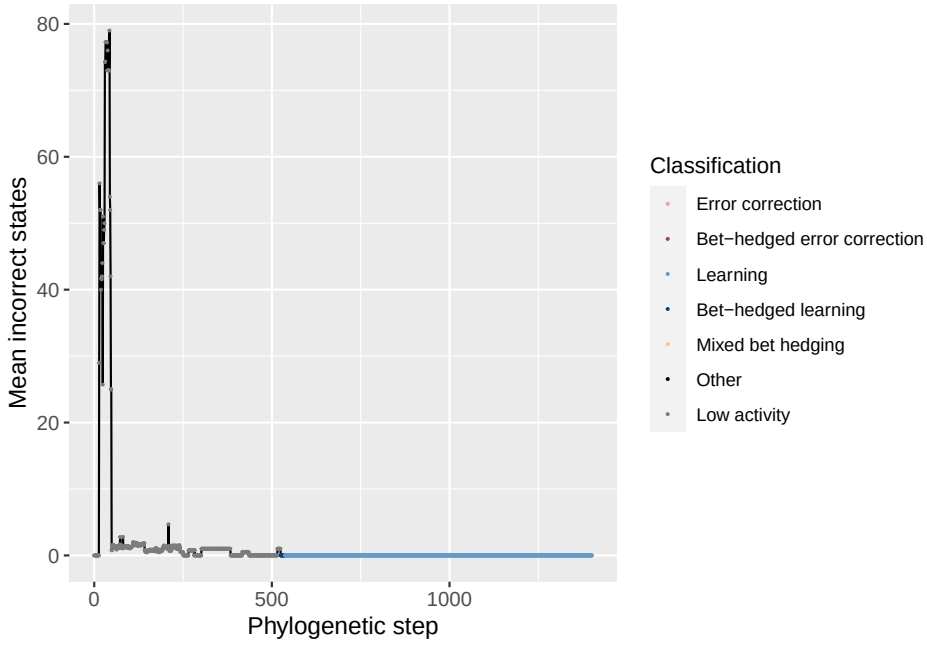
We see no noticable change in the number of correct states around the potentiating mutation.

3.3.3 Incorrect states

Just like our analysis of correct states, we can also look at the average *incorrect* states over time. By incorrect states, we mean the wrong movements while in a *left*, *right*, or *forward* state. Incorrect movements from a *backward* state are tracked separately.

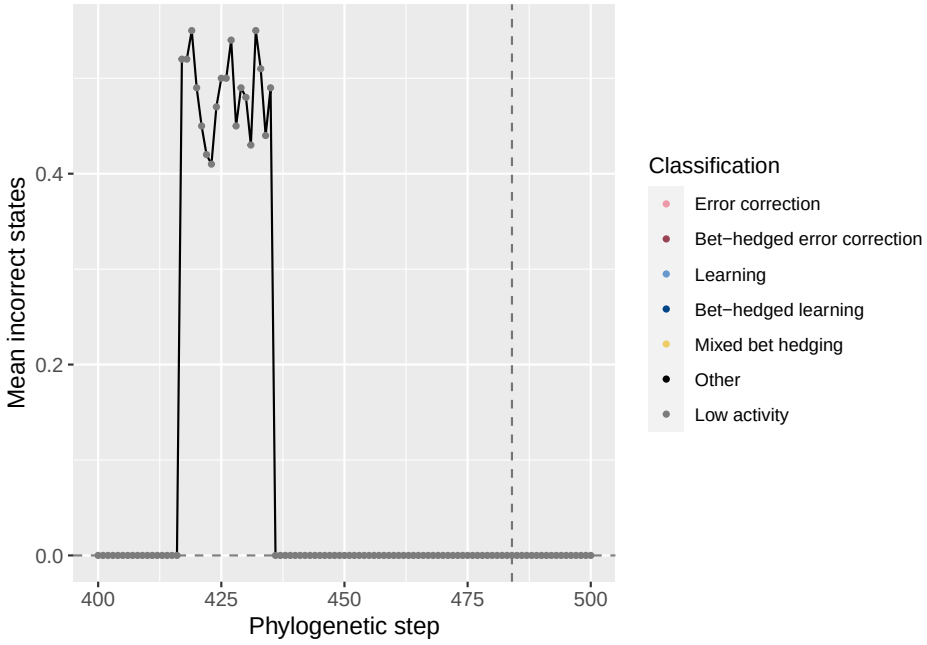
Overall:

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = incorrect_doors_mean)) +
  geom_point(aes(y = incorrect_doors_mean, color = seed_classification), size = 0.25) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean incorrect states') +
  xlab('Phylogenetic step')
```



Potentiation window:

```
target_step_incorrect_doors_mean = df_lineage[df_lineage$depth == potentiation_target,]
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_incorrect_doors_mean), linetype = 'dashed', a
  geom_line(aes(y = incorrect_doors_mean)) +
  geom_point(aes(y = incorrect_doors_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean incorrect states') +
  xlab('Phylogenetic step')
```



3.4 Replay data

Now that we know what the lineage looks like, we can begin to break down the replay data.

3.4.1 Exploratory replays

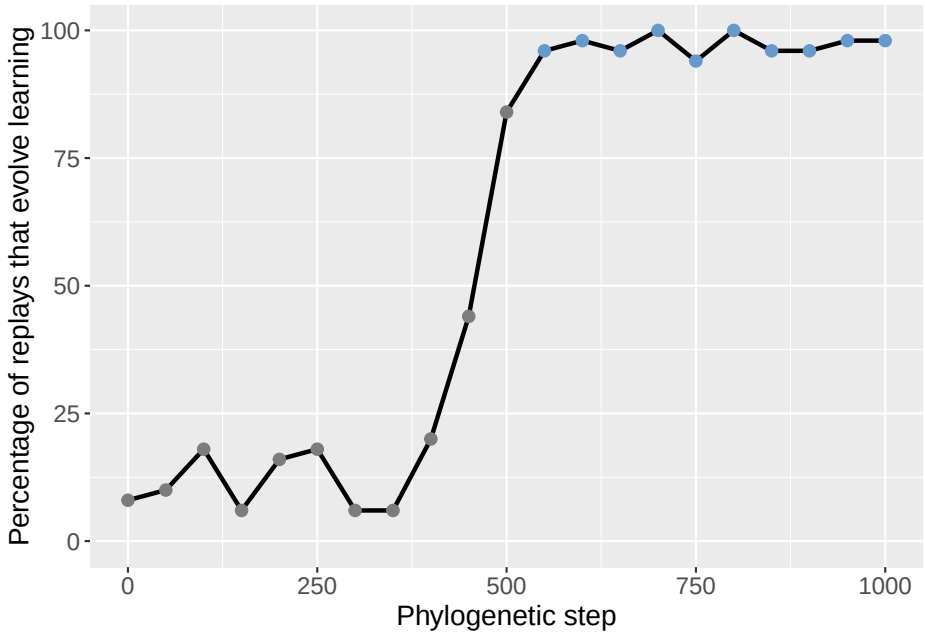
We started by seeding 50 replay replicates for every 50th step along the lineage in the window [50, 1000]. The data for step 0 comes from the initial 200 replicates.

Here we show the potentiation over time for those exploratory replays:

```
df_learning_summary = classification_summary[classification_summary$seed_classification == seed_c
df_learning_summary$lineage_classification = NA
# Assign each depth its classification from the lineage data
for(depth in unique(df_learning_summary$depth)){
  df_learning_summary[df_learning_summary$depth == depth,]$lineage_classification = df_lineage[df
}
# Create masks
mask_learning_summary_focal = df_learning_summary$depth %in% potentiation_window_start:potentiati
mask_learning_summary_initial = df_learning_summary$depth %% 50 == 0

ggplot(df_learning_summary[mask_learning_summary_initial,], aes(x=depth)) +
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size = 2.5) +
```

```
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```

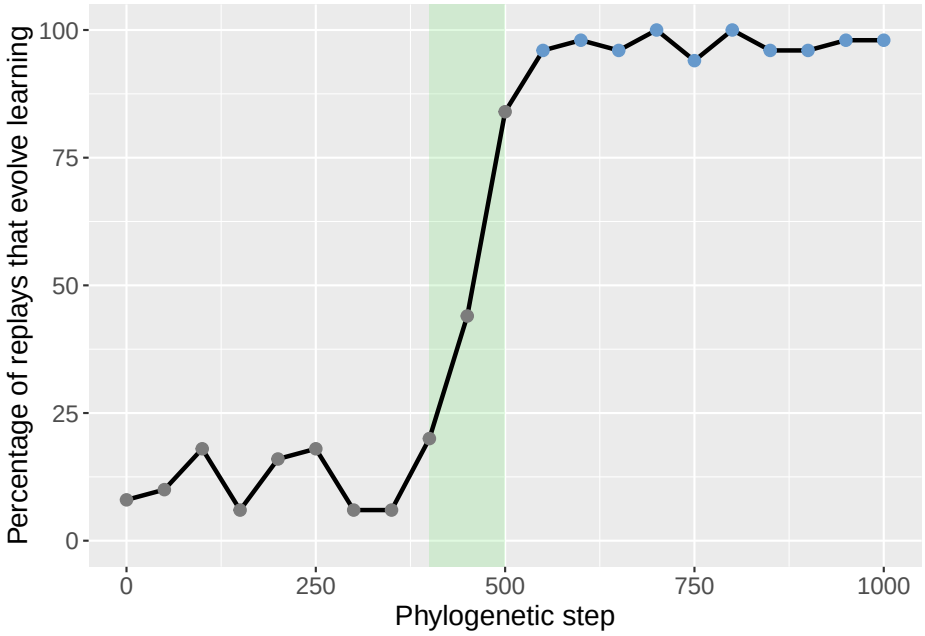


We can then identify the potentiation window based on the points with the most potentiation gain.

Here we selected steps 400-500 (green shaded region):

```
ggplot(df_learning_summary[mask_learning_summary_initial,], aes(x=depth)) +
  annotate("rect", xmin=potentiation_window_start, xmax=potentiation_window_stop, ymin=
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
```

```
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```



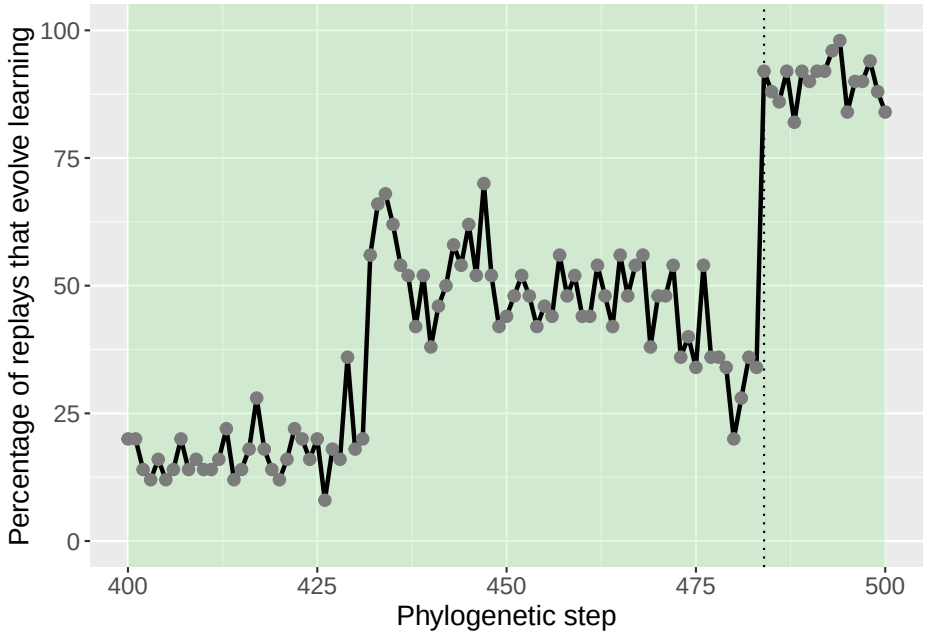
3.4.2 Targeted replays

With the potentiation window identified, we seeded targeted replays.

For the targeted replays, we seeded 50 replay replicates from *every* step in the potetiation window. This gives us an idea of how individual steps in the phylogney affected potentiation.

```
ggplot(df_learning_summary[mask_learning_summary_focal,], aes(x=depth)) +
  annotate("rect", xmin=potentiation_window_start, xmax=potentiation_window_stop, ymin=-Inf, ymax
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dotted') +
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size = 2.5) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
```

```
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```



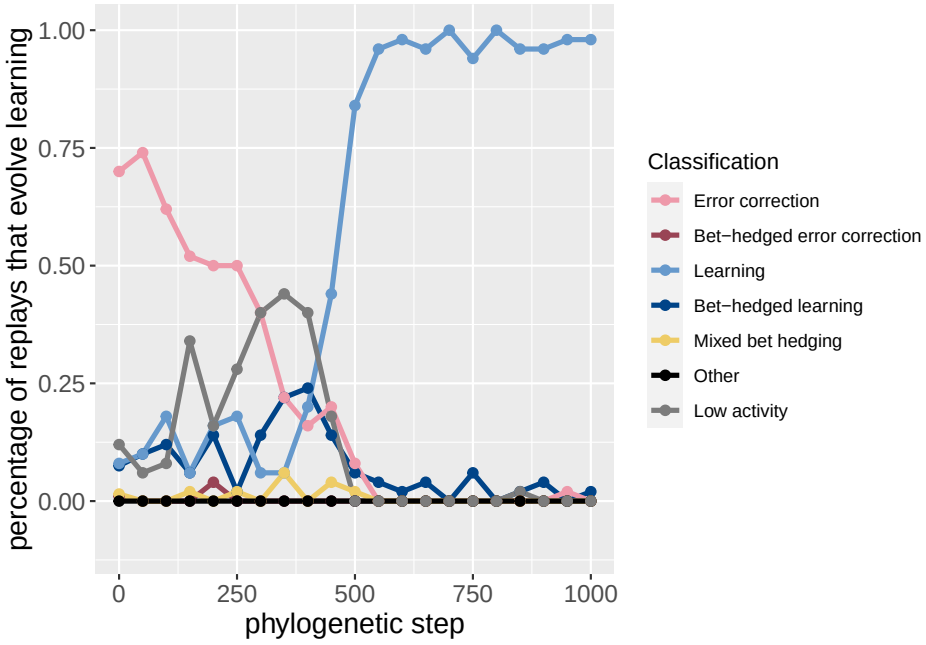
This plot was extensively discussed in the paper, so we refrain from additional discussion here.

3.4.3 Potentiation of other behaviors

While the paper focused on learning, it can be useful to see how the potentiation of other behaviors changes before learning takes over.

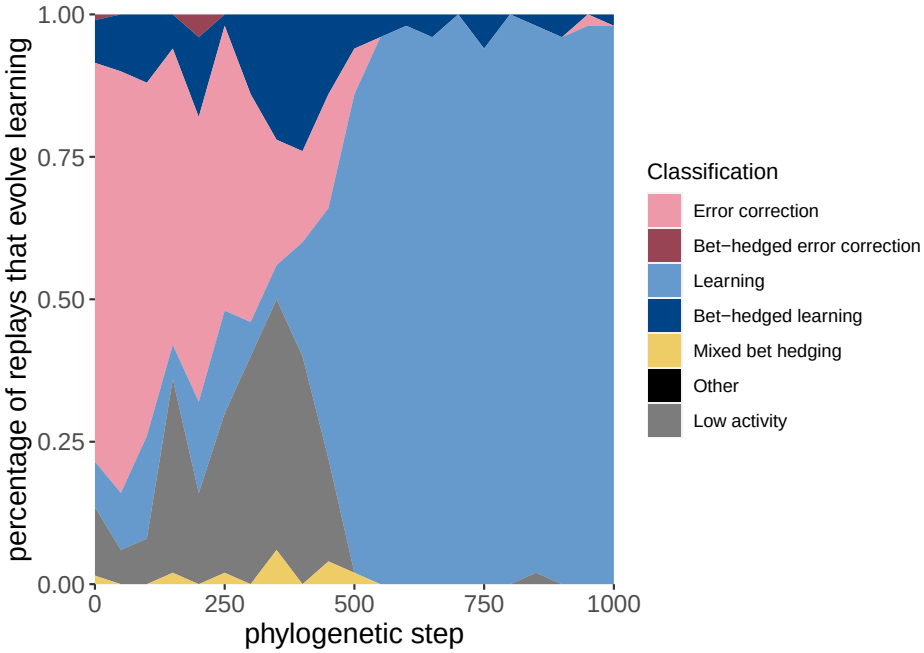
These plots are included as inspiration for future work.

```
mask_focal = classification_summary$depth %in% potentiation_window_start:potentiation_w
mask_initial = classification_summary$depth %% 50 == 0
ggplot(classification_summary[mask_initial,], aes(x=depth, color = seed_classification_
  geom_line(mapping=aes(y = frac), size=1.2) +
  geom_point(mapping=aes(y = frac), size = 2) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,1)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```



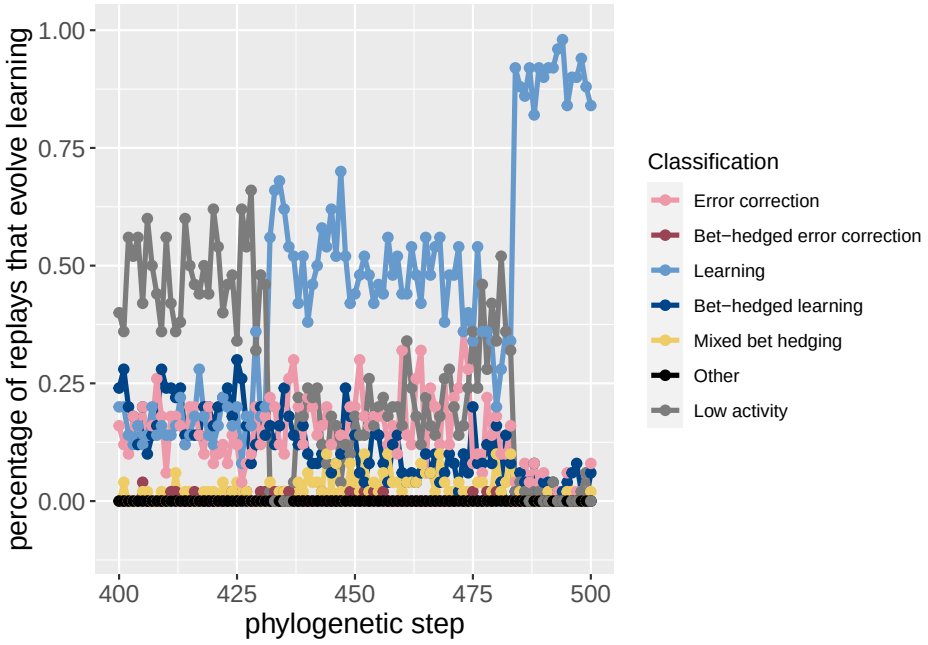
Here is the same data, but plotted as an area plot:

```
ggplot(classification_summary[mask_initial,], aes(x = depth, y = frac, fill = seed_classificatior
  geom_area() +
  scale_fill_manual(values = color_map) +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```

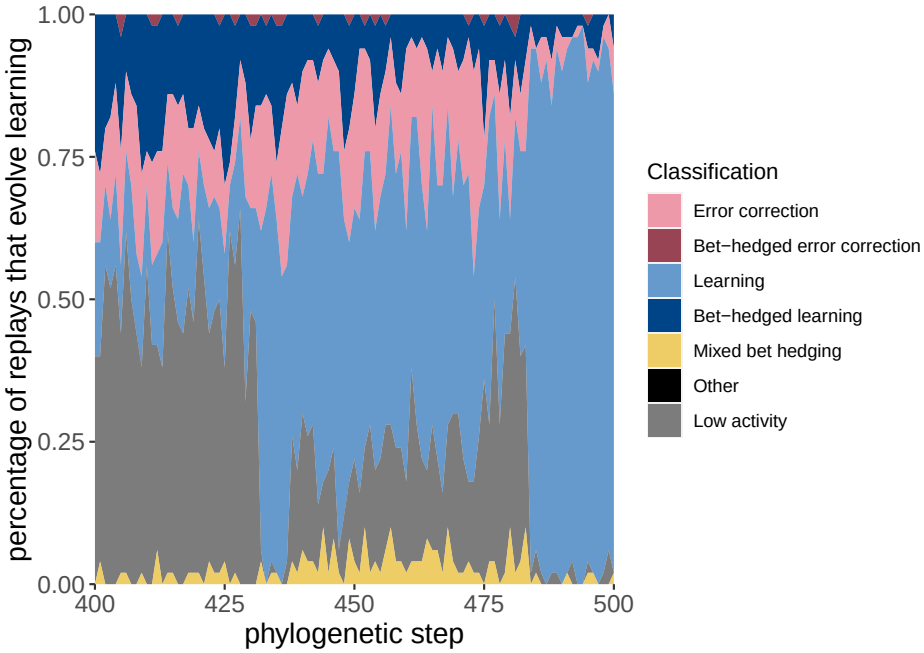


Here are the same two plots showing the potentiation window and the targeted replays:

```
ggplot(classification_summary[mask_focal,], aes(x=depth, color = seed_classification_fa
  geom_line(mapping=aes(y = frac), size=1.2) +
  geom_point(mapping=aes(y = frac), size = 2) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,1)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```




```
ggplot(classification_summary[mask_focal,], aes(x = depth, y = frac, fill = seed_classification_f
  geom_area() +
  scale_fill_manual(values = color_map) +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```


1.00
0.75
0.50
0.25
0.00
percentage of replays that evolve learning
400
425
450
475
500
phylogenetic step
Classification
Error correction
Bet−hedged error correction
Learning
Bet−hedged learning
Mixed bet hedging
Other
Low activity

Chapter 4

Lineage B (Seed 4)

Here we plot more details of Lineage B than would fit in the paper. If you are looking through the experiment files, Lineage B was seed 4. All data for Lineage B can be found in the `data/reps/4/` directory.

4.1 Dependencies

```
rm(list = ls())
# External
library(ggplot2)
library(dplyr)
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce9

# Configuration variables
base_repo_dir = '../..'
exp_dir = paste0(base_repo_dir, '/experiments/alife_2023/2023_02_21_01__initial_10_cooldown')

seed = 4
potentiation_window_start = 50
potentiation_window_stop = 150
potentiation_target = 104

# Internal
source(paste0(base_repo_dir, '/global_shared_files/constant_vars__three_cues_one_set.R'))
source(paste0(base_repo_dir, '/global_shared_files/shared_funcs__three_cues_one_set.R'))
```

4.2 Loading data

To save time, we load the pre-processed data.

To see how the data gets processed, please see `./analysis/replay_analysis.R` from the experiment directory.

```
# Data for all trials from all replay replicates
df = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/processed_data/processed_
# One line per replay replicate that summarizes all 100 trials
df_summary = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/processed_data/pro
# Sumamrize each category, e.g., how many replicates evolved error correction?
classification_summary = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/proces
# Also grab lineage data
df_lineage = read.csv(paste0(exp_dir, '/data/reps/', seed, '/dominant_lineage_summary.c
# Because lineage data was collected on the cluster before we renamed a few classifica
df_lineage[df_lineage$seed_classification == 'Bet-hedged imprinting',]$seed_classificat
df_lineage[df_lineage$seed_classification == 'Bet-hedged imprinting',]$seed_classificat
df_lineage[df_lineage$seed_classification == 'Small',]$seed_classification_factor = see
df_lineage[df_lineage$seed_classification == 'Small',]$seed_classification = seed_class
```

4.3 Lineage data

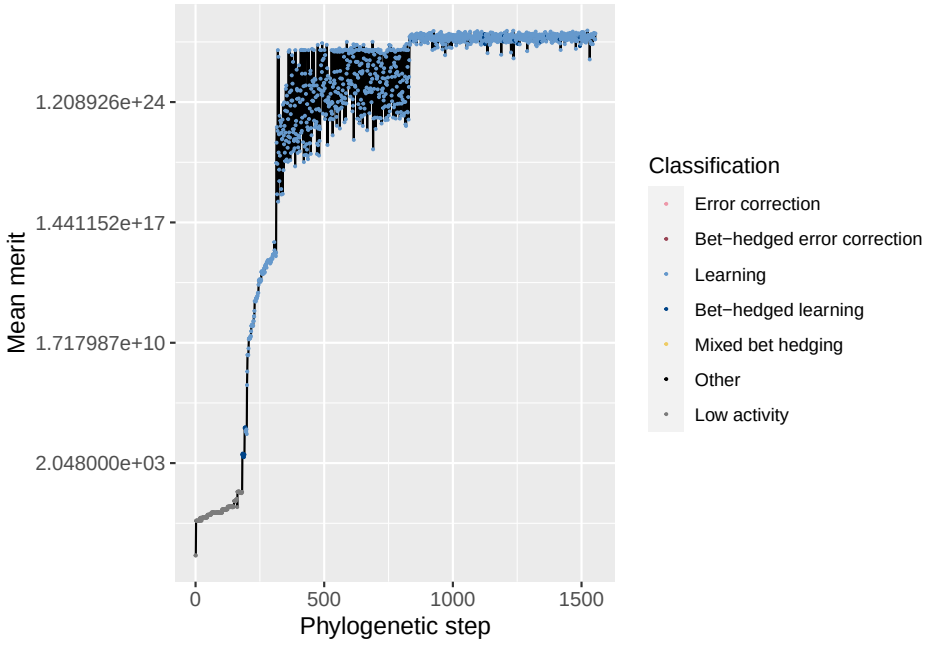
While not shown in the paper due to space limitations, it is important to look at the initial replicate's lineage and how it changes over time.

The code for these plots is copied from the `single_seed_lineage_analysis.R` script.

4.3.1 Merit

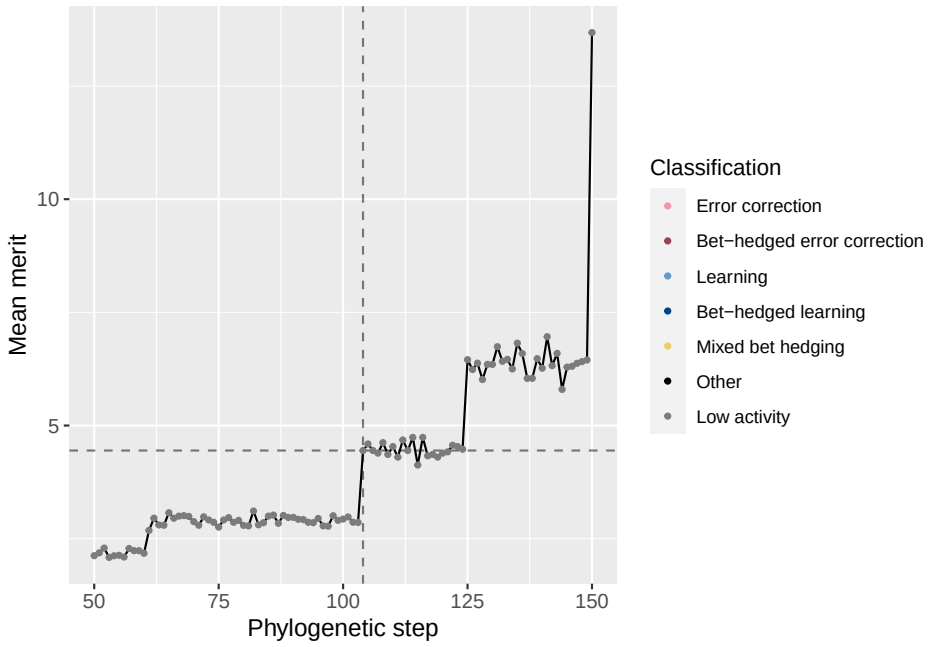
First, we look at the lineage as a whole and how average merit changes over time (note the y-axis is logarithmic)

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = merit_mean)) +
  geom_point(aes(y = merit_mean, color = seed_classification), size = 0.25) +
  scale_y_continuous(trans = 'log2') +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```



Additionally, we can zoom in to view just the potentiation window. The vertical dashed line shows the main potentiating step, step 104, while the horizontal dashed line shows average merit at that step for easier comparisons. Note that this plot is *not* logarithmic.

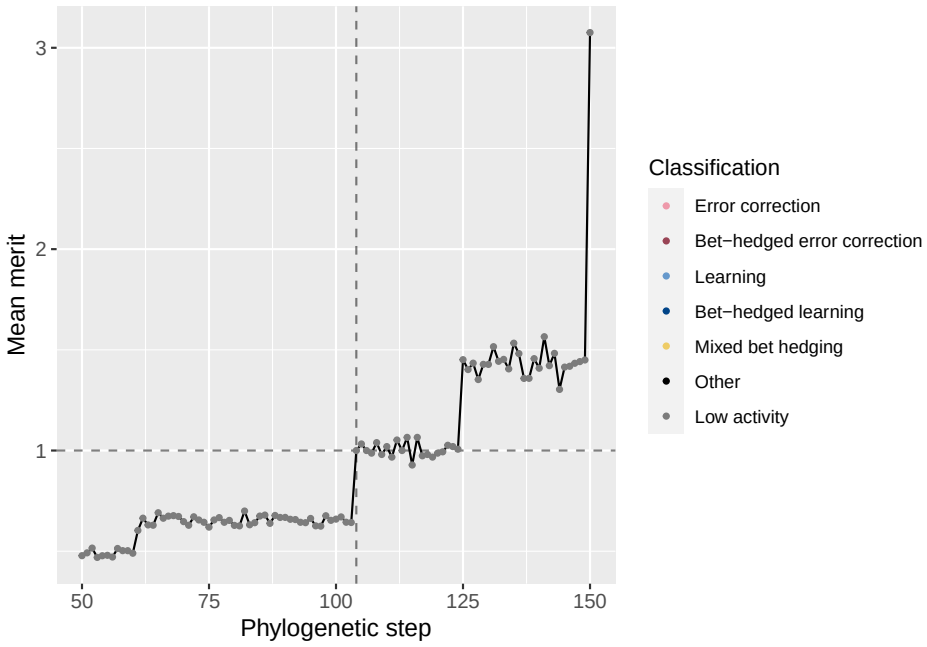
```
potentiation_mask = df_lineage$depth >= potentiation_window_start & df_lineage$depth <= potentiat
target_step_merit_mean = df_lineage[df_lineage$depth == potentiation_target,]$merit_mean
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_merit_mean), linetype = 'dashed', alpha = 0.5) +
  geom_line(aes(y = merit_mean)) +
  geom_point(aes(y = merit_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```



We see a small increase in merit at the potentiation mutation.

We can also normalize the merit based on the potentiating step. Remember that one additional correct state confers a merit bonus of 10%. We see this jump is larger than 10%.

```
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = 1), linetype = 'dashed', alpha = 0.5) +
  geom_line(aes(y = merit_mean / target_step_merit_mean)) +
  geom_point(aes(y = merit_mean / target_step_merit_mean, color = seed_classification)
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```

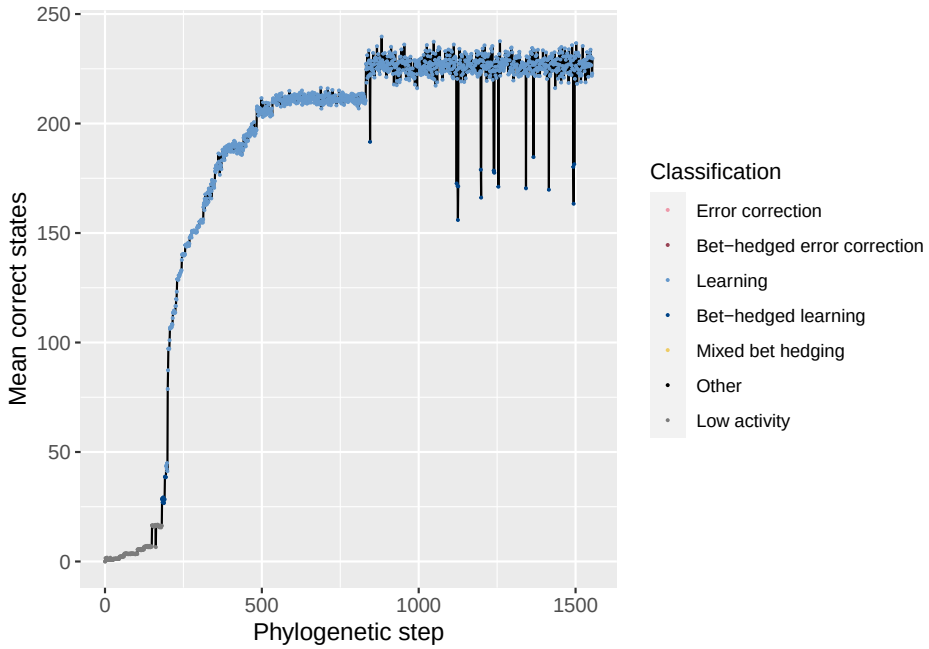


4.3.2 Correct states

Similarly, we can look at the number of correct states both overall and in the potentiation window:

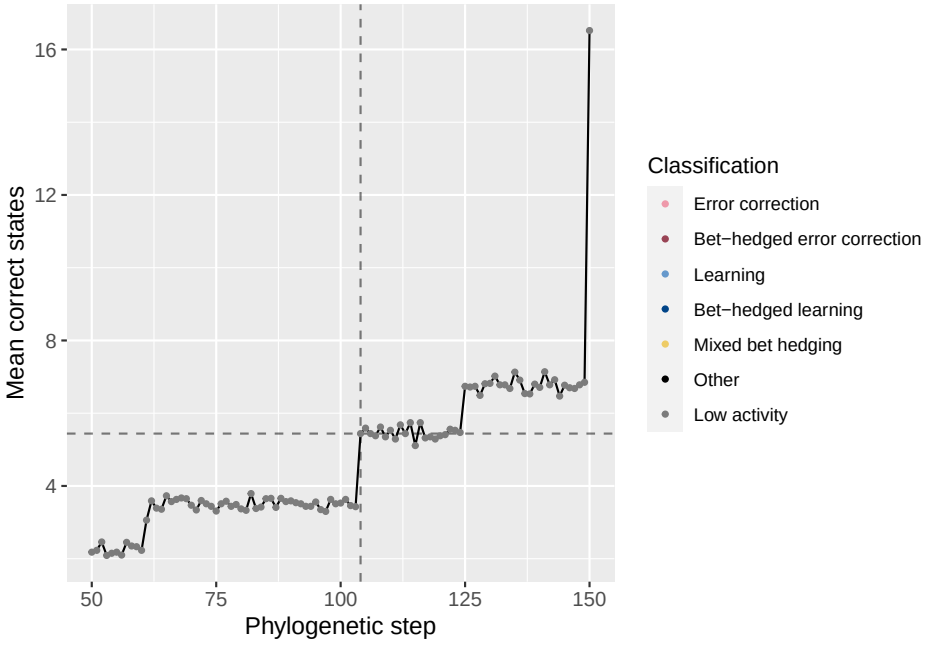
Overall:

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = correct_doors_mean)) +
  geom_point(aes(y = correct_doors_mean, color = seed_classification), size = 0.25) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean correct states') +
  xlab('Phylogenetic step')
```



Potentiation window (dashed lines show the main potentiating mutation):

```
target_step_correct_doors_mean = df_lineage[df_lineage$depth == potentiation_target,]$c
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_correct_doors_mean), linetype = 'dashed', alp
  geom_line(aes(y = correct_doors_mean)) +
  geom_point(aes(y = correct_doors_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean correct states') +
  xlab('Phylogenetic step')
```



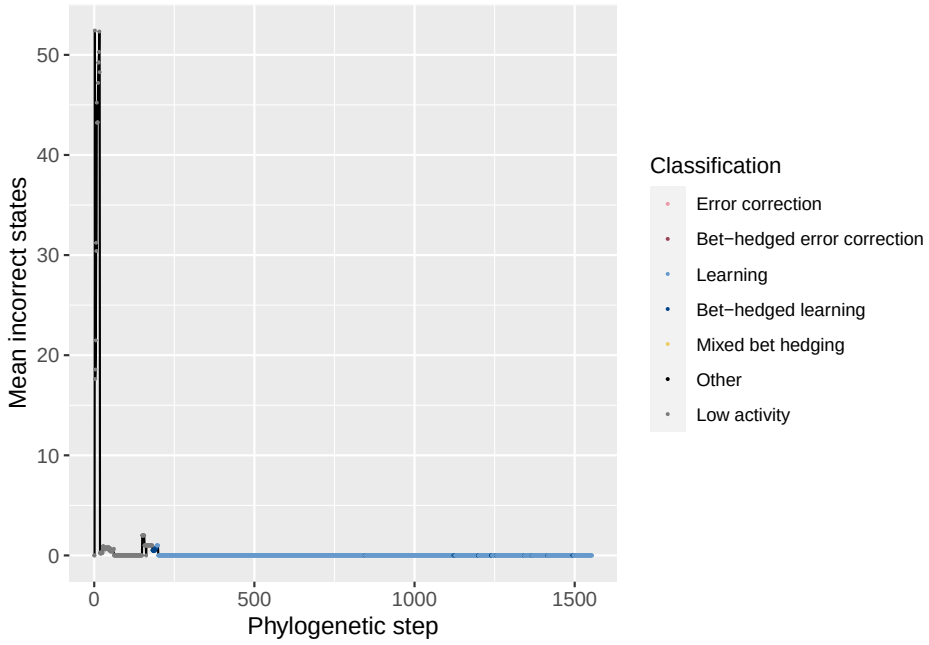
We see also an increase in correct states at the potentiating mutation.

4.3.3 Incorrect states

Just like our analysis of correct states, we can also look at the average *incorrect* states over time. By incorrect states, we mean the wrong movements while in a *left*, *right*, or *forward* state. Incorrect movements from a *backward* state are tracked separately.

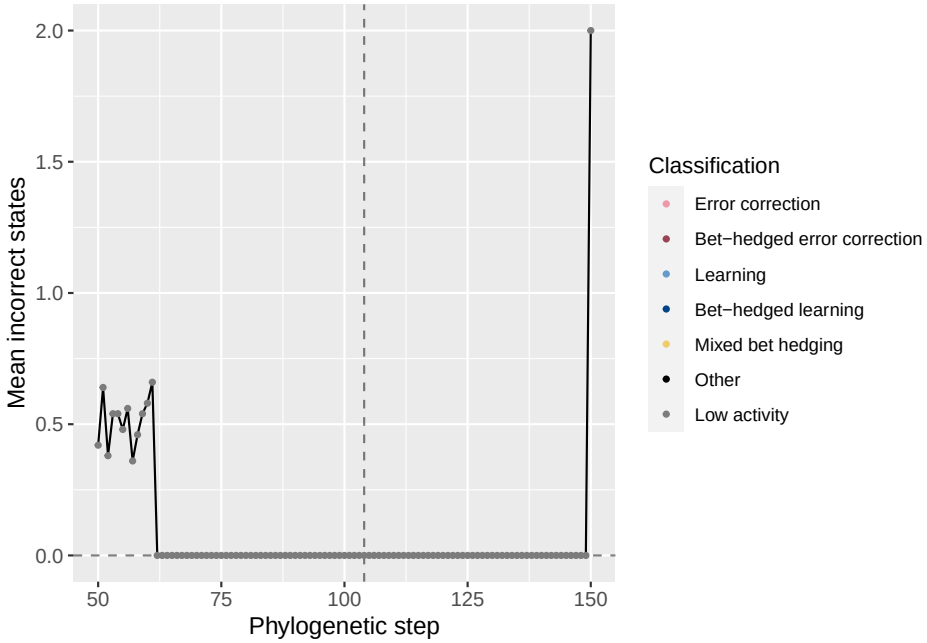
Overall:

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = incorrect_doors_mean)) +
  geom_point(aes(y = incorrect_doors_mean, color = seed_classification), size = 0.25) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean incorrect states') +
  xlab('Phylogenetic step')
```



Potentiation window:

```
target_step_incorrect_doors_mean = df_lineage[df_lineage$depth == potentiation_target,]
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_incorrect_doors_mean), linetype = 'dashed', a
  geom_line(aes(y = incorrect_doors_mean)) +
  geom_point(aes(y = incorrect_doors_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean incorrect states') +
  xlab('Phylogenetic step')
```



We see that the potentiating mutation was able to increase the correct states without increasing the number of incorrect states.

4.4 Replay data

Now that we know what the lineage looks like, we can begin to break down the replay data.

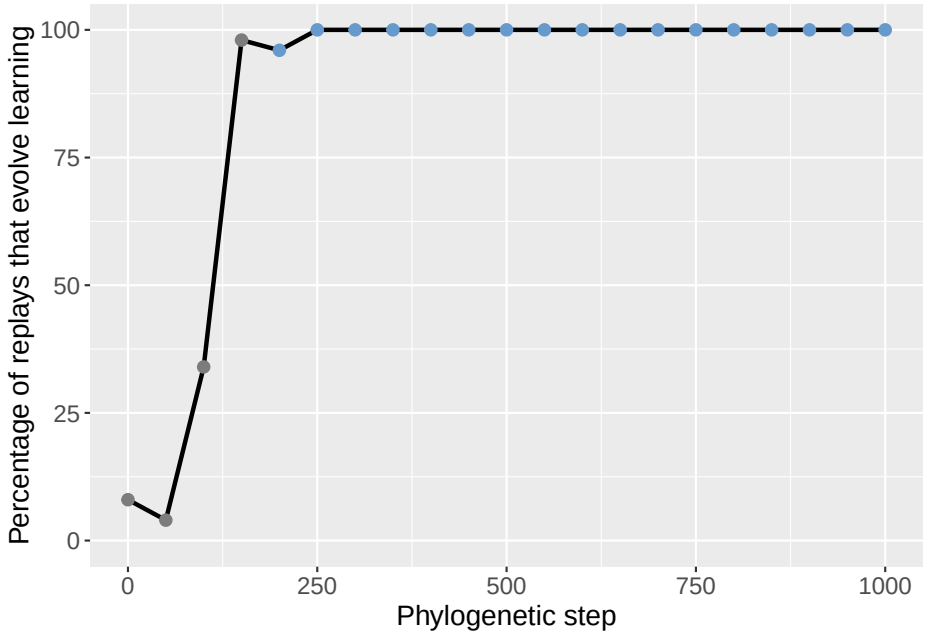
4.4.1 Exploratory replays

We started by seeding 50 replay replicates for every 50th step along the lineage in the window [50, 1000]. The data for step 0 comes from the initial 200 replicates.

Here we show the potentiation over time for those exploratory replays:

```
df_learning_summary = classification_summary[classification_summary$seed_classification == seed_c
df_learning_summary$lineage_classification = NA
# Assign each depth its classification from the lineage data
for(depth in unique(df_learning_summary$depth)){
  df_learning_summary[df_learning_summary$depth == depth,]$lineage_classification = df_lineage[df
}
# Create masks
mask_learning_summary_focal = df_learning_summary$depth %in% potentiation_window_start:potentiati
mask_learning_summary_initial = df_learning_summary$depth %% 50 == 0
```

```
ggplot(df_learning_summary[mask_learning_summary_initial,], aes(x=depth)) +
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```

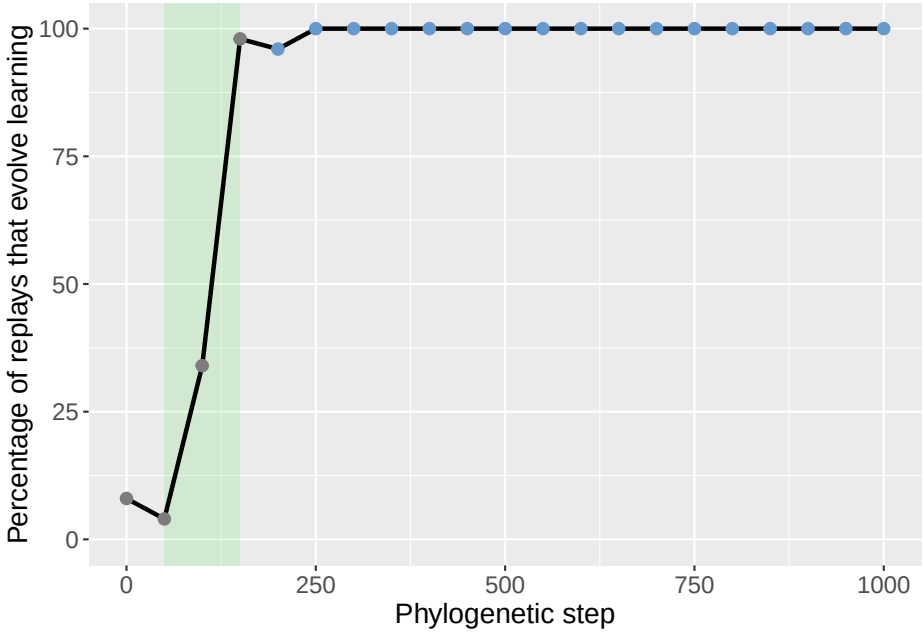


We can then identify the potentiation window based on the points with the most potentiation gain.

Here we selected steps 50-150 (green shaded region):

```
ggplot(df_learning_summary[mask_learning_summary_initial,], aes(x=depth)) +
  annotate("rect", xmin=potentiation_window_start, xmax=potentiation_window_stop, ymin=
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
```

```
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```



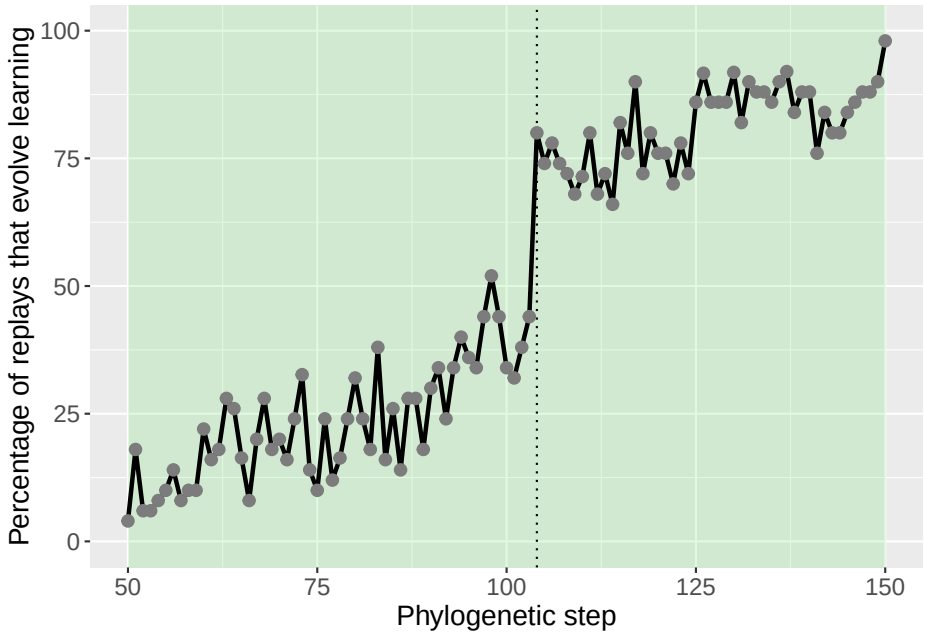
4.4.2 Targeted replays

With the potentiation window identified, we seeded targeted replays.

For the targeted replays, we seeded 50 replay replicates from *every* step in the potetiation window. This gives us an idea of how individual steps in the phylogney affected potentiation.

```
ggplot(df_learning_summary[mask_learning_summary_focal,], aes(x=depth)) +
  annotate("rect", xmin=potentiation_window_start, xmax=potentiation_window_stop, ymin=-Inf, ymax
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dotted') +
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size = 2.5) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
```

```
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```



This plot was extensively discussed in the paper, so we refrain from additional discussion here.

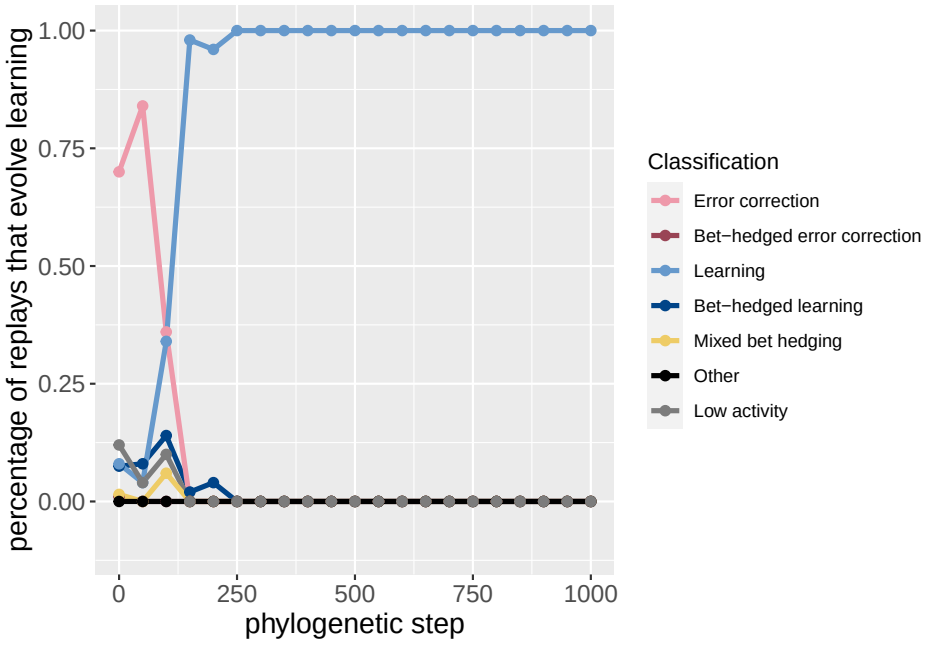
4.4.3 Potentiation of other behaviors

While the paper mostly focused on learning, it can be useful to see how the potentiation of other behaviors changes before learning takes over.

These plots are included as inspiration for future work.

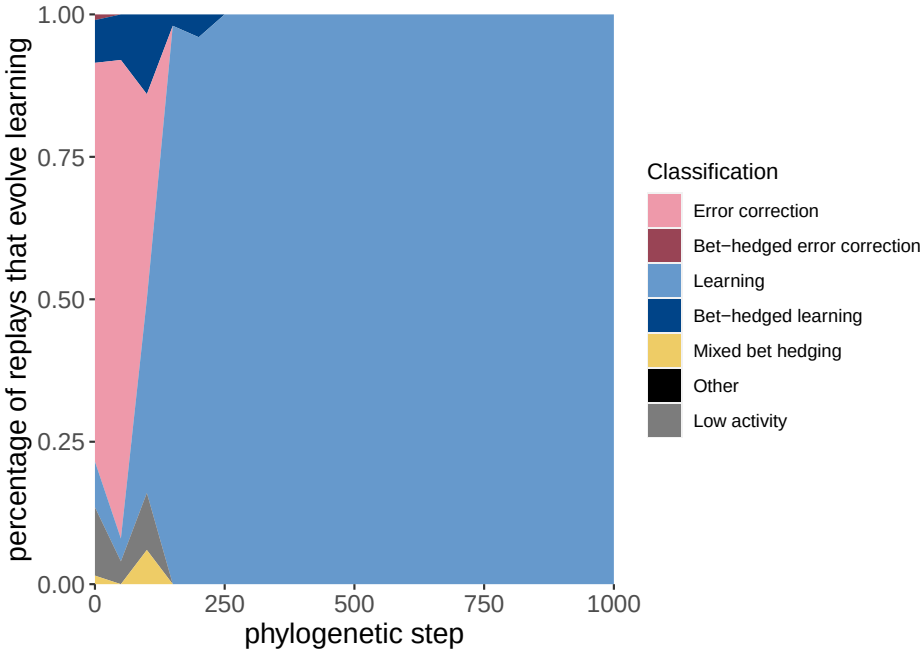
```
mask_focal = classification_summary$depth %in% potentiation_window_start:potentiation_w
mask_initial = classification_summary$depth %% 50 == 0
ggplot(classification_summary[mask_initial,], aes(x=depth, color = seed_classification_
  geom_line(mapping=aes(y = frac), size=1.2) +
  geom_point(mapping=aes(y = frac), size = 2) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,1)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
```

```
  labs(color = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```



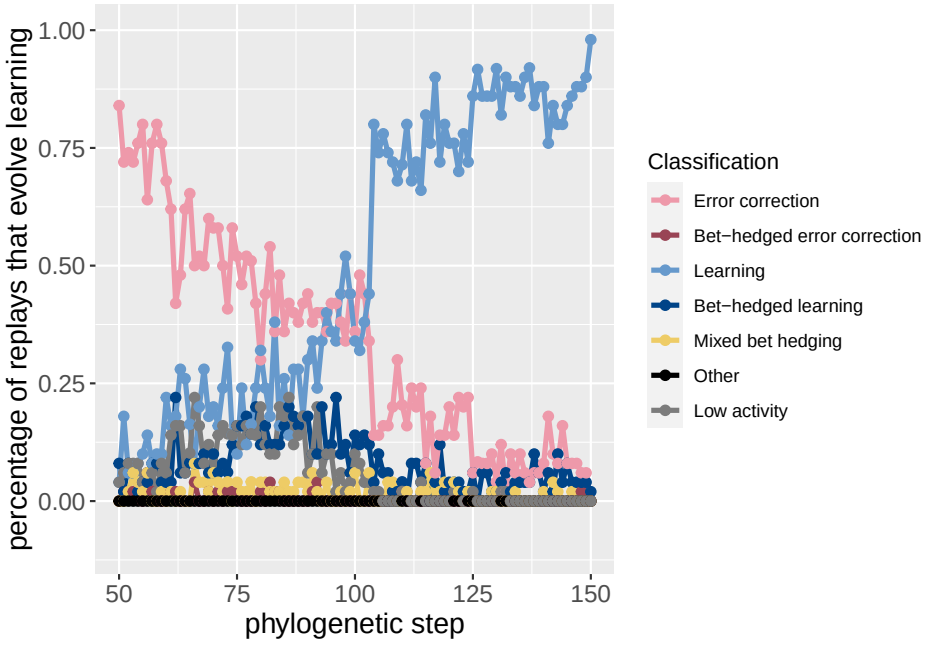
Here is the same data, but plotted as an area plot:

```
ggplot(classification_summary[mask_initial,], aes(x = depth, y = frac, fill = seed_classificatior
  geom_area() +
  scale_fill_manual(values = color_map) +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```

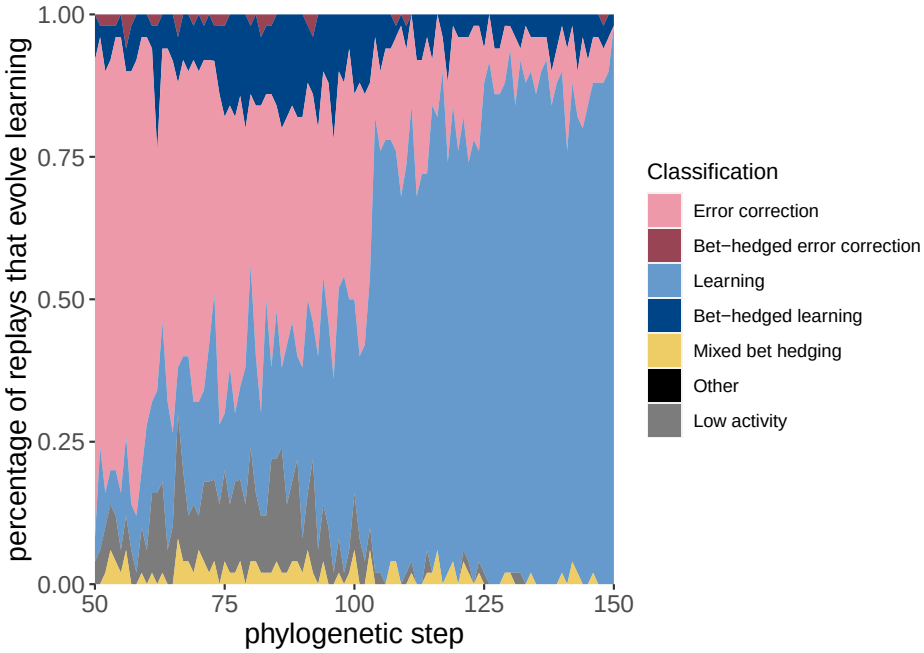


Here are the same two plots showing the potentiation window and the targeted replays:

```
ggplot(classification_summary[mask_focal,], aes(x=depth, color = seed_classification_fa
  geom_line(mapping=aes(y = frac), size=1.2) +
  geom_point(mapping=aes(y = frac), size = 2) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,1)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```




```
ggplot(classification_summary[mask_focal,], aes(x = depth, y = frac, fill = seed_classification_f
  geom_area() +
  scale_fill_manual(values = color_map) +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```


1.00
0.75
0.50
0.25
0.00
percentage of replays that evolve learning
50
75
100
125
150
phylogenetic step
Classification
Error correction
Bet−hedged error correction
Learning
Bet−hedged learning
Mixed bet hedging
Other
Low activity

Chapter 5

Lineage C (Seed 15)

Here we plot more details of Lineage C than would fit in the paper. If you are looking through the experiment files, Lineage C was seed 15. All data for Lineage C can be found in the `data/reps/15/` directory.

5.1 Dependencies

```
rm(list = ls())
# External
library(ggplot2)
library(dplyr)
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce9

# Configuration variables
base_repo_dir = '../..'
exp_dir = paste0(base_repo_dir, '/experiments/alife_2023/2023_02_21_01__initial_10_cooldown')

seed = 15
potentiation_window_start = 250
potentiation_window_stop = 300
potentiation_target = 279

# Internal
source(paste0(base_repo_dir, '/global_shared_files/constant_vars__three_cues_one_set.R'))
source(paste0(base_repo_dir, '/global_shared_files/shared_funcs__three_cues_one_set.R'))
```

5.2 Loading data

To save time, we load the pre-processed data.

To see how the data gets processed, please see `./analysis/replay_analysis.R` from the experiment directory.

```
# Data for all trials from all replay replicates
df = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/processed_data/processed_
# One line per replay replicate that summarizes all 100 trials
df_summary = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/processed_data/pro
# Sumamrize each category, e.g., how many replicates evolved error correction?
classification_summary = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/proces
# Also grab lineage data
df_lineage = read.csv(paste0(exp_dir, '/data/reps/', seed, '/dominant_lineage_summary.c
# Because lineage data was collected on the cluster before we renamed a few classifica
df_lineage[df_lineage$seed_classification == 'Bet-hedged imprinting',]$seed_classificat
df_lineage[df_lineage$seed_classification == 'Bet-hedged imprinting',]$seed_classificat
df_lineage[df_lineage$seed_classification == 'Small',]$seed_classification_factor = see
df_lineage[df_lineage$seed_classification == 'Small',]$seed_classification = seed_class
```

5.3 Lineage data

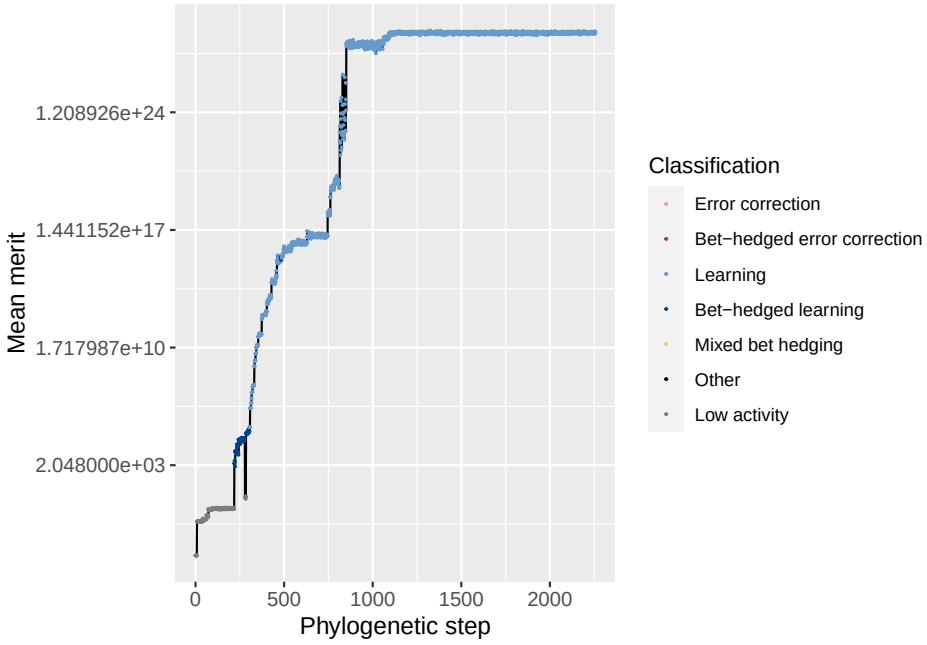
While not shown in the paper due to space limitations, it is important to look at the initial replicate's lineage and how it changes over time.

The code for these plots is copied from the `single_seed_lineage_analysis.R` script.

5.3.1 Merit

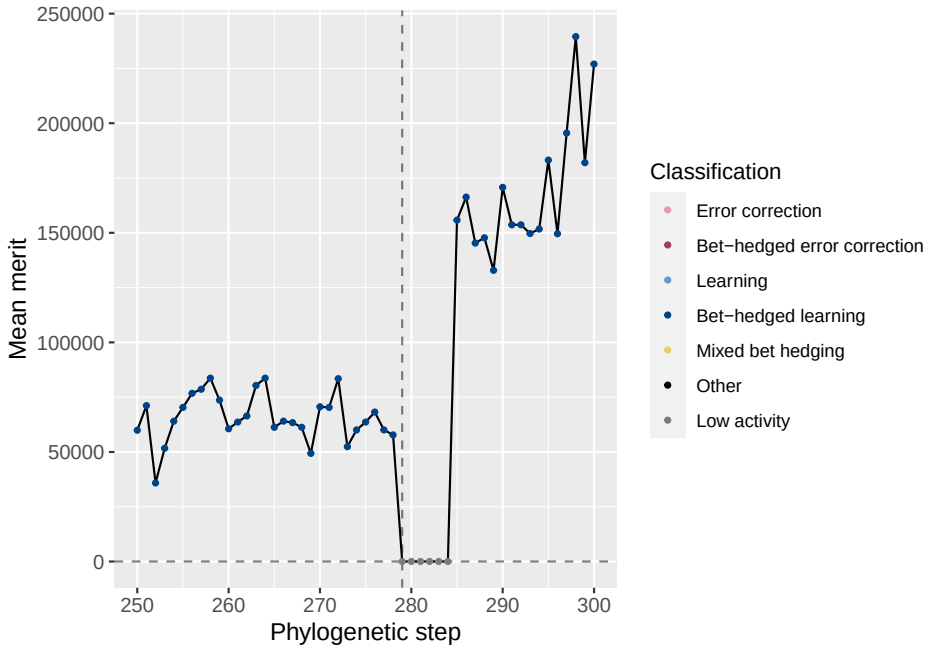
First, we look at the lineage as a whole and how average merit changes over time (note the y-axis is logarithmic)

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = merit_mean)) +
  geom_point(aes(y = merit_mean, color = seed_classification), size = 0.25) +
  scale_y_continuous(trans = 'log2') +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```



Additionally, we can zoom in to view just the potentiation window. The vertical dashed line shows the main potentiating step, step 279, while the horizontal dashed line shows average merit at that step for easier comparisons. Note that this plot is *not* logarithmic.

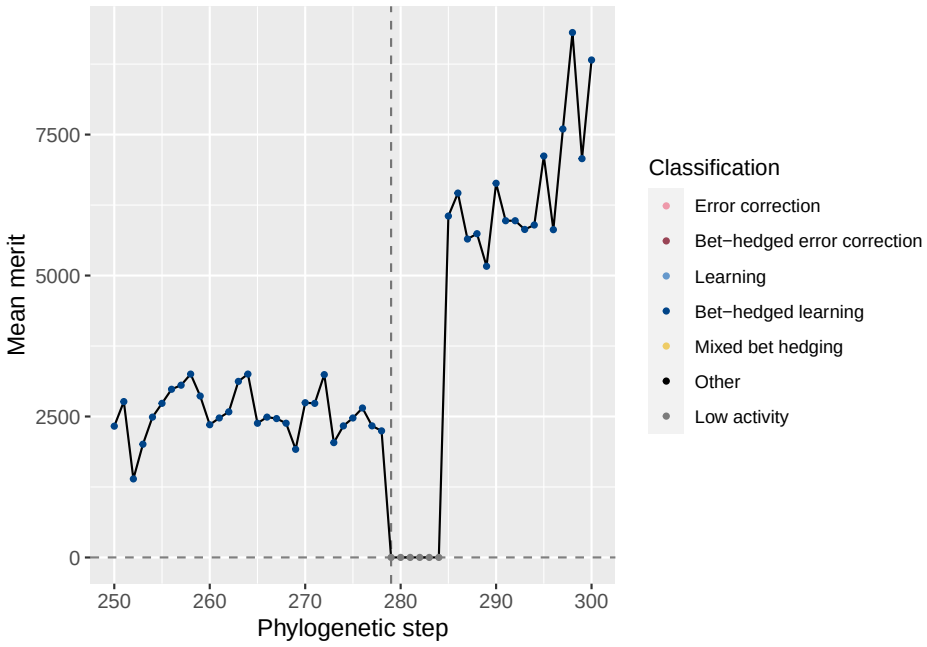
```
potentiation_mask = df_lineage$depth >= potentiation_window_start & df_lineage$depth <= potentiat
target_step_merit_mean = df_lineage[df_lineage$depth == potentiation_target,]$merit_mean
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_merit_mean), linetype = 'dashed', alpha = 0.5) +
  geom_line(aes(y = merit_mean)) +
  geom_point(aes(y = merit_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```



We see a *drastic* drop in merit at the potentiating mutation. Merit does increase to an even higher point a few steps after the potentiating mutation, however.

We can also normalize the merit based on the potentiating step. Remember that one additional correct state confers a merit bonus of 10%. Here, we can see that merit was over 2,000 times higher before the potentiating mutation.

```
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = 1), linetype = 'dashed', alpha = 0.5) +
  geom_line(aes(y = merit_mean / target_step_merit_mean)) +
  geom_point(aes(y = merit_mean / target_step_merit_mean, color = seed_classification)
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```

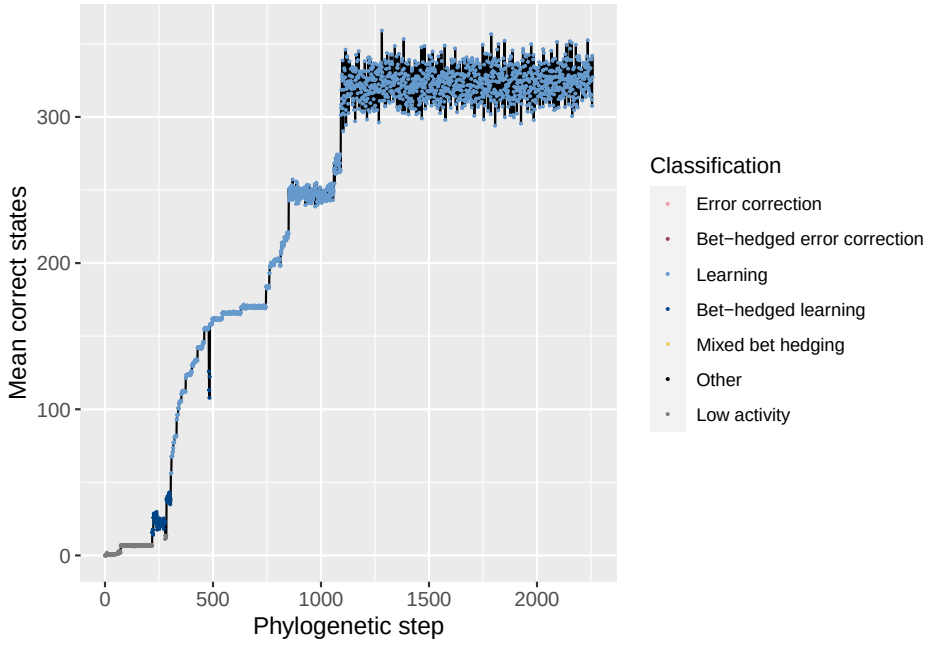


5.3.2 Correct states

Similarly, we can look at the number of correct states both overall and in the potentiation window:

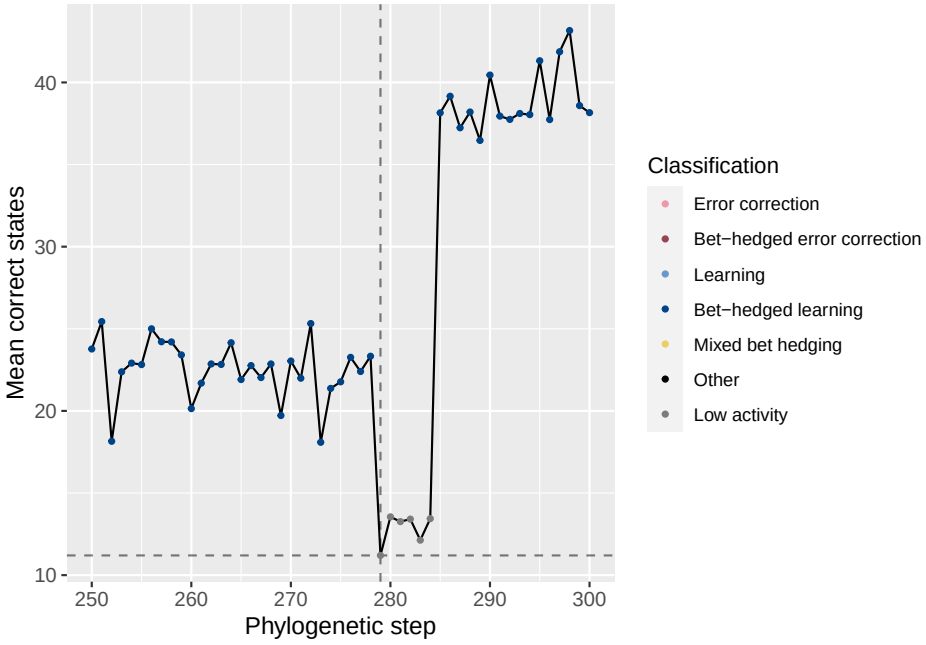
Overall:

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = correct_doors_mean)) +
  geom_point(aes(y = correct_doors_mean, color = seed_classification), size = 0.25) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean correct states') +
  xlab('Phylogenetic step')
```



Potentiation window (dashed lines show the main potentiating mutation):

```
target_step_correct_doors_mean = df_lineage[df_lineage$depth == potentiation_target,]$c
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_correct_doors_mean), linetype = 'dashed', alp
  geom_line(aes(y = correct_doors_mean)) +
  geom_point(aes(y = correct_doors_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean correct states') +
  xlab('Phylogenetic step')
```



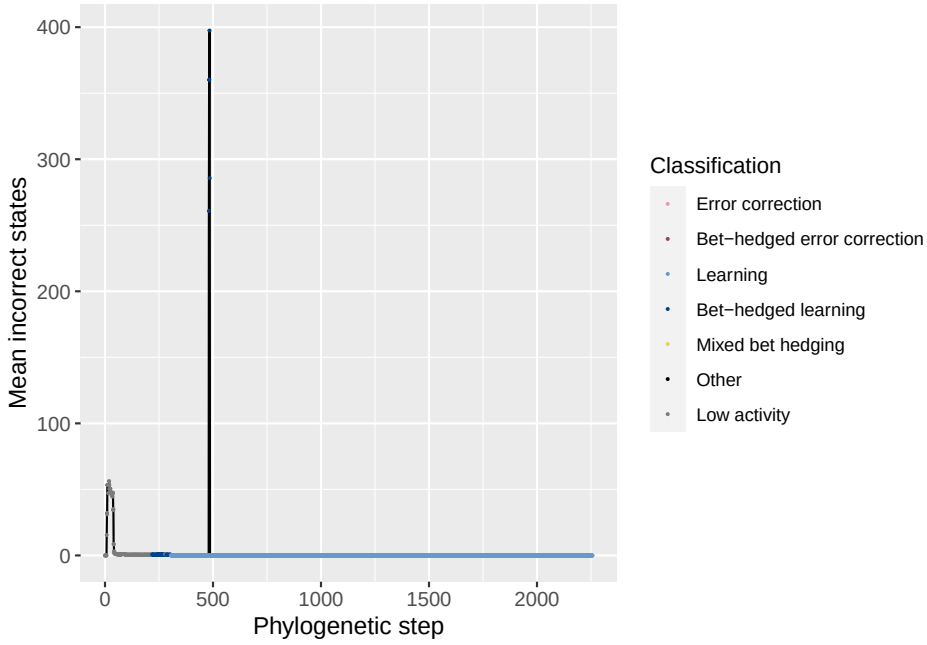
We see a noticable drop in correct states at the potentiating mutation, and we also see the increase in correct states a few steps later.

5.3.3 Incorrect states

Just like our analysis of correct states, we can also look at the average *incorrect* states over time. By incorrect states, we mean the wrong movements while in a *left*, *right*, or *forward* state. Incorrect movements from a *backward* state are tracked separately.

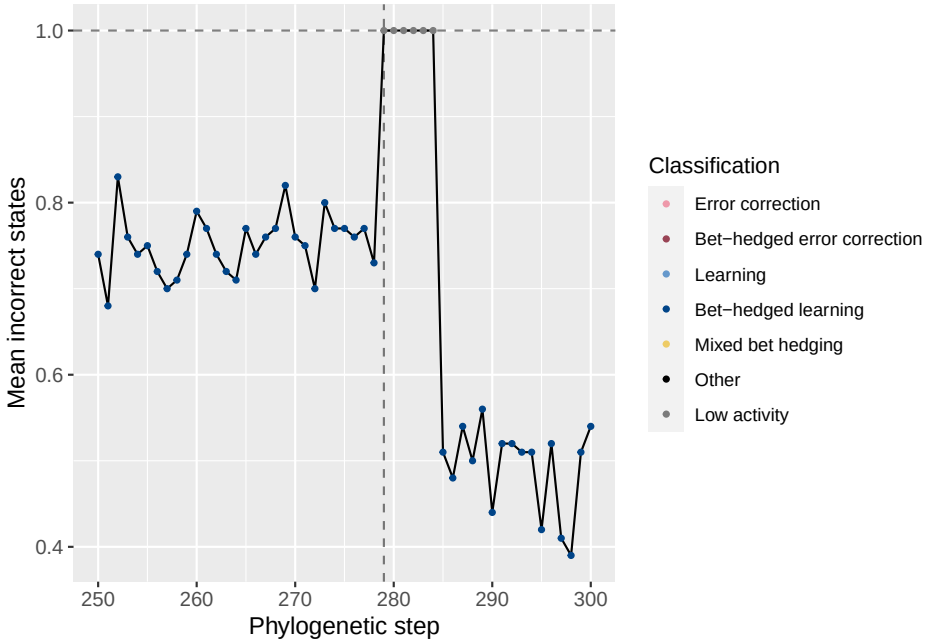
Overall:

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = incorrect_doors_mean)) +
  geom_point(aes(y = incorrect_doors_mean, color = seed_classification), size = 0.25) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean incorrect states') +
  xlab('Phylogenetic step')
```



Potentiation window:

```
target_step_incorrect_doors_mean = df_lineage[df_lineage$depth == potentiation_target,]
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_incorrect_doors_mean), linetype = 'dashed', a
  geom_line(aes(y = incorrect_doors_mean)) +
  geom_point(aes(y = incorrect_doors_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean incorrect states') +
  xlab('Phylogenetic step')
```



We see an increase in incorrect states at the potentiating mutation, but only to an average of 1.

5.4 Replay data

Now that we know what the lineage looks like, we can begin to break down the replay data.

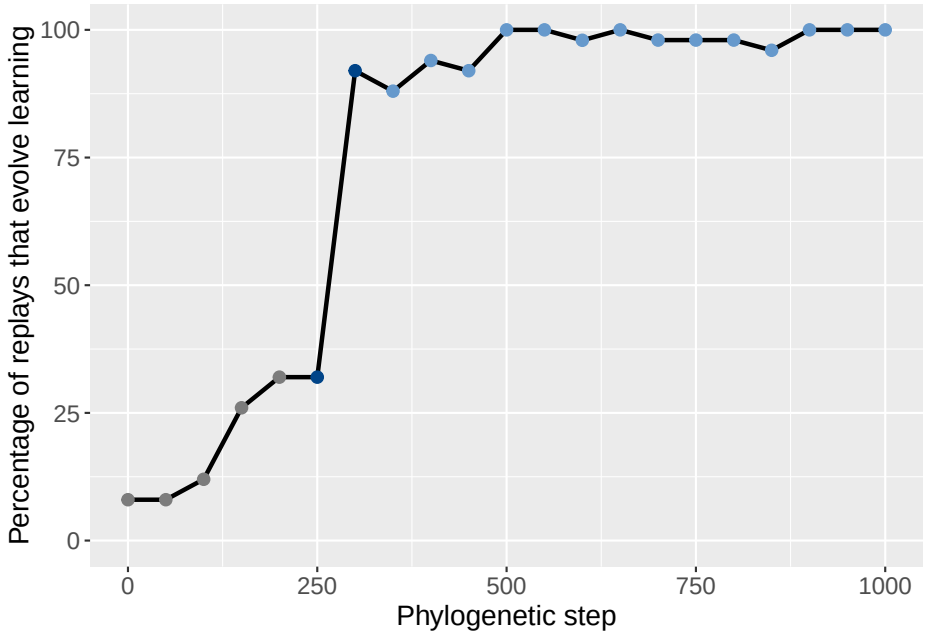
5.4.1 Exploratory replays

We started by seeding 50 replay replicates for every 50th step along the lineage in the window [50, 1000]. The data for step 0 comes from the initial 200 replicates.

Here we show the potentiation over time for those exploratory replays:

```
df_learning_summary = classification_summary[classification_summary$seed_classification == seed_c
df_learning_summary$lineage_classification = NA
# Assign each depth its classification from the lineage data
for(depth in unique(df_learning_summary$depth)){
  df_learning_summary[df_learning_summary$depth == depth,]$lineage_classification = df_lineage[df
}
# Create masks
mask_learning_summary_focal = df_learning_summary$depth %in% potentiation_window_start:potentiati
mask_learning_summary_initial = df_learning_summary$depth %% 50 == 0
```

```
ggplot(df_learning_summary[mask_learning_summary_initial,], aes(x=depth)) +
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```

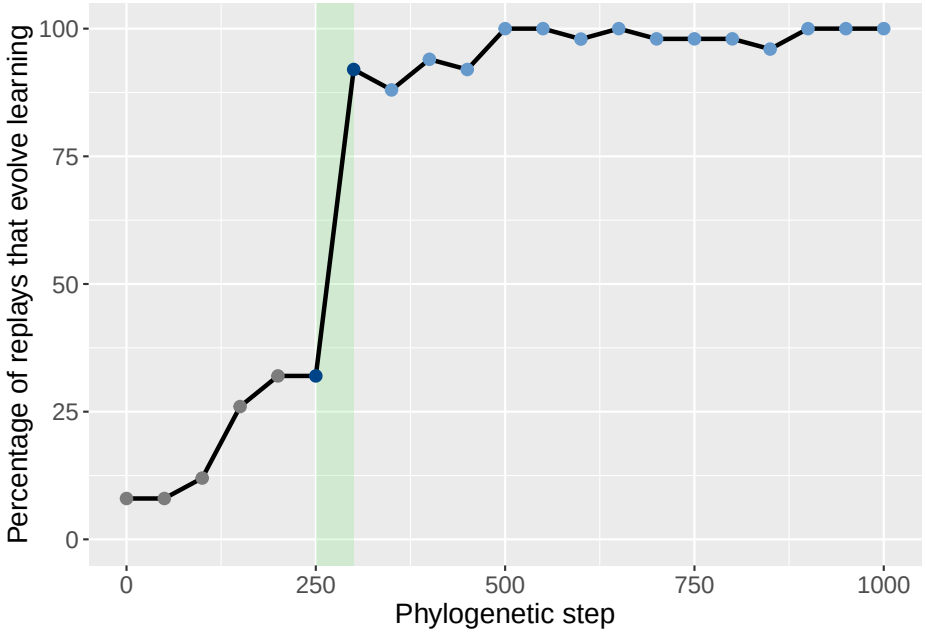


We can then identify the potentiation window based on the points with the most potentiation gain.

Here we selected steps 250-300 (green shaded region):

```
ggplot(df_learning_summary[mask_learning_summary_initial,], aes(x=depth)) +
  annotate("rect", xmin=potentiation_window_start, xmax=potentiation_window_stop, ymin=
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
```

```
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```



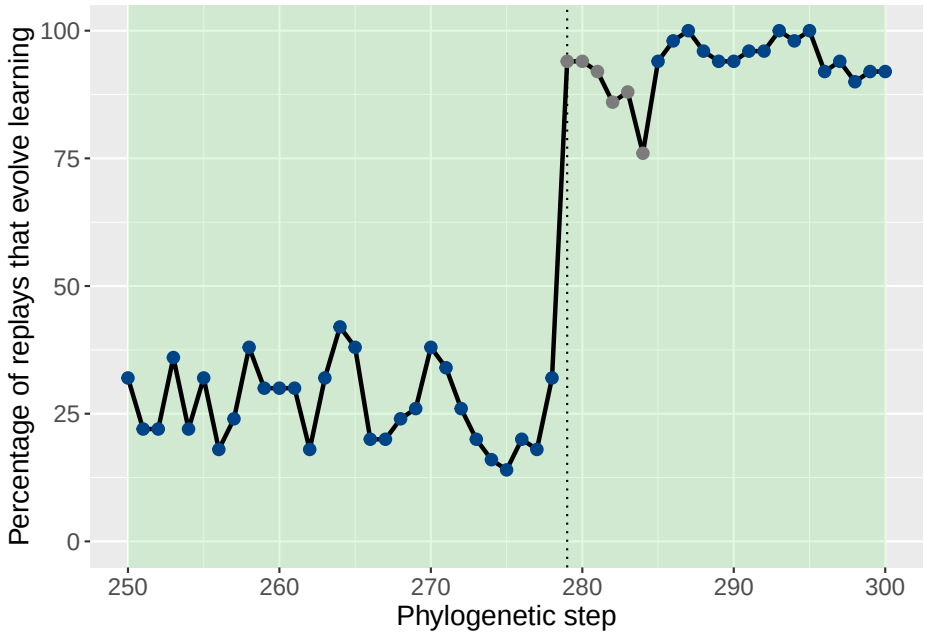
5.4.2 Targeted replays

With the potentiation window identified, we seeded targeted replays.

For the targeted replays, we seeded 50 replay replicates from *every* step in the potetiation window. This gives us an idea of how individual steps in the phylogney affected potentiation.

```
ggplot(df_learning_summary[mask_learning_summary_focal,], aes(x=depth)) +
  annotate("rect", xmin=potentiation_window_start, xmax=potentiation_window_stop, ymin=-Inf, ymax
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dotted') +
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size = 2.5) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
```

```
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```



This plot was extensively discussed in the paper, so we refrain from additional discussion here.

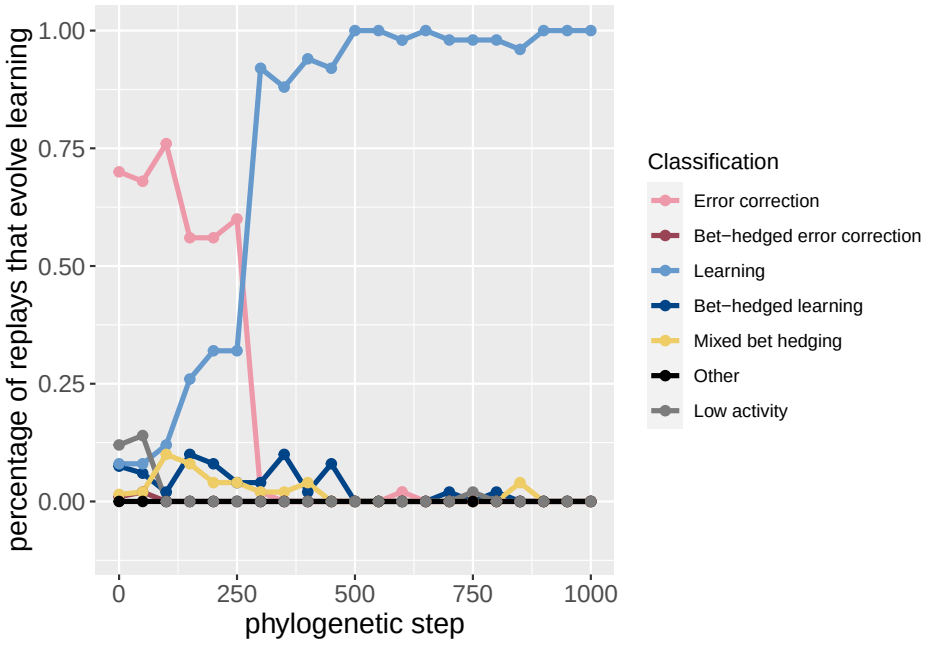
5.4.3 Potentiation of other behaviors

While the paper mostly focused on learning, it can be useful to see how the potentiation of other behaviors changes before learning takes over.

These plots are included as inspiration for future work.

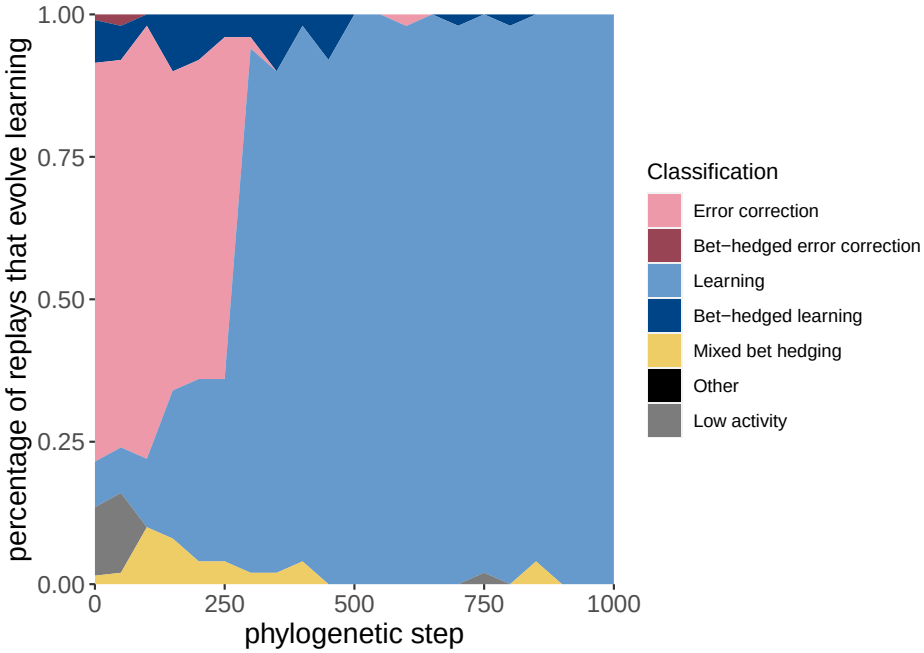
```
mask_focal = classification_summary$depth %in% potentiation_window_start:potentiation_w
mask_initial = classification_summary$depth %% 50 == 0
ggplot(classification_summary[mask_initial,], aes(x=depth, color = seed_classification_
  geom_line(mapping=aes(y = frac), size=1.2) +
  geom_point(mapping=aes(y = frac), size = 2) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,1)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
```

```
  labs(color = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```



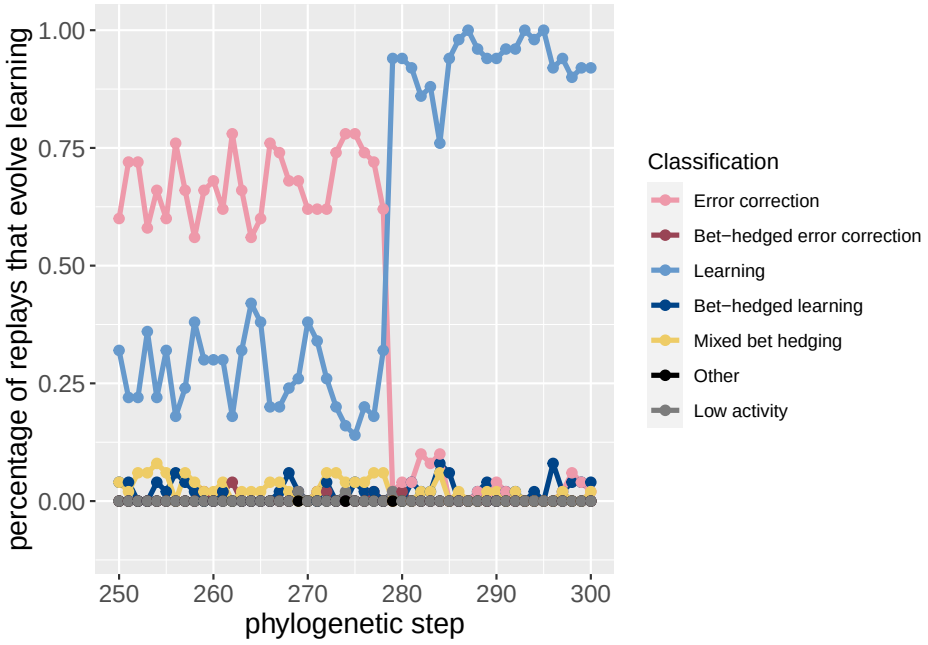
Here is the same data, but plotted as an area plot:

```
ggplot(classification_summary[mask_initial,], aes(x = depth, y = frac, fill = seed_classificatior
  geom_area() +
  scale_fill_manual(values = color_map) +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```

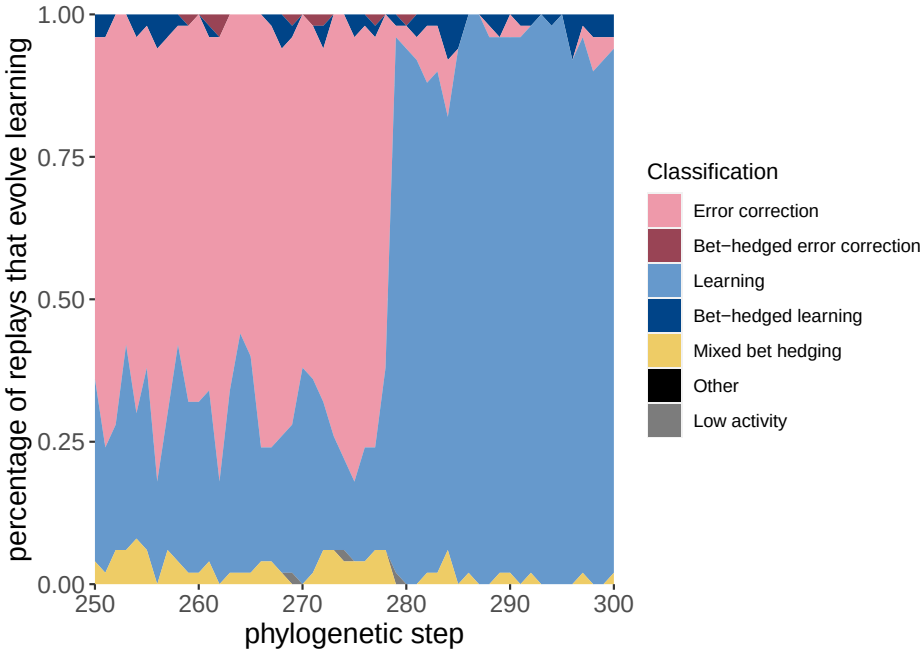


Here are the same two plots showing the potentiation window and the targeted replays:

```
ggplot(classification_summary[mask_focal,], aes(x=depth, color = seed_classification_fa
  geom_line(mapping=aes(y = frac), size=1.2) +
  geom_point(mapping=aes(y = frac), size = 2) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,1)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```




```
ggplot(classification_summary[mask_focal,], aes(x = depth, y = frac, fill = seed_classification_f
  geom_area() +
  scale_fill_manual(values = color_map) +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```


1.00
0.75
0.50
0.25
0.00
percentage of replays that evolve learning
250
260
270
280
290
300
phylogenetic step
Classification
Error correction
Bet−hedged error correction
Learning
Bet−hedged learning
Mixed bet hedging
Other
Low activity

Chapter 6

Lineage D (Seed 6)

Here we plot more details of Lineage D than would fit in the paper. If you are looking through the experiment files, Lineage D was seed 6. All data for Lineage D can be found in the `data/reps/6/` directory.

6.1 Dependencies

```
rm(list = ls())
# External
library(ggplot2)
library(dplyr)
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce9

# Configuration variables
base_repo_dir = '../..'
exp_dir = paste0(base_repo_dir, '/experiments/alife_2023/2023_02_21_01__initial_10_cooldown')

seed = 6
potentiation_window_start = 500
potentiation_window_stop = 550
potentiation_target = 548

# Internal
source(paste0(base_repo_dir, '/global_shared_files/constant_vars__three_cues_one_set.R'))
source(paste0(base_repo_dir, '/global_shared_files/shared_funcs__three_cues_one_set.R'))
```

6.2 Loading data

To save time, we load the pre-processed data.

To see how the data gets processed, please see `./analysis/replay_analysis.R` from the experiment directory.

```
# Data for all trials from all replay replicates
df = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/processed_data/processed_
# One line per replay replicate that summarizes all 100 trials
df_summary = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/processed_data/pro
# Sumamrize each category, e.g., how many replicates evolved error correction?
classification_summary = read.csv(paste0(exp_dir, '/data/reps/', seed, '/replays/proces
# Also grab lineage data
df_lineage = read.csv(paste0(exp_dir, '/data/reps/', seed, '/dominant_lineage_summary.c
# Because lineage data was collected on the cluster before we renamed a few classifica
# Note: no seeds were classified as bet-hedged learning, so we must skip these lines
#df_lineage[df_lineage$seed_classification == 'Bet-hedged imprinting',]$seed_classific
#df_lineage[df_lineage$seed_classification == 'Bet-hedged imprinting',]$seed_classific
df_lineage[df_lineage$seed_classification == 'Small',]$seed_classification_factor = see
df_lineage[df_lineage$seed_classification == 'Small',]$seed_classification = seed_class
```

6.3 Lineage data

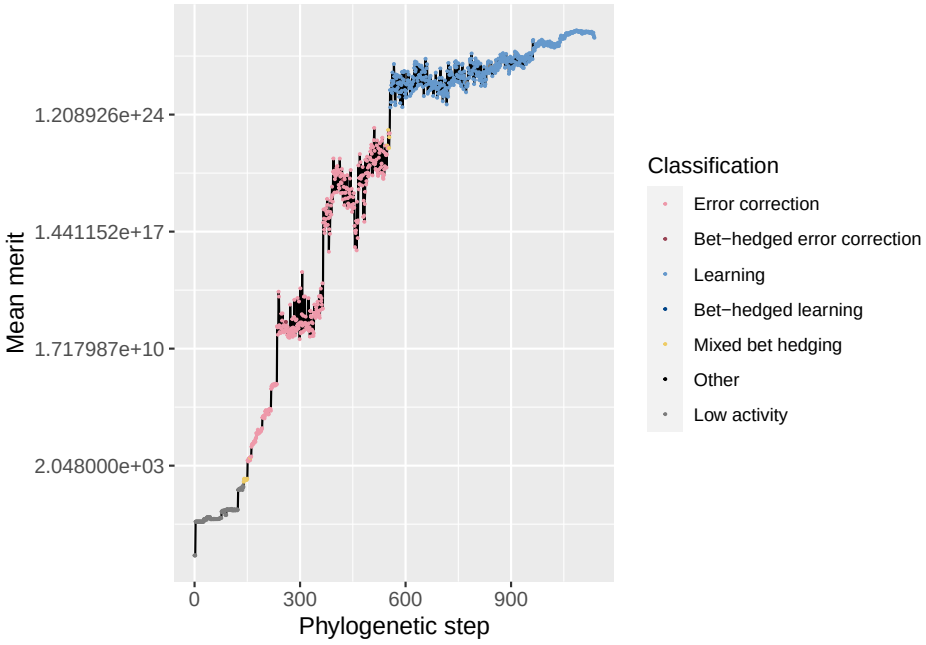
While not shown in the paper due to space limitations, it is important to look at the initial replicate's lineage and how it changes over time.

The code for these plots is copied from the `single_seed_lineage_analysis.R` script.

6.3.1 Merit

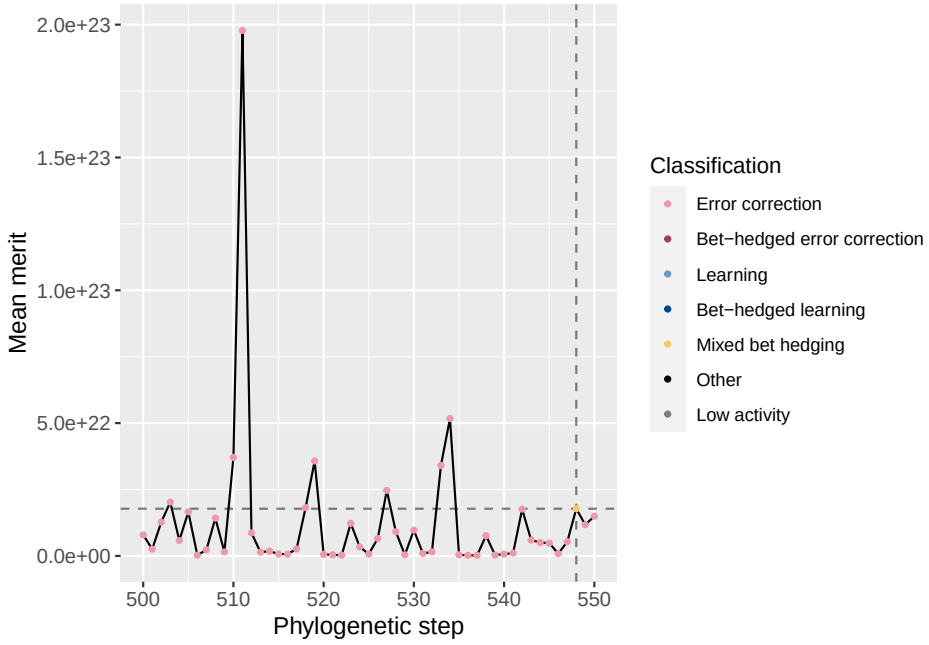
First, we look at the lineage as a whole and how average merit changes over time (note the y-axis is logarithmic)

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = merit_mean)) +
  geom_point(aes(y = merit_mean, color = seed_classification), size = 0.25) +
  scale_y_continuous(trans = 'log2') +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```



Additionally, we can zoom in to view just the potentiation window. The vertical dashed line shows the main potentiating step, step 548, while the horizontal dashed line shows average merit at that step for easier comparisons. Note that this plot is *not* logarithmic.

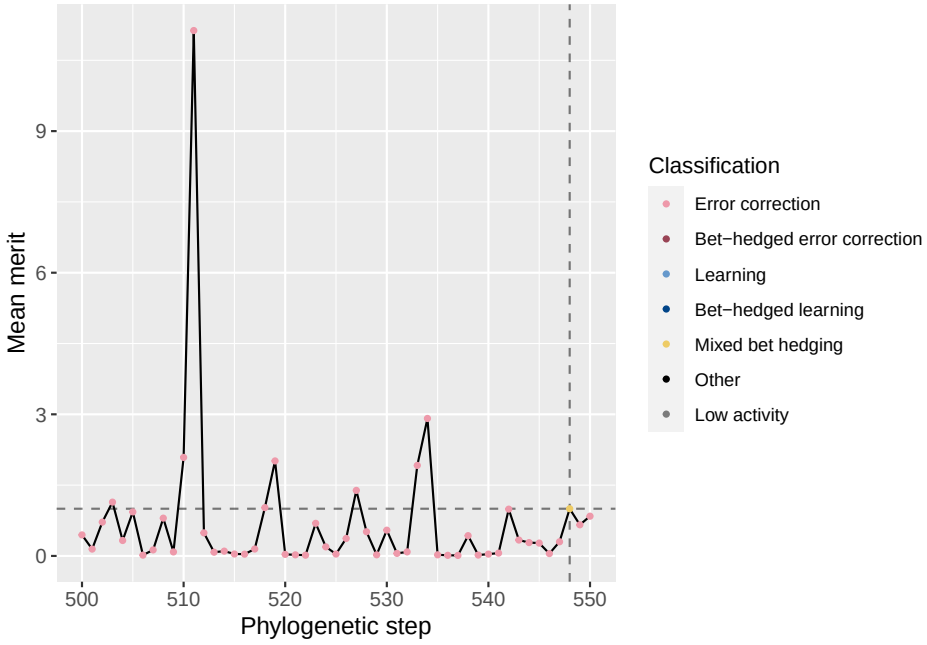
```
potentiation_mask = df_lineage$depth >= potentiation_window_start & df_lineage$depth <= potentiat
target_step_merit_mean = df_lineage[df_lineage$depth == potentiation_target,]$merit_mean
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_merit_mean), linetype = 'dashed', alpha = 0.5) +
  geom_line(aes(y = merit_mean)) +
  geom_point(aes(y = merit_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```



We do not see drastic differences in fitness around step 484.

We can also normalize the merit based on the potentiating step. Remember that one additional correct state confers a merit bonus of 10%. Here, we do see a slight increase in merit.

```
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = 1), linetype = 'dashed', alpha = 0.5) +
  geom_line(aes(y = merit_mean / target_step_merit_mean)) +
  geom_point(aes(y = merit_mean / target_step_merit_mean, color = seed_classification)
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean merit') +
  xlab('Phylogenetic step')
```

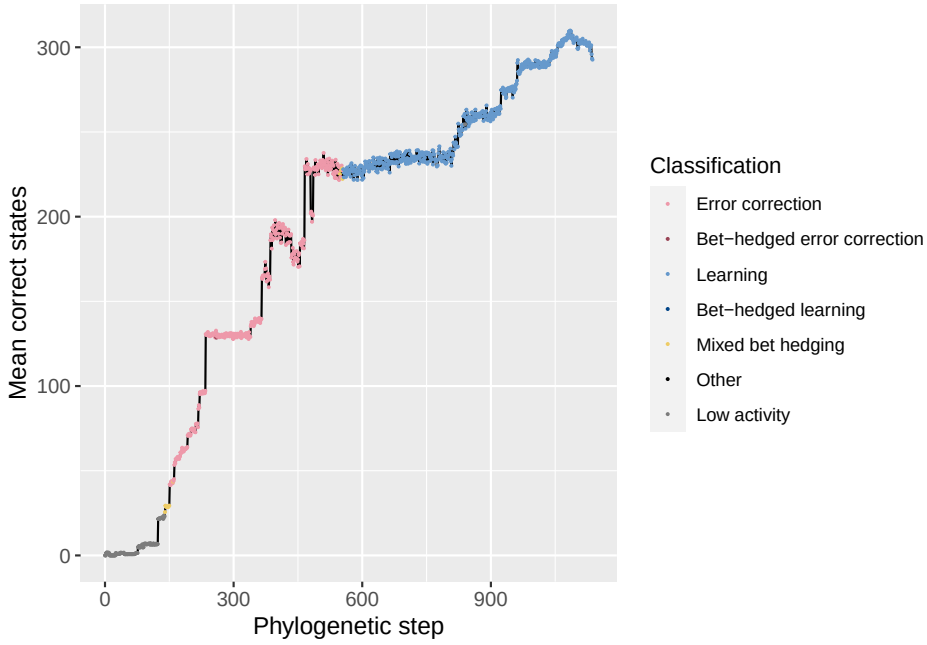


6.3.2 Correct states

Similarly, we can look at the number of correct states both overall and in the potentiation window:

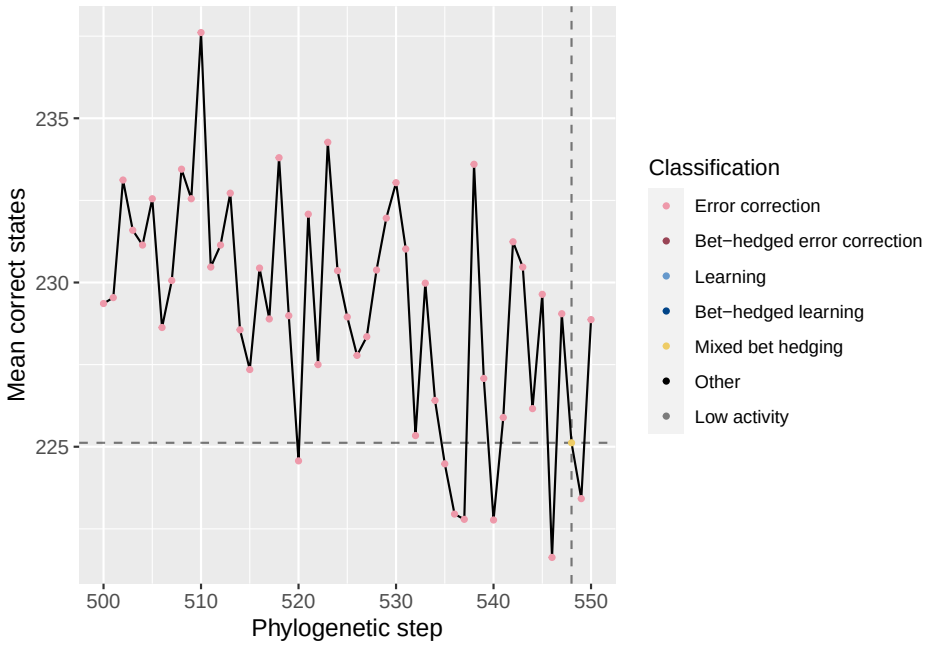
Overall:

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = correct_doors_mean)) +
  geom_point(aes(y = correct_doors_mean, color = seed_classification), size = 0.25) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean correct states') +
  xlab('Phylogenetic step')
```



Potentiation window (dashed lines show the main potentiating mutation):

```
target_step_correct_doors_mean = df_lineage[df_lineage$depth == potentiation_target,]$c
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_correct_doors_mean), linetype = 'dashed', alp
  geom_line(aes(y = correct_doors_mean)) +
  geom_point(aes(y = correct_doors_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean correct states') +
  xlab('Phylogenetic step')
```



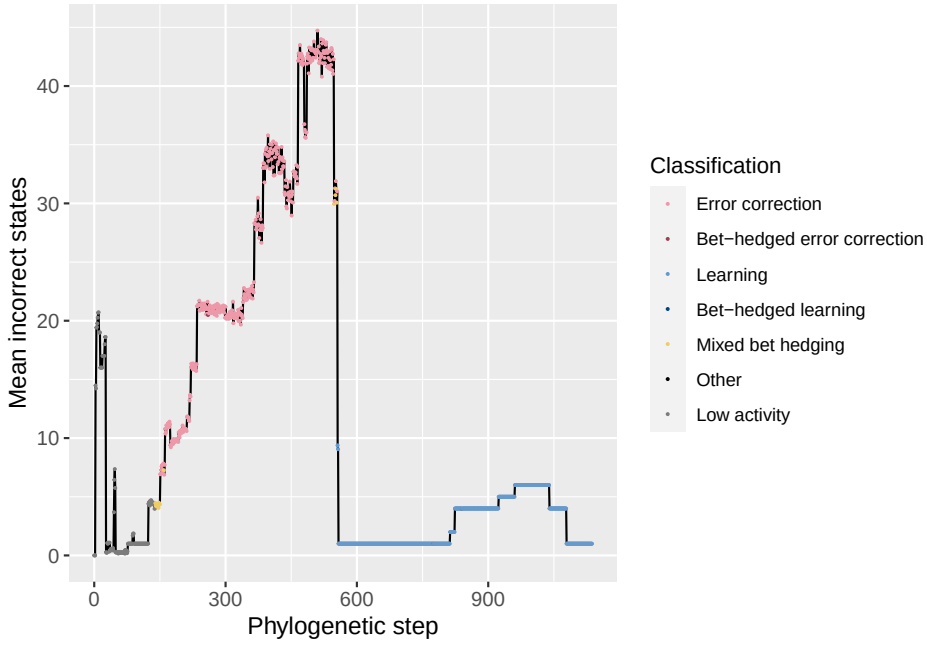
We see no noticable change in the number of correct states around the potentiating mutation.

6.3.3 Incorrect states

Just like our analysis of correct states, we can also look at the average *incorrect* states over time. By incorrect states, we mean the wrong movements while in a *left*, *right*, or *forward* state. Incorrect movements from a *backward* state are tracked separately.

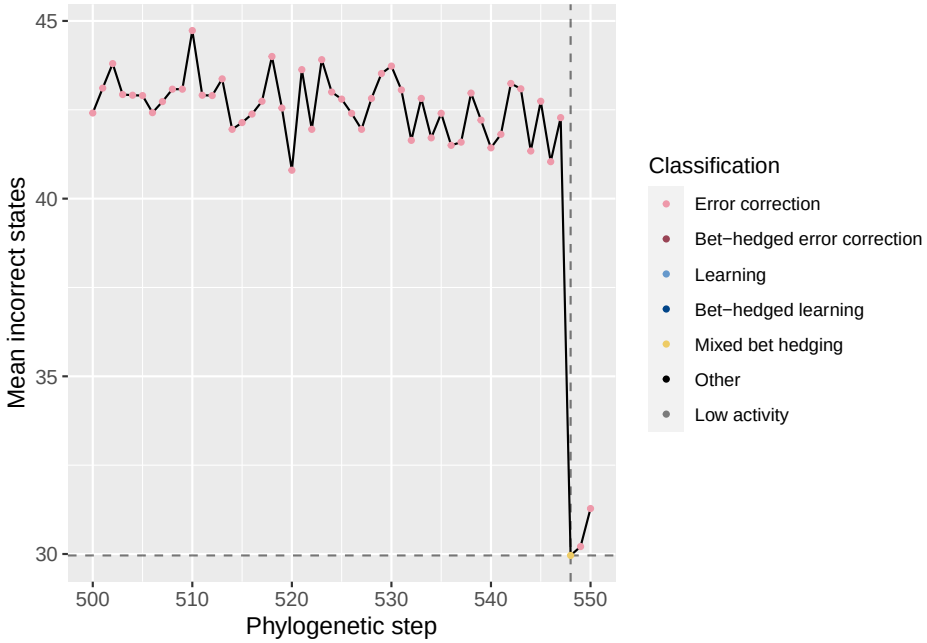
Overall:

```
ggplot(df_lineage, aes(x = depth)) +
  geom_line(aes(y = incorrect_doors_mean)) +
  geom_point(aes(y = incorrect_doors_mean, color = seed_classification), size = 0.25) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean incorrect states') +
  xlab('Phylogenetic step')
```



Potentiation window:

```
target_step_incorrect_doors_mean = df_lineage[df_lineage$depth == potentiation_target,]
ggplot(df_lineage[potentiation_mask,], aes(x = depth)) +
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dashed', alpha = 0.5) +
  geom_hline(aes(yintercept = target_step_incorrect_doors_mean), linetype = 'dashed', a
  geom_line(aes(y = incorrect_doors_mean)) +
  geom_point(aes(y = incorrect_doors_mean, color = seed_classification), size = 1) +
  scale_color_manual(values = color_map) +
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  labs(color = 'Classification') +
  ylab('Mean incorrect states') +
  xlab('Phylogenetic step')
```



We do see a noticeable decrease in the average number of incorrect states at the potentiating mutation. Also, in the full plot, we can see the clear shift from error correction to learning when the number of incorrect states quickly drops to 0.

6.4 Replay data

Now that we know what the lineage looks like, we can begin to break down the replay data.

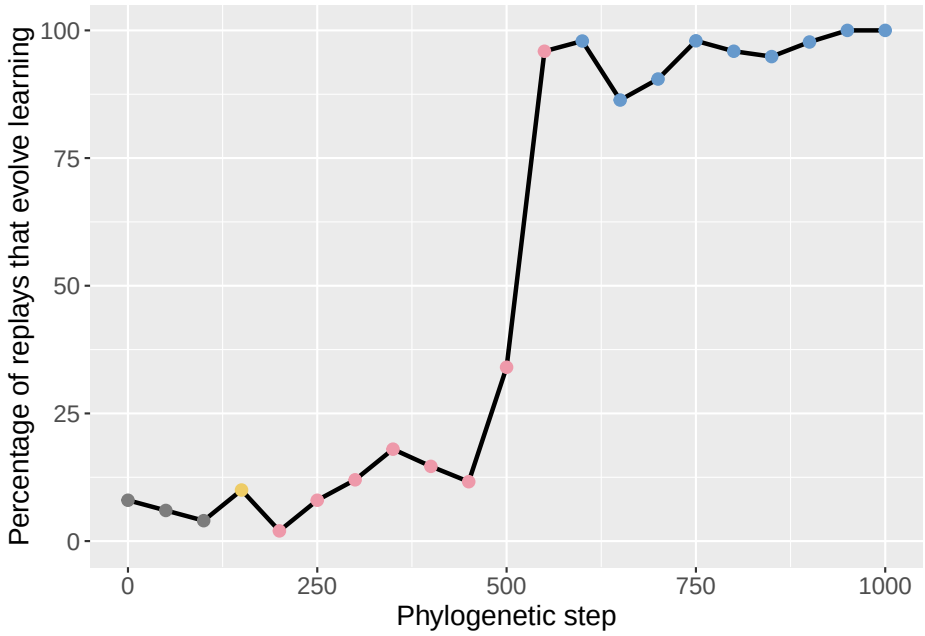
6.4.1 Exploratory replays

We started by seeding 50 replay replicates for every 50th step along the lineage in the window [50, 1000]. The data for step 0 comes from the initial 200 replicates.

Here we show the potentiation over time for those exploratory replays:

```
df_learning_summary = classification_summary[classification_summary$seed_classification == seed_c
df_learning_summary$lineage_classification = NA
# Assign each depth its classification from the lineage data
for(depth in unique(df_learning_summary$depth)){
  df_learning_summary[df_learning_summary$depth == depth,]$lineage_classification = df_lineage[df
}
# Create masks
mask_learning_summary_focal = df_learning_summary$depth %in% potentiation_window_start:potentiati
mask_learning_summary_initial = df_learning_summary$depth %% 50 == 0
```

```
ggplot(df_learning_summary[mask_learning_summary_initial,], aes(x=depth)) +
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```

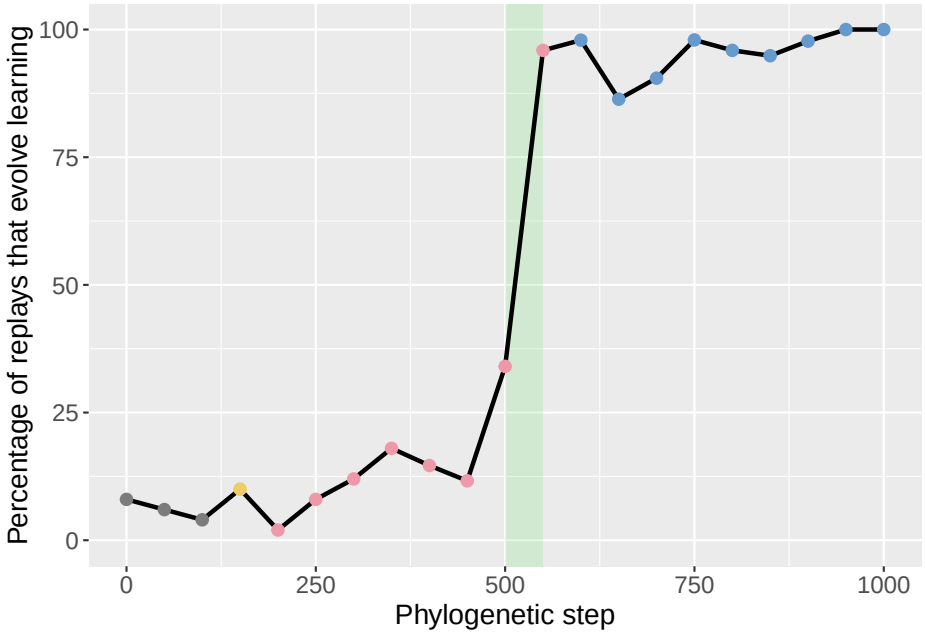


We can then identify the potentiation window based on the points with the most potentiation gain.

Here we selected steps 500-550 (green shaded region):

```
ggplot(df_learning_summary[mask_learning_summary_initial,], aes(x=depth)) +
  annotate("rect", xmin=potentiation_window_start, xmax=potentiation_window_stop, ymin=
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
```

```
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```



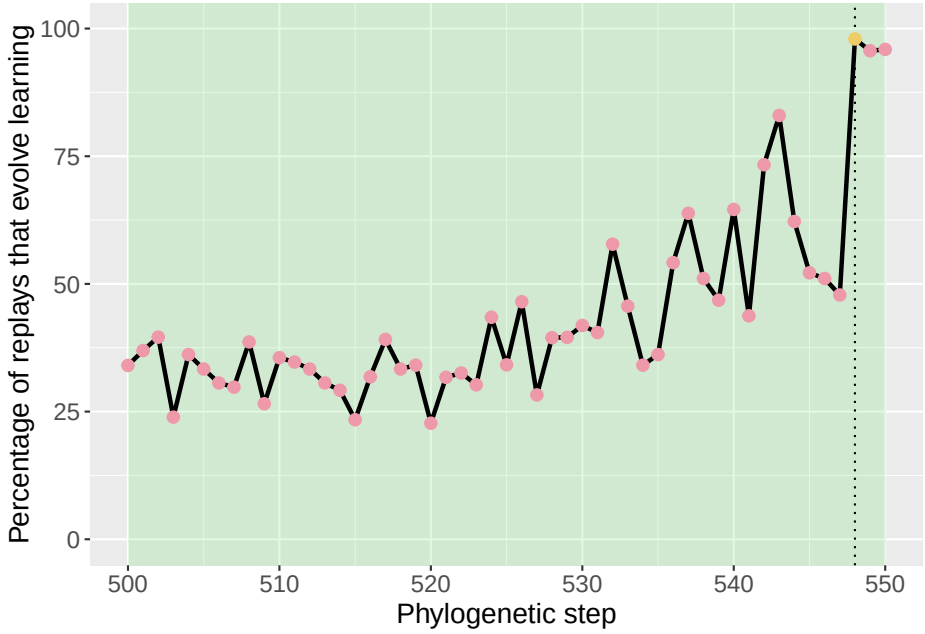
6.4.2 Targeted replays

With the potentiation window identified, we seeded targeted replays.

For the targeted replays, we seeded 50 replay replicates from *every* step in the potetiation window. This gives us an idea of how individual steps in the phylogney affected potentiation.

```
ggplot(df_learning_summary[mask_learning_summary_focal,], aes(x=depth)) +
  annotate("rect", xmin=potentiation_window_start, xmax=potentiation_window_stop, ymin=-Inf, ymax
  geom_vline(aes(xintercept = potentiation_target), linetype = 'dotted') +
  geom_line(mapping=aes(y = frac*100), size = 1.05) +
  geom_point(mapping=aes(y = frac*100, color = as.factor(lineage_classification)), size = 2.5) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,100)) +
  xlab('Phylogenetic step') +
```

```
  ylab('Percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14)) +
  theme(legend.position = 'none')
```



This plot was extensively discussed in the paper, so we refrain from additional discussion here.

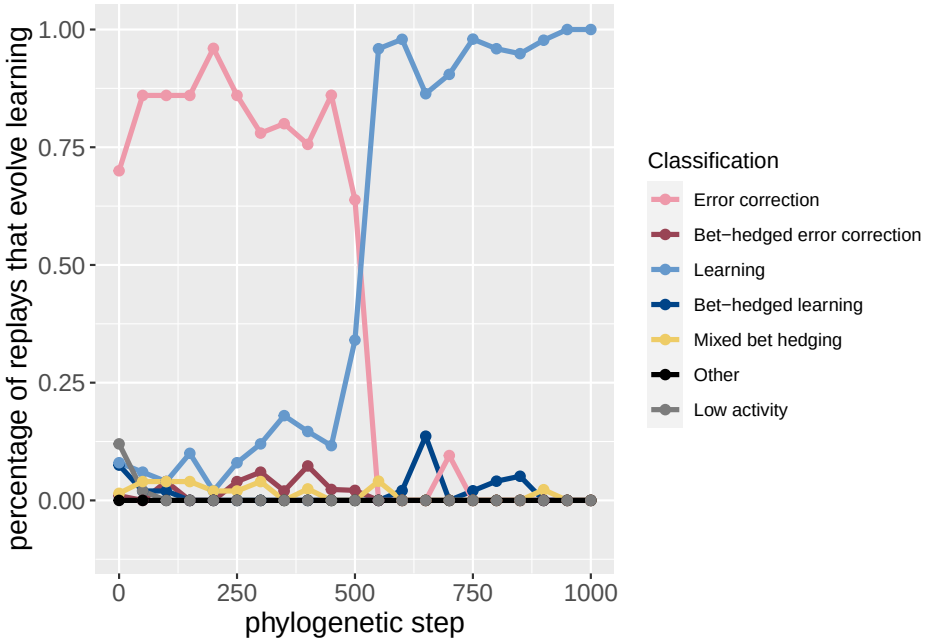
6.4.3 Potentiation of other behaviors

While the paper mostly focused on learning, it can be useful to see how the potentiation of other behaviors changes before learning takes over.

These plots are included as inspiration for future work.

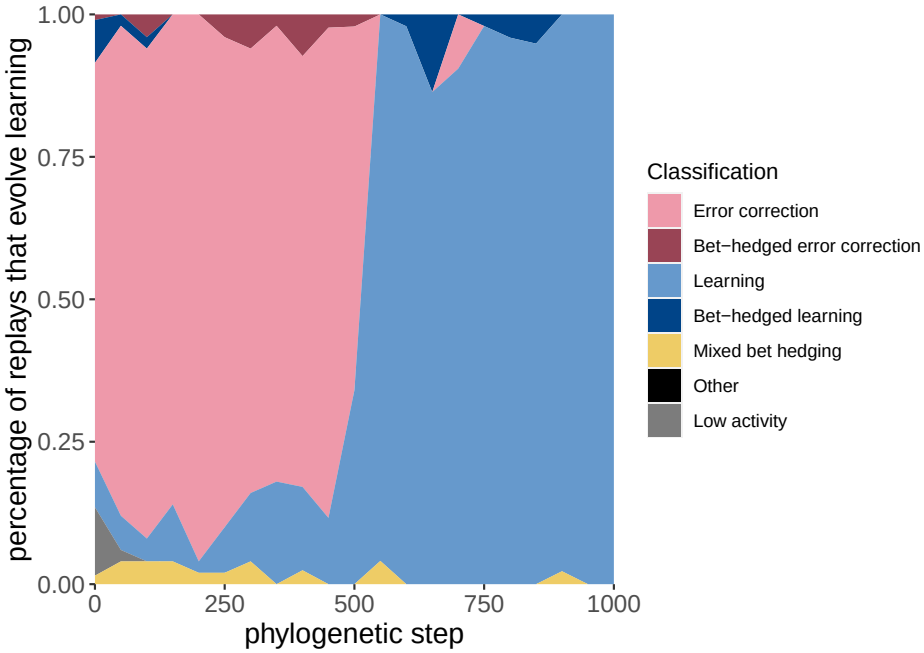
```
mask_focal = classification_summary$depth %in% potentiation_window_start:potentiation_w
mask_initial = classification_summary$depth %% 50 == 0
ggplot(classification_summary[mask_initial,], aes(x=depth, color = seed_classification_
  geom_line(mapping=aes(y = frac), size=1.2) +
  geom_point(mapping=aes(y = frac), size = 2) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,1)) +
  xlab('phylogenetic step') +
```

```
  ylab('percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```



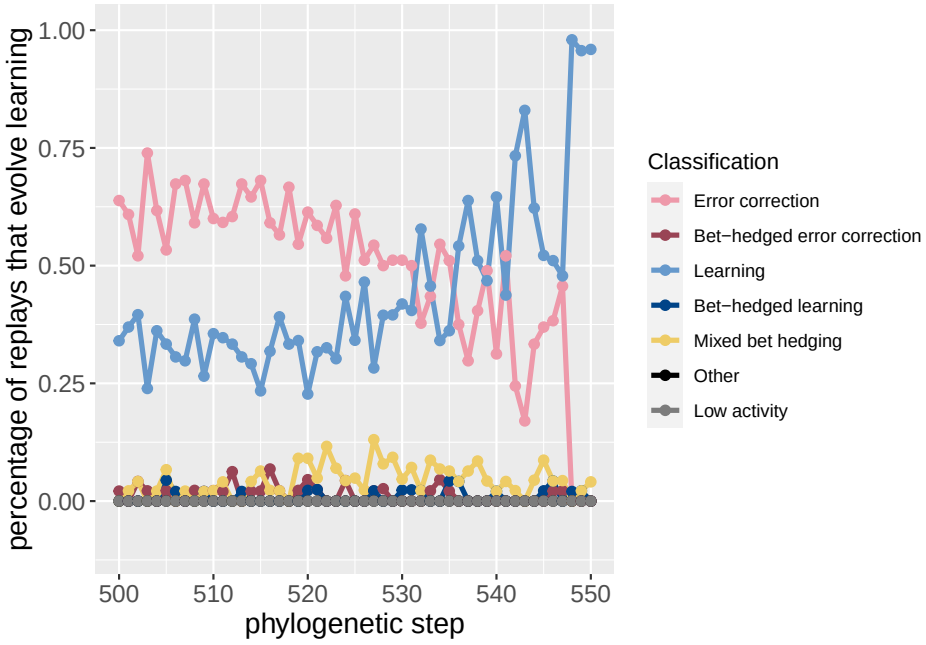
Here is the same data, but plotted as an area plot:

```
ggplot(classification_summary[mask_initial,], aes(x = depth, y = frac, fill = seed_classificatior
  geom_area() +
  scale_fill_manual(values = color_map) +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```

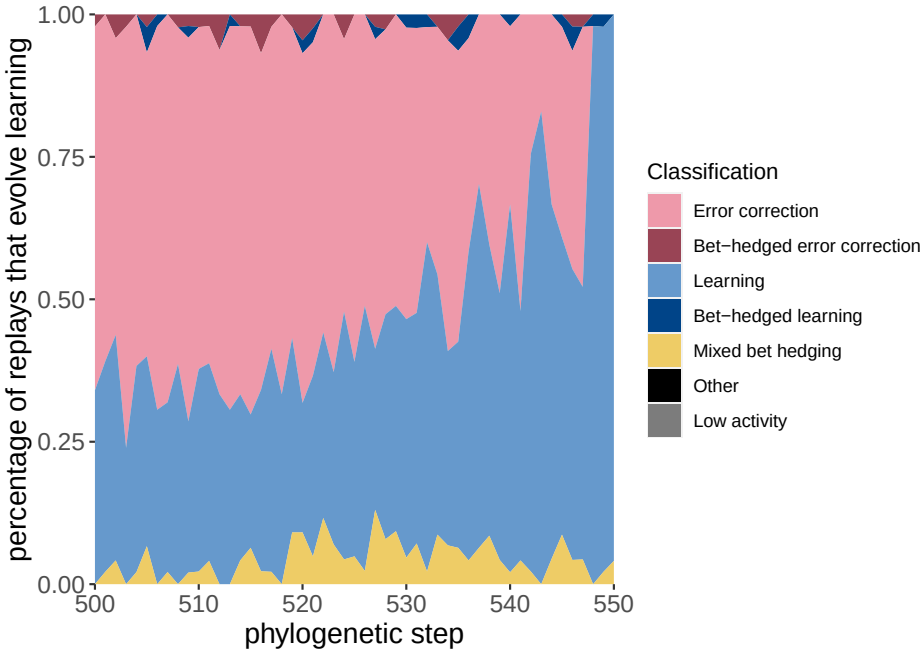


Here are the same two plots showing the potentiation window and the targeted replays:

```
ggplot(classification_summary[mask_focal,], aes(x=depth, color = seed_classification_fa
  geom_line(mapping=aes(y = frac), size=1.2) +
  geom_point(mapping=aes(y = frac), size = 2) +
  scale_color_manual(values = color_map) +
  scale_fill_manual(values = color_map) +
  scale_y_continuous(limits = c(-0.1,1)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(color = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```




```
ggplot(classification_summary[mask_focal,], aes(x = depth, y = frac, fill = seed_classification_f
  geom_area() +
  scale_fill_manual(values = color_map) +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('phylogenetic step') +
  ylab('percentage of replays that evolve learning') +
  labs(fill = 'Classification') +
  theme(axis.text = element_text(size = 12)) +
  theme(axis.title = element_text(size = 14))
```


1.00
0.75
0.50
0.25
0.00
percentage of replays that evolve learning
500
510
520
530
540
550
phylogenetic step
Classification
Error correction
Bet−hedged error correction
Learning
Bet−hedged learning
Mixed bet hedging
Other
Low activity