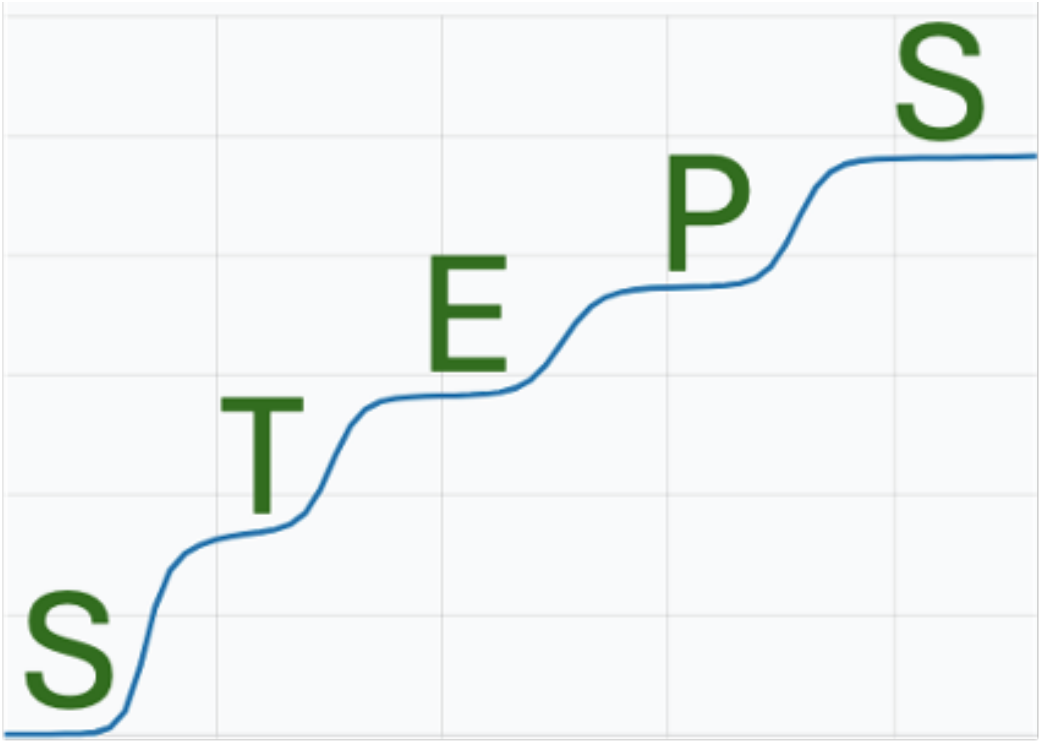

STEPS User Manual

Version 2.0.0

Devin M. Lake
Zachary W. D. Matson
Richard E. Lenski

August 5, 2025

Contents

1 Overview

1.1 What is STEPS?

STEPS stands for **S**erially **T**ransferred **E**volving **P**opulation **S**imulator. That's a mouthful, so let's unpack things a bit.

1.1.1 Experimental Evolution

Evolution is the biological process by which populations of organisms change over time as mutations occur and spread through a population. Some mutations are called beneficial because they enhance an organism's fitness. Beneficial mutations tend to increase in frequency over time because the mutants grow faster than their progenitors.

People usually think of evolution as something that happened in the past. In fact, however, evolution is ongoing in the world around us. With some organisms, one can even perform experiments to study evolution in action. Bacteria and other microbes are often used in evolution experiments because of their fast generations, large populations, and compact size.

One such experiment is called the Long-Term Evolution Experiment[1], or LTEE for short. That experiment started in 1988 and continues to this day, with 12 populations of *E. coli* bacteria propagated daily in a simple laboratory environment. Over the decades, the bacteria have undergone more than 80,000 generations.

The LTEE serves as a touchstone for STEPS because the program's implementation reflects the procedures used in the LTEE, as discussed below. Moreover, many of the default parameters in STEPS have been estimated from the LTEE. The STEPS program also allows one to vary the parameters in order to study their effects on the dynamics of evolution.

1.1.2 Serial Transfers and Population Growth

The microbial populations in evolution experiments are commonly propagated by a method called serial transfer. The microbes grow and multiply until they deplete the nutrients present in a liquid medium. They then sit there at some final population size until a specific fraction of the culture is removed and transferred to fresh medium. This procedure is then repeated daily (or at some other regular interval).

The dilution factor defines the fraction of the culture that is serially transferred. For example, a dilution factor of 100 means that 1%, or $\frac{1}{100}$, of the population is moved to fresh medium at each transfer event. The culture volume (i.e., amount of medium) is kept the same at every transfer.

[1] the-ltee.org

After each transfer, the population starts with an initial population size that is equal to the final population size divided by the dilution factor. The population then grows in the fresh medium until it reaches the same final population size. The population stops growing because, given the same medium and culture volume, the cells will deplete the available nutrients. In other words: initial population size x dilution factor = final population size. For example, the final population size in the LTEE is ~5 x 10^8 cells and the dilution factor is 100, so that each population starts with ~5 x 10^6 cells after the daily transfer.

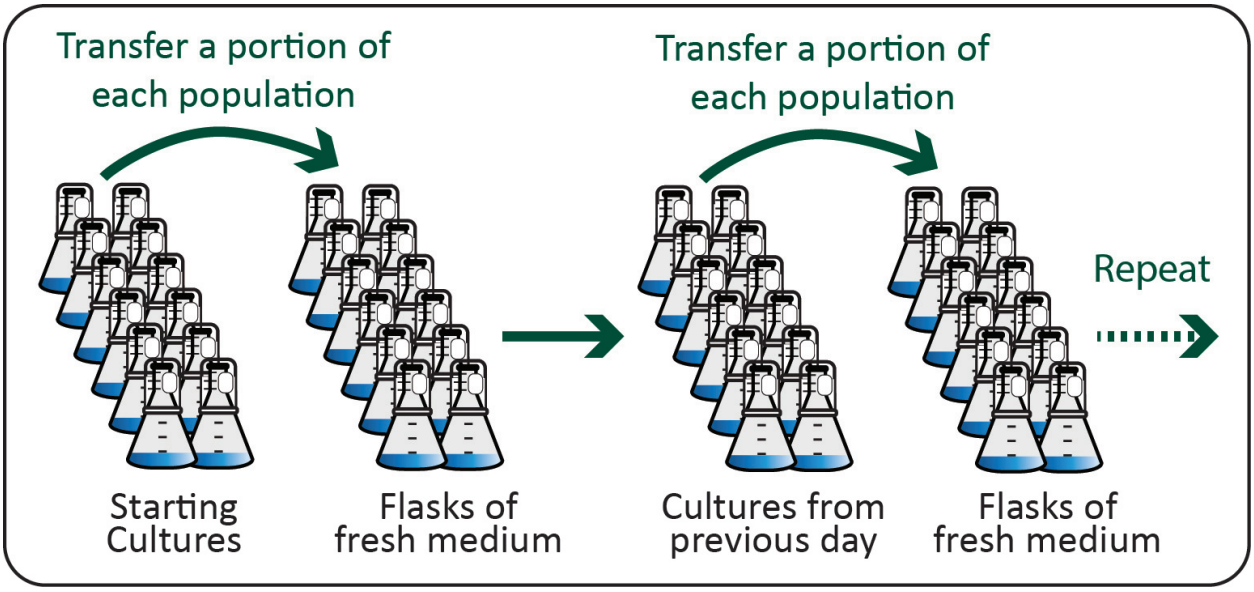

Figure 1: Serial transfer protocol.

Most microbes reproduce by binary fission, such that a growing cell doubles in size and divides into two daughter cells. In that case, we can calculate the number of generations between successive transfers using the dilution factor: generations = $\log_2$(dilution factor). Note that the number of generations is an integer only if the dilution factor is some power of two (e.g., 2, 4, 8, 16, ...). For example, the 100-fold dilution and regrowth in the LTEE allows $\log_2(100) \approx 6.6$ generations between each daily transfer. Note also that some cells in the initial population may undergo more doublings than other cells. Thus, the number of generations calculated using the dilution factor is an average for the population as a whole.

Figure 1 shows schematically the serial-transfer protocol. The dilutions and population regrowth could take place even without evolution, but that would not be very interesting. The reason for doing repeated serial transfers is because it allows evolution to occur as individuals with new mutations appear in the population. In addition to repeated transfers, most evolution experiments include replicate populations that start from the same ancestral strain, but which then evolve independently over time. The STEPS software also allows one to run multiple replicate populations.

1.1.3 Mutations and Fitness

Bacteria and some other microbes reproduce asexually, which is sometimes called clonal reproduction. That means that both daughter cells are identical and exact genetic copies of their parent—unless a mutation happens during the replication process. (In nature, microbes can sometimes also acquire genetic differences by picking up bits of DNA from other cells. The STEPS software does not incorporate this process.)

When mutations occur, they can have many different effects. In STEPS, the focus is on how a mutation affects the microbe's fitness, which is defined here (and in many experimental studies) as the growth rate of the mutant genotype relative to the ancestral strain used to start the simulation (or experiment). Some mutations are called beneficial because they increase fitness, others deleterious because they reduce fitness, and still others neutral because they have no effect on fitness. Beneficial mutations are often considered the most interesting ones because they fuel the process of adaptation by natural selection, which allows organisms to become more fit to their environments over time. Deleterious mutations, by contrast, tend to disappear precisely because they reduce the rate at which progeny cells are produced.

STEPS allows one to include and define the rates for one or more of these three classes of mutation. For example, a beneficial mutation rate of 1×10^{-6} means that one new beneficial mutation will occur, on average, in 10^6 (i.e., million) progeny cells. Each mutation occurs in an individual that is chosen at random from the population. For both beneficial and deleterious mutations, STEPS also allows one to specify the expected distribution of fitness effects. For example, mutations that increase fitness by 1% tend to be more common than those that increase fitness by 10%. These distributions will be discussed later in this user manual.

1.2 Core Assumptions

The STEPS software implements several core assumptions. These include:

- Populations grow to a maximum population size, at which point growth stops.
- Populations are diluted by a fixed factor, and then growth resumes until the population again reaches the maximum size.
- Mutations occur as a population grows, generating new lineages. The mutations occur stochastically (i.e., in individuals chosen at random from the entire population).
- A mutation may increase or decrease the fitness (i.e., growth rate) of individuals that have the mutation, or it may have no effect.
- After every transfer, each lineage grows deterministically at a rate that reflects the net effect of any and all mutations that it has accumulated. All of the lineages stop

growing when the total population reaches its maximum size, at which point the next transfer occurs.

- The individuals that get transferred are chosen at random from the population. Thus, mutations can affect an organism's growth rate, but they have no effect on the chance of surviving the periodic dilutions.

1.3 Two Versions of STEPS

In order to make STEPS as broadly accessible as possible, the software supports two different versions for running the simulations. One version uses a web-based graphical user interface, while the other version uses the command line and provides some additional options. Both versions use the same underlying code for running the actual simulations.

The web-based version is designed with the following points in mind:

- It allows the user to run simulations quickly and easily, which can help develop the user's intuition about the evolutionary process and the resulting dynamics.
- The web-based version produces graphical results that are easily visualized, which can be valuable in educational settings.
- It provides access to many features of the simulation package for those researchers who are less computationally adept.
- The web-based version also allows the data generated during the simulations to be downloaded, so that students and researchers alike can analyze the results using statistical and other familiar tools.

The command-line version provides access to more advanced options for research:

- For example, the command-line version allows one to use different effect-size distributions for deleterious mutations, which is not available with the web-based interface.
- By providing more time and/or computational power, one can run simulations for longer (more generations), with more replicate populations, with larger population sizes, and/or with higher mutation rates (leading to more lineages that must be tracked) than would be practical using the web-based version.
- Similarly, one can use the command-line version to set up and run simulation experiments that systematically explore large spaces for multiple parameters.

Because both versions of STEPS use the same underlying code, they can be used during different stages of a project as well. For example, one can use the web-based version to see how long a small simulation takes to run, and then use that information to estimate the time required for a larger simulation using the command-line version.

1.4 Roadmap for the User Manual

Chapter 2 presents the web-based version of STEPS. It describes the parameters that can be varied, and it walks the user through several examples of basic simulations with different inputs. It then presents a few more advanced options that are initially hidden, but which can be used even with the web-based version. Finally, it explains how the data from the simulations can be saved during the runs and then downloaded for later analysis using other tools.

Chapter 3 describes the operation of the command-line version of STEPS using simple examples. It then explains all of the parameters and options (including both inputs and outputs), some of which are not available in the web-based version.

Chapter 4 provides a detailed explanation of the model choices and computations that power the dynamical heart of the STEPS software. The mathematics behind the growth dynamics, mutation distributions, and other important features are also described in this chapter.

2 Running STEPS on the Web

2.1 Getting Started

Turn on your computer, open a web browser, and navigate to the STEPS Portal[2]. You should see a page that looks like Figure 2.

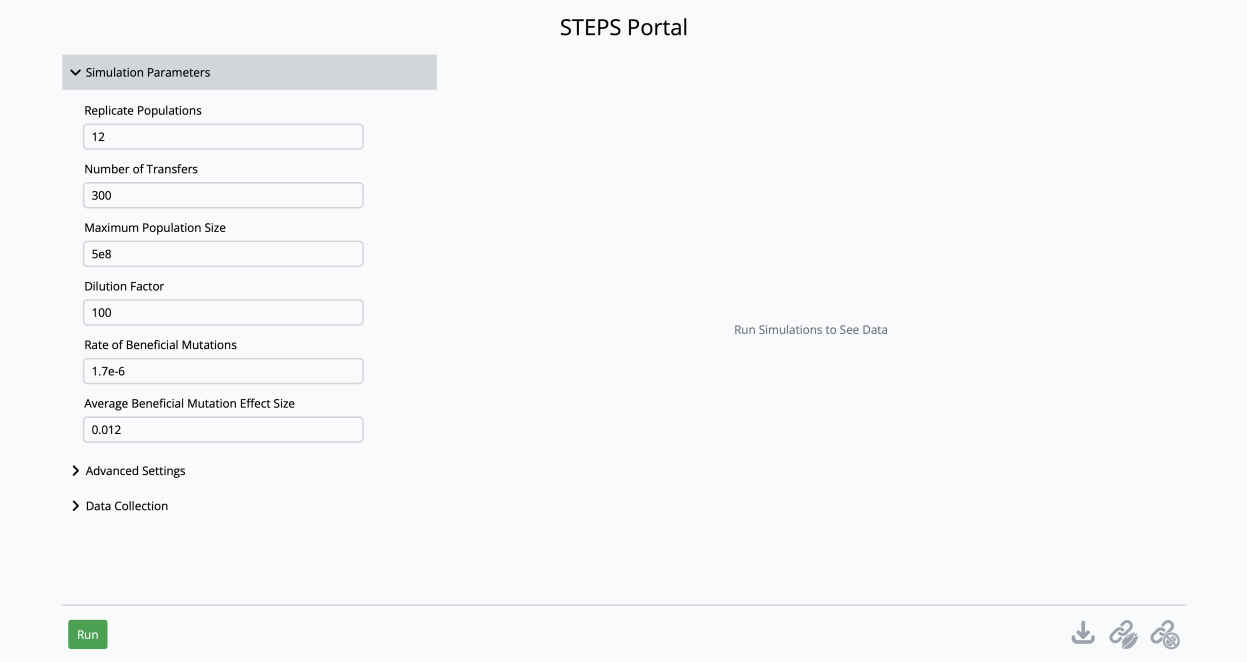

Figure 2: Screenshot showing the STEPS portal and menu.

On the left, you will see a list of Simulation Parameters. You can click on Advanced Settings and Data Collection to see more parameters and other options. To start, we will leave things with the default settings.

2.1.1 Running STEPS

Click on the green button at the lower left that says Run. Voila, you are running a simulated evolution experiment! After only a few seconds, you should see something like what is shown below in Figure 3. The default parameters shown on the portal correspond to the LTEE, and the number of daily transfers represent the first 10 months of that experiment.

Your runs won't look *exactly* like those in Figure 3, because each population uses random numbers to generate mutations and to sample the individual cells that survive the dilutions during each of the 300 transfers. There are 12 trajectories because there are 12 Replicate Populations, as you can see in the top field of the menu for the Simulation Parameters. The Number of Transfers is shown in the second field of that menu.

[2] `https://steps-portal.pages.dev`

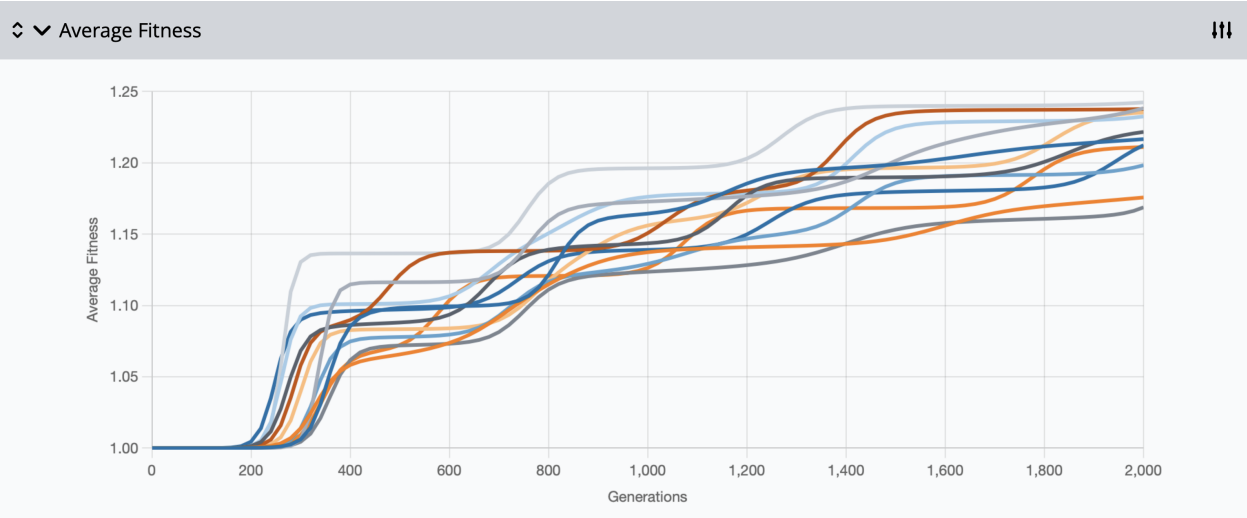

Figure 3: Screenshot showing trajectories for Average Fitness with default parameters.

2.1.2 Transfers and Generations

You will also see the Dilution Factor is set to 100 in the menu's fourth field. Recall from Section 1.1.2 that the number of generations between successive transfers is ~6.6 when the Dilution Factor is 100. That means that ~2,000 generations have elapsed after the 300 daily transfers, as shown along the x-axis.

The y-axis shows the Average Fitness of a population. In STEPS, fitness is expressed on a relative scale, starting from the ancestral value of 1. A value above 1 means that a genotype grows faster than the ancestor. The simulations show that the Average Fitness is increasing in all 12 populations over time. At the end of the runs, most of the evolved cells are growing more than 20% faster than their ancestor. Beneficial mutations arise and spread through a population because the cells with those mutations grow faster than the cells without them.

2.1.3 Caution: Avoiding Slow and Failed Runs

Before moving on, here's a note of caution about choosing the parameter values for your simulations. STEPS runs will generally take longer to finish in rough proportion to four inputs that you control: Replicate Populations, Number of Transfers, Maximum Population Size, and the total Mutation Rate (including beneficial, neutral, and deleterious mutations). A 10-fold increase in any of these parameters will cause your simulations to take about 10 times as long to complete.

This slowdown happens because the STEPS program tracks all the lineages that are generated, and the number of lineages increases when there are more populations, more individuals per population, more mutations per individual, and/or more transfers. In the exercises below, we adjust the numbers of populations and transfers to avoid exceedingly

slow runs, while still illustrating biologically important features of the STEPS program. The actual time required will also depend somewhat on your computer's speed. In any case, if you find that a run is proceeding too slowly, you can stop it by pressing the gray Pause button; then change one or more of the input values, and press the green Restart button.

Certain other parameter settings can also cause problems. For example, some runs will fail when the Beneficial Mutation Size is too large (above ~0.3) or the minimum population size (i.e., Maximum Population Size / Dilution Factor) is too small (below ~25). If your run fails, try changing the input parameters to less extreme values.

2.2 How Many Beneficial Mutations?

That's a simple question, but it has two answers, depending on how we approach the question.

2.2.1 How Many Occur?

In the menu, you can see that the Maximum Population Size is 5e8 (scientific notation for 500 million). With the Dilution Factor of 100, the starting population after the dilution is reduced to 5e6. Then, 4.95e8 new cells are produced as the population grows back to its maximum size before the next serial transfer.

The menu also shows that the Rate of Beneficial Mutations is 1.7e-6 (1.7 per million cells). With 4.95e8 new cells produced during the regrowth between successive transfers, that means that ~840 beneficial mutations occur in each population during every transfer cycle, and ~250,000 over the course of 300 cycles.

2.2.2 How Many Accumulate?

However, most of the beneficial mutations that occur are lost from the population, for two reasons. First, many of them are lost during the dilution process before they have become common. For example, a beneficial mutation that arose during the last generation before a transfer is present in only one cell. That mutation has a 99% chance of going extinct during the next 100-fold dilution. Even if the mutation survives, and even if its lineage grows somewhat faster than other lineages, it might still only have, say, 110 descendants after the population as a whole has increased 100-fold, and so that lineage could be a bit unlucky and go extinct at the next dilution. These extinctions reflect the random process called genetic drift.

Second, some lineages with beneficial mutations go extinct when they are out competed by other lineages with mutations that are even more beneficial. As explained in the previous subsection, hundreds of beneficial mutations occur in every transfer cycle, and many more than that occur over the course of many cycles. Imagine a beneficial mutation that gives a 5% advantage, and that is lucky enough to survive several dilution cycles, so that it becomes

firmly established in the population. Now imagine that another beneficial mutation that arose in a different lineage confers a 10% advantage, and it too survives and begins to sweep through the population. At first, both lineages compete mostly against their common ancestor, but as they increase in abundance they will increasingly compete with one another. Over time, the lineage with the 10% benefit will out compete the one with the 5% benefit (unless later mutations further complicate the picture). This kind of extinction reflects a non-random process called clonal interference.

To see how many beneficial mutations actually survive and accumulate over time, scroll down and you should see a second panel with data. (If the second panel is missing, go to the Data Collection menu, make sure the boxes are checked for both Average Fitness and Average Accumulated Mutations, and press the green Run button.) As seen in Figure 4 below, this second panel shows the trajectories for the Average Accumulated Mutations in the 12 populations.

So, there's the second answer: Of the ~250,000 beneficial mutations that occurred in each lineage during the 300 transfers and 2,000 generations, a typical lineage has accumulated only a handful of mutations.

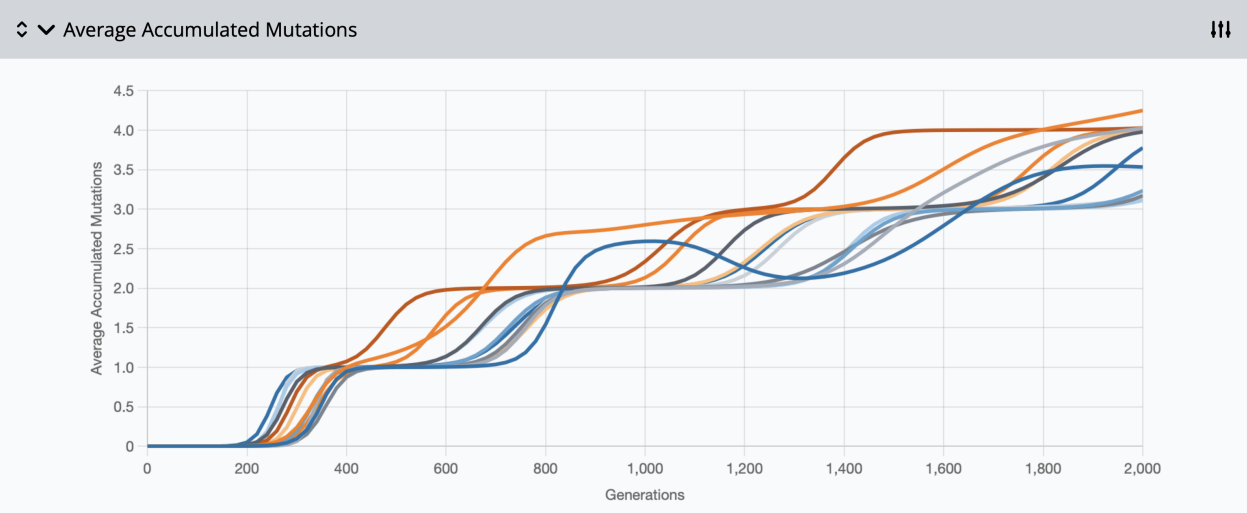

Figure 4: Trajectories for Average Accumulated Mutations with default parameters.

2.3 Why Do the Trajectories Increase in a Stepwise Manner?

You probably noticed that Average Fitness (Figure 3) and Average Accumulated Mutations (Figure 4) typically follow step-like trajectories, especially in the early generations. When a new beneficial mutation first occurs, it starts at an extremely low frequency. Let's consider a mutant that survives the first dilution as a single cell. In that case, its frequency is 1 / 5,000,000 or just 0.00002% of the population. Even if it grows, say, 10% faster than the other cells and survives later dilutions, it will take many generations to reach 0.0002%, and then many more generations to reach 0.002%, and so on. Only after it becomes, say, 20%

of the entire population will the Average Fitness have increased by 2% (20% of 10%), at which point the fitness trajectory is moving upwards at an appreciable rate.

The second step-like increase typically requires two things. First, the lineage with the first beneficial mutation that ultimately prevails has to become a large enough subpopulation that it is likely to get another beneficial mutation. It's unlikely to get one when that lineage has reached 50,000 cells, for example, because the beneficial mutation rate is less than 1 in 500,000. Second, when that lineage eventually gets the second beneficial mutation, the double mutant again starts out as a single cell, and the mathematical considerations explained in the preceding paragraph come into play again. And so on and so forth as evolution proceeds. These step-like dynamics occur not only in the STEPS runs, they are also seen in the LTEE and other evolution experiments with microbes.

2.3.1 Reconciling the Fitness Gains and Average Beneficial Effect Size

You can see that the first few steps in the populations led to roughly 5-10% fitness gains. And if you have a keen eye, perhaps you saw in the menu of Simulation Parameters that the Average Beneficial Effect Size is 0.012, or just 1.2%. Why are the steps clearly larger than the average that was input?

Not all beneficial mutations are equally beneficial because they have different effect sizes. When a new beneficial mutation occurs in a STEPS run, its effect size is drawn at random from an exponential distribution. That distribution has a long tail, in which a few mutations have much larger effect sizes than the average. Figure 5 below shows the distribution when the Average Beneficial Effect Size is 0.012.

Now recall the process of clonal interference that was explained in Section 2.2.2 above. Given the default Maximum Population Size and Rate of Beneficial Mutations, hundreds of beneficial mutations occur in each growth cycle between transfers. Even though most of them are lost due to random genetic drift caused by the dilutions, there are still many that survive. And those surviving beneficial mutations will compete with one another as they increase in frequency, with only the most fit—those with beneficial mutations with the largest effect sizes—ultimately prevailing.

2.4 Changing Some Parameters

So far, we have used the default simulation parameters. We will now see what happens when we change some of those parameters.

2.4.1 Extending the Runs

Let's start by increasing the Number of Transfers, thus allowing the simulated populations to evolve for much longer. Go to the second field in the menu and change it from 300 to 7500. Then press the green Run button again.

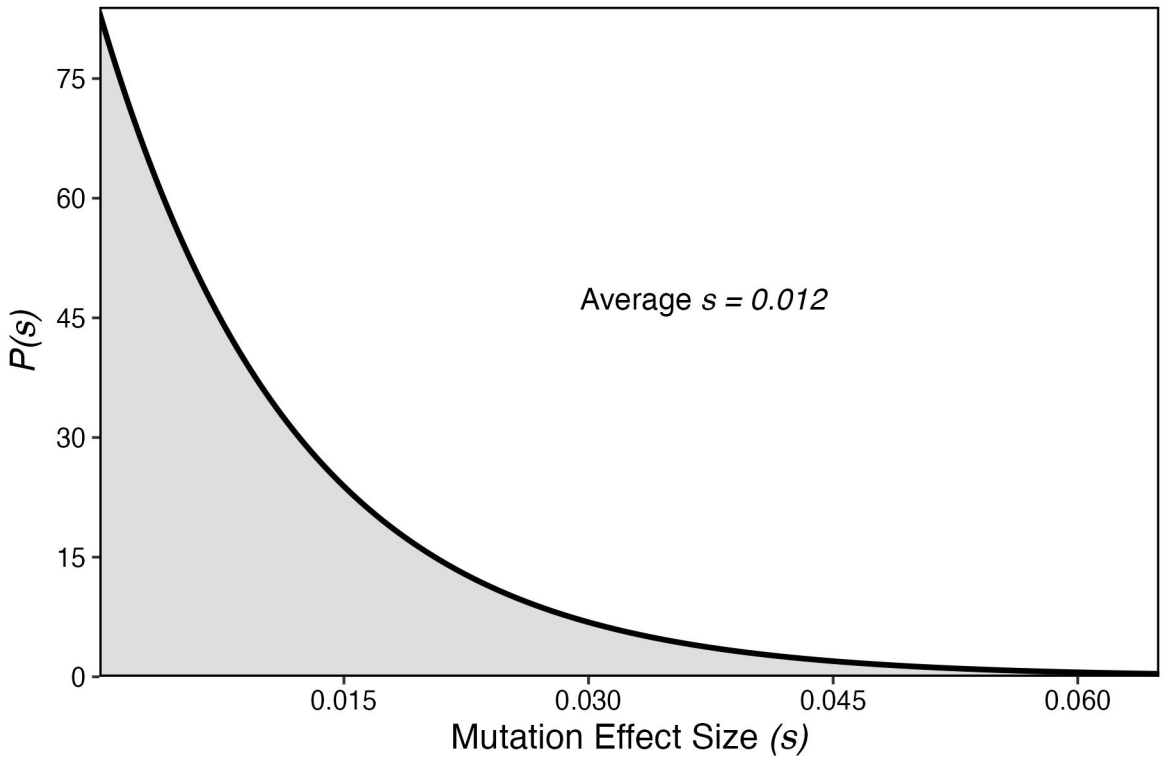

Figure 5: Distribution of effect sizes for beneficial mutations with the default parameters. The y-axis shows the relative probabilities of the various effect sizes. The x-axis has been truncated in this figure.

With the same 100-fold dilutions and regrowth, the 7,500 transfers correspond to 50,000 generations. That required over 20 years of daily transfers in the actual LTEE, but the simulated runs will take only a few minutes. You'll generate a picture that looks something like Figure 6 below.

There's a big change in the fitness trajectories for the later generations compared to the first 2,000 generations—the rate of increase slows way down. While fitness increased by about 20-25% in the first 2,000 generations (Figure 3), it reached only about 40-50% by 10,000 generations and about 60-70% after 50,000 generations (Figure 6).

Once again, these trajectories are similar to those seen in the LTEE with bacteria. The slowdown occurs because, as organisms become more fit, it usually becomes harder to make further gains. This effect is called diminishing-returns epistasis. Epistasis is a word from genetics; it refers to situations where the effect of a later mutation depends on the presence of some earlier mutation or mutations. In this context, the beneficial effects of later mutations become smaller when more beneficial mutations have already accumulated.

There are two more things you might notice when you compare the fitness trajectories from the longer and shorter runs. First, you might think the step-like dynamics have disappeared

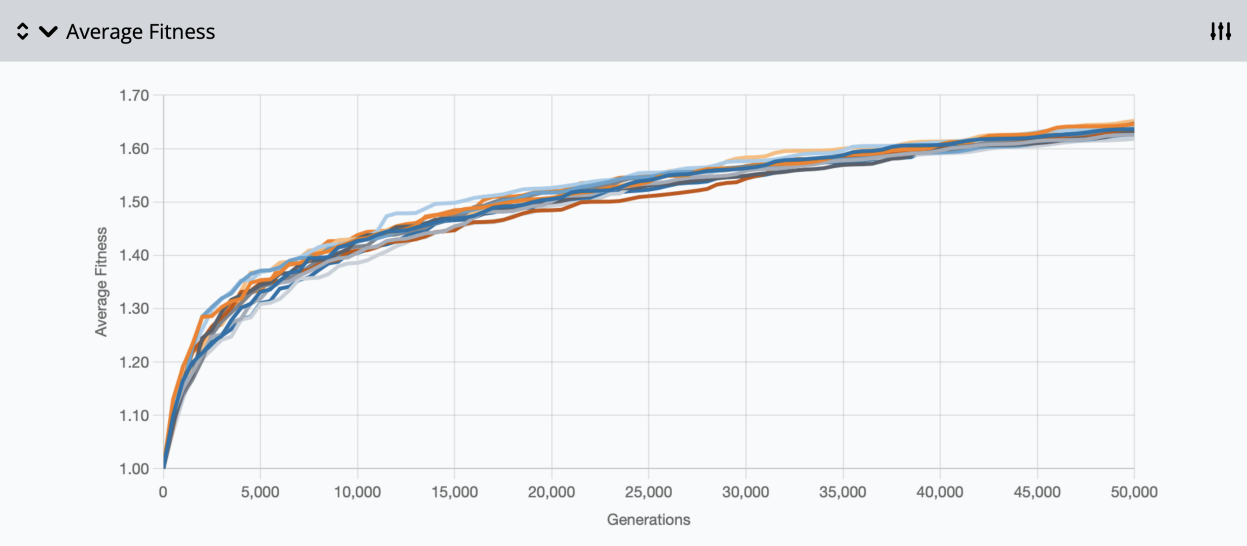

Figure 6: Trajectories for Average Fitness with Number of Transfers increased to 7,500.

in the longer trajectories. In fact, however, the stepwise gains are still there, but they aren't as obvious visually when there are more of them, and especially because the steps tend to become smaller due to the diminishing-returns epistasis. Second, you might think the trajectories for the replicate populations look more similar in the longer runs. Again, that difference is mostly a visual artifact. Look closely at the first 2,000 generations in the longer runs (Figure 6), and you will see that the 12 trajectories lie close to one another early as well as late. It's only when you zoom in on a short interval that you see the random effects—the timing of beneficial mutations, their effect sizes, and their survival during the transfer dilutions—that cause the replicate populations to vary.

2.4.2 Optional: Playing with Epistasis

There's a parameter in the Advanced Settings called Strength of Epistasis. The default value was estimated from the fitness trajectories seen in the LTEE. The phenomenon of diminishing-returns epistasis has been observed in many evolution experiments with microbes, but it is not universal and the strength of the effect, when it occurs, will depend on the organism and the environment.

If you're curious, you can try setting the Strength of Epistasis to 0. If you do that, however, you should also reduce the Number of Transfers back to 300. You will see that average fitness increases to levels roughly similar to what you saw after 7,500 transfers with the Strength of Epistasis set to its the default value of 6.0. If you now run it for 7,500 transfers with the Strength of Epistasis set to 0, you will see that fitness increases exponentially, and you would need to plot the data on a logarithmic scale in order to make sense of it.

In fact, the web version of STEPS allows you to change the scaling of the graphical results. To do so, click on the three slider bars at the far right of the Average Fitness graph. Then

select Log for the y-axis scale. You will see the fitness trajectories are now approximately linear, because a logarithmic transformation linearizes exponential data.

2.5 Adding Neutral and Deleterious Mutations

Up to this point, we have run simulations with only beneficial mutations. That often makes sense, because neutral mutations, by definition, have no effect on fitness. And deleterious mutations, while they have fitness effects, tend to be eliminated by natural selection.

On the other hand, beneficial mutations in biological organisms are usually pretty rare—much rarer, in fact, than deleterious and neutral mutations. So, it's important to examine and understand when they can be ignored, and when they must be taken into account.

From a practical standpoint, the STEPS software runs more slowly when deleterious and neutral mutations are included. The slowdown occurs because STEPS tracks every lineage. If we use realistic rates of deleterious and neutral mutations, then STEPS runs a lot slower, unless we also reduce the population size. So, at first, we will reduce the number of transfers, and we will run only 3 populations instead of the default 12 replicates.

We will then examine something interesting that happens with deleterious mutations when we reduce populations to very small sizes.

Note: One can use the command-line version of the STEPS software, described in Chapter 3, to run much longer, larger, and more complex simulations.

2.5.1 Neutral Mutations Only

In this experiment, we will turn off beneficial mutations while adding neutral mutations. Go the STEPS menu and set the Replicate Populations to 3, the Number of Transfers to 300, and the Rate of Beneficial Mutations to 0. Then open the Advanced Setting and set the Rate of Neutral Mutations to 0.001 (or 1e-3). This may seem like a small number, but it is more than 500 times the rate of beneficial mutations used in our previous runs. That means the STEPS software must keep track of many more lineages. Also, scroll down to the Data Collection menu and make sure that both Average Accumulated Mutations and Average Fitness are checked.

Press the Run button. If you watch the top data panel, you'll see that Average Fitness doesn't change at all. That's what we expect, because all the mutations are neutral, meaning they have no effect on fitness. Now look at the bottom panel. You will see three nearly identical straight lines that rise from 0 to ~2 Average Accumulated Mutations at 2,000 generations (300 transfers). That is also what we expect. Without selection, neutral mutations should accumulate in a steady clock-like manner. With a mutation rate of 0.001 for 2,000 generations, we expect about 0.001 x 2000 = 2 mutations in the average individual, just as we see in the simulations.

2.5.2 Deleterious Mutations Only

Now let's run a similar experiment, except with deleterious mutations on and both neutral and beneficial mutations turned off. Set the Replicate Populations to 3, the Number of Transfers to just 30, and the Rate of Beneficial Mutations to 0. In the Advanced Settings, set the Rate of Neutral Mutations to 0 and the Rate of Deleterious Mutations to 0.01. Finally, make sure again that both Average Fitness and Average Accumulated Mutations are checked in the Data Collection menu, and then press Run.

The web-based version of STEPS uses a uniform, or flat, distribution for the effect sizes of deleterious mutations. That means that a deleterious mutation is just as likely to reduce fitness by 80% as it is by 20%, or by any other value between 0% and 100%. The average negative effect size across the uniform distribution is 50%.

The top data panel in your runs should look like Figure 7, with three nearly identical trajectories for Average Fitness. Without beneficial mutations, fitness can only decline or stay flat. While fitness declines slightly in these populations, the losses are far less than the gains when beneficial mutations occur (Figure 3), even though both the mutation rate and average effect size are much larger for the deleterious mutations than for the beneficial mutations in the earlier runs. Natural selection eliminates most deleterious mutations because cells with those mutations grow slower than the ancestor.

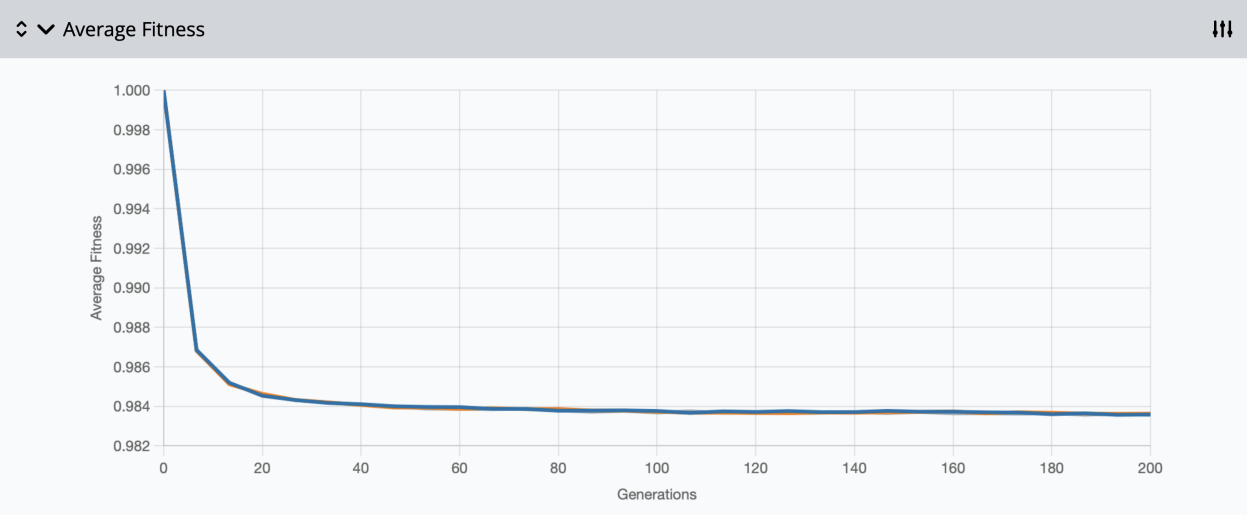

Figure 7: Trajectories for Average Fitness in populations evolving with deleterious mutations only.

Now scroll to the second panel that shows the average number of mutations accumulating over time (Figure 8). These data confirm the efficacy of selection in eliminating deleterious mutations. After 30 transfers, millions of deleterious mutations have occurred in each population, but most cells remain entirely free of mutations since the average number is well below 1. The occasional deleterious mutations that are present either arose recently or cause only small fitness losses, so they have not yet been eliminated by selection.

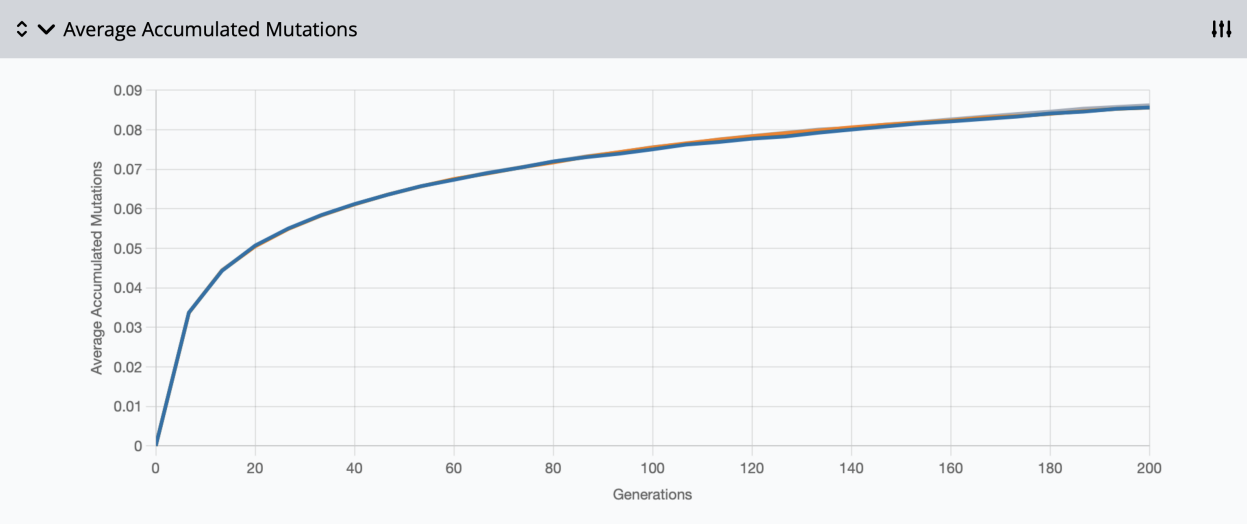

Figure 8: Trajectories for Average Accumulated Mutations in populations evolving with deleterious mutations only.

2.5.3 Beneficial, Neutral, and Deleterious Mutations Combined

Let's put the three kinds of mutations together now. Start with the default parameters in the main menu, except set the Replicate Populations to 6, so that the runs don't take too long. You can always run more later. In the advanced menu, set both the neutral and deleterious mutation rates equal to 0.001. In the data menu, make sure both Average Fitness and Average Accumulated Mutations are checked. Then press the Run button.

The fitness trajectories should look similar to those in Figure 3, where the neutral and deleterious mutation rates were 0. Again, that's because neutral mutations have no effect on fitness, while most deleterious mutations are eliminated by natural selection.

Now scroll down to the trajectories for accumulating mutations, which should look similar to Figure 9 below. The trajectories show step-like increases, just as we saw with only beneficial mutations (Figure 4). However, the average number of mutations is a bit higher now because other mutations can "hitchhike" with beneficial mutations. Imagine, for example, that a highly beneficial mutation occurs in a cell that already has a neutral mutation. As the beneficial "driver" spreads through the population, the neutral "passenger" will also increase because both mutations are in the same genome.

2.5.4 Caution: Avoiding Failed Runs with Neutral and Deleterious Mutations

In Section 2.1.3, we discussed certain basic parameter values that can cause STEPS to run slowly or not at all. Now that we have added two more types of mutations, we can expand on the parameter values to avoid. Specifically, we recommend the following two limits: the Rate of Deleterious Mutations should be no greater than 0.3; and the product of Maximum Population Size and the total mutation rate (sum of neutral, deleterious, and

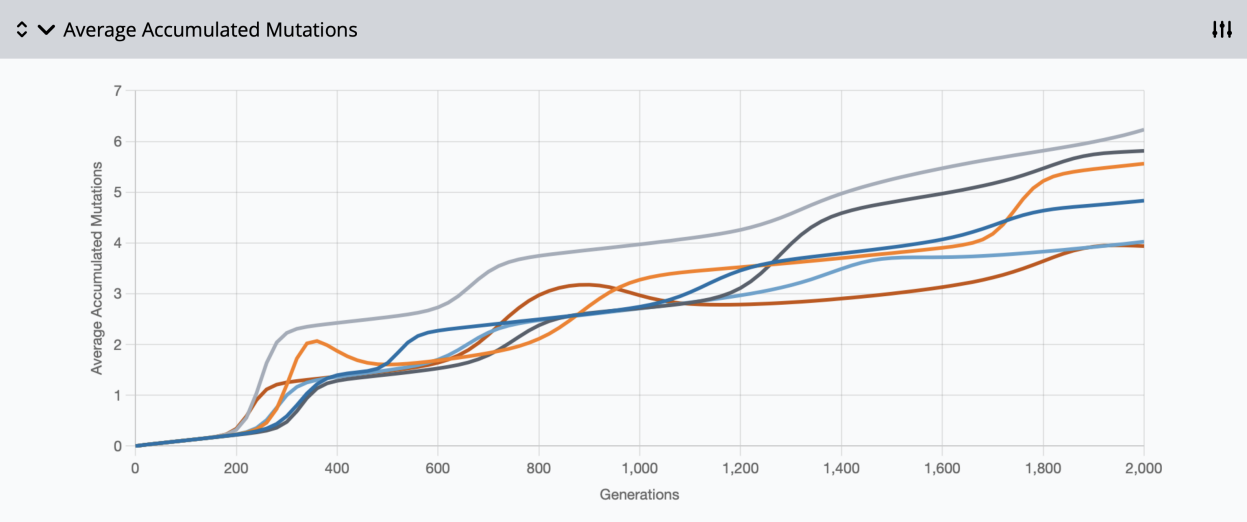

Figure 9: Trajectories for Average Accumulated Mutations in populations evolving with beneficial, neutral, and deleterious mutations.

beneficial rates) should be no greater than 1e8. If you want to study mutation rates above these limits, then you should use the command-line version of STEPS (see Chapter 3).

2.5.5 Optional: Severe Bottlenecks Cause Fitness to Decline

For this experiment, keep the parameters and settings as used above (section 2.5.3), with four differences: reduce the Maximum Population Size to 100, reduce the Dilution Factor to 10, set the Rate of Neutral Mutations to 0, and increase the Rate of Deleterious Mutations to 0.1. The first two values mean that a population has only ~10 cells after each transfer. With so few individuals, and the low Rate of Beneficial Mutations (1.7e-6), beneficial mutations are very unlikely to occur.

Worse yet, natural selection has become ineffective at eliminating the far more common deleterious mutations. Any mutation, even a deleterious one, can get lucky in a small sample, such as the few cells that survive each transfer. While that mutation might be lucky, the process is bad news for the evolving population, which will accumulate deleterious mutations over time, resulting in a declining fitness trajectory as seen in Figure 10 below.

2.6 Saving and Collecting Data

In our experiments with STEPS, we can watch the trajectories as they evolve and see the results after the runs have ended. We could also print screenshots of the graphs. However, once we start another experiment, the data from the previous experiment are lost—unless, that is, we choose to save and collect the data.

To do so, click the arrow next to Data Collection to expand that menu. Then select the box labeled CSV Download along with at least one of the boxes below Tracked Statistics.

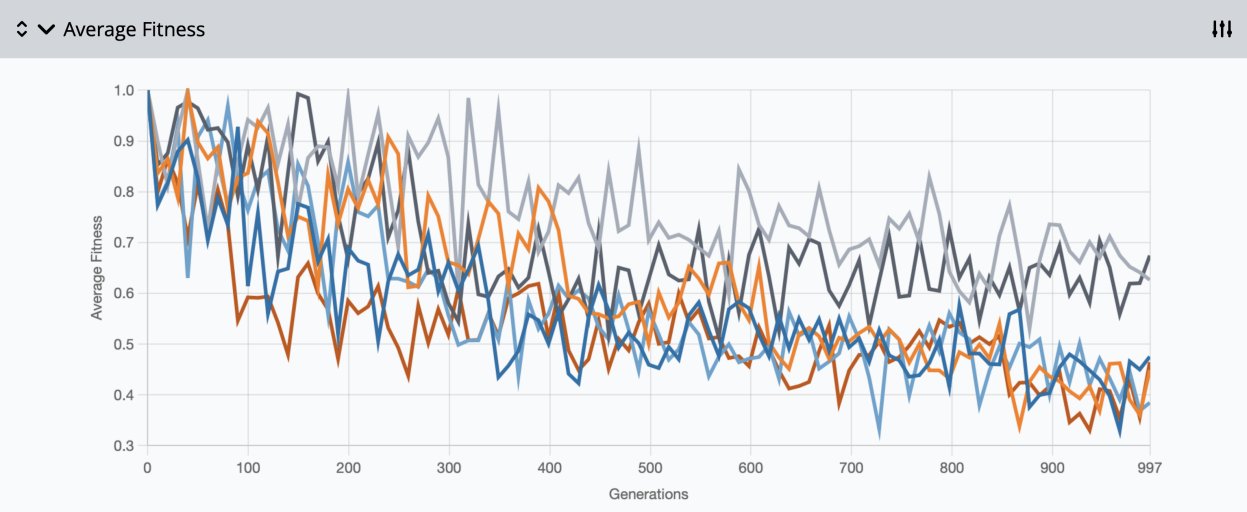

Figure 10: Trajectories for Average Fitness in populations evolving with severe bottlenecks.

(Average Fitness and Average Accumulated Mutations are turned on by default.) Make sure the other parameters are set to the values you intend for your experiment. Then hit the Run button, as usual.

After the runs finish, you will see a symbol like this at the lower right-hand side of the STEPS screen. (The symbol is always there, but if you haven't checked the box for CSV Download, it appears as light gray.) If you hover over the symbol, it will say Download CSV. Now click on the symbol to save a file with Comma Separated Values to your computer.

You can open the file using various programs, such as Microsoft Excel, Google Sheets, or Apple Numbers. The top row of the file contains some basic information, such as the version of the STEPS program that was used. The second row lists all of the parameter settings that were used, including the random number seed (discussed in the next section below). The third row shows the headings for the various data collected, beginning with the replicate and transfer number and followed by, for example, Average Fitness, Average Accumulated Mutations, and any other values you chose in the Tracked Statistics menu. All the following rows show the data over time for replicate population 1, then replicate population 2, and so on.

You can analyze your data as desired using any relevant programs. You will probably also want to rename your files to be reminded of the experiment and its purpose, or to avoid confusion if your data will be compared or combined in a classroom setting.

2.7 Random Number Seeds

The STEPS software uses random numbers in several processes including generating mutations and their effect sizes, placing mutations in lineages, and determining the number

of individuals (if any) in each lineage that survive a serial dilution.

The default mode of STEPS starts with a different random number, or seed, each time you hit the Run button. That means that if you run the same simulation several times, you will get results that differ, at least slightly, because the initial seed affects the entire sequence of random numbers that are used in a simulation.

These differences are appropriate in most contexts. For example, if you are teaching a class where many students run the same experiment, you probably want them to have their own data. That way, the class as a whole can discuss both the general patterns they see and why the results vary somewhat between them. And if everyone collects their own data, the data can be pooled to allow analyses of the combined dataset.

However, if something unusual happens—because of the random processes that occur during evolution, or perhaps because of an error in setting parameters of the simulation—then it can be useful to be able to reproduce the exact same runs and results. One way of doing that would be to have each student use, for example, the last 4 digits of their phone number as the seed for their simulations. To do so, they would simply enter that number in the box below Randomization Seed in the Advanced Settings menu.

A course instructor can also use the same seed in order to generate identical results to show different sets of students and point to some specific result at different times.

2.8 Optional: Adding Markers to Track Lineages

With the default settings, every STEPS population starts out with all of the individuals being identical. In biological experiments, it can be useful to start populations with two (or more) lineages that differ by a genetic marker that is neutral (i.e., the marker doesn't affect fitness). Markers allow one to examine certain processes that are strongly influenced by random events. The following subsections present experiments that use two marked lineages to illuminate the effects of random mutation and random genetic drift, respectively.

2.8.1 Optional: Marker Divergence in Large Populations

In this experiment, we will use all of the default settings plus two optional settings. (You can restore the default settings by reloading STEPS in your browser.) In the menu for Advanced Settings, change the Number of Initial Markers from 1 to 2; in the Data Collection menu, select the Marker 1 Ratio option. Then press the Run button.

Each population starts with an equal mix of two genotypes that are identical in every respect, except they have markers that allow their abundances to be separately tracked. If you look at the trajectories for Average Fitness, they should be comparable to the trajectories you obtained in your very first run (see Figure 3). The exact trajectories will differ slightly because of different random numbers, but nothing of substance will change.

Now take a look at Figure 11 below, which shows the trajectories for the marker ratio. The trajectories look pretty wild and unpredictable. Let's examine what the data show and why the trajectories are so unpredictable. To start, each trajectory shows the $\log_2$ of the ratio of the abundance of cells with the first marker relative to the abundance of cells with the second marker. Let's call that R = M/N, where M and N are the abundances of cells with the first and second marker, respectively. On a logarithmic scale, a value of 0 means that the ratio is 1, hence M = N (i.e., equal numbers of the two marked lineages). Values above 0 or below 0 mean that that first marker or the second marker is more common, respectively.

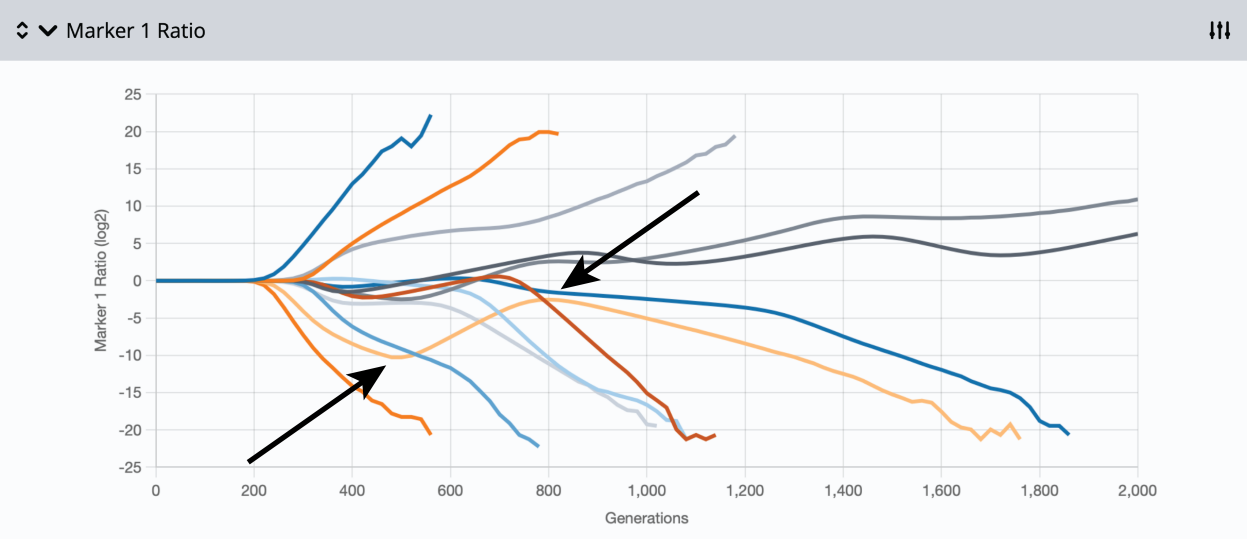

Figure 11: Trajectories for Marker 1 Ratio with the Number of Initial Markers set to 2, showing the effect of beneficial mutations in large populations. The arrows mark a double reversal in the trajectory for one of the populations (yellow).

Now look at just the first 200 or so generations of the trajectories in Figure 11. All of them lie on top of one another, right at the log ratio of 0. That makes sense, because the two marked lineages have equal fitness, and random fluctuations in their proportions will be tiny in populations that number five million even after the serial dilutions (5e8 / 100 = 5e6).

But then what happens? Recall the earlier discussion (Section 2.3) of why the fitness trajectories increase in a stepwise manner. In brief, there's a long lag between when a beneficial mutation first occurs, and when the mutant lineage becomes common enough in a large population to cause an appreciable increase in the average fitness. For the same reason, a beneficial mutation that occurs in one of the marked lineages must increase for a similarly long time before it can noticeably impact the ratio of the two lineages.

Therefore, the sudden changes in the marker ratio after 200 generations or so reflect beneficial mutations that occurred in the first few tens of generations, but which required compounding (like growth in a bank account) to become sufficiently common to both perturb the ratio of the two lineages and cause the step-like increases in average fitness.

Notice also that some of the trajectories break upward, while others break downward. That happens because the first beneficial mutation that reaches high frequency is just as likely to arise in one marked lineage as the other. You can also see that most or all of the trajectories end well before 2,000 generations. The fitness trajectories don't end prematurely, only those for the marker ratio. The trajectories for the ratio end when one or the other lineage goes extinct. The ratio is plotted on a logarithmic scale, and so it is undefined when either the numerator or the denominator hits 0.

Finally, notice there are some reversals of fortune, where the initial break in the trajectory starts in one direction and then reverses, in some cases even reversing again. These reversals occur when beneficial mutations appear in both marked lineages; for example, a mutation in lineage 1 provides a 5% advantage and reaches some substantial frequency, but before lineage 2 goes extinct it gets a mutation with a 10% benefit and ultimately prevails. A double reversal can occur if lineage 1 then gets another highly beneficial mutation before it is eliminated. In Figure 11, we've added arrows pointing to a double reversal in the marker-ratio trajectory in one of the replicate populations.

2.8.2 Optional: Genetic Drift in Small Populations

For this experiment, set the Number of Transfers to 150, the Maximum Population Size to 5e3 (or 5000), the Dilution Factor to 100, the Rate of Beneficial Mutations to 0 (main menu), the Number of Initial Markers to 2 (advanced menu), and select the Marker 1 Ratio option (data menu). Then press Run.

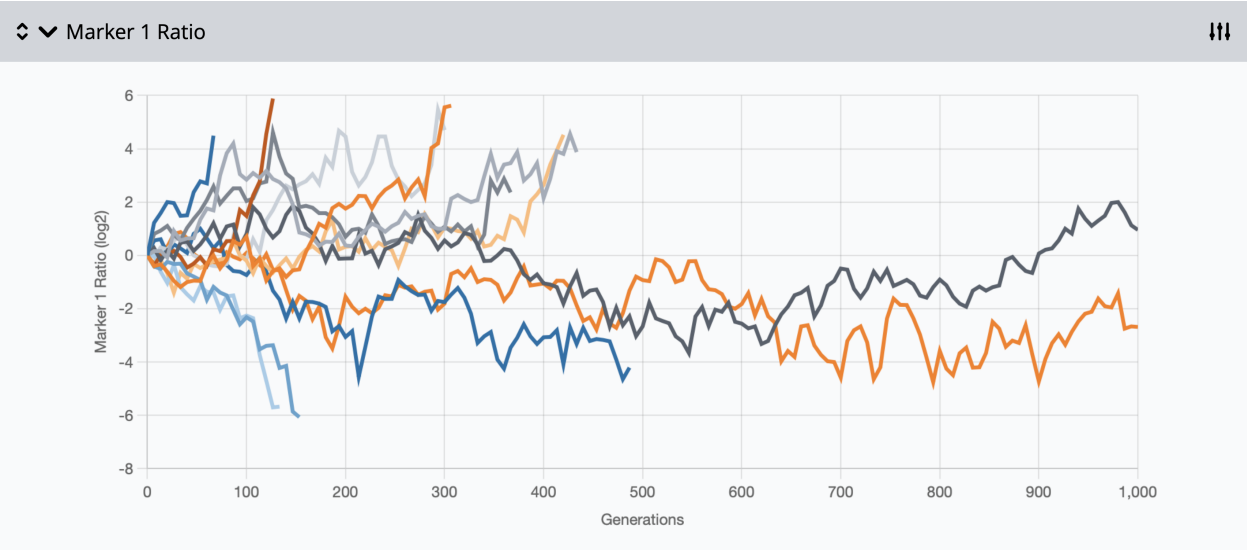

Figure 12: Trajectories for Marker 1 Ratio with the Number of Initial Markers set to 2, showing the substantial effect of genetic drift in small populations.

You will see something like Figure 12 above. Again, each population begins with an equal mixture of two lineages that are identical except for markers that allow their abundances

to be tracked. From the very start, the ratio fluctuates wildly, and by 1,000 generations one lineage or the other has gone extinct in most populations. With a Maximum Population Size of just 5000 and a Dilution Factor of 100, there are only 50 surviving individuals, on average, after each serial transfer. The survivors are chosen at random, and so the ratio of the two lineages exhibits a random walk that reflects the process called genetic drift. The ratio of the two lineages can't deviate by more than about 50 in either direction (<6 on a $\log_2$ scale) before there are no individuals remaining in one of the lineages.

Now change the Number of Transfers to 300 and Maximum Population Size to 5e8, which is the usual setting, and press Run again. You should now see something like Figure 13 below. At first glance, you might think there are still some pretty big fluctuations in the marker ratio. But look carefully at the y-axis scale. Most of the fluctuations are within + or – 0.05, which reflect changes of <5% in the ratio of the two lineages, even after 2,000 generations. Indeed, recall the imperceptible fluctuations during the first 200 generations in the runs performed with beneficial mutations (Figure 11).

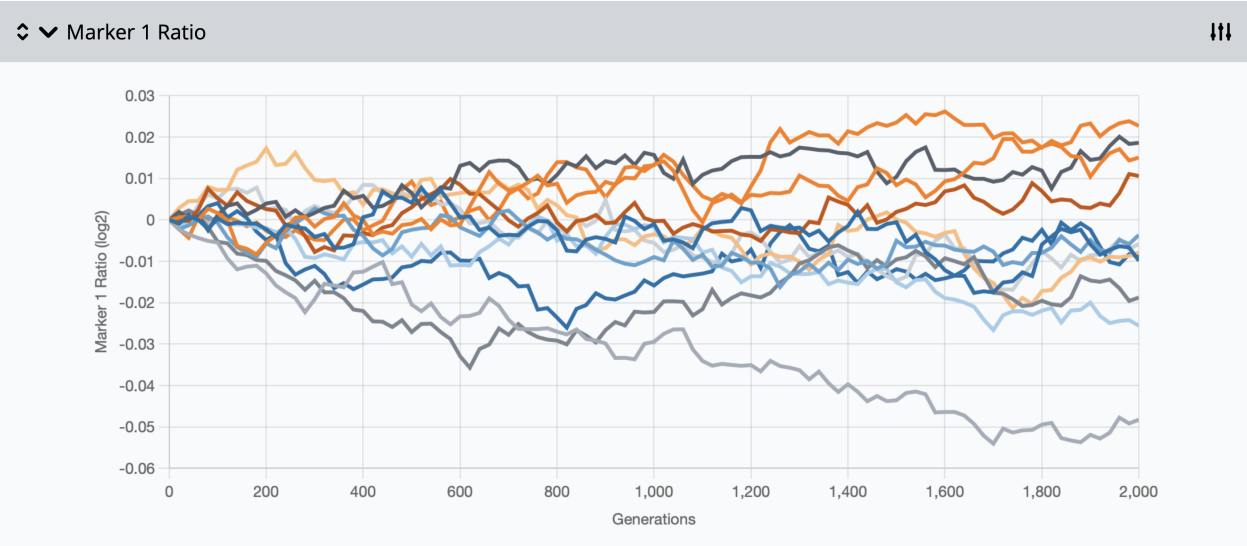

Figure 13: Trajectories for Marker 1 Ratio with the Number of Initial Markers set to 2, showing the tiny effect of genetic drift in large populations.

2.9 Wrapping Up

You've now had a pretty thorough introduction to the web-based version of STEPS. There's a lot that you can do with this version.

The command-line version provides even more possibilities, as described in Chapter 3. One can run simulations with additional options that aren't available with the web-based interface. By providing more time and/or computational power, the command-line version enables one to run simulations for longer (more generations), with more replicate populations, with larger population sizes, and/or with higher mutation rates (leading to more

lineages that must be tracked) than is practical with the web-based version. One can also use the command-line version to set up and run simulations that systematically explore large spaces for multiple parameters, which can be useful for designing or analyzing the results of biological experiments.

And because STEPS is open source, one can modify the code to run simulations in other ways, if so desired.

3 Command-Line STEPS

The command-line version of STEPS provides an advanced program designed with researchers in mind. The web-based version can also be used for research purposes, but the advanced version offers a wider variety of adjustable parameters and more data types that can be saved. Also, the simulations using the command-line version will typically run faster than the web-based version for the same parameter settings.

Unlike Chapter 2, this chapter focuses on the inputs and outputs, rather than illustrating the results of simulations. Therefore, you may want to skip ahead to specific sections depending on whether you are running this version for the first time (Section 3.1), need to brush up on starting a run (Section 3.2), or want to explore the parameters that are available as inputs (Section 3.3).

3.1 Setup

3.1.1 Required Software

Before you can run the command-line version of STEPS, you will need to install the latest versions of Rust and Cargo. You can do so by going to the rustup website[3] and following the instructions there. You will also want to make sure that you have downloaded STEPS from the GitHub repository[4].

3.1.2 Compiling the Code

In order to run STEPS, you must first compile the code. Begin by opening a command-line interface (CLI), such as Terminal on a Mac or the Command prompt in Windows, and navigating to your downloaded STEPS directory. Then run one of the following three sets of commands (shown as white text on black background) to compile the code. Once the code has been compiled, you can run as many simulations as you want with any number of different inputs. You should only need to recompile the code if you've updated any of the relevant software, upgraded your OS, or changed to a different machine.

The basic compile command is as follows:

```
cargo build --release
```

If you want to optimize the compilation to your machine, use the following command:

```
RUSTFLAGS="-C target-cpu=native" cargo build --release
```

[3] https://www.rust-lang.org/tools/install
[4] https://github.com/zachmatson/STEPS.git

If you need to optimize the compilation to a specific architecture, you can specify it in place of 'native' in the previous command. For example, you can compile to skylake using this command:

```
RUSTFLAGS="-C target-cpu=skylake" cargo build --release
```

After you have successfully compiled the code, there should be a folder called *target* in your STEPS directory. If warning messages popped up during the compilation, they can generally be ignored. However, if there were error messages, or if there is no *target* folder in your STEPS directory, then something went wrong. Try re-downloading the code for STEPS and following the instructions to compile it again. If that still doesn't work, visit the issues page[5] on our GitHub site. Provide as much information as possible about your machine and the error messages that occurred, and we will try to provide a fix.

3.2 The Run Basics

When running the command-line version of STEPS, you must provide all input options, simulation parameters, and output details at the start of the run. We will begin with an example to show the different parts of a typical script for running STEPS.

Open a CLI window and navigate to the directory that contains your compiled code. Then enter the code shown below to begin a short simulation that will create an output file, named *BasicOutput.csv*, in that same folder.

```
./target/release/steps simulate --Ub 1.7e-6 --replicates 3 --transfers 4
    --dilution-factor 100 --Nmax 1e8 --stdev-w --summary-output-path
    BasicOutput.csv
```

You should find the *BasicOutput.csv* file and open it to confirm that something was saved to the file and that the run has completed. Now that you've successfully run STEPS using the command-line version, let's break down what you told the simulation to do. The first two commands instruct your computer to find the compiled code and run it.

```
./target/release/steps simulate
```

If you have already saved a file created using STEPS, you can use the command 'reproduce' instead of 'simulate' to extract the inputs from that file and use them in a new run. You would still need to indicate an output file name, but you wouldn't have to provide any other inputs. You can also use the command 'help' in place of the 'simulate' command, and STEPS will print information about the simulations and any inputs you provided. (More detailed information on each of these can be found in Section 3.3.3.)

[5] https://github.com/zachmatson/STEPS/issues

The next five inputs tell STEPS to use the values that you provide (immediately after each one) for specific parameters in the runs.

```
--Ub 1.7e-6 --replicates 3 --transfers 4 --dilution-factor 100 --Nmax 1e8
```

Section 3.3.1 explains the many options that are available for input parameters. Note that each of these inputs requires both the name of the parameter and a numerical value. The run will fail if both are not provided.

The last two inputs in this example concern the output file. The first of these instructs STEPS that the output file should contain the standard deviation of fitness for each replicate population.

```
--stdev-w
```

Unlike the parameter inputs, this one doesn't require any further information. Listing an output is sufficient to indicate that this data type should be included the summary output.

The final input tells STEPS that you want a summary output file and what it should be named.

```
--summary-output-path BasicOutput.csv
```

Be sure to include an output file type and name in all of your runs; if you don't, the simulation will still run, but none of the results will be saved for you to analyze.

There is another output type that can also be saved. It provides, in essence, the metagenomic data for the run. If you include '--mutation-summary-output' and provide a file name, STEPS will save the frequency of each mutation over time. Either or both types of output data (standard and metagenomic) can be saved for any run.

The next sections provides additional information and a list of all available options for inputs and outputs. Before we jump into those options, however, you may have noticed a brief flash in the window after you ran the above example. Figure 14 shows what it might look like.

The window actually shows a progress bar that appeared at the start of the run and then disappeared at the end. With simulations that take longer to complete, this progress bar can be used to judge how much longer you must wait until the run finishes. If the progress bar is moving very slowly, you might want to reconsider your parameter settings. Alternatively, you might run the simulations on a high-performance computer cluster rather than on your local machine.

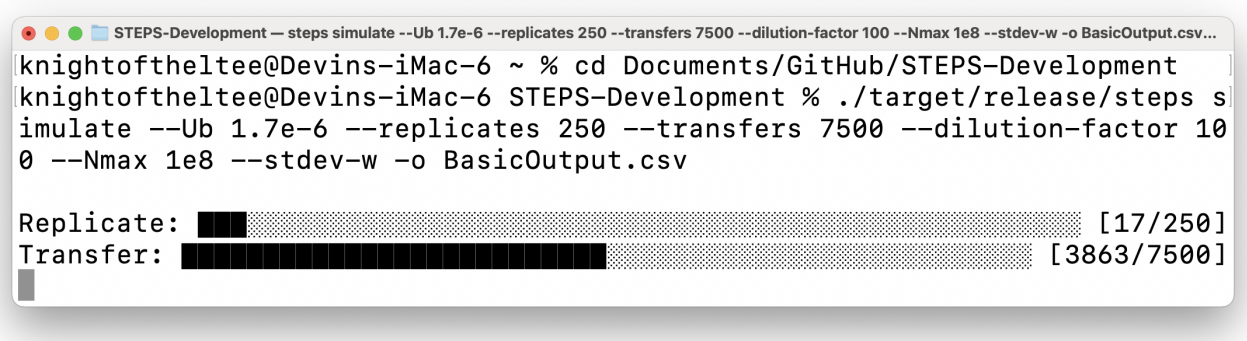

Figure 14: Example of STEPS running in a terminal window.

3.3 Input Options

The input options can be grouped into three categories, and they are listed alphabetically in the corresponding sections below. The section on Simulation Parameters includes the inputs that directly affect the simulations themselves. All of these inputs require a numeric value when used. This section explains the purpose of the inputs and gives the range of acceptable parameters. For further explanation of how the inputs impact the simulations as a whole, see Chapter 4. The section on Optional Outputs presents the various data that can be saved by the user. You can simply list any of them and the relevant data will be added to the summary output file. The third section pertains to several inputs that are different from the others. They include the Simulate command itself, two different kinds of output files, and a few other options.

3.3.1 Simulation Parameters

Note that the default values for the simulation parameters in the command-line version of STEPS are the same values as used in the web-based version. Note also that the range of allowable parameter values may include ones that cause STEPS to run very slowly; some allowable values might even cause the simulations to fail during the run. These ranges simply indicate input values that allow the simulations to get started. Also, some of the available parameters have two ways of being entered; both ways will do the same thing, but one is typically more explicit while the other is easier to type.

Beneficial Mutation Rate —The rate at which beneficial mutations occur in the population. Input values as mutations per individual per generation.

Usage: **--Ub**

Values: between 0 (inclusive) and 1 (exclusive).

Default: 1.7e-6

Beneficial Selection Coefficient, Initial —The effect size of each beneficial mutation is sampled from an exponential distribution. This input sets the initial mean of that distribution.

Usage: **--Sb**

Values: between 0 (inclusive) and 1 (exclusive).

Default: 0.012

Deleterious Mutation Rate —The rate at which deleterious mutations occur in the population. Input values as mutations per individual per generation.

Usage: **--Ud**

Values: between 0 (inclusive) and 1 (exclusive).

Default: 0

Dilution Factor —The inverse of the multiplicative factor applied to the final Maximum Population Size. The dilution is applied after the growth phase is complete, and it produces the bottleneck associated with each serial transfer event. The binomial distribution is used to determine the number of individuals in each lineage that are chosen.

Usage: **--dilution-factor** or **-D**

Values: must be greater than 1

Default: 100

Diminishing Returns Epistasis, Strength of —Controls how the mean of the distribution of fitness effects (DFE) for beneficial mutations, α^{-1}, changes after each successive beneficial or deleterious mutation occurs in a lineage. For beneficial mutations, STEPS uses the model presented by Wiser et al. (2013, *Science* 342, 1364–1367), as follows: $\alpha_{n+1} = \alpha_n(1 + g \times s_{n+1})$. Here, α_n is the inverse of the mean of the DFE after the first *n* mutations have occurred, *g* is the strength of the diminishing returns, and s_n is the effect size of the n^{th} mutation. STEPS uses a complementary model for deleterious mutations (see Section 4.6.2 for more information). Note that the value of α is tracked separately for each lineage, as they all have unique mutational histories.

Usage: **--diminishing-returns-epistasis-strength** or **-g**

Values: must be greater than or equal to 0

Default: 6.0

Markers —The number of distinctly labeled but otherwise identical subpopulations in each starting population. All of the subpopulations begin at the same frequency.

Usage: **--markers**

Values: any positive integer less than the minimum population size (i.e., N_{max} / D)

Default: 1

Maximum Population Size —Maximum number of individuals in the population, at which point growth ceases and the next serial dilution is applied.

Usage: **--Nmax**

Values: at least 10

Default: 5e8

Neutral Mutation Rate —The rate at which neutral mutations occur in the population. Input values as mutations per individual per generation.

Usage: **--Un**

Values: between 0 (inclusive) and 1 (exclusive).

Default: 0

Randomization Seed —A saved value that allows you to repeat the exact same simulations and obtain the same results. It can be useful in situations where you want to change the duration of a previous simulation by changing the number of transfers; collect data at a new temporal resolution by adjusting the sampling frequency; save additional information to your summary file from the available options; and/or add the optional metagenomic data as an output file. Note, however, that if *any* other aspects of the simulation inputs are changed (e.g, the maximum population size or any of the mutation rates), then the same seed will *not* produce the same results.

Usage: **--seed**

Values: any positive integer

Replicates —Number of replicate populations to run. (Note that each replicate population will experience a unique series of random events even if a seed is provided for the overall simulation.)

Usage: **--replicates**

Values: any positive integer

Default: 12

Sampling Frequency —The frequency at which output data are saved, with values given as the number of serial transfer events.

Usage: **--sampling-frequency** or **-f**

Values: any positive integer less than the total number of transfers

Default: 1

Transfers —The number of times that each replicate population is serially transferred.

Usage: **--transfers**

Values: any positive integer

Default: 300

3.3.2 Optional Outputs

Accumulated Mutations, Maximum —The maximum number of mutations accumulated by any individual in the population.

Usage: **--max-accumulated-muts**

Default: false

Accumulated Mutations, Mean —The average number of mutations accumulated across all individuals in the population.

Usage: **--mean-accumulated-muts**

Default: true

Accumulated Mutations, Minimum —The minimum number of mutations accumulated by any individual in the population.

Usage: **--min-accumulated-muts**

Default: false

Accumulated Mutations, Standard Deviation —The standard deviation of the number of mutations among individuals in the population.

Usage: **--stdev-accumulated-muts**

Default: false

Fitness, Average —The mean fitness across all individuals in the population.

Usage: **--avg-W**

Default: true

Fitness, Standard Deviation —The standard deviation of fitness values across individuals in the population.

Usage: **--stdev-W**

Default: false

Fitness, Maximum —The maximum fitness value of any individual in the population.

Usage: **--max-W**

Default: false

Genotype Count —The number of different genotypes (i.e., lineages) in the population.

Usage: **--genotype-count**

Default: false

Marker Ratio —The ratio of the subpopulation that is descended from the ancestor with the first marker, relative to the sum of all other subpopulations.

Usage: **--marker-1-ratio**

Default: false

Shannon Diversity —The Shannon diversity index of the different genotypes calculated across all individuals in the population.

Usage: **--shannon-diversity**

Default: false

3.3.3 Other Inputs

Deleterious Selection Coefficient, Fixed —Changes the distribution of deleterious effects from a uniform distribution to all having the same size as the provided value.

Usage: **--Sd**

Values: any positive value less than or equal to 1

Help —Prints the help information for running the STEPS simulations. It is used instead of the simulate or reproduce commands.

Usage: **--help** or **-h**

Output, Summary —Outputs the summary data to a CSV file with the name provided at the designated path. The first two lines of the file list the run parameters and provide information about the STEPS version used to generate the data. From there, the file is like a typical CSV file, with column names in the first row and data in the later rows.

Usage: **--summary-output-path** or **-o**

Output, Sequencing —Outputs the frequency of each mutation to a CSV file with the name provided at the designated path. The first two lines of the file provide the run parameters and the STEPS version. From there, column names are shown in the first row and data in the later rows. Note that generating this output file will substantially slow the runs, much more so than adding additional statistics to the Summary output. Also, the Sequencing file can be huge, so we recommend running smaller tests with this optional output before using it for larger projects.

Usage: **--mutation-summary-output**

Reproduce —Rather than provide a series of inputs to start a new STEPS run, you can use the summary csv file from a previous run for which you would like to use the same

parameters. This command is used in place of the simulate command. You must still provide an output type along with the file name and path for the new output.

Usage: **reproduce**

Simulate —Runs the simulation with the parameters that are provided. For those parameters that are not provided, the default values are used.

Usage: **simulate**

4 Simulation Mechanics

The processes implemented in the STEPS program reflect biological processes, and the mechanics of the simulations use a mathematical framework built on models that have been tested experimentally. This chapter will explain what's going on "under the hood" during the simulations. It will also explain the reasoning behind various model choices and identify some potential limitations of those choices.

4.1 Fitness

The concept of fitness was explained in Chapter 1 and then used throughout the illustrative examples in Chapter 2. In general, fitness is used to describe the expected reproductive success of a specific genotype, and it is typically expressed in relative terms. In STEPS, the fitness of a genotype is defined as its growth rate relative to that of its ancestor. We use the letter W for fitness and define the fitness of the ancestor to be $W_{anc} = 1$. As the population grows, mutations are introduced as new individuals are added to the growing population. The size of the effect that a mutation has on the fitness of an individual is called the selection coefficient s. We'll go into more detail about the different kinds of mutations in later sections.

For now, let's first look at a simple example of how a genotype's fitness is used to calculate the number of individuals that a lineage will have after a single timestep. The duration of the timestep we will use corresponds to a generation, or population doubling, for organisms like bacteria that reproduce by binary fission. Thus, after a single timestep:

$$N_{final} = N_{initial} \times 2^{W \times t} \tag{1}$$

where $N_{initial}$ is the current number of individuals in a lineage, N_{final} is number after the timestep, W is the genotype's fitness, and t is the duration of the timestep that allows the total number of individuals in the population to double. (In Section 4.4.1, we will discuss how the duration of the timestep is estimated in STEPS.) If this lineage is the ancestral genotype in the run, it will have $W = 1$ and, for this example, let's say $N_{initial} = 100$. Like fitness, the time in STEPS is relative to the ancestor. Thus, we define $t = 1$ as the amount of time it takes for the ancestral population to double in size. Keeping with this example, that means:

$$\begin{aligned} N_{final} &= N_{initial} \times 2^{W \times t} \\ &= 100 \times 2^{1 \times 1} \\ &= 200 \end{aligned}$$

Let's now consider a new genotype and its resulting lineage, L, that has accumulated ℓ mutations. To calculate L's fitness, we use:

$$W_L = W_{anc} \times \prod_{i=1}^{\ell} (1 + s_i) \tag{2}$$

where s_i is the selection coefficient of the i^{th} mutation. This model is called the multiplicative fitness model. There are other fitness models (e.g., the additive model) that are sometimes used in the literature. STEPS uses the multiplicative model for several reasons. First, it is an extremely simple model. For example, for deleterious mutations with $s < 1$, it prevents a fitness value from becoming negative, which would add various complications. Second, it has been shown that the long-term fitness trend in the LTEE is well-described by a trajectory that has no upper limit. The multiplicative model doesn't directly impose an upper limit to fitness, but it leaves room for other potential factors to create a limit. Third, the multiplicative model means that the genetic background in which a new mutation occurs impacts the resulting change in fitness. For example, consider two lineages, *A* and *B*, with fitness $W_A = 1.0$ and $W_B = 1.5$. A new mutation with effect size $s = 0.1$ would produce new lineages that increase fitness by 0.1 and 0.15, respectively. This multiplicative model, when combined with other features of STEPS discussed in a later section, generates path-dependent fitness effects that are also seen in the LTEE and other evolution experiments.

4.2 Population Structure

In evolution experiments with microbes, the population size of a single replicate can reach billions of individuals. Keeping track of so many individuals would be slow and inefficient, so most simulation programs find ways to simplify things. In STEPS, we take advantage of the clonal nature of the microbes that we model by putting all the individuals with the exact same genotype into a group called a lineage. Thus, we only need to track the number of distinct lineages in a population, rather than every individual.

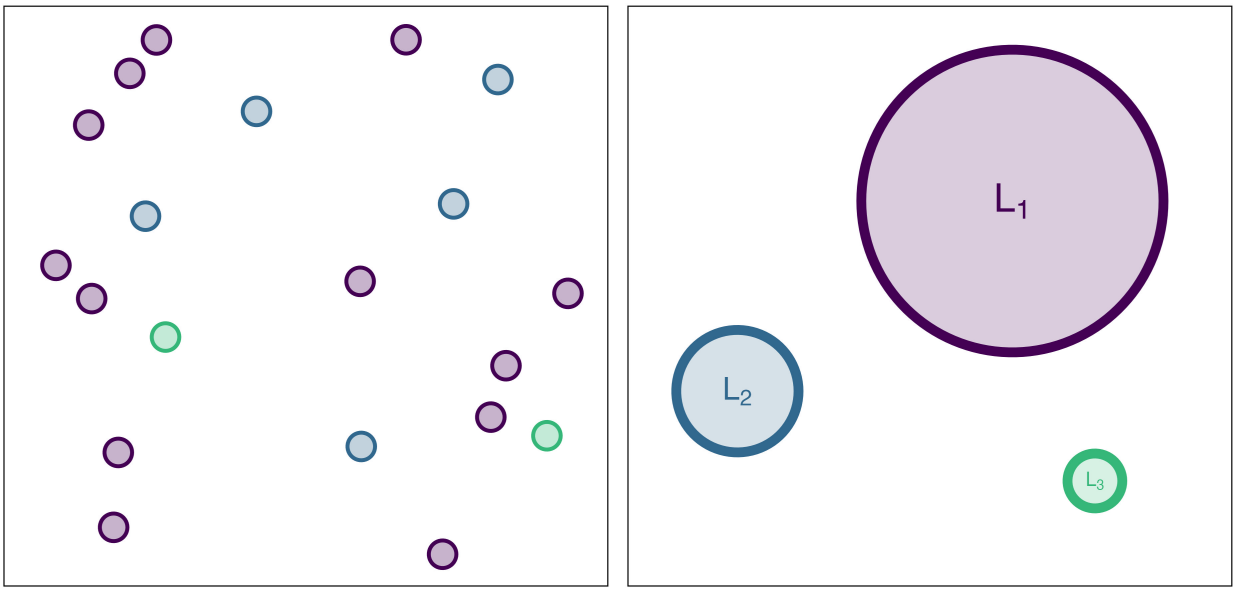

Figure 15: Comparison of individuals and lineages in a simple hypothetical population. Left panel: each individual's color indicates its genotype. Right panel: STEPS groups the individuals with identical genotypes into three distinct lineages.

Figure 15 illustrates this consolidation of individuals into lineages. On the left, each symbol represents an individual in the population, and its color indicates its genotype. On the right, the same population is shown as three genetically distinct lineages, with the size of the symbol scaling with the number of individuals in that lineage. In addition to keeping track of the number of individuals in each lineage, STEPS also tracks other properties for each lineage including its relative fitness, its parent (i.e., the lineage from which it was derived by mutation), the number of mutations it has accumulated, and certain other information.

4.3 Initial State of a Population

Before we can simulate growth, mutation, and evolution, we need a starting population. Each replicate population in STEPS starts with the same number of genetically identical individuals (see 4.3.1 for an exception). That initial number, N_0, is calculated as follows:

$$N_0 = \lfloor \frac{N_{max}}{D} \rceil$$

where N_{max} is the maximum population size and D is the dilution factor. Both of these parameters must be specified in a command; otherwise, the default values will be used. (The floor and ceiling brackets indicate rounding in the event that the ratio is not an integer.)

4.3.1 Neutral Markers

STEPS also allows one to embed two or more neutral markers into the starting population. Because the markers are neutral, they have no effect on fitness. However, they allow the subpopulations with different marker states to be tracked over the course of a simulation. The default number of neutral markers, M, is set to 1, such that all of the individuals in each starting population are identical, as discussed above. If $M > 1$ is used as an input, then the starting population will be evenly divided among the M marked subpopulations.

As is the case with only one marker, the equation for the initial population size with multiple markers also rounds after the division.

$$N_0 = M \times \lfloor \frac{N_{max}}{D} / M \rceil$$

The rounding here can cause slight deviations in the value of N_0. For example, if $N_{max} = 1000$ and $D = 100$, then $N_0 = 10$ when $M = 1$. However, if $M = 3$, the population would start with $N_0 = 9$ (i.e., 3 marked subpopulations each with 3 individuals). This rounding should have a negligible effect unless the ratio shown between the floor and ceiling brackets is quite small.

4.4 The Transfer Cycle

As explained in Chapter 1, the populations in a typical serial transfer experiment experience periodic bottlenecks when a specified fraction of the population is diluted into fresh culture

medium. From there, the cells will grow and divide until the limiting resource is exhausted, at which point the population awaits the next transfer event. The STEPS program follows the same basic cycle. However, STEPS uses some procedures and approximations that improve computational efficiency and thereby allow STEPS to run faster, as explained here.

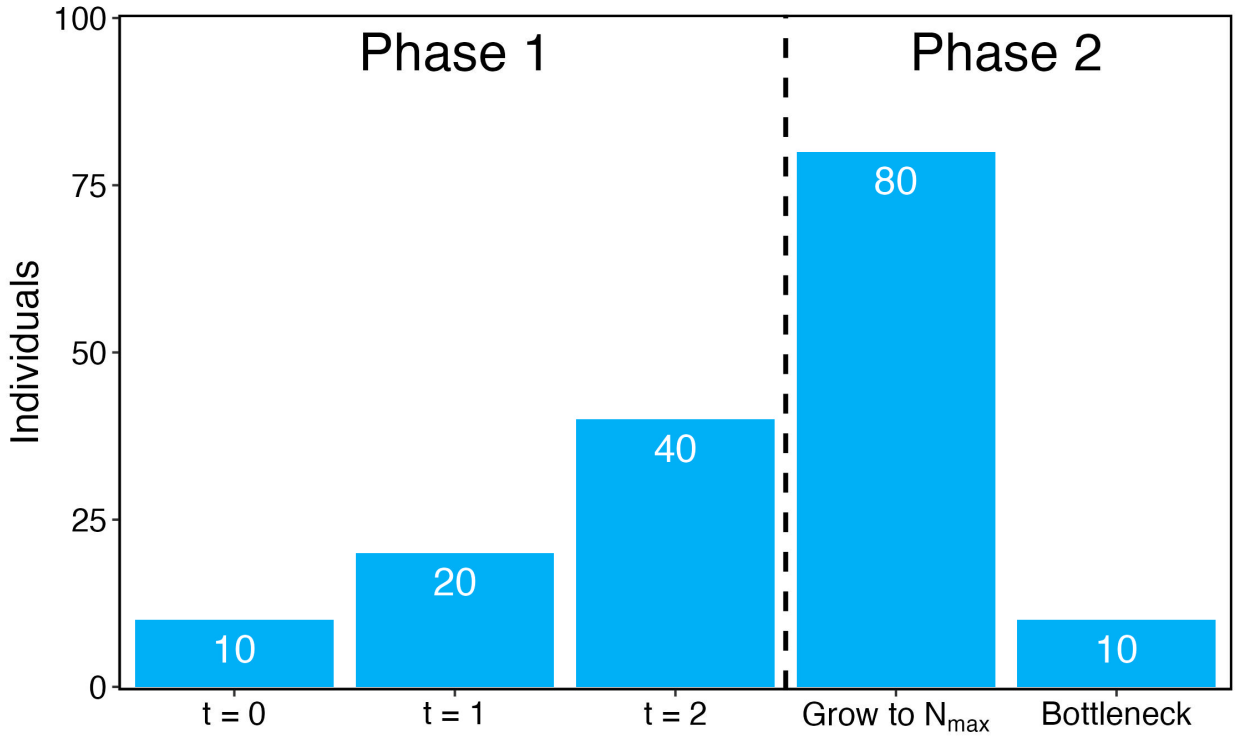

Figure 16: Illustrative example of the two sequential phases of population growth, as implemented in STEPS, for a single transfer cycle with D = 8 and N_{max} = 80.

The growth portion of each transfer cycle in STEPS is designed to mimic how cells grow and divide in an actual population, but it is split into two phases in the simulation with different numbers of timesteps in each. The first phase has a predetermined number of timesteps, which are designed to allow the population roughly to double in size in each step. The second phase has only a single timestep that takes the population from its size at the end of the first phase to its maximum size. The population is then subjected to a transfer bottleneck at the end of the second phase, after which the first growth phase of the next transfer cycle begins (unless the specified number of transfers has been reached). Figure 16 illustrates how the population size changes over the course of a complete transfer cycle in STEPS.

The total number of timesteps is directly related to the number of generations, *gen*, that would occur during a single transfer cycle. The number of generations equals the number of doublings required for the population to go from its minimum size to its maximum size. The ratio of those two numbers is also the dilution factor, so the number of generations

can be calculated using either input:

$$\begin{aligned} gen &= log_2(\frac{N_{max}}{N_{min}}) \\ &= log_2(D) \end{aligned} \tag{3}$$

4.4.1 Growth Phase 1

Growth Phase 1 will usually contain the majority of the timesteps, while Growth Phase 2 allows the remaining growth that is necessary to reach the maximum population size. To prevent a population from exceeding its maximum size during Growth Phase 1, STEPS requires that at least half of a generation (i.e., population doubling) must occur in Growth Phase 2. Thus, the number of timesteps in Growth Phase 1, n_t, is calculated as follows:

$$n_t = \begin{cases} gen - \gamma, & \text{if } \gamma \geq 0.5 \\ gen - (\gamma + 1), & \text{otherwise} \end{cases} \tag{4}$$

where $\gamma = gen - \lfloor gen \rfloor$ represents the remaining partial generations after all full generations have occurred.

Of course, as mutant lineages appear in the population, the average fitness will also change. As a consequence, the timestep required to double the number of individuals must be adjusted commensurate with the change in average fitness. As shown in equation 2, the variable t determines the size of the timestep needed to double the population. This unit of time is defined so that $t = 1$ corresponds to the length of time required for the ancestor to double. A lineage with $W = 2$ would require a timestep of $t = \frac{1}{2}$ to double in number. In order to allow a heterogeneous population to double, STEPS calculates the average fitness of the whole population, and that value is then inverted to determine the length of the timestep that is applied during each of the doublings in Growth Phase 1.

Let's consider an example using the same setup as in Figure 16. We will start with 9 ancestral individuals and 1 individual with a fitness of 1.5. The average fitness at the start of the transfer cycle is therefore $\overline{W} = \frac{1\times 9+1.5\times 1}{9+1} = 1.05$, which means that the timestep is set to $t = \overline{W}^{-1} \approx 0.95$. The ancestral lineage will less than double in size because of this smaller timestep ($2^{(1\times 0.95)} < 2$), while the mutant lineage will more than double in size ($2^{(1.5\times 0.95)} > 2$). This differential growth drives the increase in frequency of beneficial mutations; similarly, it reduces the frequency of deleterious mutations. Figure 17 illustrates the impact of the fitness difference across a full transfer cycle.

4.4.2 Growth Phase 2

The purpose of Growth Phase 1 is to get the simulated population near the maximum size. Growth Phase 2 then gets the population to that maximum size, after which the population is reduced to its minimum "bottleneck" size. To take the population from its current size

at the end of Phase 1 to the maximum size, STEPS calculates the necessary timestep, as follows:

$$t_{final} = log_2(\frac{N_{max}}{N_{current}})/\overline{W} \quad (5)$$

If $\frac{N_{max}}{N_{current}} \approx 2$, which will occur when $\gamma \approx 0$, then the final timestep will be $t_{final} \approx \overline{W}^{-1}$ as in Growth Phase 1. This correspondence makes sense when the dilution factor is a power of 2 and thus $\gamma = 0$, because there should be a full population doubling to reach the maximum size, just as there was during each timestep in Phase 1.

4.4.3 The Bottleneck

At this point in the transfer cycle, $N \approx N_{max}$ and the population growth is completed. It is now time to apply the bottleneck and dilute the population to its minimum size. Until this point, the lineages in the population could have both whole and fractional individuals. To apply the bottleneck to the population using a binomial distribution, however, one can have only whole individuals. STEPS thus rounds the number of individuals in each lineage to the nearest integer after the final timestep. To apply the bottleneck, a binomial distribution is sampled for each lineage using the newly rounded individual count as the number of trials and the inverse of the dilution factor as the probability of surviving the transfer event. As a consequence, each lineage will have only integer numbers of individuals at the start and end of each transfer cycle, as seen in Figure 17. This bottleneck procedure also leads to variation in the number of individuals that start each transfer cycle. On average, there are $N_{min} = \frac{N_{max}}{D}$ individuals after each transfer, but the actual number may be more or less. Because the number of survivors can be less than N_{min}, a population may go extinct if N_{min} is small. The STEPS program will stop running if a population goes extinct, so you should keep this fact in mind when choosing the parameters of a run.

4.4.4 Summary Statistics

STEPS calculates and tracks many values throughout the transfer cycle, but only those at the end of the transfer cycle are available as outputs. Thus, all data in the output files (e.g., Average Fitness) reflect values after the bottleneck was applied, rather than when the population was at its maximum size. This approach also reflects the timing of when mutations are added during the transfer cycle, as discussed in the next section below.

4.4.5 Transfers and Generations Revisited

The Summary Output file reports time in terms of the number of transfers, not generations. One can calculate the corresponding number of generations as the product of the number of transfers and $\log_2(D)$, where D is the Dilution Factor. This same conversion to generations is used as the default when showing dynamical trajectories in the web-based version of STEPS, with one small deviation. Namely, when $D = 100$ exactly, then STEPS shows the number of

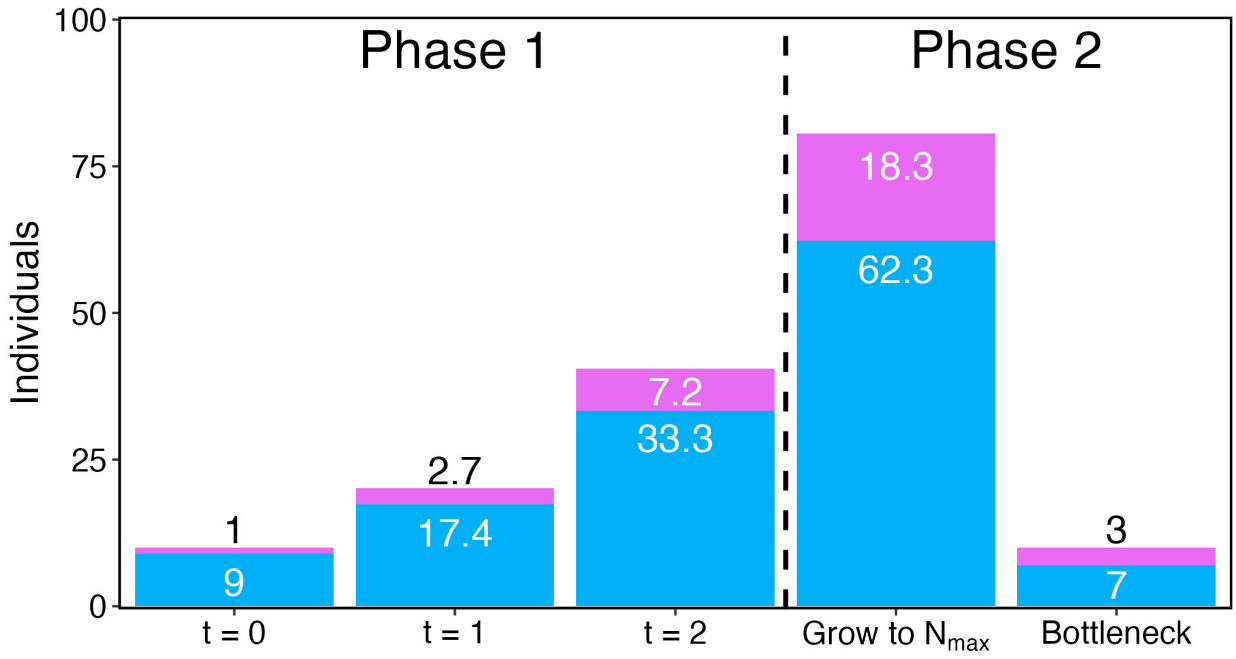

Figure 17: Illustration of population growth for a single transfer cycle with $D = 8$ and $N_{max} = 80$. The blue lineage is the ancestor, and the purple lineage starts from a single mutant with a relative fitness of 1.5.

generations on the x-axis as the number of transfers times $6\frac{2}{3}$. This approximation is used in this one case so that the number of generations matches the approximation used when reporting data from the LTEE, with which many simulated datasets will be compared.

4.5 Mutations

Mutations are a core process in STEPS; without them, there would be no variation on which selection could act. There were many important factors to consider when designing the mutational process in STEPS including when to add them, the different types available, how they might interact with each other, the distributions of their effect sizes, and the rates at which they occur.

One fundamental choice when designing STEPS was to use an "infinite alleles" model. This model assumes that there are so many possible mutations that it is effectively impossible for the exact same mutation to occur and then become established in independently evolving lineages. This model also means we can describe each mutation by a single number, namely the proportional effect that it has on the growth rate of a lineage. This choice does impose some limitations. For example, STEPS cannot be used to model certain situations that can occur in real life, such as mutations that allow organisms to access new traits or interact with other individuals in the population via metabolic byproducts.

Another consequence of the infinite alleles model is that mutations cannot be reverted.

In biology, reversions occur when an existing mutation mutates back to its ancestral (sometimes called "wild-type") state. In STEPS, a mutation doesn't have a physical location on the genome, and so there's no way for a mutation to be undone in this manner. When we describe deleterious mutations later on, we explain how it is nonetheless possible for a later mutation to effectively cancel the impact of an earlier mutation.

Mutations in STEPS fall into three broad categories, namely beneficial ($s > 0$), neutral ($s = 0$), and deleterious ($s < 0$). We will discuss each of these types more in later sections, but first we explain when and how the STEPS program adds new mutations to the population.

4.5.1 Timing of New Mutations

New mutations occur during both growth phases. In Growth Phase 1, mutations are added to the population immediately after each doubling. In the example shown in Figure 16, there are two population doublings in that phase and therefore two opportunities for new mutations to occur. With larger values of D, there would be more doublings and opportunities for new mutations. There is only one opportunity for new mutations in Growth Phase 2. The mutations are not added when the population reaches N_{max}, but instead they are introduced immediately after the population bottleneck at the end of that phase. Adding these mutations only after the bottleneck speeds up the STEPS program because it avoids computing and tracking those new mutations that would immediately be lost during the dilution at the next transfer event.

4.5.2 Number of New Mutations and Placement in Lineages

The expected number of new mutations to be added to the population at each opportunity (as described in the section above) is equal to the total mutation rate of each lineage multiplied by the number of individuals being added to that lineage and then summed over all lineages:

$$Mutations = \sum_{i=1}^{\#of\ Lineages} \mu_i \times N_i \tag{6}$$

where $\mu = \mu_{beneficial} + \mu_{neutral} + \mu_{deleterious}$ is the sum of all three mutation rates for a lineage. The expected value is then used in a Poisson distribution to determine the actual number of mutations that are added to the population.

Each lineage contributes a different number of individuals to the population growth, and therefore the lineages have different probabilities of generating one or more new mutations. The STEPS program weights the probability of a lineage receiving one of the actual mutations by the expected proportion of mutated individuals that lineage contributes to the total population.

When a new mutation is added to any lineage, the type of mutation is chosen from those available with a probability equal to its relative rate. For example, if $\mu_{beneficial} = 0.01$, $\mu_{neutral} =$

0.09, and $\mu_{deleterious}$ = 0.1, then the overall mutation rate is μ = 0.2 mutations per individual. Thus, there is a 5% chance that the new mutation is beneficial, a 45% chance it is neutral, and a 50% chance it is deleterious.

4.5.3 Neutral Mutations

Neutral mutations have no effect on the growth rate of lineages. However, when an individual gets a neutral mutation, a new lineage is created and its count of accumulated mutations is increased accordingly.

4.5.4 Beneficial Mutations

Beneficial mutations increase the growth rates of the lineages in which they occur. The magnitude of the benefit is drawn at random from a distribution of fitness effects (DFE) that depends on the lineage in which the new beneficial mutation occurs. This dependency is caused by epistasis between the new mutation and the genetic background in which it occurs (see Section 4.6 for more details). In STEPS, the DFE for beneficial mutations is always an exponential distribution with a mean value of α^{-1}, but that mean can change as a lineage acquires mutations that alter its fitness. More specifically, the mean of the DFE declines as a lineage becomes more fit, reducing the expected effect size of subsequent beneficial mutations. The STEPS program tracks the value of α for every lineage in the population.

4.5.5 Deleterious Mutations

There are two modes for implementing deleterious mutations in STEPS. The default mode draws the effect size of the mutation from a uniform distribution with values that range from 0 to 1. With the other option, all deleterious mutations have the same effect size. Both options are explained in more detail below.

Uniform Distribution The DFE for beneficial mutations has no upper bound, as the exponential distribution allows for arbitrarily large effects, albeit at increasingly diminished probabilities. By contrast, the DFE for deleterious mutations in STEPS has a lower limit that corresponds to a growth rate of 0 ($s = -1$).

Fixed Effect A fairly common practice in theoretical population biology is to set the effect size of deleterious mutations to some fixed constant, which makes the relevant equations easier to solve analytically. STEPS also allows this possibility in the simulated runs.

4.5.6 Multiple Mutations

In STEPS, as in biology, a newly produced individual may have multiple new mutations. When this occurs, each mutation is introduced to the genome one at a time, and all relevant parameters are updated before the next mutation is introduced. The mutations that are added to the affected individual are not limited to one type (beneficial, neutral, or deleterious). For example, a triple mutant could have three mutations of the same type, all three of the different types, or any other combination of the three types.

4.6 Epistasis

Epistasis refers to the non-additive interaction between two or more genes that influences the resulting phenotype. In nature, epistasis can be manifest in many different ways. In STEPS, the only phenotype is how fast an organism can grow, and epistasis describes how the DFE for beneficial mutations changes depending on the fitness effects of previous mutations in the lineage.

4.6.1 Diminishing Returns

The LTEE and other evolution experiments often exhibit diminishing returns over time as populations are propagated under constant conditions. When organisms face a new environment, they generally have far-from-optimal traits, and many different mutations may give large benefits, thus allowing rapid adaptation. As the organisms adapt to their new conditions, however, it typically becomes harder to find ways of making further improvements. In the STEPS program, the tendency for diminishing returns is implemented by updating the DFE whenever a new lineage appears with one or more new mutations. We can use the LTEE to illustrate. The DFE for beneficial mutations in the ancestral strain had an average $s = 0.012$. However, the first mutations to fix typically had much larger effect sizes, owing to the fact that mutants with larger benefits out-compete those with smaller benefits. When a mutation with $s = 0.1$ occurs, for example, the average of the resulting lineage's DFE is reduced to 0.0063. Figure 18 shows the DFEs of both the ancestor (purple) and the mutant lineage (light blue). Note that the figure only shows s values up to about 0.06, but mutations with even greater beneficial effects can occur in large populations. While the two distributions overlap, the DFE for the beneficial mutant is shifted much farther to the left than the ancestral one. With each subsequent beneficial mutation, the DFE continues to move leftward, with diminishing returns as a consequence.

The equation that STEPS uses to adjust the mean of the DFE comes from a paper by Wiser et al. (2013, *Science* 342, 1364–1367). It uses the inverse of the current DFE's average selection coefficient ($\alpha_n = s_n^{-1}$), the effect size of the new mutation (s_{n+1}), and the strength of the diminishing returns (g) to calculate the new lineage's alpha value (α_{n+1}), as follows:

$$\alpha_{n+1} = \alpha_n \times (1 + g \times s_{n+1}) \tag{7}$$

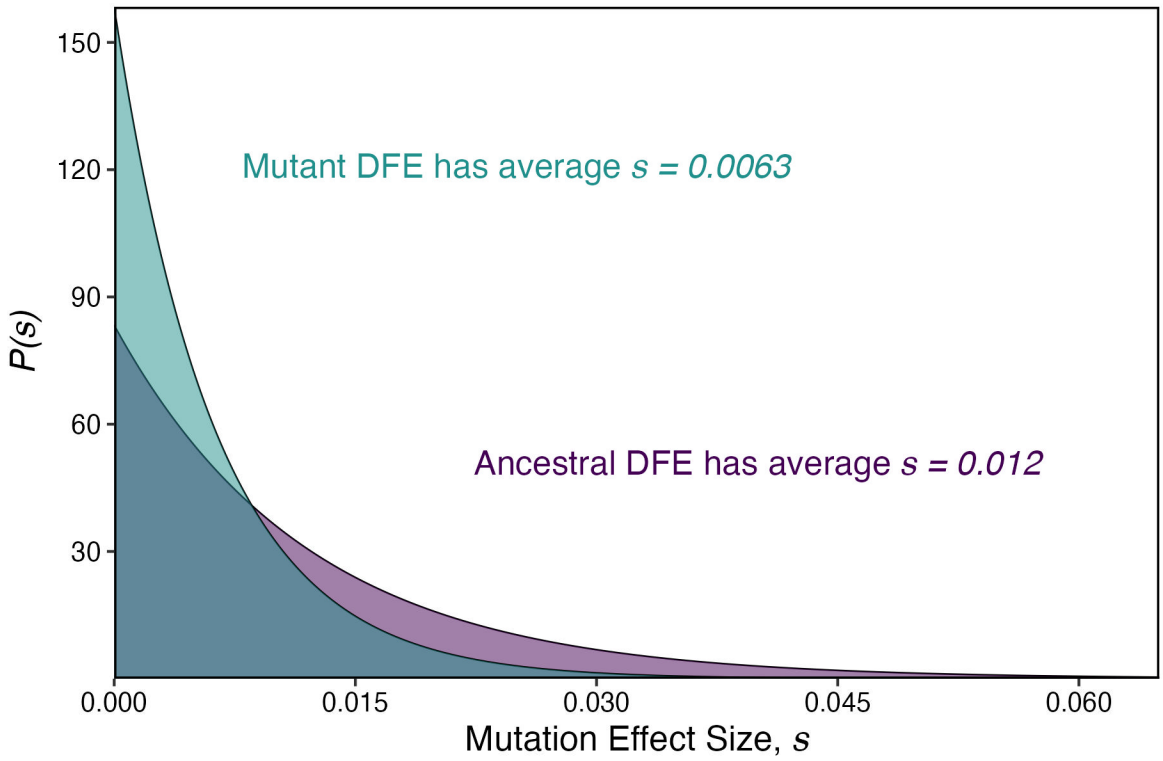

Figure 18: Illustration of how a beneficial mutation reduces the mean of the DFE, thereby shifting the entire exponential distribution leftward. The x-axis has been truncated. The y-axis shows relative probabilities with the same area under the curve for both DFEs.

As previously mentioned, the STEPS program keeps track of the current α value for each lineage. That value allows the proper DFE to be sampled when descendant lineages acquire new mutations, and it is used to calculate the new α values for the descendant lineages.

4.6.2 Epistasis with Deleterious Mutations

The original equation describing the form of diminishing returns epistasis applies to beneficial mutations only. STEPS also allows deleterious mutations, and so we modified the equation by making an additional assumption—if a lineage gets two new mutations, one that reduces fitness and the other that restores fitness to its prior value, then the DFE should also return to its prior state. To that end, we calculate the strength of epistasis for deleterious mutations, g^*, that depends on the values of g and the strength of the deleterious mutation s, as follows:

$$g^* = \frac{g}{-g + s + 1} \tag{8}$$

4.7 Things That Can Go Wrong

We designed the STEPS program to be as robust as possible, but runs may still fail or otherwise behave weirdly under certain conditions. We explained previously that runs may be slow or even fail when the population size and/or mutation rates are so large that the number of lineages that must be tracked exceeds the capacity of the web-based version (see Sections 2.1.3 and 2.5.4). We describe an additional failure mode in the section below and explain how to avoid it.

4.7.1 Caution: Mutations with Extreme Effects

If s_0 is too large or if $g < 0$, then the effect size of beneficial mutations can violate assumptions that are implicit in the growth model we use (see Figures 16 and 17). For example, imagine 999 individuals in the ancestral lineage and a single mutant with a fitness of 10. The total population size would roughly triple during the first timestep that was meant to allow the population to double. This discrepancy occurs because the growth model assumes that the range in fitness values at any timestep is not too extreme relative to the average fitness. In such cases, a simulated population might exceed its maximum size during Growth Phase 1, which would imply a negative timestep for Growth Phase 2. In some cases, STEPS may give an error message when this problem occurs, but it is possible to violate the assumption in less extreme cases without causing the simulation to fail completely.

4.7.2 Reporting Other Problems You Might Encounter with STEPS

If you come across other situations where the STEPS program fails to run or otherwise behaves in a way that does not make sense, you can report the issue via this GitHub link[6]. Please describe the problem and provide the relevant input and output files. We will do our best to either correct the problem in the code or add a relevant caution to the User Manual.

[6] https://github.com/zachmatson/STEPS/issues

5 Acknowledgments

5.1 People

We thank Minako Izutsu for important contributions during the development of STEPS; indeed, she is a coauthor of the STEPS software. We thank Ben Good for valuable discussions during the design of STEPS. We thank Kyle Card and Nkrumah Grant for helpful discussions, and for testing early versions of STEPS. The LTEE inspired us to create STEPS, and we thank everyone who has worked on that experiment over the several decades that it has been running.

5.2 Funding

The development of the STEPS package has been supported by the National Science Foundation (DEB-1951307), a USDA Hatch Grant (MICL12143), and the John A. Hannah professorial endowment at Michigan State University.

5.3 Figures

The cover image was produced by adding the letters to a fitness trajectory generated in a STEPS run. Figure 1 was adapted from files provided by Zachary Blount using Adobe Illustrator. Figures 2–4 and 6–13 are screenshots of the STEPS web portal. Those figures that show data used the indicated parameters and the random number seed 606, except Figure 11, which used 1234 as the seed. The screenshots were taken on an iMac desktop computer using Safari as the browser. Figure 14 is a screenshot of the terminal window on an iMac desktop during a STEPS run. Figures 5 and 15–17 were produced using R (version 4.3.0) and the tidyverse package (version 2.0.0).

5.4 Citing the STEPS Package and User Manual

If you present results from simulations using the STEPS software, we would appreciate being cited in any resulting publications, documents, and presentations.

If you employ the new methods, concepts, or examples presented in this user manual, it can be cited as follows: Devin M. Lake, Zachary W. D. Matson, and Richard E. Lenski. 2025. STEPS User Manual (v2.0.0). *Zenodo*. 10.5281/zenodo.16690374