Directional NK Paper Supplement

Sonia Singhal and Emily Dolson

Contents

0.1 Introduction

This document (created using Rmarkdown) contains: 1) all analysis code used to generate figures for this paper, 2) supplemental figures that did not fit in the main body of the paper.

First, we set up our R environment:

```
library(ggplot2)
library(reshape)
library(dplyr)
library(tidyr)
library(readr)
library(ggraph)
library(tidygraph)
library(captioner)

# Set up figure numbers
fig_nums <- captioner()
fig_nums("surviving", "Proportion of replicates surviving until end")
fig_nums("max_fitness", "Final maximum fitness")
fig_nums("avg_fitness", "Final average fitness")
fig_nums("max_fitness_ts", "Maximum fitness over time")
fig_nums("avg_fitness_ts", "Average fitness over time")
fig_nums("population", "Population size over time")
fig_nums("uniques", "Unique genotypes over time")
fig_nums("normalized_uniques",
  "Unique genotypes over time, normalized by population")
fig_nums("corr", "Landscape correlation in evolved genomes")
fig_nums("repro_landscape", "Full reproductive fitness landscape")
fig_nums("shock_landscape", "Full shock fitness landscape")
```

Next, we load in all our data, which are expected to be in the directory above the one where this script is.

```
# NOTE: This cell will take quite a while to run

base <- "../"

# Read in the file that contains the names of the output documents
files <- read.table(paste(base, "files.txt", sep = ""), stringsAsFactors = F)
```

```
nval <- 15
ntimes = nrow(files)

# Compile the summary statistics, populations, and (reproductive) landscapes
# into individual data frames row <- 1
filename <- files[1,]
subfolder <- strsplit(filename, "/")[[1]][1]
subfile <- strsplit(filename, "/")[[1]][2]

# read in the statistics
trace <- read.table(paste(base, subfolder, "/stats-", subfile, sep = ""),
                    header = T, stringsAsFactors = F)

# Read in the population output
pops <- read.table(paste(base, subfolder, "/popt-", subfile, sep = ""),
                   header = T, stringsAsFactors = F,
                   col.names =
                     c("Gen", "Genome", "Count", "Fitness", "ShockFitness"))

# get N, K, mu, replicate (seed), and regime arguments
subpieces <- strsplit(subfile, "-")
kval <- as.numeric(substring(subpieces[[1]][2], 2))

# if seed is followed by another argument
# seed <- as.numeric(substring(subpieces[[1]][3], 2))

# if seed ends argument list
seed <- as.numeric(substring(strsplit(subpieces[[1]][3], "[.]")[[1]][1], 2))
muval <- ifelse(length(subpieces[[1]]) == 5,
                as.numeric(substring(subpieces[[1]][4], 2)),
                as.numeric(paste(substring(subpieces[[1]][4], 2),
                                 subpieces[[1]][5], sep = "-")))
regparam <- subpieces[[1]][length(subpieces[[1]])]
regval <- ifelse(length(regmatches(regparam, gregexpr("S", regparam))[[1]]) > 0,
                 "Sudden",
                 "Gradual")

# read in the reproductive landscape (requires the N value and the K value).
# For the shock landscape, use "/sland-" instead of "/land-".
peaks <- read.table(paste(base, subfolder, "/land-", subfile, sep = ""),
                    skip = (6+nval+2^(kval+1)),
                    col.names = c("Genome", "Fitness"))
shock_peaks <- read.table(paste(base, subfolder, "/sland-", subfile, sep = ""),
                          skip = (6+nval+2^(kval+1)),
                          col.names = c("Genome", "Fitness"))

# Assign treatment parameters to each data frame
trace$k <- kval
pops$k <- kval
peaks$k <- kval
shock_peaks$k <- kval
trace$n <- nval
pops$n <- nval
```

```
peaks$n <- nval
shock_peaks$n <- nval
trace$seed <- seed
pops$seed <- seed
peaks$seed <- seed
shock_peaks$seed <- seed
trace$regime <- regval
pops$regime <- regval
peaks$regime <- regval
shock_peaks$regime <- regval
trace$mu <- muval
pops$mu <- muval
peaks$mu <- muval
shock_peaks$mu <- muval
for (f in 2:nrow(files)){
  filename <- files[f,]
  subfolder <- strsplit(filename, "/")[[1]][1]
  subfile <- strsplit(filename, "/")[[1]][2]
  thistrace <- read.table(paste(base, subfolder, "/stats-", subfile, sep = ""),
                          header = T,
                          stringsAsFactors = F)
  thispop <- read.table(paste(base, subfolder, "/popt-", subfile, sep = ""),
                        header = T,
                        stringsAsFactors = F,
                        col.names =
                          c("Gen", "Genome", "Count",
                            "Fitness", "ShockFitness"))
  subpieces <- strsplit(subfile, "-")
  kval <- as.numeric(substring(subpieces[[1]][2], 2))
  #seed <- as.numeric(substring(subpieces[[1]][3], 2))
  seed <- as.numeric(substring(strsplit(subpieces[[1]][3], "[.]")[[1]][1], 2))
  regparam <- subpieces[[1]][length(subpieces[[1]])]
  regval <- ifelse(length(regmatches(regparam,
                                     gregexpr("S", regparam))[[1]]) > 0,
                   "Sudden",
                   "Gradual")
  #print(paste(seed, regval))
  muval <- ifelse(length(subpieces[[1]]) == 5,
                  as.numeric(substring(subpieces[[1]][4], 2)),
                  as.numeric(paste(substring(subpieces[[1]][4], 2),
                                   subpieces[[1]][5], sep = "-")))
  thispeak <- read.table(paste(base, subfolder, "/land-", subfile, sep = ""),
                         skip = (6+nval+2^(kval+1)),
                         col.names = c("Genome", "Fitness"))
  thisshockpeak <- read.table(paste(base,
                                    subfolder,
                                    "/sland-",
                                    subfile,
                                    sep = ""),
                              skip = (6+nval+2^(kval+1)),
                              col.names = c("Genome", "Fitness"))
  thistrace$k <- kval
  thispop$k <- kval
```

```
  thispeak$k <- kval
  thisshockpeak$k <- kval
  thistrace$n <- nval
  thispop$n <- nval
  thispeak$n <- nval
  thisshockpeak$n <- nval
  thistrace$seed <- seed
  thispop$seed <- seed
  thispeak$seed <- seed
  thisshockpeak$seed <- seed
  thistrace$regime <- regval
  thispop$regime <- regval
  thispeak$regime <- regval
  thisshockpeak$regime <- regval
  thistrace$mu <- muval
  thispop$mu <- muval
  thispeak$mu <- muval
  thisshockpeak$mu <- muval
  trace <- rbind(trace, thistrace)
  pops <- rbind(pops, thispop)
  peaks <- rbind(peaks, thispeak)
  shock_peaks <- rbind(shock_peaks, thisshockpeak)
}

# Create regime label for future plotting
shock_peaks$Landscape <- "shock"
peaks$Landscape <- "reproduction"
```

Next, we do some additional post-processing:

```
# Check number of persisting generations
ntimes <- length(unique(trace$seed))
survivals <- data.frame(n = rep(NA, ntimes),
                        k = rep(NA, ntimes),
                        seed = rep(NA, ntimes),
                        regime = rep(NA, ntimes),
                        cutoff = rep(NA, ntimes),
                        gen = rep(NA, ntimes),
                        uniques = rep(NA, ntimes),
                        average = rep(NA, ntimes),
                        stdev = rep(NA, ntimes),
                        max = rep(NA, ntimes),
                        shockcut = rep(NA, ntimes),
                        shockAvg = rep(NA, ntimes),
                        shockStd = rep(NA, ntimes),
                        shockMax = rep(NA, ntimes),
                        mu = rep(NA, ntimes),
                        diversity = rep(NA, ntimes),
                        evenness = rep(NA, ntimes),
                        population = rep(NA, ntimes))
for(s in 1:ntimes){
  sd <- unique(trace$seed)[s]
  sub <- subset(trace, seed == sd)
  survivals[s, ] <- subset(sub,
```

```
                            gen == max(sub$gen),
                            select = c(n, k, seed, regime, cutoff, gen, uniques,
                                       average, stdev, max, shockCut, shockAvg,
                                       shockStd, shockMax, mu,
                                       diversity, evenness, population))
}

# Adjust significant digits on mutation rate for plotting
survivals$mu <- format(survivals$mu, digits=3, drop0trailing = TRUE, format="g")
survivals <- survivals %>% mutate(mu = case_when(
  mu=="0.066667" ~ "0.0667",
  TRUE ~ mu
  )
)
```

0.2 Analysis

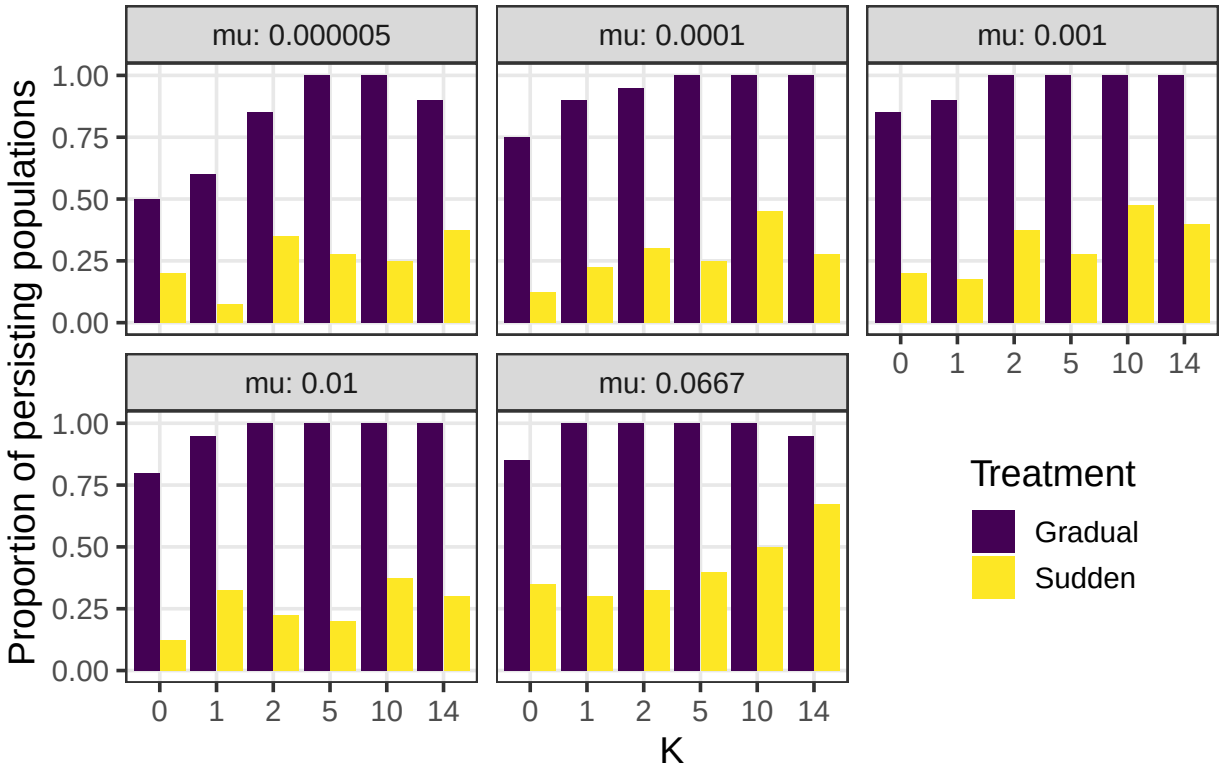
0.2.1 Survival probability

First, we assess the effect of different evolutionary regimes on the probability of a population surviving until the end of the experiment:

```
pd <- position_dodge(.5)
ggplot(data = survivals %>%  # Do some additional processing of the data
                # Only grab data points from the final generation
                # (eliminates populations that went extinct)
                filter(gen == 51) %>%
                # Calculate percentages per condition
                # (combo of k, regime, and mu)
                group_by(k, regime, mu) %>%
                summarise(
                  # The sudden regime had 40 replicates
                  # while the gradual regime had 20
                  percent_survive = ifelse(
                                      regime == "Sudden",
                                      n()/40,
                                      n()/20
                                    )
                ),
      # Plot each regime as a different color with k as x axis
      aes(x = as.factor(k), fill = regime)) +
      scale_x_discrete() +
      # Plot data as bars indicating proportion of population that survived
      geom_bar(
        aes(y=percent_survive),
        position = position_dodge(.92),  # Put bars for each regime side by side
        stat = "identity"
      ) +
      theme_bw(base_size = 14) +
      theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "grey91"),
        strip.text.y = element_text(face = "italic"),
        legend.position = c(.85, .25)) +
      # Make a panel for each mutation rate
```

```
      facet_wrap(~mu, labeller = labeller(mu=label_both)) +
      scale_fill_viridis_d(name = "Treatment") +
      ylab("Proportion of persisting populations") +
      xlab("K") +
      ggtitle(fig_nums("surviving"))
```

Figure 1: Proportion of replicates surviving until end



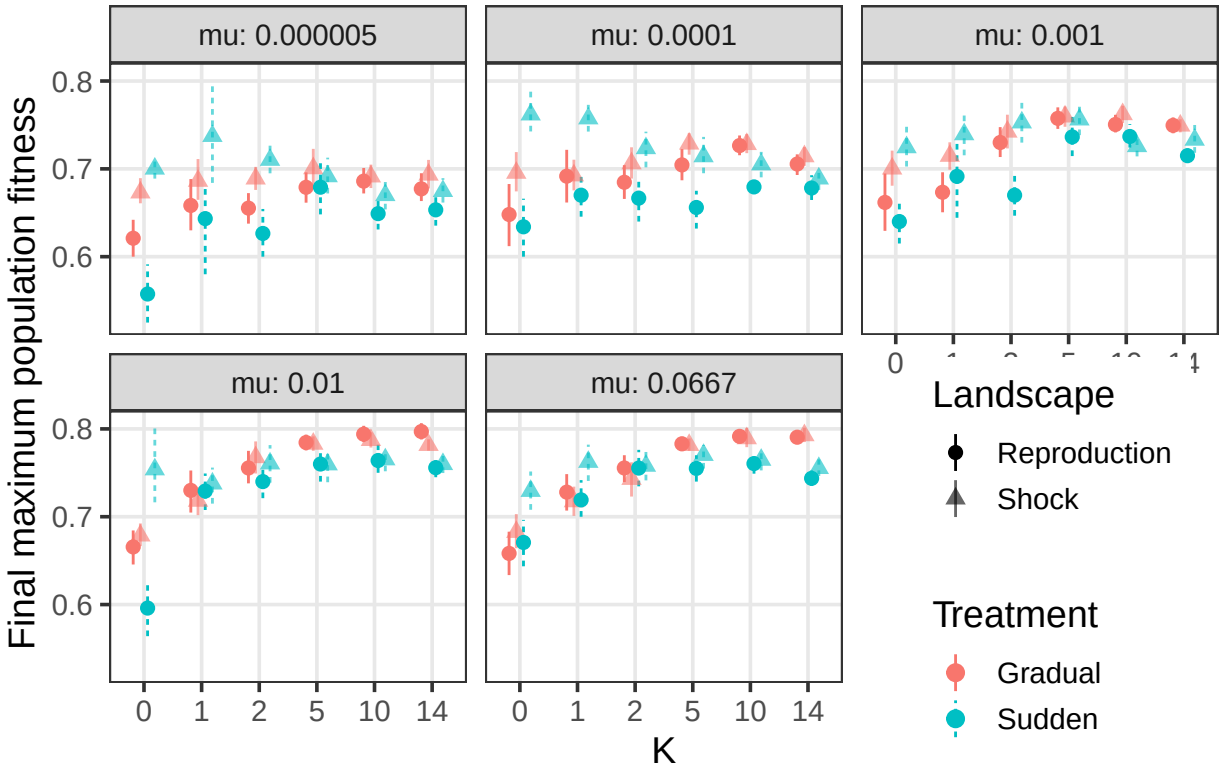
0.2.2 Final fitness

Next, we compare the maximum fitness at the end of the run (in both landscapes) across conditions (for runs that did not go extinct):

```
# Pivot shock and reproductive fitness into a single column
# so we can easily plot a comparison
ggplot(data = survivals %>%
                # Grab only final generation,
                # to exclude populations that went extinct early
                filter(gen==51) %>%
                pivot_longer(c(max, shockMax),
                             names_to="Landscape",
                             values_to="max_fitness") %>%
                mutate(Landscape = case_when(
                                    Landscape == "max" ~ "Reproduction",
                                    Landscape=="shockMax" ~ "Shock")
                       ),
       aes(x = as.factor(k),
           y = max_fitness,
           color = regime,
           shape=Landscape,
```

```
          linetype=regime, # Give regimes different linetype for greyscale
          alpha=Landscape  # Give landscapes different opacity for greyscale
        )
) +
# Plot a bootstrapped confidence interval around the mean
stat_summary(fun.data = "mean_cl_boot", geom = "pointrange", position = pd) +
theme_bw(base_size = 14) +
theme(panel.grid.minor = element_blank(),
      panel.grid.major = element_line(color = "grey91"),
      strip.text.y = element_text(face = "italic"),
      legend.position = c(.85, .20)) +
scale_color_hue(name = "Treatment") +
scale_alpha_discrete(range = c(1,.6)) +  # Set alpha levels the landscapes
facet_wrap(~mu, labeller = labeller(mu = label_both)) +
ylab("Final maximum population fitness") +
scale_linetype("Treatment") +
xlab("K") +
ggtitle(fig_nums("max_fitness"))
```

Figure 2: Final maximum fitness

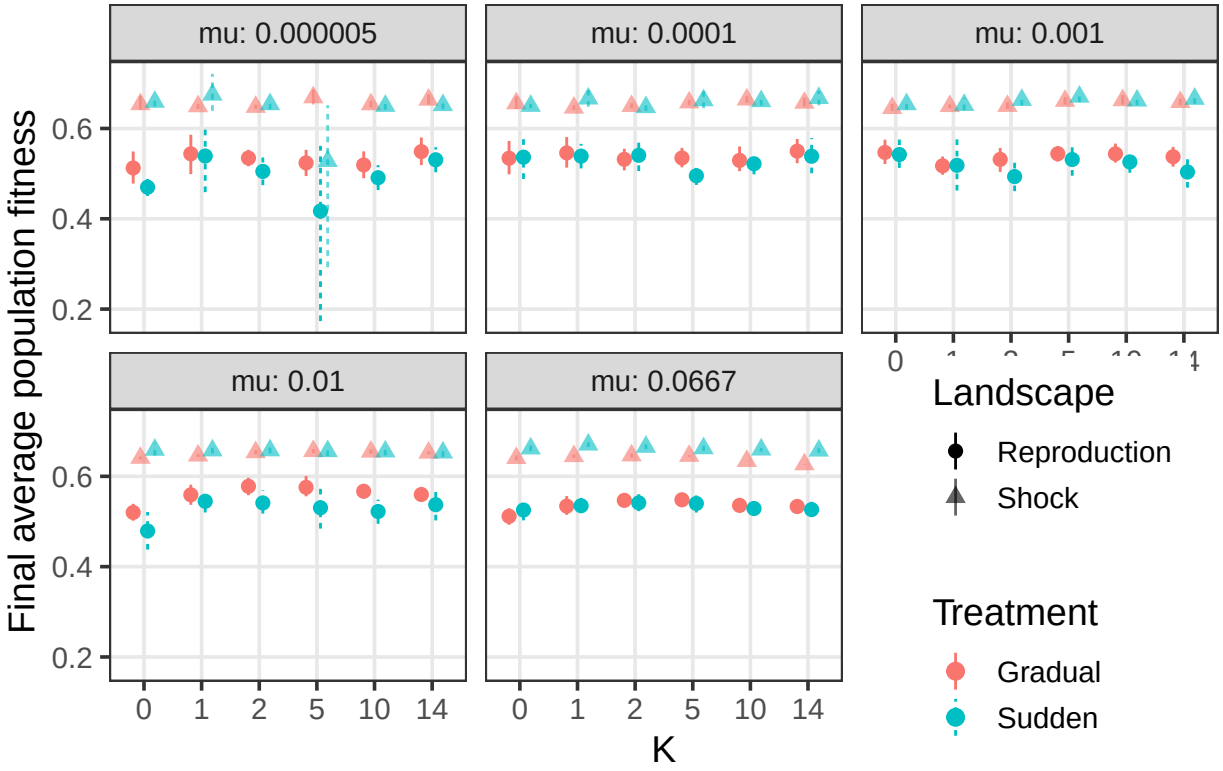


We make the same plot for average fitness (recall that there the shock landscape is the only source of death, so there are many low fitness individuals in the population at any point in time):

```
ggplot(data = survivals %>%
                filter(gen==51) %>%
                pivot_longer(c(average, shockAvg),
                             names_to="Landscape",
                             values_to="max_fitness") %>%
                mutate(Landscape = case_when(
```

```
                                        Landscape == "average" ~ "Reproduction",
                                        Landscape=="shockAvg" ~ "Shock")
                ),
       aes(x = as.factor(k),
           y = max_fitness,
           color = regime,
           shape=Landscape,
           linetype=regime, alpha=Landscape)
       ) +
  stat_summary(fun.data = "mean_cl_boot", geom = "pointrange", position = pd) +
  theme_bw(base_size = 14) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "grey91"),
        strip.text.y = element_text(face = "italic"),
        legend.position = c(.85, .2)) +
  scale_color_hue(name = "Treatment") +
  facet_wrap(~mu, labeller = labeller(mu = label_both)) +
  ylab("Final average population fitness") +
  scale_alpha_discrete(range = c(1,.6)) +
  scale_linetype("Treatment") +
  xlab("K") +
  ggtitle(fig_nums("avg_fitness"))
```

Figure 3: Final average fitness

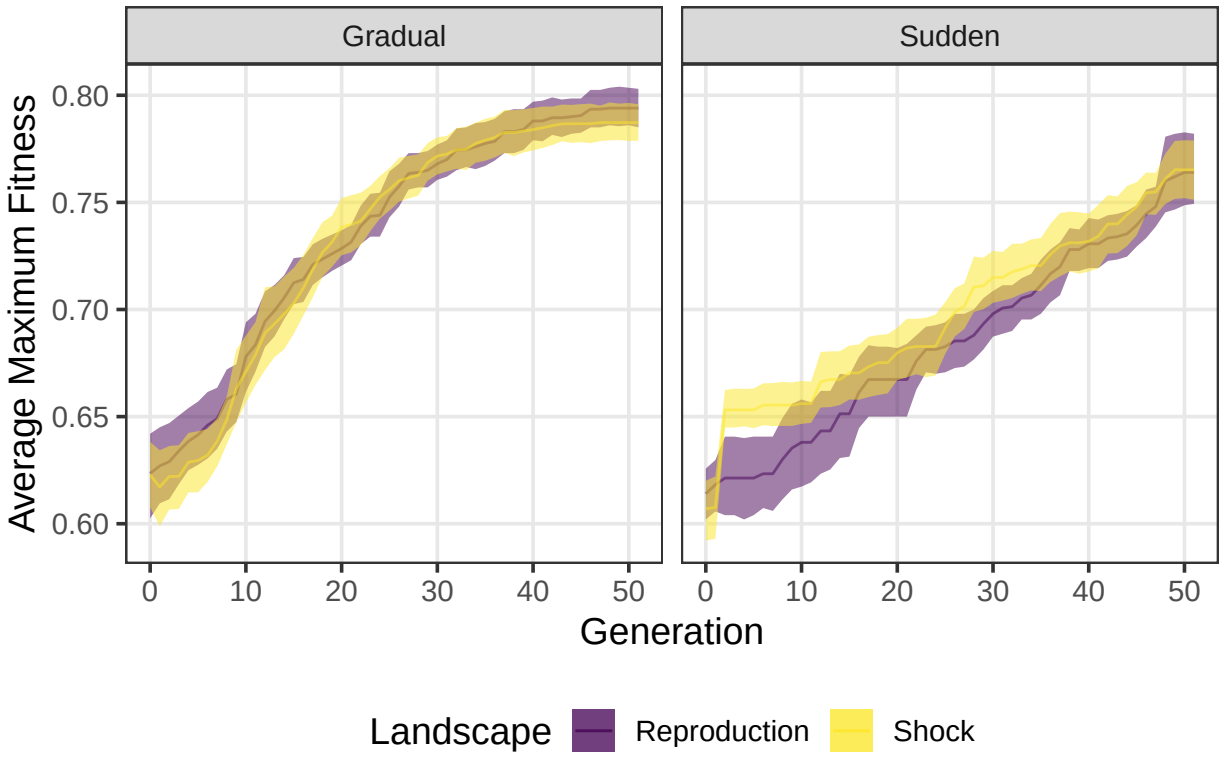


0.2.3 Fitness timeseries analysis

Next, we analyze the behavior of maximum fitness (within each individual population) over time:

```
ggplot(data = trace %>%
         # Plot a single representative combination of mu and k
         filter(mu==.01, k==10),
       aes(x = gen, y = max)
  ) +
  facet_wrap(~regime) +  # A panel for each shock regime
  # Bootstrapped 95% CI for repro landscape
  stat_summary(fun.data="mean_cl_boot",
               geom="ribbon",
               alpha=.5, aes(fill="Reproduction")) +
  # Avg max fitness for repro landscape
  stat_summary(fun.data="mean_cl_boot",
               geom="line",
               alpha=.5,
               aes(color="Reproduction")) +
  # Bootstrapped 95% CI for shock landscape
  stat_summary(aes(y=shockMax, fill="Shock"),
               fun.data="mean_cl_boot",
               geom="ribbon",
               alpha=.5) +
  # Avg max fitness for shock landscape
  stat_summary(aes(y=shockMax, color="Shock"),
               fun.data="mean_cl_boot",
               geom="line",
               alpha=.5) +
  theme_bw(base_size = 14) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "grey91"),
        strip.text.y = element_text(face = "italic"),
        legend.position = "bottom")  +
  # Clean up legend
  scale_color_manual("Landscape",
                     values = c("Reproduction"="#440154", "Shock"="#fde725")) +
  scale_fill_manual("Landscape",
                    values = c("Reproduction"="#440154", "Shock"="#fde725")) +
  ylab("Average Maximum Fitness") +
  xlab("Generation") +
  ggtitle(fig_nums("max_fitness_ts"))
```

Figure 4: Maximum fitness over time

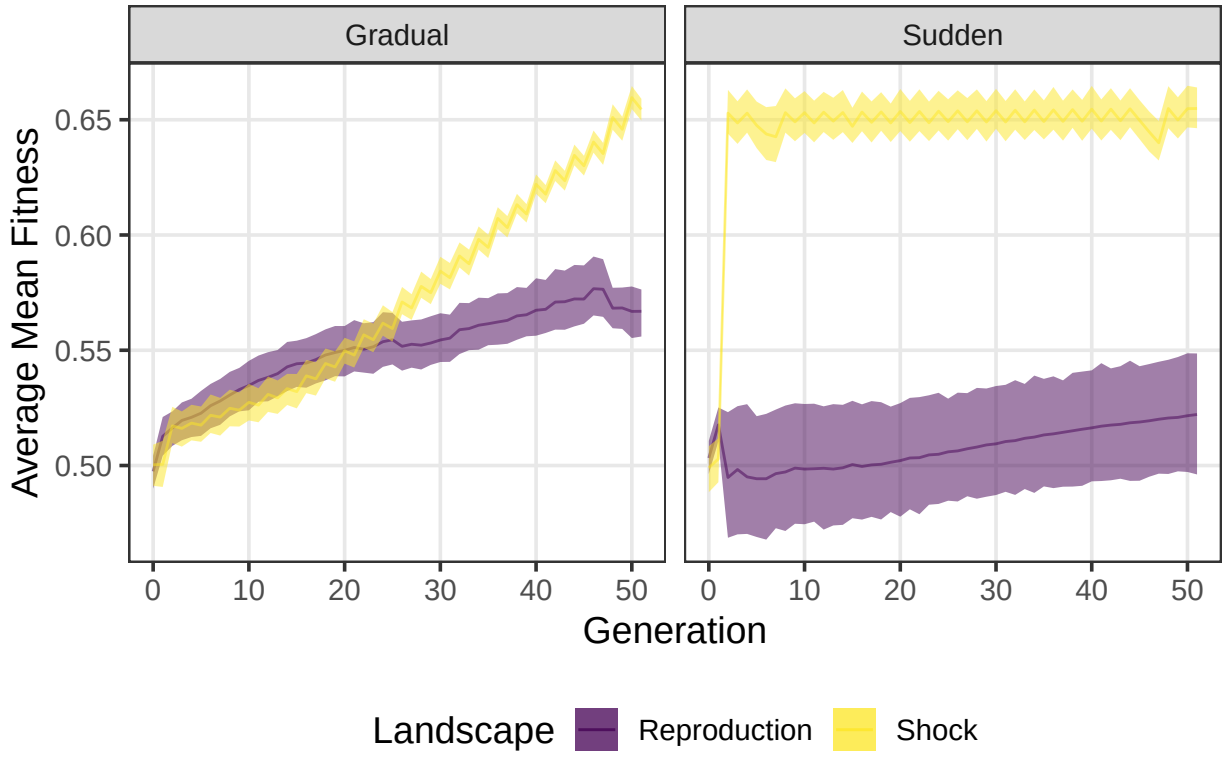


And we do the same for average fitness:

```
ggplot(data = trace %>%
         filter(mu==.01, k==10),
       aes(x = gen, y = average)) +
  facet_wrap(~regime) +
  stat_summary(fun.data="mean_cl_boot",
               geom="ribbon",
               alpha=.5,
               aes(fill="Reproduction")) +
  stat_summary(fun.data="mean_cl_boot",
               geom="line",
               alpha=.5,
               aes(color="Reproduction")) +
  stat_summary(aes(y=shockAvg, fill="Shock"),
               fun.data="mean_cl_boot",
               geom="ribbon",
               alpha=.5) +
  stat_summary(aes(y=shockAvg, color="Shock"),
               fun.data="mean_cl_boot",
               geom="line",
               alpha=.5) +
  theme_bw(base_size = 14) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "grey91"),
        strip.text.y = element_text(face = "italic"),
        legend.position = "bottom")  +
  scale_color_manual("Landscape",
```

```
                      values = c("Reproduction"="#440154", "Shock"="#fde725")) +
  scale_fill_manual("Landscape",
                    values = c("Reproduction"="#440154", "Shock"="#fde725")) +
  ylab("Average Mean Fitness") +
  xlab("Generation") +
  ggtitle(fig_nums("avg_fitness_ts"))
```

Figure 5: Average fitness over time

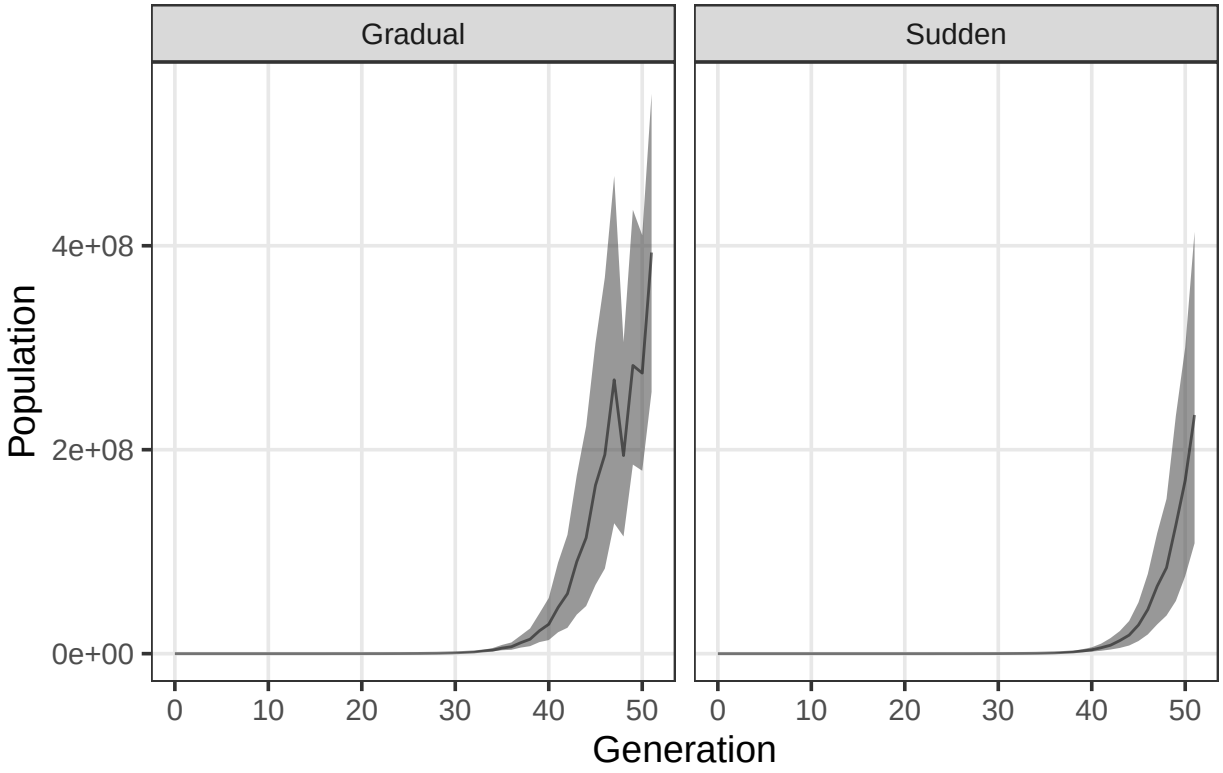


0.2.4 Population size

A major part of what is happening here appears to be the effect of the gradual regime allowing a larger population than the sudden regime. Here we plot population:

```
ggplot(data = trace %>%
         filter(mu==.01, k==10),
       aes(x = gen, y = population)) +
  facet_wrap(~regime) +  # One panel for each regime
  # Bootstrapped 95% CI around mean count of unique genotypes
  stat_summary(fun.data="mean_cl_boot", geom="ribbon", alpha=.5) +
  # Mean count of unique genotypes
  stat_summary(fun.data="mean_cl_boot", geom="line", alpha=.5) +
  theme_bw(base_size = 14) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "grey91"),
        strip.text.y = element_text(face = "italic"),
        legend.position = "none")  +
  ylab("Population") +
  xlab("Generation") +
  ggtitle(fig_nums("population"))
```

Figure 6: Population size over time

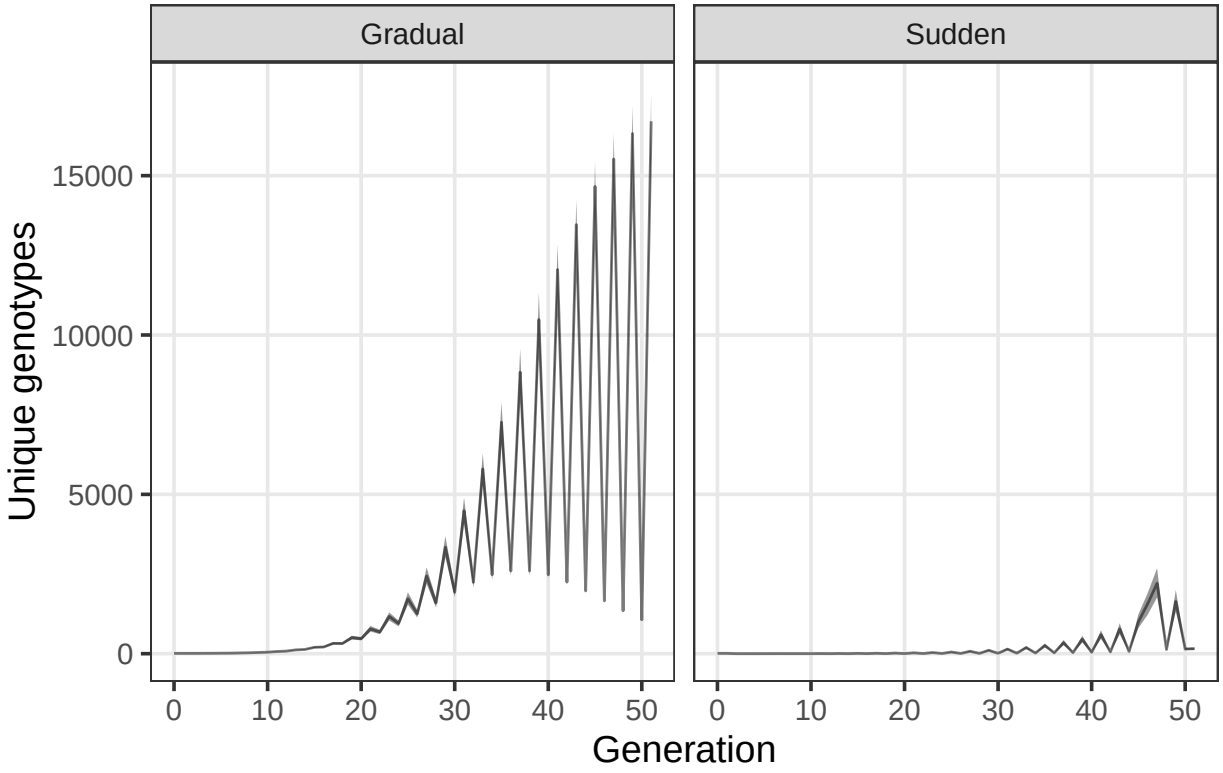


0.2.5 Diversity

Diversity clearly plays an important role in the dynamics we observe. Here, we plot the number of unique genotypes over time in each condition:

```
ggplot(data = trace %>%
         filter(mu==.01, k==10),
       aes(x = gen, y = uniques)) +
  facet_wrap(~regime) +  # One panel for each regime
  # Bootstrapped 95% CI around mean count of unique genotypes
  stat_summary(fun.data="mean_cl_boot", geom="ribbon", alpha=.5) +
  # Mean count of unique genotypes
  stat_summary(fun.data="mean_cl_boot", geom="line", alpha=.5) +
  theme_bw(base_size = 14) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "grey91"),
        strip.text.y = element_text(face = "italic"),
        legend.position = "none")  +
  ylab("Unique genotypes") +
  xlab("Generation") +
  ggtitle(fig_nums("uniques"))
```

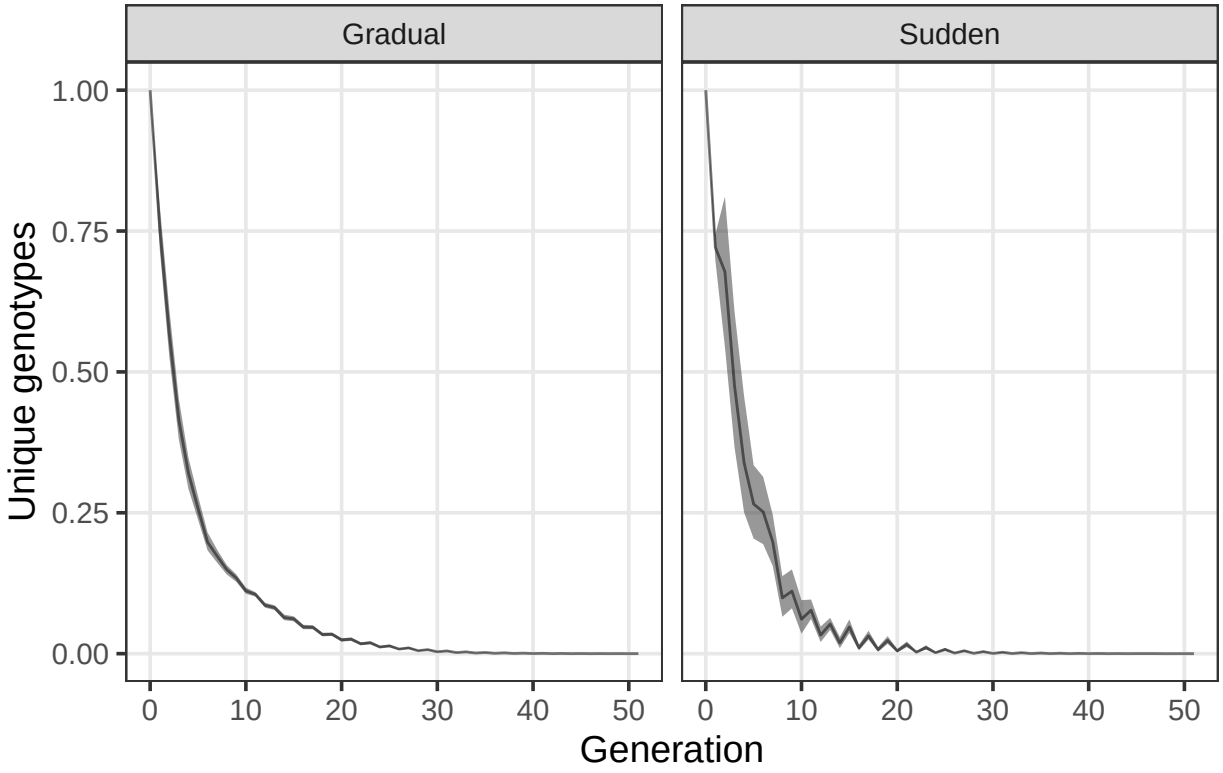
Figure 7: Unique genotypes over time



A lot of this trend is likely being driven by population size, so it may also be worth looking at the same plot normalized by population:

```
ggplot(data = trace %>%
         filter(mu==.01, k==10),
       aes(x = gen, y = uniques/population)) +
  facet_wrap(~regime) +  # One panel for each regime
  # Bootstrapped 95% CI around mean count of unique genotypes
  stat_summary(fun.data="mean_cl_boot", geom="ribbon", alpha=.5) +
  # Mean count of unique genotypes
  stat_summary(fun.data="mean_cl_boot", geom="line", alpha=.5) +
  theme_bw(base_size = 14) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "grey91"),
        strip.text.y = element_text(face = "italic"),
        legend.position = "none")  +
  ylab("Unique genotypes") +
  xlab("Generation") +
  ggtitle(fig_nums("normalized_uniques"))
```

Figure 8: Unique genotypes over time, normalized by p



0.2.6 Landscape analysis

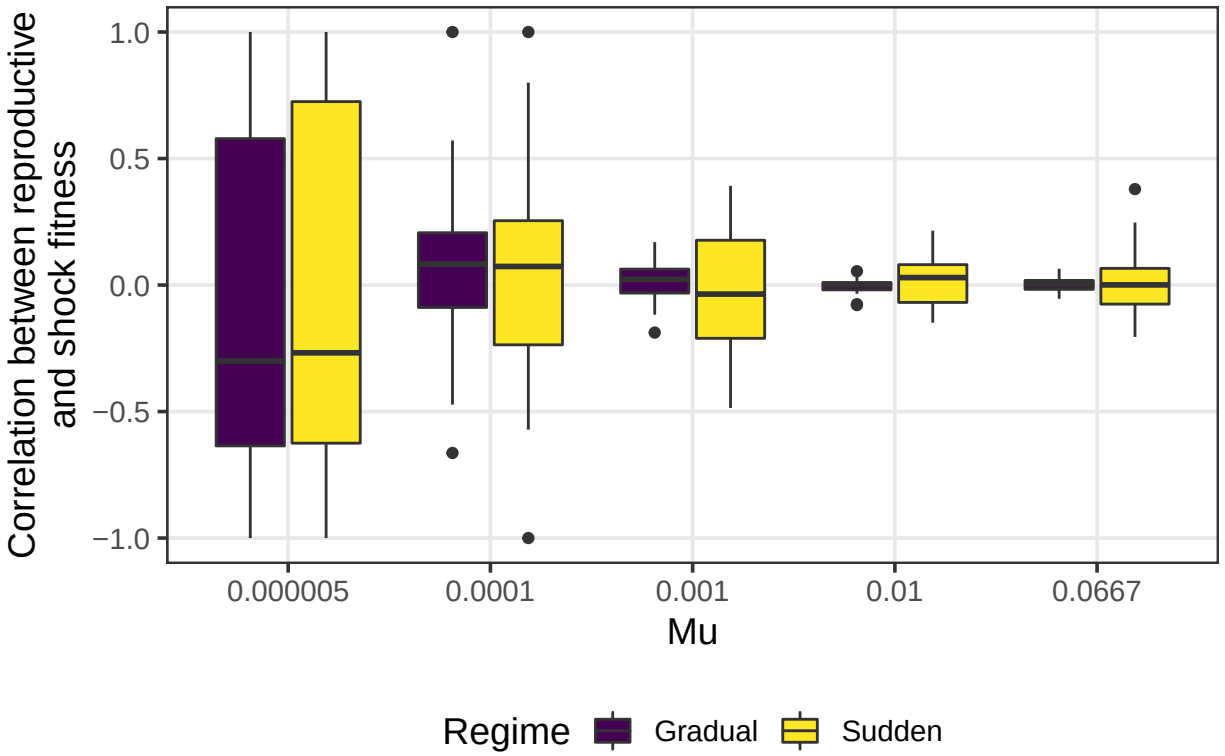
We see relatively high fitnesses in both landscapes. The next obvious set of questions to ask concern the way the population is traversing the two landscapes.

One possibility is that successful populations are finding regions of the fitness landscape that are highly fit in both landscapes. To assess this possibility, we measure the correlation between reproductive fitness and shock fitness among evolved genotypes in the population. We measure this correlation for each population:

```
# Make new dataframe with all the correlations
df <- pops %>%
        # Group by run (as indicated by random seed used for each run)
        # and by genome within run
        group_by(seed, Genome) %>%
        arrange(Gen) %>%  # Sort by generation so time series are in order
        # Group by run (as indicated by random seed used for each run)
        # and by genome within run
        filter(last(Gen)==50, first(Gen) < 48, row_number() == 1) %>%
        # Condense each genome into a single row -
        # these values should all be identical across rows, so first is as good
        # as any other value
        summarise(Gen = first(Gen),
                  Fitness= first(Fitness),
                  ShockFitness=first(ShockFitness),
                  regime=first(regime),
                  k=first(k),
                  mu=first(mu),
                  .groups = "drop_last"  # Keep data frame grouped by run
```

```
                    ) %>%
        # Calculate correlation between fitness in each environment for run
        summarise(r=cor(Fitness, ShockFitness, method="spearman"),
                  regime=first(regime),
                  k=first(k),
                  mu=first(mu)
                  )

# Clean up significant digits on mutation rate for label
df$mu <- format(df$mu, digits=3, drop0trailing = TRUE, format="g")
df <- df %>% mutate(mu = case_when(
  mu=="0.066667" ~ "0.0667",
  TRUE ~ mu
  )
)

# Actually plot the data

ggplot(df %>% filter(k== 10)) +
  geom_boxplot(aes(x=as.factor(mu), y=r, fill=regime)) +
  xlab("Mu") +
  scale_fill_viridis_d("Regime") +
  ylab("Correlation between reproductive\nand shock fitness") +
  theme_bw(base_size = 14) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "grey91"),
        strip.text.y = element_text(face = "italic"),
        legend.position = "bottom") +
  ggtitle(fig_nums("corr"))
```

Figure 9: Landscape correlation in evolved genomes



This implies that different genotypes have high fitness in each landscape rather than a single genotype being good at both. To understand better what is happening, it would be helpful to more directly examine the traversal of these landscapes. Unfortunately, doing so is very challenging with an N=15 landscape due to the high dimensionality. To build intuition, we undertake a case study in a lower-dimensional landscape.

0.3 Lower dimensional case study

Here, we use a pair of N=6, K=1 fitness landscapes. We plot these landscapes as graphs in which noddes represent genotypes and edges represent single-mutation adjaceny between genotypes.

First, we set up the data:

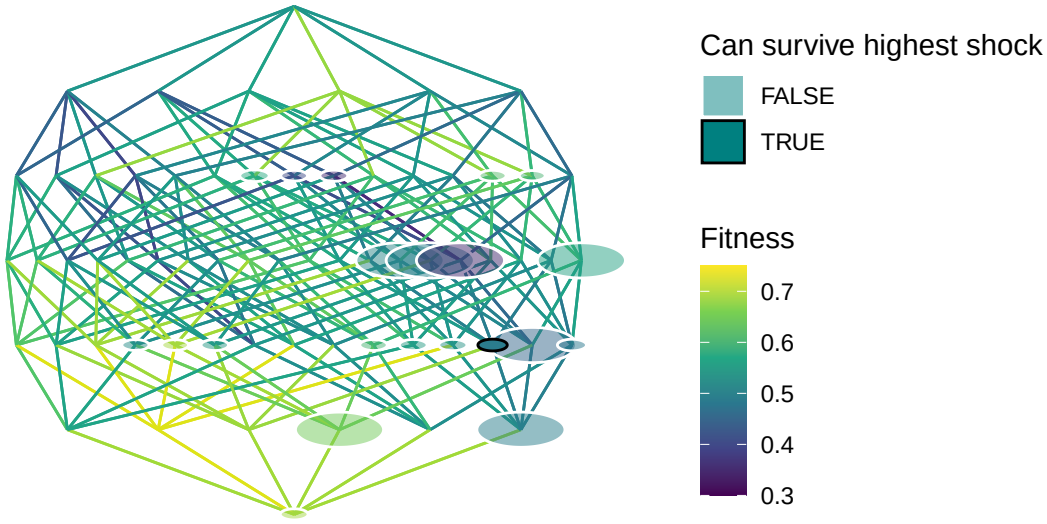
```
# Read in a file containing all the edges for this fitness landscape graph
edges <- read_csv("edges_6.csv")

# Read in fitnesses from reproduction fitness landscape
repro_fits <- read_csv("repro_fits_1963.csv")

# Read in fitnesses from shock landscape
shock_fits <- read_csv("shock_fits_1963.csv")

# Clean up dataframes
colnames(repro_fits)[1] <- "name"
colnames(shock_fits)[1] <- "name"
colnames(shock_fits)[3] <- "shock_fitness"

# Make sure name is the same type in all data frames for easy joining
repro_fits$name <- as.character(repro_fits$name)
shock_fits$name <- as.character(shock_fits$name)
```

```
edges$from <- as.character(edges$from)
edges$to <- as.character(edges$to)

# Read in population time series data from run of evolutioon
pop <- read_table("../outs/popt-N6-k1-s1963-r0.001-Gradual.txt",skip = 3)

# Clean up data frame
colnames(pop) <- c("gen", "genome", "count", "fitness",
                   "shockfitness", "placeholder")
pop$placeholder <- NULL
pop$genome <- as.character(pop$genome)

# For now we're only worrying about the last generation
pop <- pop %>%filter(gen == max(gen))

# Build tidygraph data frame containing all the data we need
graph <- as_tbl_graph(edges)  # Make graph object
graph <- graph %>% full_join(repro_fits) %N>%  # Join in repro landscape data
                   # Set up data for positioning nodes in an intelligible way
                   group_by(ones) %>%
                   arrange(as.numeric(name)) %>%
                   mutate(pos = row_number(), total = n()) %>%
                   ungroup()
# Join in shock landscape data
graph <- graph %>% full_join(shock_fits, by = c("name", "ones"))
graph <- graph %>%  # Join in population data frame
            full_join(pop %>%  # Only want one row per genome so we summarize
                        group_by(genome) %>%
                        arrange(gen) %>%
                        summarise(count=max(count), gen=first(gen)),
                                  by = c("name" = "genome")) %N>%
                        mutate(count=ifelse(is.na(count), -1, count),
                               gen=ifelse(is.na(gen), -1, gen)) %>%
                        mutate(count=ifelse(count > 1000, 1000, count))

# Set up edge data
graph <- graph %E>%
            mutate(
              fitness = if_else(.N()$fitness[to] > .N()$fitness[from],
                                .N()$fitness[to],
                                .N()$fitness[from]),
              shock_fitness = if_else(.N()$shock_fitness[to] >
                                        .N()$shock_fitness[from],
                                      .N()$shock_fitness[to],
                                      .N()$shock_fitness[from]),
              followed = if_else(is.na(.N()$count[to]) |
                                   is.na(.N()$count[from]),
                                 0,
                                 if_else(.N()$count[to] > .N()$count[from],
                                         .N()$count[to],
                                         .N()$count[from]))
              )
```

Next, we visualize the data for the reproductive landscape:

```
ggraph(graph, layout = 'manual', x=max(ones)/2 * (pos)/(total+1), y=ones) +
    geom_edge_link(aes(color=fitness)) +
    geom_node_circle(aes(fill=fitness,
                         r = (.05*(count>0)) + sqrt(count)/200,
                         alpha=shock_fitness>=.66,
                         color=shock_fitness>=.66)) +
    theme(legend.position = "none") +
    # Set up legend and appearance
    theme_graph(base_family = 'Helvetica') +
    scale_fill_viridis_c("Fitness", limits=c(.3,.75)) +
    coord_fixed(ratio = .4) +
    scale_edge_color_viridis("Fitness", limits=c(.3,.75)) +
    scale_alpha_discrete("Can survive highest shock", range = c(.5,1)) +
    scale_color_manual("Can survive highest shock",
                       values=c("FALSE"="white", "TRUE"="black"),
                       guide = guide_legend(override.aes =
                                              list(fill = "#008080") )) +
    guides(edge_color=FALSE) + # Edge color legend is same as fill
    ggtitle(fig_nums("repro_landscape"))
```

Figure 10: Full reproductive fitness landscape

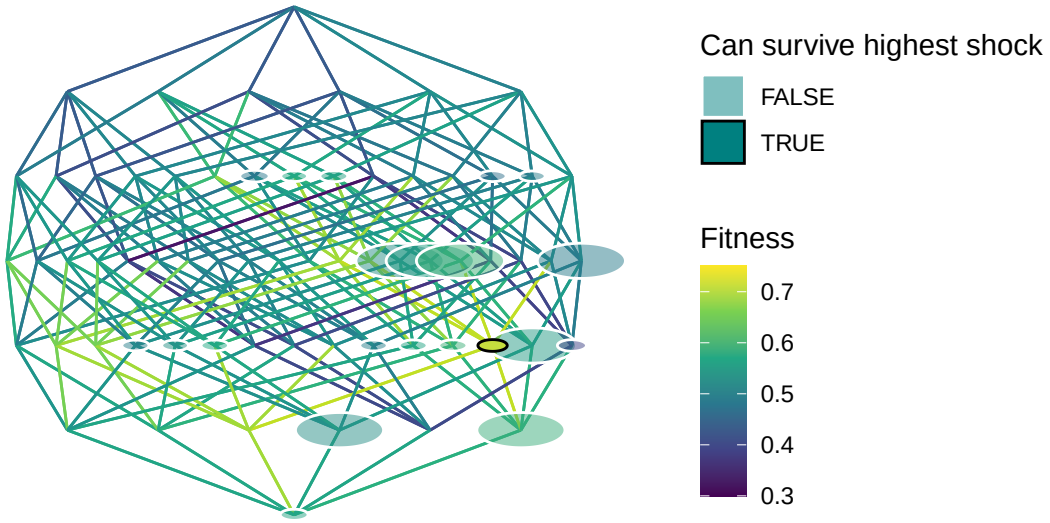


And the shock landscape:

```
ggraph(graph, layout = 'manual', x=max(ones)/2 * (pos)/(total+1), y=ones) +
    geom_edge_link(aes(color=shock_fitness)) +
    geom_node_circle(aes(fill=shock_fitness,
                         r = (.05*(count>0)) + sqrt(count)/200,
                         alpha=shock_fitness>=.66,
                         color=shock_fitness>=.66)) +
    theme(legend.position = "none") +
    # Set up legend and appearance
    theme_graph(base_family = 'Helvetica') +
    scale_fill_viridis_c("Fitness", limits=c(.3,.75)) +
    coord_fixed(ratio = .4) +
    scale_edge_color_viridis("Fitness", limits=c(.3,.75)) +
```

```
    scale_alpha_discrete("Can survive highest shock", range = c(.5,1)) +
    scale_color_manual("Can survive highest shock",
                       values=c("FALSE"="white", "TRUE"="black"),
                       guide = guide_legend(override.aes =
                                              list(fill = "#008080") )) +
    guides(edge_color=FALSE) +  # Edge color legend is same as fill
    ggtitle(fig_nums("shock_landscape"))
```

Figure 11: Full shock fitness landscape



Note that in these graphs: node color indicates fitness in the reproductive and shock fitness landscapes respectively, edge color also indicates fitness in the landscape being visualized (specifically, the fitness of the fitter of the two genotypes being connected by the edge), and node size indicates final population size of that genotype. Node opacity indicates whether that genotype is capable of surviving the maximum shock in the shock landscape, as does node outline color.

From this data visualization, it appears that (in this run at least) the genotypes that are successful in each landscape are indeed very different. The population of genotypes that are fit in the reproductive landscape grows quickly but is wiped out at each shock event and most then be repopulated by mutations from the genotypes that can survive the shocks. This phenomenon explains the pattern of oscillations with increasing amplitude observed in the graphs of unique genotypes and population size.