REvoSim Documentation

Release 3.0.0

Mark D. Sutton, Russell J. Garwood, Alan R.T. Spencer, Euan Furr

Aug 09, 2023

Contents

The [R]apid [Evo]lutionary [Sim]ulator program.

REvoSim is an individual-based evolutionary model, using a simplified first-principles evolutionary model to facilitate high computational efficiency, allowing the simulation of large populations incorporating space, over geological time, using modest computer hardware. It can simulate populations of 10^5 –10^7 digital organisms over geological timescales, and incorporates spatial and temporal environmental variation, recombinant or asexual reproduction, mutation and dispersal. Speeds attainable depend on the computer hardware in use, the size of the populations simulated, and details of the experimental setup (most notably on whether species tracking and fitness recalculation are activated). With a typical 2018 desktop computer, speeds of between 500,000 and 1,000,000 iterations per hour can be achieved for populations of around 250,000.

REvoSim has been in development since 2008, and has been released with the intention that it can be used as a multipurpose platform for the study of many evolutionary phenomena. While it was designed with macroevolutionary studies in mind, it is also applicable to microevolutionary problems. As such it is complementary to the many other approaches of studying evolution on a range of different timescales. It is continually developed by the core team to expand its capabilities.

t:@palaeoware

e:palaeoware@gmail.com

w:https://github.com/palaeoware

CHAPTER 1

Relevant references

1.1 Software references:

Garwood, R.J., Spencer A.R.T. and Sutton, M.D., 2019. REvoSim: Organism-level simulation of macro- and microevolution. Palaeontology 62(3),339-355. https://doi.org/10.1111/pala.12420

Furness, E.N., Garwood, R.J. and Sutton, M.D., 2023. REvoSim v3.0.0: A fast evolutionary simulation tool with ecological processes. JOSS. Submitted.

1.2 Other references:

Furness, E.N., Garwood, R.J., Mannion, P.D. and Sutton, M.D., 2021. Evolutionary simulations clarify and reconcile biodiversity-disturbance models. Proceedings of the Royal Society B, 288(1949), p.20210240. https://doi.org/10.1098/rspb.2021.0240

Furness, E.N., Garwood, R.J., Mannion, P. D. & Sutton, M.D. 2021. Productivity, niche availability, species richness and extinction risk: Untangling relationships using individual-based simulations. Ecology and Evolution 11(13): 8923-8940. https://doi.org/10.1002/ece3.7730

CHAPTER 2

Table of Contents

2.1 Introduction

2.1.1 Overview

REvoSim can be used to study a range of evolutionary processes. It is based on digital organisms (each is a binary string of a user-defined length), within an environment defined by the RGB values of a two-dimensional image. It is highly abstracted, and is designed for computational efficiency. For versatility there are a large number of user-defined variables: an overview is provided below. This assumes the software has already been installed (instructions can be found on the page *Compiling, Installation, and Requirements*). We recommend reading the paper below for full discussion of REvoSim's approach, potential limitations, and its strengths:

Garwood, R.J., Spencer A.R.T. and Sutton, M.D., 2019. REvoSim: Organism-level simulation of macro- and microevolution. Palaeontology. https://doi.org/10.1111/pala.12420

That paper is based on version 2.0.0, versions from 3.0.0 have a number of additional capabilities including, for example, multi-length genomes (in v2.0.0 these were 64 bits). These changes are documented in the paper:

Furness, E.N., Garwood, R.J. and Sutton, M.D., 2023. REvoSim v3.0.0: A fast evolutionary simulation tool with ecological processes. JOSS. Submitted.

A utility program, EnviroGen, is available to generate environments for REvoSim and thus provide a high level of control over the nature of the environment used for simulations. This is documented separately.

In brief, controls for the simulation are found on the toolbar at the top of the main window. Hovering a mouse over each toolbar button (or any other areas for user input in REvoSim) will provide an overview of what it does. The buttons are as follows:

Run Launch a simulation

Run for Launch a simulation and then allow it to continue for a set number of iterations.

Batch Repeat Run for *n* times.

Pause Pause a simulation.

Stop Cancel a simulation.

Reset Reset the simulation and reseed with a random digital organism in the central pixel.

Reseed Launch a dialogue to allow the simulation to be reseeded with a known genome, with two individuals that share a (random or user defined) genome, or with a large number of organisms with fixed genome with known properties (see *Interactions*).

Genome Launch Genome Comparison Dock which allows genomes to be inspected and compared.

Settings Launch Settings Dock which allows variables to be defined.

Logging Output options are included in a separate dock, launched by clicking this button.

Tests Switch REvoSim to test mode and run software tests.

About Launch dialogue with information about REvoSim.

The main part of the window comprises two panels:

Population view Provides an overview of the population alive at any given polling iteration. The information shown can be selected with a drop down menu at the top.

Environment Shows the RGB environment which is used to calculate organism fitness, or - if present - image stacks being used to control the underlying variables.

Below this is the *Information Bar*, which shows a number of statistics for the given run, updated each polling iteration. These include population size, number of species, iterations and speed. You can find more information on the *Window Layout* pages.

2.1.2 Simulation setup

Variables can be defined within the settings dock on the right. Full descriptions of these and their implications can be found in the REvoSim paper. Clicking settings on the toolbar at the top of a window toggles the visibility of this dock. At the bottom of the dock are three tabs, each of which has variables associated with different aspects of the simulation.

Organism tab This includes the variables which dictate the behaviour of the digital organisms in a REvoSim run. This includes chance of mutation, starting age (i.e. length of life), breed threshold and cost, mode of breeding, and breed settings. More information: *Configuring your Organisms*

Simulation tab This includes the settings for the environment and associated files, simulation size, fitness target (i.e. the nature of the fitness lanscape), energy input, settle tolerance, and species tracking. More information: *Setting up the Simulation*

Interactions tab Here you can find the settings for interactions between individuals, and also settings for the predator system in REvoSim. More information: *Interactions*

2.1.3 Output setup

The output can be configured within the logging dock on the left, which has two tabs at the bottom of the dock.

Output tab This includes output options for the simulation: save directory, refresh rate, and logging/output options. More information: *Configuring Outputs and Run End Log*

Running log From v3.0.0 REvoSim includes a highly customisable running log, the contents of which can be defined using this tab. More information: *Configuring Outputs and Run End Log*

2.1.4 Quick start

A simulation - using default settings and environment - can be started by hitting the Run button. In addition to the visualisation, runs can be analysed using log files which are placed by default in a folder called *REvoSim_output* on the desktop for all operating systems. A log is written during a run when "Write Log Files" (Logging dock, Output tab) is checked, and the phylogenetic tree and other more detailed statistics for a run can be written at any point by clicking the button "Write data (including tree) for current run".

2.2 Compiling, Installation, and Requirements

2.2.1 Installation

Pre-compiled binary releases and packaged installers can be downloaded from the REvoSim GitHub repository. For Windows users we provide both a portable binary release (.zip) - which just needs extracting to a convenient location - and a self contained installer. For Mac we provide a zip containing the REvoSim program that can be downloaded from the REvoSim GitHub repository. To install the software, drag and drop the required .app folder(s) into the Applications folder. You may be required to the approve the software in security and privacy settings before it will launch. For Linux users, the instructions below will allow the software to be built using a limited number of lines of bash. Please contact palaeoware@gmail.com if you encounter any issues or would like an app image.

2.2.2 System requirements

REvoSim has no minimum requirements as such, and will run on most standard systems (Windows/Linux/Mac); it has not been tested, however, on versions of Windows older than Windows 10, Ubuntu 16.04, and macOS High Sierra. Performance will benefit from high processor speed and increased number of processor cores, with large amounts (>4GB) of available RAM recommended for large simulations. Graphics card performance is not relevant as GPUs are not currently used in the program's calculation pipeline. A fast hard drive (e.g. SSD) is recommend when intensive logging is enabled; as slow I/O response time can affect the iteration cycle speed.

We recommend a minimum of 1GB RAM and a 1.8 GHz or faster, ideally multicore processor. We also recommend a minimum screen resolution of 1280x720 if using the software without the genome comparison docker (and 1920x1080 if this is enabled).

2.2.3 Compiling from Source

If you wish to use REvoSim it is generally easiest to use the resources linked above. However, if you wish to develop features, the software can be compiled as follows.

Windows 64-bit

QT Creator + QT >v5.11 using MSYS2 (64-bit) and MinGW (64-bit). We recommend you install and use MSYS2 (64-bit) a Windows package manager, based on modern Cygwin (POSIX compatibility layer) and MinGW-w64, that allows easy installation of QT v5.x 64-bit.

1. Download and run the latest version of MSYS2 for 64-bit Windows. This will be name "mysys2-x86_64-…" for the 64-bit installer.
2. Follow the install instructions. We have used the default install location of "C:mysys64" and it is here that includes required in the .pro files point. If you install MSYS2 to another location the .pro files will need to be updated to your install location.

3. Once installed open up MSYS2 shell and run the pacman update command: pacman -Syu Note that as this will almost certainly update pacman itself you may have to close down and restart the MYSYS2 shell before re-running the command to finish.
4. Once MSYS2 and pacman are fully updated run the following command to install QT 5.x and its dependencies: pacman -S mingw-w64-x86_64-qt-creator mingw-w64-x86_64-qt5
5. Optional - if you intend on debugging the software in QT and wish to use GDB then run the following to install the matching GBD debugger: pacman -S mingw-w64-x86_64-gdb
6. **At this stage you should have the following under the MYSYS2 install location:**
    - {install location}/mingw64 (Main ming64 folder)
    - {install location}/mingw64/bin/qmake.exe (QMake for QT version)
    - {install location}/mingw64/bin/g++.exe (C++ complier)
    - {install location}/mingw64/bin/gcc.exe (C complier)
    - {install location}/mingw64/bin/gdb.exe (Debugger | OPTIONAL)
7. You should now be able to find the required libraries under "{install location}/mingw64/bin" and the required header (.h) files for QT v5.x.
8. Open the .pro file in QT Creator, and then use the information above to setup a new 64-bit ming64 kit. Follow standard QT Creator debug/release procedure.

Ubuntu 18.04/20.04/22.04 64-bit - QT Creator + QT >v5.11 using GCC (64-bit)

To compile from command line.

1. Install GCC and Qt using system packages:

```
sudo apt-get install build-essential libgl1-mesa-dev
sudo apt install qt5-default
```

2. Download source code and navigate to folder, or alternatively clone using Git:

```
git clone https://github.com/palaeoware/revosim.git
cd revosim
```

3. Within REvoSim folder create makefile:

```
qmake ./revosim.pro
```

4. Build by running the make command:

```
make
```

5. Navigate to bin folder (e.g. revosim/bin) and launch software by double clicking on file.

On older operating systems, the OS Qt distribution will result in a large number of compile warnings. None impact on the functioning of the software.

Using Qt creator.

1. Install Q5.X on your system by running the installer from Qt: https://www.qt.io/download Further instructions are available here: https://wiki.qt.io/Install_Qt_5_on_Ubuntu
2. Download source code, launch Qt Creator, and open the .pro file. Configure build and follow standard debug/release procedure.

MacOS

QT Creator + QT >v5.11

The above (Linux, using Qt Creator) approach should also work for MacOS builds. This will require xcode to be installed, which you can do using the app store, followed by QtCreator, which can be achieved through the Qt online installer. To build the software, download source code, launch Qt Creator, and open the .pro file. Configure build and follow standard debug/release procedure.

2.3 Concepts and Example Usage

2.3.1 Concepts

There are a number of underlying concepts that are useful to consider when using REvoSim. Please bear in mind that the model itself is fully described in the publications listed on the *Introduction* page. Prior to publishing any work using the software, or for more details on the following, please do check these out. To see how the software fits into the wider field, the paper Digital Evolution for Ecology Research: A Review provides a useful overview.

If it would be beneficial to have further concepts outlined below, please contact the authors.

Species concept

REvoSim is an individual-based simulation, and thus during runs population(s) of digital organisms evolve under a set of rules. As they evolve, species can emerge. These stem from the species concept employed by REvoSim which is based upon reproductive isolation, and is thus akin to the biological species concept. The implementation of this (the species algorithm) is described in the 2019 paper documenting the model, but involves a pairwise comparison of all individuals within a simulation every polling iteration to identify reproductively isolated clusters, and then a tracking function between these iterations. The level of dissimilarity required for reproductive isolation to occur is under user control: the option Maximum difference to breed dictates how many bits different two genomes are before they are considered isolated by this algorithm. Lowering this number creates more speciation-prone simulations.

Within this framework, runs in REvoSim - that typically start with a single genome - also start with a single species, that will go on to dominate the simulation. As a simulation runs, however, the population evolves, and under most settings species start to appear when populations specialise to a point when none can interbreed with any other organism alive within the simulation. Sometimes these will be sympatric (i.e. they occur in a single block of colour) but often, as a simulation runs, species will evolve in, and then start to track, particular colours (~niches) in their environment.

Time

Time in a REvoSim simulation is measured in iterations. Every iteration, the algorithms that comprise the model are completed once. In order to link this to real world time units, it is useful to consider the average time per generation. Exactly how iterations map to real time, however, depends on the settings you use for any given run. You can investigate this using REvoSim's *Running Log* system: the *gridGeneration* tag can be used for any settings to calculate and output the average age of all organisms successfully breeding in a polling iteration. This works on the assumption – which is true for the majority of settings – that organisms will breed once in their lifetime. This assumption can be tested (and a correction factor calculated and applied if so desired) by using the *gridNumberAlive* and *gridBreedSuccess* log outputs to calculate the proportion of breeds for the grid per unit population.

Using this approach demonstrates, for example, that the average generation time for default REvoSim settings is between 11 and 12 iterations.

2.3.2 Example Usage

Setting up REvoSim simulations to test particular eco-evolutionary hypotheses involves several steps. This page will guide you through setting up simulations to test the "more-individuals" hypothesis of species richness: the hypothesis

that the species richness of ecosystems is controlled by the number of individuals that they contain. REvoSim is used to test this hypothesis in Furness et al. (2021).

Is REvoSim suitable?

A variety of different tools exist to simulate eco-evolutionary processes under different environmental conditions (see Dolson & Ofria (2021) for a review). Each one of these tools makes trade offs between complexity and computing efficiency, and so each will be suitable for answering different research questions. As an individual-based eco-evolutionary simulator, optimised to operate over geological timescales, REvoSim is relatively well placed to test hypotheses regarding the process of speciation within large populations. It is therefore suitable to investigate the "more-individuals" hypothesis, but be aware that other tools, such as gen3sis, may be more appropriate if the phenomena of interest are operating at the level of the population, rather than the individual.

Choosing logging options

In order for our REvoSim runs to produce useful data, we must select appropriate logging options before running any simulations. The "v2.0.0 log" option provides a useful default choice for log content, which includes species richness in every logged iteration and the total number of organisms alive in each logged iteration. The v2.0.0 log is therefore suitable for testing the more-individuals hypothesis, and also serves as a useful starting point for building a logging output in general. We must also check the "write to file" box, in order to produce a log in each experiment, and we must specify a directory to which the log is to be written.

Choosing an environment

REvoSim's default environment of three vertical blue bars is useful as a tool for demonstrating adaptation of organisms to specific environments. However, it is not a very realistic environment: it contains three distinct habitat types (i.e.colours), with no gradational change between colours and no temporal variability. We might wish to replace it with a more realistic environment, such as a lights-type environment generated by the EnviroGen tool. Alternatively, we might wish to replace the default environment with an environment designed to maximise species richness, such as a noise environment from the EnviroGen tool, so that we can more easily observe any effects of the number of individuals on the number of species. In the course of a research project, we probably want to run simulations using both of these options.

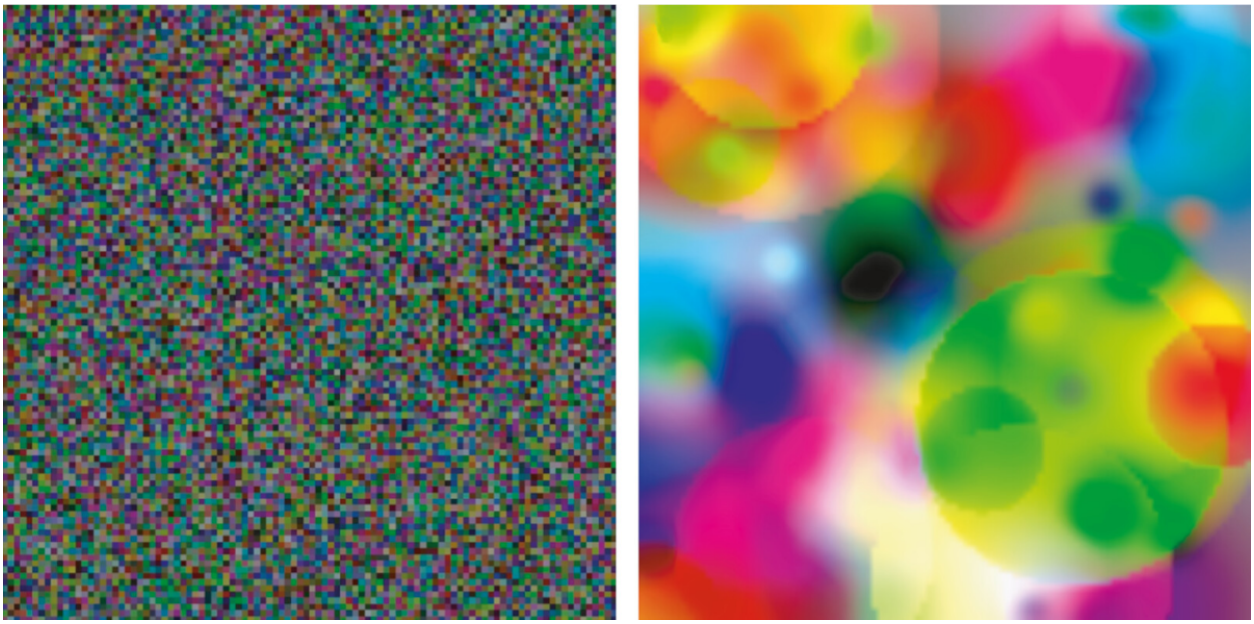


Fig. 1: Above: examples of the noise (left) and lights (right) environments from EnviroGen.

Choosing simulation settings

The default settings in REvoSim have been shown to produce realistic evolutionary phenomena such as adaptation towards fitness peaks. Consequently, choosing simulation settings for experiments in REvoSim is usually as simple as modifying settings that are to be treated as variables in the experiments while leaving other settings as defaults. To test the more-individuals hypothesis, we need to vary the number of individuals in the simulation, which means varying the energy supplied to the organisms in the simulation for reproduction. We can control this by changing the "energy input" setting from its default of 2000 to either, for example, a low value of 1000 or a high value of 4000examples. We may also wish to change the "environment mode" setting: the "bounce" setting works well with lights-type environments, and "static" may be appropriate for a noise environment if we want to allow for the highest possible species diversity.

Running the simulation

Now that our logging options are set, our environment is chosen, and our settings are modified as required, we need to run the simulation. To ensure that all is functioning as intended, or if only a few runs are needed, we can use the "Run for" button in the GUI. This will produce a dialogue box that asks us how many iterations we want to run our simulation for. Generation times under default settings are 10-15 iterations, so 50,000 iterations, representing as many as 5,000 generations, is likely to be sufficient to reach equilibrium. While the simulation runs, it will continuously write logging information to a text file (REvoSim_output.txt) in the specified logging directory.

If we want to run a large number of replicate simulations, then running each individually would take a lot of effort. The "Run Batch" button allows us to run several identical simulations back-to-back, outputting logs for each. Alternatively, if we want to run a large number of simulations with slightly different settings (for example, slightly different energy levels, representing a gradient from high to low energy), we can use REvoSim's *Command Line Options*. Below is an example of a single instruction to the command line in the Windows operating system that runs a REvoSim simulation. It will do so with a non-default environment (whatever images are in the folder at the filepath following the "-e" tag) and non-default simulation settings ("-m Static" means that the environment will not change over time, "-n 4000" means that the simulation will have an energy input of 4000 units, "-v2log True" means that the simulation will produce a v2.0.0 log output to the filepath specified after the "-j" tag, and "-auto 50000" means that the simulation will end after 50000 iterations have elapsed):

```
C:\Users\Guest\Desktop\revosim\bin\revosim.exe -e C:\Users\Guest\Desktop\Folder_
↪Containing_Environment_Files\ -j C:\Users\Guest\Desktop\Folder_To_Store_Output -m␣
↪Static -n 4000 -v2log True -auto 50000
```

This command line approach is often the most useful for running actual experiments in REvoSim, because of its ability to modify variables along a gradient.

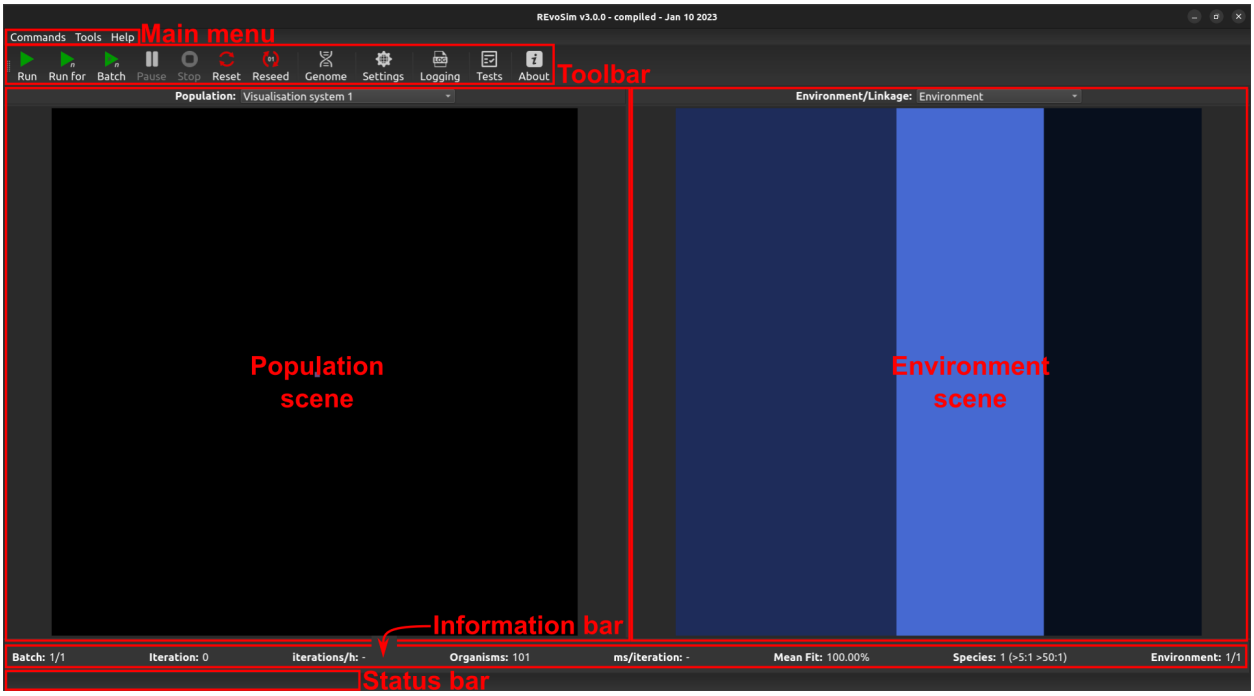
Processing logs

REvoSim is relatively flexible in terms of the format of the logs it produces. However, some processing of these logs will be necessary in order to visualise or analyse the data that they contain. REvoSim does not include any software for this processing, but since logs are output as text files, they can be easily manipulated by a variety of other tools. String handling in Python, or any other appropriate coding language, is a straightforward way of converting data in the REvoSim log file into a useable dataset. Examples of this code can be found in the supporting information of Furness et al. (2021), but keep in mind that different logging setups will require different processing code (and note also with the custom log functionality, logs can be output as e.g. csv files to be loaded into spreadsheet software if desired).

By writing a short piece of code that extracts from each REvoSim log file the number of species and number of individuals present in each REvoSim simulation in its final iteration, we can build up a picture of the relationship between these two variables. This allows us to directly test the predictions of the more-individuals hypothesis.

2.4 Window Layout

Each of the above mentioned features is documented in more detail on the following pages:



2.4.1 Main Menu

The Main Menu, located at the top of the program window, allows access to all program actions, functions, and settings. The menu is currently sub-divided into three sections:

1. *Commands*
2. *Tools*
3. *Help*

Commands

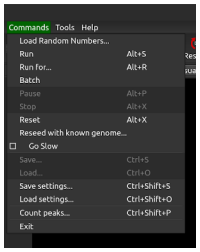


Fig. 2: The commands menu

The Commands menu holds the majority of available actions and program functions related to running of one or more simulations. The options are as follows:

Load Random Numbers Allows user-selected random numbers to be loaded through a custom file, if you so wish. See *Custom Random Numbers*.

Run .... Reseed with known These options are provided as alternatives to the buttons on the top toolbar of the GUI.See *Main Toolbar*.

Go slow This option slows the simulation to allow environmental changes and the visualisation in the population view to be viewed more clearly. It achieves this by adding a 30ms delay to every iteration.

Save This saves the current state of the REvoSim simulation, allowing it to be loaded later, including the masks, organisms, and all settings. This is saved as a binary file to allow the minimum file size possible.

Load This loads the above REvoSim file.

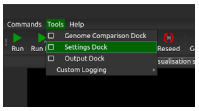
Save settings This saves the settings of REvoSim in a given state. This includes all user-defined variables, but nothing else. These are saved as a human-readable XML file.

Load settings Loads a settings file.

Count peaks This is provided to help the user understand the fitness landscape of their run (albeit in simple terms). See *Count peaks*.

Exit Quits REvoSim.

Tools



The Tools menu allows access to the built-in dockable widgets (called 'Docks') which alter or extend the core program functions. This includes the main simulation settings dock, described in *Configuring your Organisms* and *Setting up the Simulation*, the output dock, covered in *Configuring Outputs and Run End Log*, and the genome comparison dock (*Genome comparison dock*).

The development version of the software also has a series of custom logs, generally written at the logging iteration, and placed in the same folder as the main log (many of these can now be achieved using the new logging tools in versions since REvoSim 3.0.0):

Fitness logging This is largely obsolete from v3.0.0 as the same functionality is possible with the custom log. It is kept here for backward compatibility.

Recombination logging This log outputs the proportion of asexual v.s. sexual breeds when variable breeding is enabled, for each iteration for the grid as a whole. It also outputs the total breed attempts and fails, and also number of living organisms.

Variable mutation logging This outputs the iteration number, and the number of ones in the part of the genome in the variable mutate system.

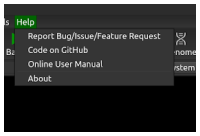
Speciation logging For every speciation event above minimum species size, this outputs iteration number, number of species in event, shared cells, and then for each species the ID, number of individuals, number of cells occupied total, and number of cells in which species is sole occupant. Requires phylogeny mode to be set to *Phylogeny and metrics*.

Disparity logging For each polling iteration this does a dump of every single genome, plus its X and Y coordinates.

All logs contain a copy of the run settings, and explanatory text.

Help

The Help menu contains links to useful program information. The first link will allow you to report a bug or request a feature. The next links to the Palaeoware code repository, and the third to REvoSim documentation. The fourth will load an about dialogue.



2.4.2 Main Toolbar

The main toolbar consists of the following buttons:

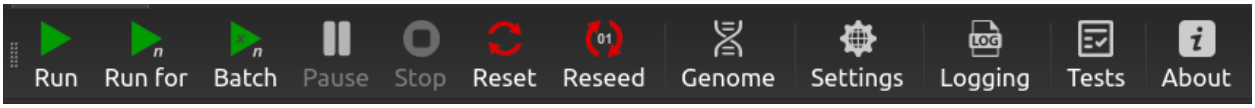


Fig. 3: Main toolbar (no simulation running).

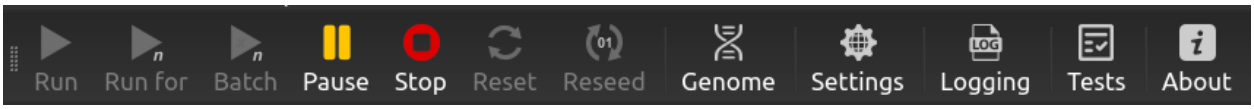


Fig. 4: Main toolbar (with simulation running).

The first four ("Run", "Run for", "Batch", "Pause" and "Stop") control the initiation and cessation of simulation runs. These commands can also be accessed from the *Commands* menu.

Run This button launches a simulation, and then runs it until it is either paused or stopped.

Run for This launches a simulation and runs it for a user-defined number of iterations.

Batch For repeated runs using the same settings, REvoSim provides a batch mode: this provides the option of repeating the environment, or continuing from the last environmental file loaded. Logs in batch mode will be labelled accordingly. The number of runs, and for how many iterations these should last, are requested on launching batch mode.

Pause Pauses a simulation, allowing it to be continued when requested.

Stop Stops a simulation and resets the GUI, but leaves the simulation in its current state.

Reset Resets the simulation by removing all digital organisms, and then placing a random individual capable of surviving in the central pixel.

Reseed Launches a dialogue to allow the simulation to be reseeded with a known genome, or with two individuals that share a (random, or user-defined) genome. Not all genomes are capable of surviving in a REvoSim run: if reseeded with a genome incapable of survival, REvoSim will provide an error. To allow reseeding with a known genome - but one which can survive in a given environment, the dialogue provides a list comprising the top ten genomes from the genome comparison dock (which can be populated prior to a given run). See *Genome comparison dock*.

Genome Launch Genome Comparison Dock, described in *Genome comparison dock*.

Settings Launch Settings Dock which allows variables to be defined. See *Configuring your Organisms*, and *Setting up the Simulation*.

Logging Launch Output Dock. See *Configuring Outputs and Run End Log*.

Tests Switch to the REvoSim test mode and run software tests.

About Launch dialogue with information about REvoSim, including version number, authors, license information, and contact details for the Palaeoware team.

The toolbar itself can be moved to one of four available positions using drag and drop: top (default), left, right, and bottom of the window. The toolbar can also be undocked from the main window and used as a floating toolbar (i.e. an independent window). To move the toolbar or to undock it as a floating window use the left mouse button on the three dotted handle (far right of the toolbar by default), then holding the mouse button down drag the window to it desired position.

2.4.3 Population Scene

The Population Scene is the program's visual output showing what is taking place within the defined simulation space. When the program is first started, or after a Reset is called, the Population Scene will default to showing a single coloured pixel in the centre of the population grid, unless Dual Reseed or 3- or 5-tier Trophic Reseed is selected is selected. By default this pixel represents the starting genome of the seeding organism: black pixels are those without any living digital organisms.

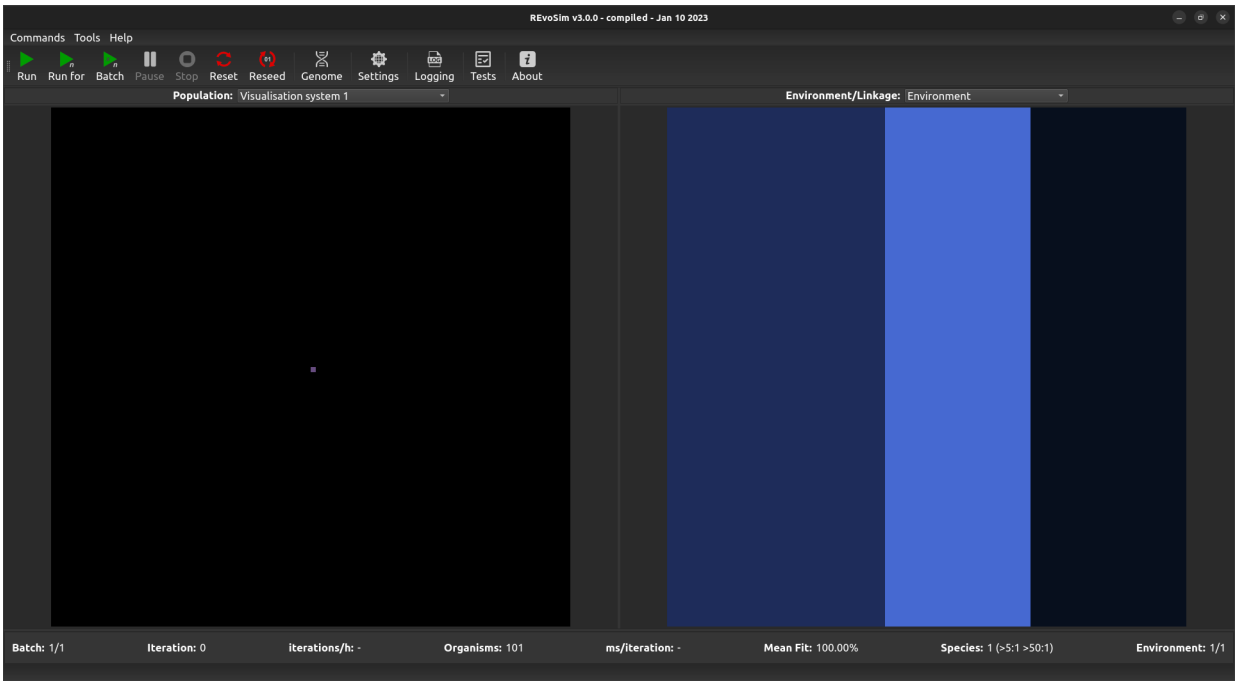


Fig. 5: Default Population Scene showing a single starting genome in the centre.

The Population Scene can be set using the drop down menu to show different output modes, as shown below:

The Population Scene can also be saved as an image stack in order to e.g. create movies or analyse runs - see *Configuring Outputs and Run End Log*. Currently supported visualisation modes are:

Population Count

Each grid-square is visualized with a grey-level representing the current number of creatures alive in the square.

Mean Fitness

Each grid-square is visualized with a grey-level representing the mean fitness of all creatures alive in the square, scaled so that white is maximum fitness (='Settle Tolerance').

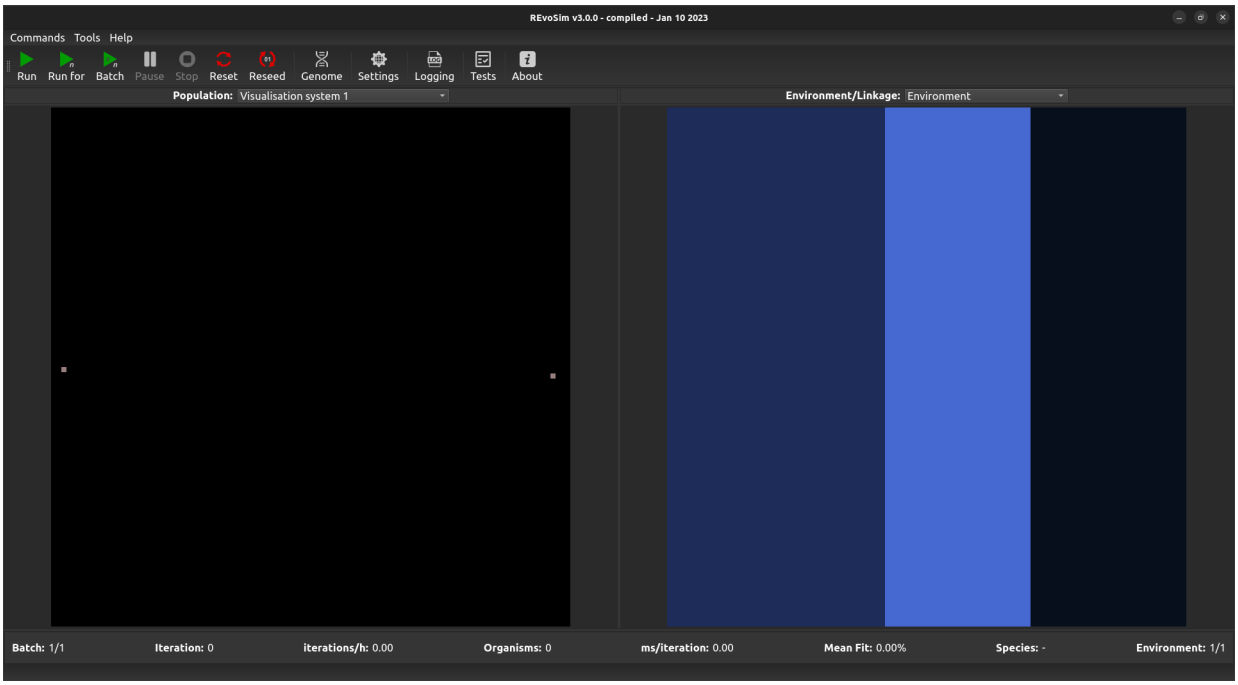
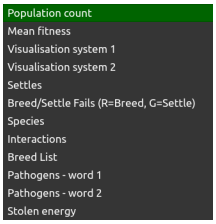
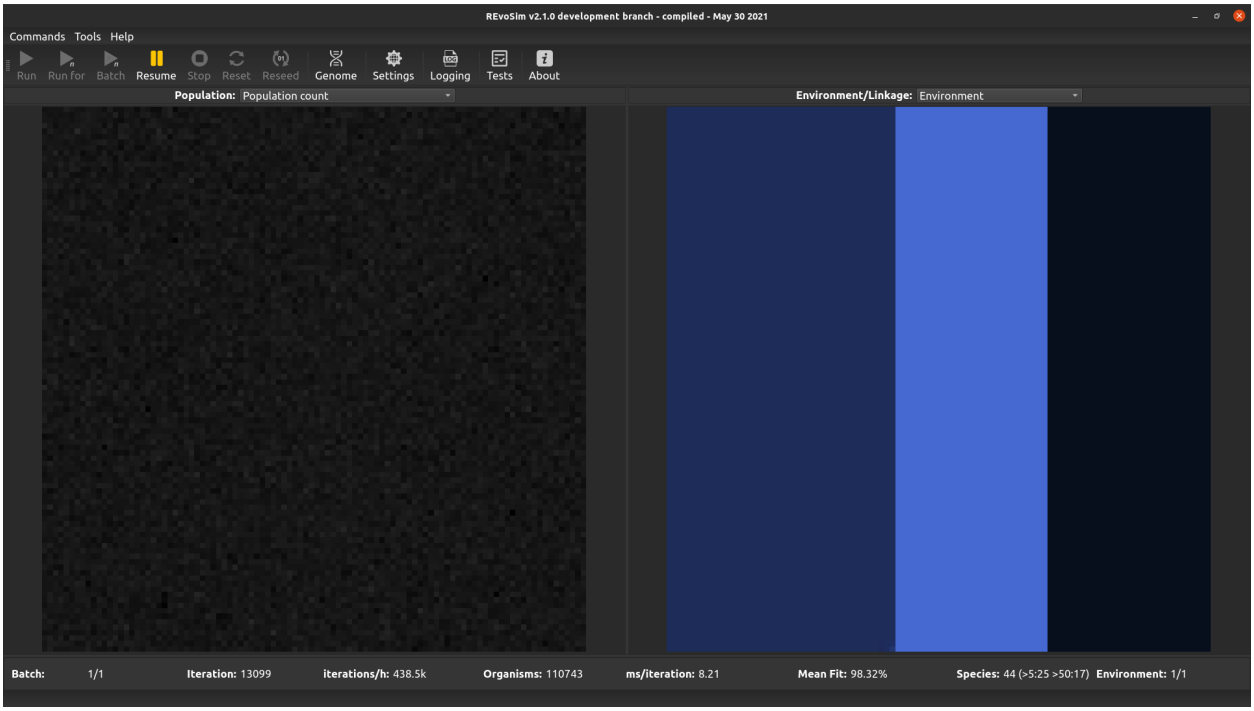
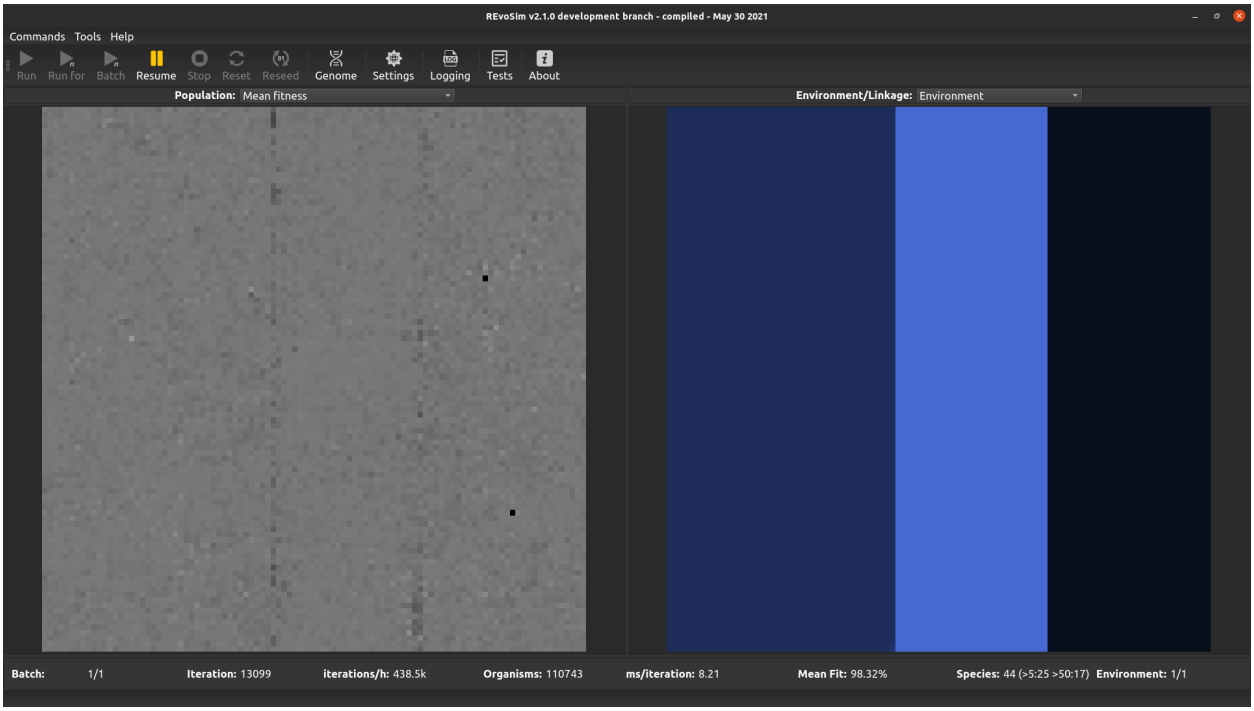


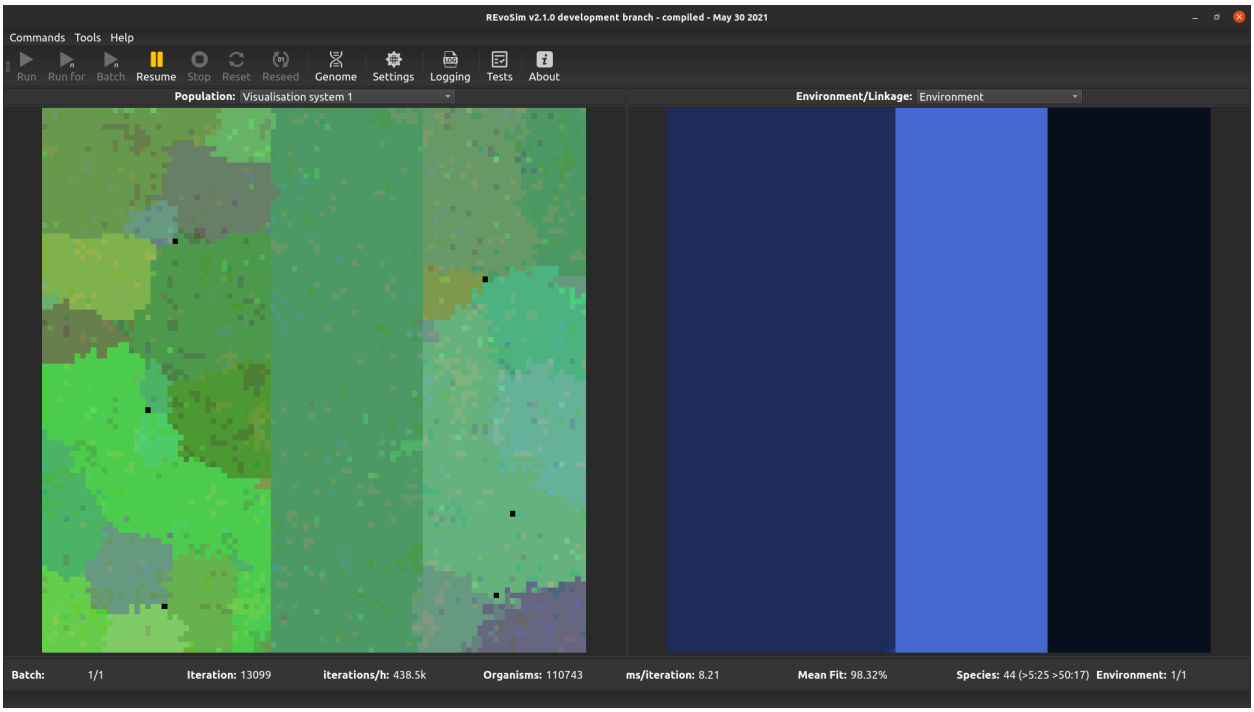
Fig. 6: Default Population Scene showing a two genomes, one on left and another on the right, as set by the Duel Reseed option.




REvoSim v2.1.0 development branch - compiled - May 30 2021
Commands Tools Help
Run Run for Batch Resume Stop Reset Reseed Genome Settings Logging Tests About
Population: Population count
Environment/Linkage: Environment
Batch: 1/1 Iteration: 13099 iterations/h: 438.5k Organisms: 110743 ms/iteration: 8.21 Mean Fit: 98.32% Species: 44 (>5:25 >50:17) Environment: 1/1


REvoSim v2.1.0 development branch - compiled - May 30 2021
Commands Tools Help
Run Run for Batch Resume Stop Reset Reseed Genome Settings Logging Tests About
Population: Mean fitness
Environment/Linkage: Environment
Batch: 1/1 Iteration: 13099 iterations/h: 438.5k Organisms: 110743 ms/iteration: 8.21 Mean Fit: 98.32% Species: 44 (>5:25 >50:17) Environment: 1/1

Visualisation system 1



From v3.0.0 REvoSim features two visualisation systems for user-defined genome words. The words for each are set in the systems section of the Simulation settings docker. These both work in the same way: for each grid-square, the most commonly occurring (modal) sequence for the selected genome words is computed. These are then converted into a colour: e.g. for a single 32-bit word the level of red is the (scaled) count of 1's in the least significant 11 bits, the level of green is the (scaled) count of '1's in the next least significant 11 bits, and the level of blue is the (scaled) count of '1's in the most significant 10 bits. The equivalent operation is applied if >1 genome words are selected. This visualization approach provides a quick means of distinguishing sequences, ensuring that small genomic changes result in small changes in colour. It does not, however, guarantee that the same colour will in all cases represent the same genome (as many very different genomes may possess the same bit-counts in section of the genome words).

Visualisation system 2

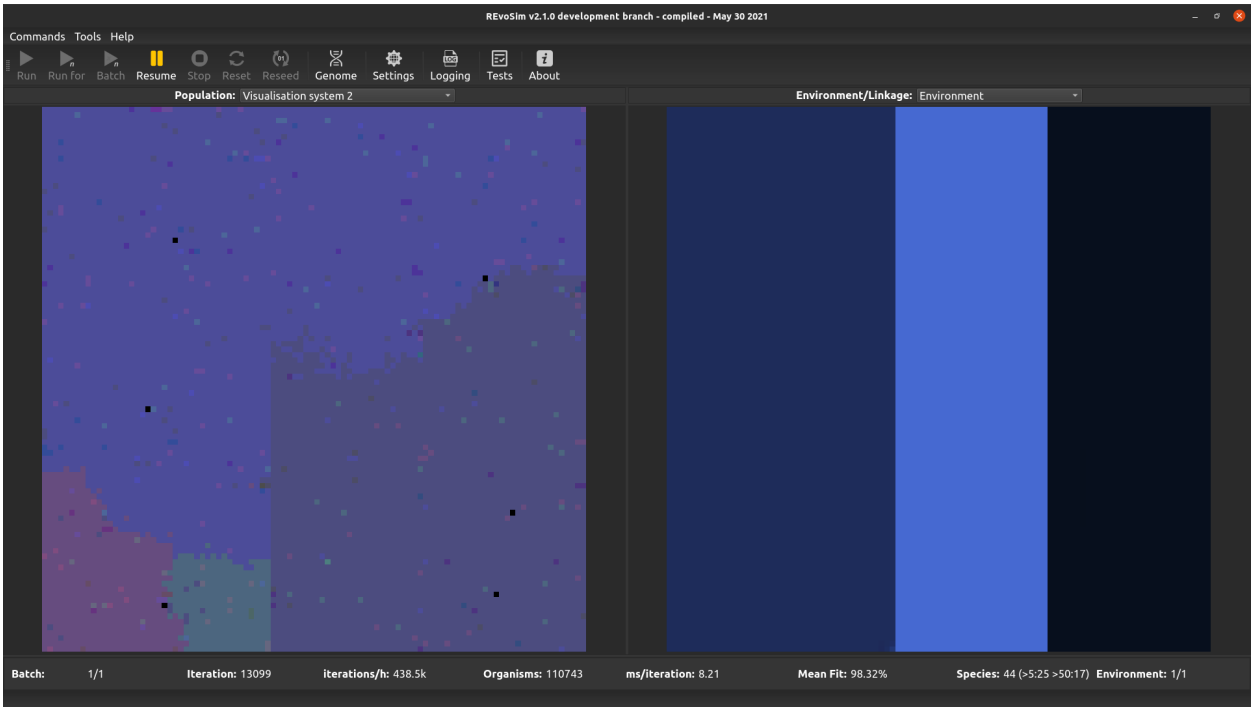
This operates in the same way as first visualisation system, but is included to allow two different word sets to be visualised (and saved as image stacks) if required.

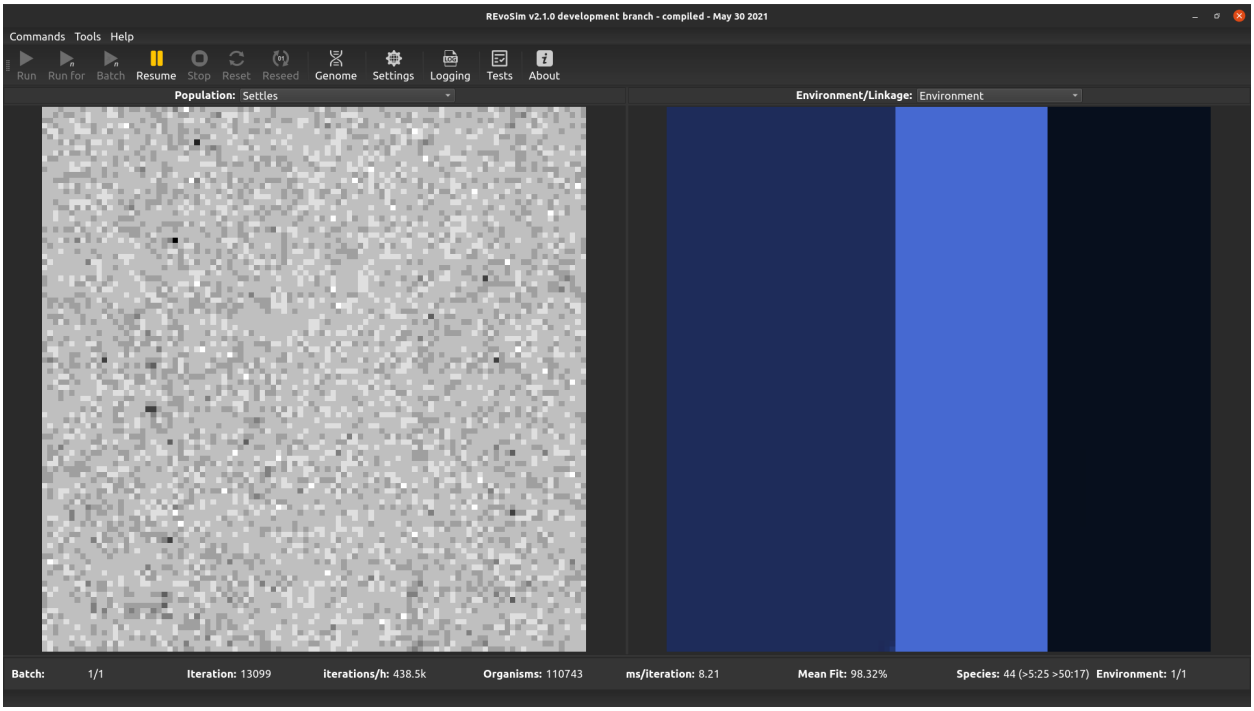
Settles

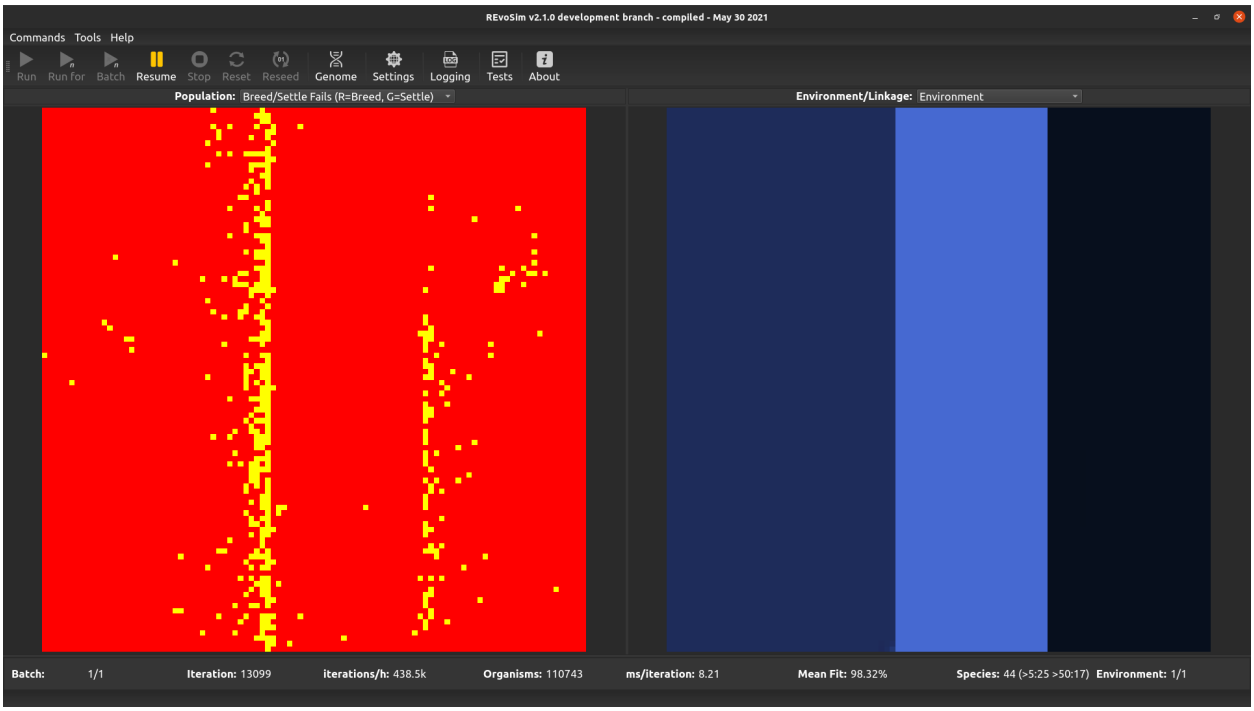
Each grid-square is visualized with a grey-level representing the number of successful 'settling' events in that square since the last visualization.

Breed/Settle Fails (R=Breed; G=Settle)

Each grid-square is assigned a colour representing the number of 'fails' since the last visualization. The number of 'breed fails' (attempts to breed that were aborted due to compatibility) provides the red level, and the number of 'settle fails' (attempts to settle that resulted in a fitness of 0 and hence the death of the settling creature) provides the green level. Fail visualizations are scaled non-linearly (using v = 100*f*0.8, where f = mean fails per iteration, and v is


REvoSim v2.1.0 development branch - compiled - May 30 2021
Commands Tools Help
Run Run for Batch Resume Stop Reset Reseed Genome Settings Logging Tests About
Population: Visualisation system 2
Environment/Linkage: Environment
Batch: 1/1 Iteration: 13099 iterations/h: 438.5k Organisms: 110743 ms/iteration: 8.21 Mean Fit: 98.32% Species: 44 (>5:25 >50:17) Environment: 1/1


REvoSim v2.1.0 development branch - compiled - May 30 2021
Commands Tools Help
Run Run for Batch Resume Stop Reset Reseed Genome Settings Logging Tests About
Population: Settles
Environment/Linkage: Environment
Batch: 1/1 Iteration: 13099 iterations/h: 438.5k Organisms: 110743 ms/iteration: 8.21 Mean Fit: 98.32% Species: 44 (>5:25 >50:17) Environment: 1/1



visualized intensity on a scale 0-255). Fail visualization maps the edge of species or subspecies ranges, as it highlights cells where gene-flow is restricted by any mechanism.

Species

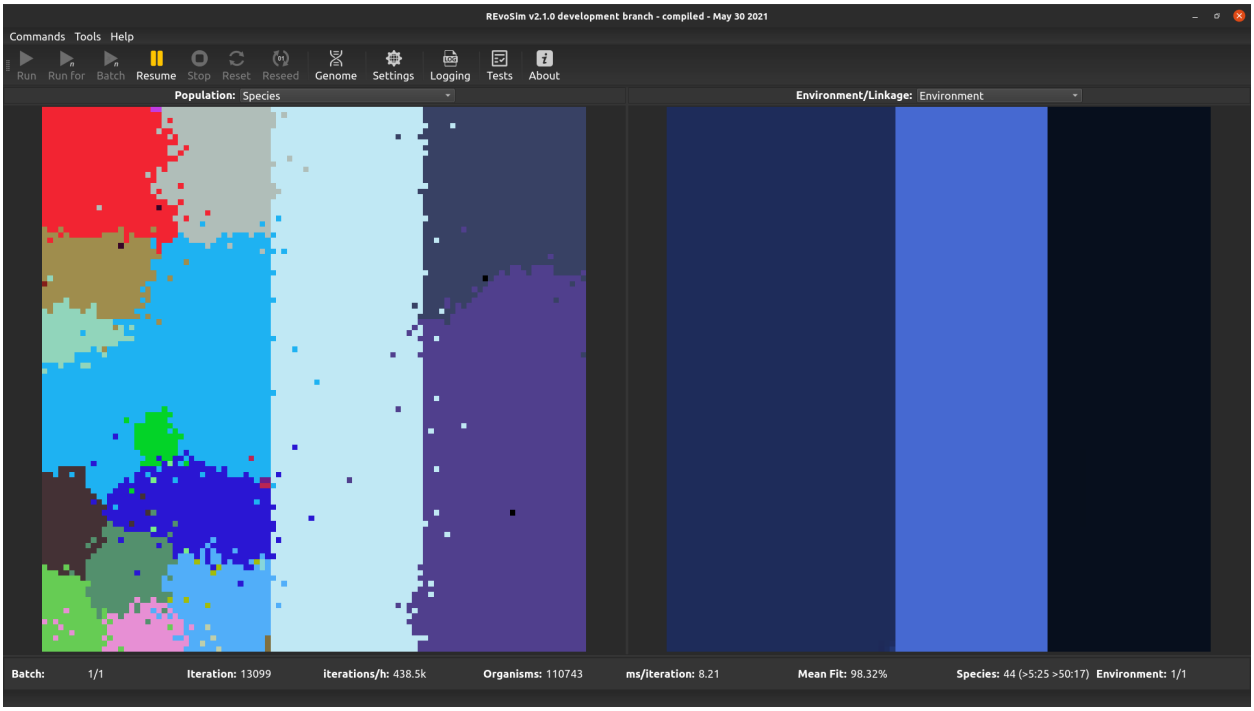
REvoSim simulations often produce discrete species, i.e. reproductively isolated gene-pools, and for many macroevolutionary studies the identification and tracking of the fate of species is a requirement. This visualisation option assigns a unique colour to each species, and colours each grid-square with the species in the lowest occupied slot of that square.

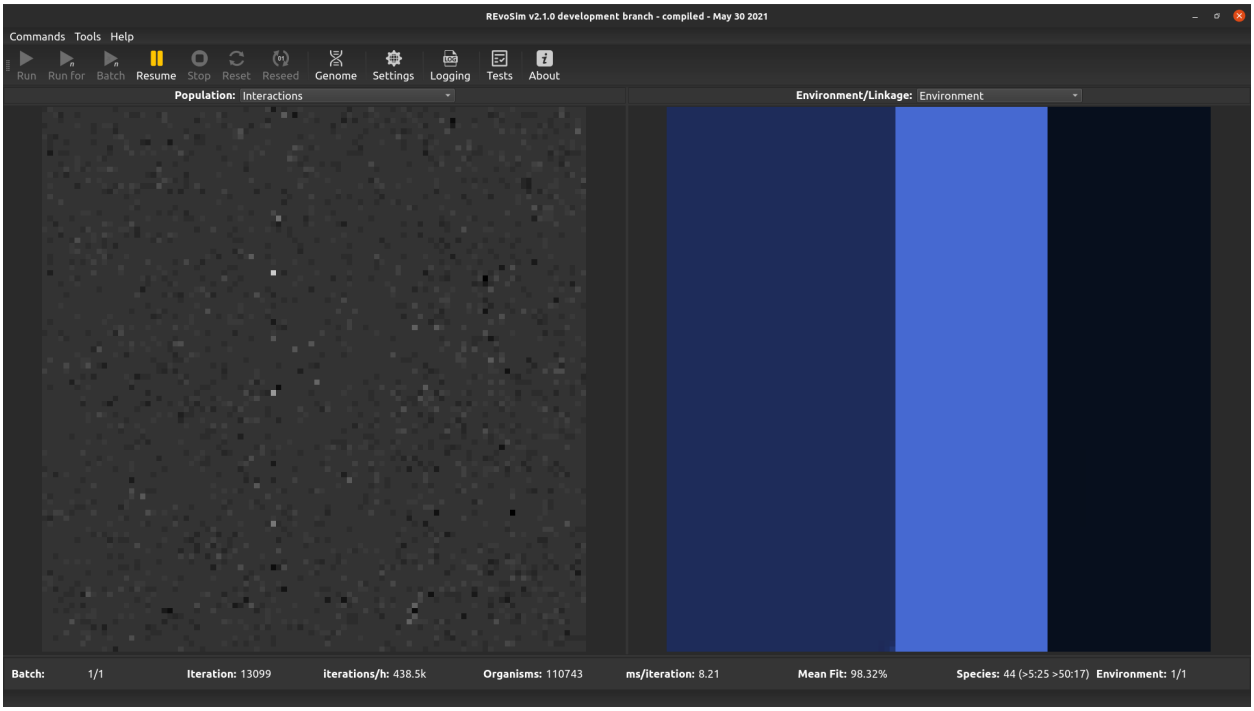
Interactions

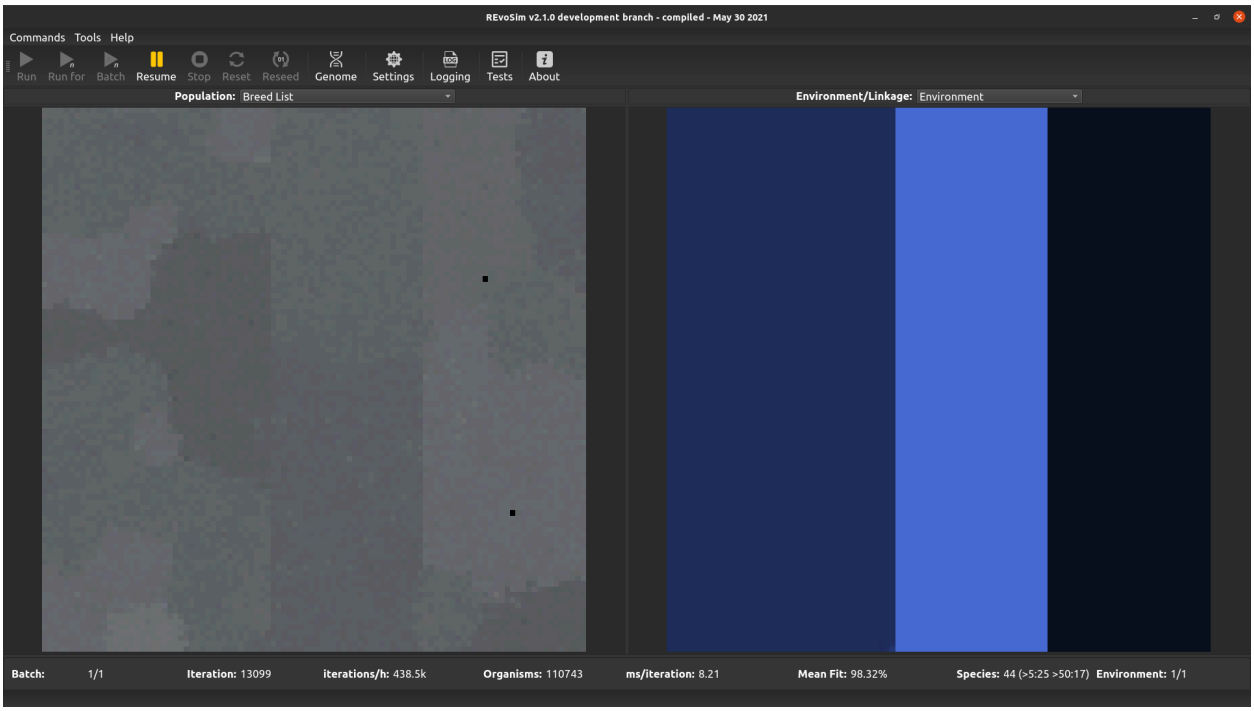
This option visualises the difference between the environmental fitness of the first organism in a grid-square, and the total fitness of that organism including interactions. It thus provides an easy overview of the impact that interactions are having on fitness when interactions are set to modify organism fitness values.

Breed list

This visualises the breed list, and is of the greatest utility when multiple breed lists are enabled (the following calculation is applied even when multiple breed lists are not enabled). When selected, every organism in a cell is inspected to determine the breed list that it is using, and the index of the most frequent breed list is used to determine the colour intensity of the red channel, the second most for green, and the third for blue. As such, cells with organisms placed in just one breed list are pure red; cells with a single but established dominant genome are grey (the 2nd and 3rd most frequent breed list are on either side of the breed list for that dominant genome, due to jitter, and their similar index values will result in similar intensities in all three colour channels); and cells with more than one dominant genome will be coloured (the different breed lists for those genomes result in different intensities in the colour channels, creating non-grey pixels).


REvoSim v2.1.0 development branch - compiled - May 30 2021
Commands Tools Help
Run Run for Batch Resume Stop Reset Reseed Genome Settings Logging Tests About
Population: Species
Environment/Linkage: Environment
Batch: 1/1 Iteration: 13099 iterations/h: 438.5k Organisms: 110743 ms/iteration: 8.21 Mean Fit: 98.32% Species: 44 (>5:25 >50:17) Environment: 1/1


REvoSim v2.1.0 development branch - compiled - May 30 2021
Commands Tools Help
Run Run for Batch Resume Stop Reset Reseed Genome Settings Logging Tests About
Population: Interactions
Environment/Linkage: Environment
Batch: 1/1 Iteration: 13099 iterations/h: 438.5k Organisms: 110743 ms/iteration: 8.21 Mean Fit: 98.32% Species: 44 (>5:25 >50:17) Environment: 1/1



Pathogens word 1

This option allows pathogens to be visualised - it follows the same approach as the visualisation systems, albeit with hard coded pathogen words. If flexibility is required, please request this. This option visualises the first word of the pathogen genome.

Pathogens word 2

This option is the same as the above, but visualises the second word of the pathogen genome.

Stolen energy

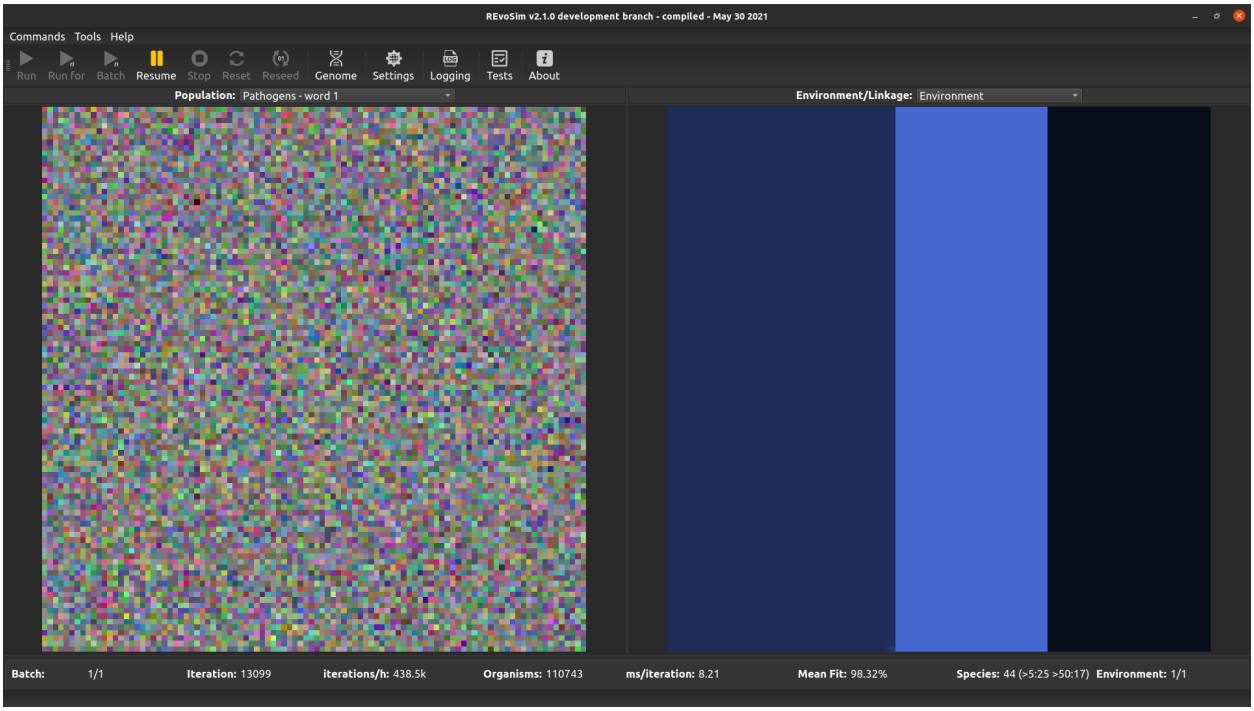
This option visualises the proprtion of the total energy collected over the lifetime of the first organism in each cell that was stolen using the interactions system (if enabled), as opposed to being sourced from the fitness algorithm. White pixels represent organisms that have obtained all of their energy through interactions with other organisms.

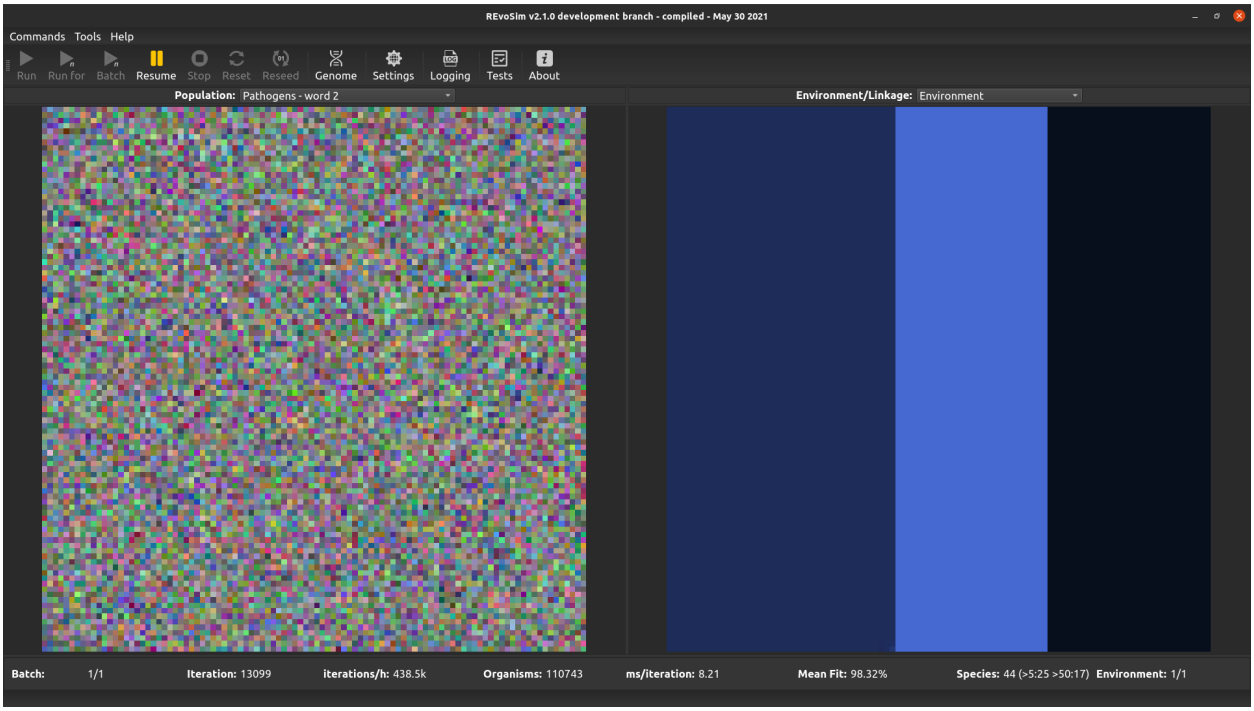
2.4.4 Environment Scene

Every grid-square of the Population Scene possess an 'environment', which consists of three integer parameters in the range 0-255. These are visualized as colour, the three parameters representing red, green and blue values (further details of how this is achieved are available in the REvoSim paper). The environment for the entire grid can thus conveniently be visualised as a raster (bitmapped) 24-bit colour image in the Environmental Scene. Environments can be static (specified by a single raster image), or dynamic (specified by a sequence of raster images).

The default environmental scene consists of a static image with three colour zones. This image can be changed from the Simulation Settings panel.

Dynamic environments can be enabled via the Simulation Settings panel. There are currently three modes of transformation between image sequences, these are:


REvoSim v2.1.0 development branch - compiled - May 30 2021
Commands Tools Help
Run Run for Batch Resume Stop Reset Reseed Genome Settings Logging Tests About
Population: Pathogens - word 1
Environment/Linkage: Environment
Batch: 1/1 Iteration: 13099 iterations/h: 438.5k Organisms: 110743 ms/iteration: 8.21 Mean Fit: 98.32% Species: 44 (>5:25 >50:17) Environment: 1/1


REvoSim v2.1.0 development branch - compiled - May 30 2021
Commands Tools Help
Run Run for Batch Resume Stop Reset Reseed Genome Settings Logging Tests About
Population: Pathogens - word 2
Environment/Linkage: Environment
Batch: 1/1 Iteration: 13099 iterations/h: 438.5k Organisms: 110743 ms/iteration: 8.21 Mean Fit: 98.32% Species: 44 (>5:25 >50:17) Environment: 1/1

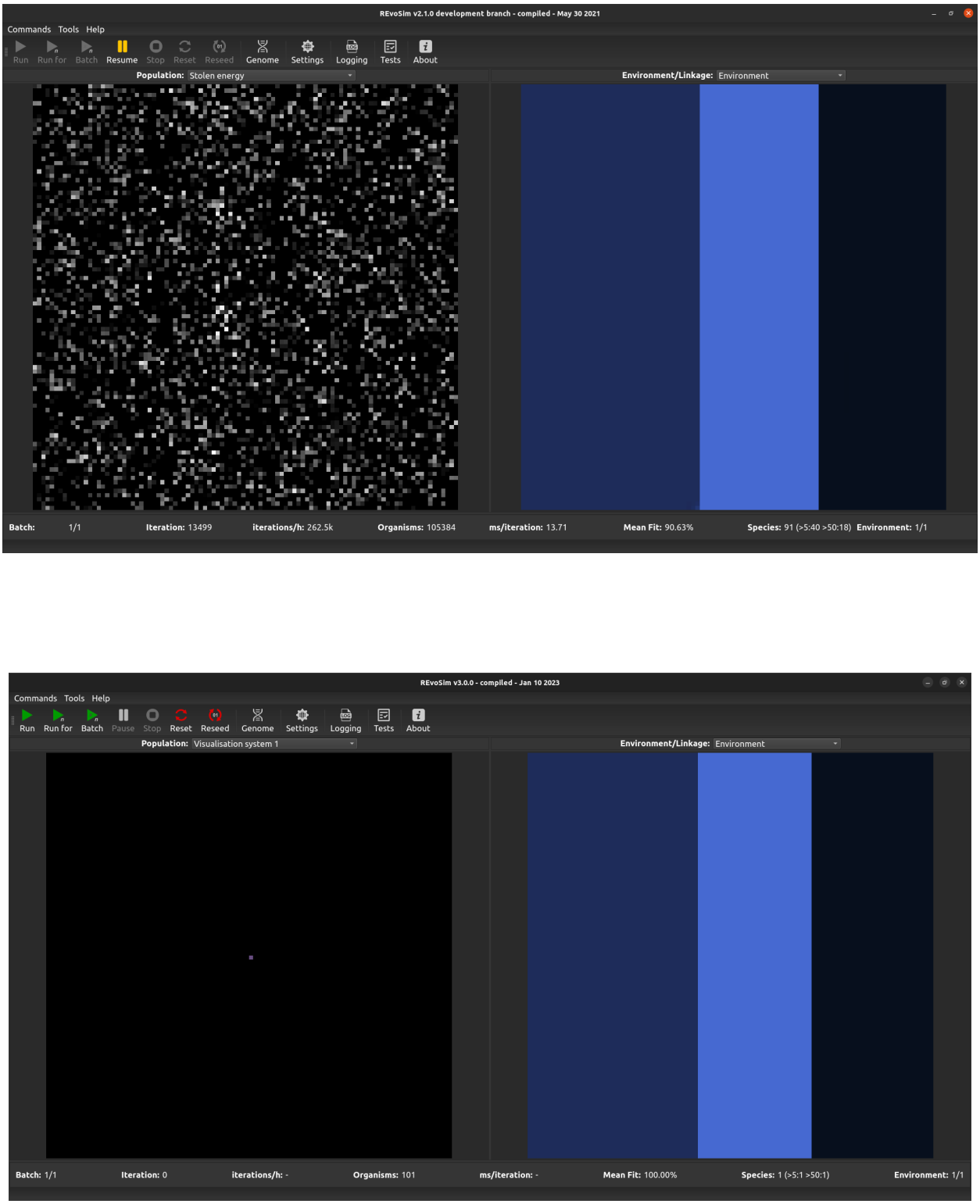


Fig. 7: Default Environmental Scene showing three blocks of colour in vertical strips.

1. Once - each image in the sequence is shown in order but only once
2. Loop - each image in the sequence is shown in order, after displaying the last image in the sequence the environment loops back to the first image (etc.)
3. Bounce - each image is shown in order, after displaying the last image the order is reversed, until the first image is once again shown, the order is then reversed (etc.)

For dynamic environments, the number of simulation iterations between image updates is configurable but by default 100 (i.e. the environment image advances to the next in sequence every 100 iterations). REvoSim can optionally interpolate between environmental images to provide smooth transitions.

2.4.5 Information Bar

At the bottom of the GUI - below the population and environment views - is an information bar, which provides an overview of a number of elements of the current simulation. This is updated every polling generation. The statistics it provides are:

Batch When RevoSim is running in batch mode, this shows how many runs are completed out of the total number requested.

Iteration The number of iterations that had been completed since the start of the simulation at the last polling interval.

Iterations per hour This provides an indication of the speed at which RevoSim is running, and thus it is realtively easy to calculate how long any given run will take.

Organisms This is a count of the total number of digital organisms alive at the last polling iteration.

Milliseconds per iteration This is an alternative measure of speed.

Mean fitness This is the mean fitness of all living organisms in the simulation at the last polling iteration.

Species If species tracking is on, this will provide the number of species at last poll once a speciation event has occurred.

Environment This is the index of the current environmental image, along with the total number that have been loaded. By default REvoSim loads with a single environmental image.

This is printed to the terminal when REvoSim simulations are initiated from the command line in Unix builds of the software (Windows lacks this functionality).

2.5 Configuring your Organisms

At the core of the REvoSim simulation are the digital organisms. A number of properties of these organisms can be defined in the suimulation, in the Organism tab of the Settings dock. These settings are as follows:

2.5.1 Organism settings

Chance of mutation This dictates the chance of a mutation when an organism breeds (if it is set to n, there is a n/256 chance of mutation). The default is 10, which thus equates to a 10/256 chance that a randomly selected bit in the words of the genome to which this is applied (see below) is flipped (from 0 to 1 or 1 to 0).

Variable mutation This uses the number of ones in a user-defined section of the genome to dictate the probability of a mutation ocurring. This uses a cumulative standard normal distribution from -3 to 3, created using the math.h complementary error function and then scaled between zero and the

maximum random number. This allows the probability of mutation occurring to be controlled by generating a random number, and applying a mutation with 99.865% probability if an individual has no ones in a 32-bit section of controlling genome, to 0.24579% with 31 ones. If you are using this you will probably want the mutation log enabled (Tools → Custom Logging → Variable mutation log). This outputs a histogram for the grid showing the number of ones in the non-coding genome (as well as providing an overview of mutation probabilities).

Start age Every iteration each organism loses one from its age counter. This setting dictates the value at which this counter is set when an organism is born. As such, this dictates generation times within the software.

2.5.2 Breed settings

Breed threshold Within a pixel, the energy provided each iteration is split between the digital organisms living within the pixel based on their fitness. A full description of how this is achieved can be found in the REvoSim paper. When a digital organism has stored enough energy to pay the breed cost and still exceed the breed threshold, it attempts to breed.

Breed cost This is the amount of resource that is removed from an organisms when it successfully breeds.

Maximum difference to breed In sexual breeding mode, if two organisms attempting to breed have a hamming distance greater than this value when genomes are compared, breeding fails.

Use max difference to breed Depending on the nature of a study, maximum difference to breed may not be desired. This tickbox dictates whether it is enforced. If unticked, in sexual modes, breeding failure does not occur on the basis of genomic distance.

Breed only within species When this checkbox is ticked, during sexual selections, digital organisms can only breed with other members of the same species. This only occurs if species tracking is turned on.

Multiple breed lists In order to promote the coexistence of different genotypes within cells (by reducing the pressure on organisms to be breed compatible with all other organisms in their cell), this option creates multiple breed lists per cell. Organisms are placed onto one of 66 breed lists based on the remainder when their bit count is divided by 67, accompanied by a level of jitter created through adding a random integer between -1 and 1, to allow mixing between adjacent lists.

Breed mode REvoSim offers multiple breed modes, given the above caveats (e.g. breed within species):

Obligate sexual Organisms can only breed with other individuals (this is the case in most animals, for example).

Facultative sexual Organisms can reproduce with other individuals, or themselves (see e.g. plants). Asexual reproduction is more common in this mode when populations are small.

Asexual mode In asexual mode, self-breeding is enforced and organisms are cloned when they have the required energy reserves to allow breeding.

Variable mode This uses the same approach as variable mutation - the number of ones in a user-defined portion of the genome is used to define the probability of breeding asexually v.s. sexually. It uses the same distribution, and when this is enabled you can select the recombination log enabled (Tools → Custom Logging → Recombination logging). Note that sexual reproduction when using this mode is facultative, so can include self pairing. The recombination log records how often each breed mode is used across all breeding organisms each polling iteration, and then reports this.

2.5.3 Settle settings

Dispersal This figure dictates the extent to which juveniles disperse on settling. Small numbers equate to significant dispersal, larger numbers increase the likelihood that juveniles settle in the same pixel as their parent. How this is achieved is described in full in the REvoSim paper.

Nonspatial settling For some evolutionary phenomena, the impact of space/dispersal may have an unknown impact and not be the element of interest within a simulation, and thus be undesirable. This tickbox allows juveniles to be randomly placed within the simulation (note that with a non-uniform environment, space will still have some impact on the simulation).

2.5.4 Genome Words and Systems

From REvoSim 3.0.0, genomes are no longer limited to 64-bits in length - rather the user can set the genome length of organisms to be between 1 and 32 words, each of which is 32-bits. All options (and associated functions) that can be applied to bits in a genome are called systems, and these systems can be applied to different words of the genome to achieve the desired outcome. Thus, for instance, to match the behaviour of REvoSim v2.0.0, organisms can be set to include a 2 word genome, the fitness system can be applied to word 0, and the breed and species ID systems can be applied to words 0 and 1. Where there is an on/off toggle in the settings for an option, systems are only applied if enabled in that toggle.

Genome Size This dictates the number of genome word for all organisms in the simulation. Its minimum is limited by the words entered in the systems below (i.e. if a system is applied to word number four, you cannot reduce this value below that word).

The options below this allow systems to be applied to words of the genome. Counts use C++ numbering, thus start from zero, and above nine continue from A-V. When a string is entered, the label will turn green if it can be applied with the current settings. If a string requires that the number of words is increased, this is implemented in the genome size option above. If an option is not possible, the label turns red. Some options, when changed, provide further prompts suggesting changes of relevant settings to achieve a viable confiugration. Options dictate the following:

Fitness Which words are used in the REvoSim fitness algorithm.

Breed Words used to calculate breeding compatibility.

Mutate Words to which mutations can be applied.

Variable Mutate Words used to dictate mutation probability.

Pathogens Words used by the pathogen system.

Species ID Which words are used in species searches for reproductively isolated clusters.

Interactions Words use for the interactions systems.

Visualisation 1 Words used for visualisation system 1 (see *Population Scene*).

Visualisation 2 Words used for visualisation system 2.

2.6 Setting up the Simulation

REvoSim provides the user with control of many elements of the simulation. These are introduced herein, and can be modified within the the Simulation tab of the Settings dock, or from the command line.

2.6.1 Environment settings

Change environmental files REvoSim launches with a default environment comprising three stripes in different shades of blue. This can be changed by clicking on this button, which will launch a file explorer. Using this allows a single, or multiple environment images (any standard image format) to be loaded. Environment image files are currently limited to 100 x 100 pixels in size.

Environment refresh rate This dictates how many iterations there are between updates of the environmental images if an image stack has been loaded, and thus dictates the environmental rate of change (small numbers are faster).

Environment mode This dictates, if multiple files are loaded, the mode with which these are updated. Static uses just the first of the chosen images as the environment throughout a run. Once will run from the start of an image stack to the end, leaving the last image as the environment once this has been loaded. Loop returns to the first environment image once the last has been reached (which may result in a significant environmental change in the environment if the first and last image in a stack are different). Bounce will move from the start to end of an environmental image stack and then back again, repeating this for the duration of a run.

Interpolate between images This provides linear interpolation of the environment between refresh iterations, preventing large stepped changes in organism fitness.

Toroidal environment By default, when a juvenile settles outside the limits of an environment, it dies. Such boundary effects can be avoided by selecting toroidal environment, which wraps around in both directions (e.g. juveniles settling off the left enter the simulation on the right; EnviroGen can simulate toroidal environments).

2.6.2 Simulation size

Grid X This dictates the size of the simulation grid in the X direction. If this is smaller than the environmental image it selects the left side of the image.

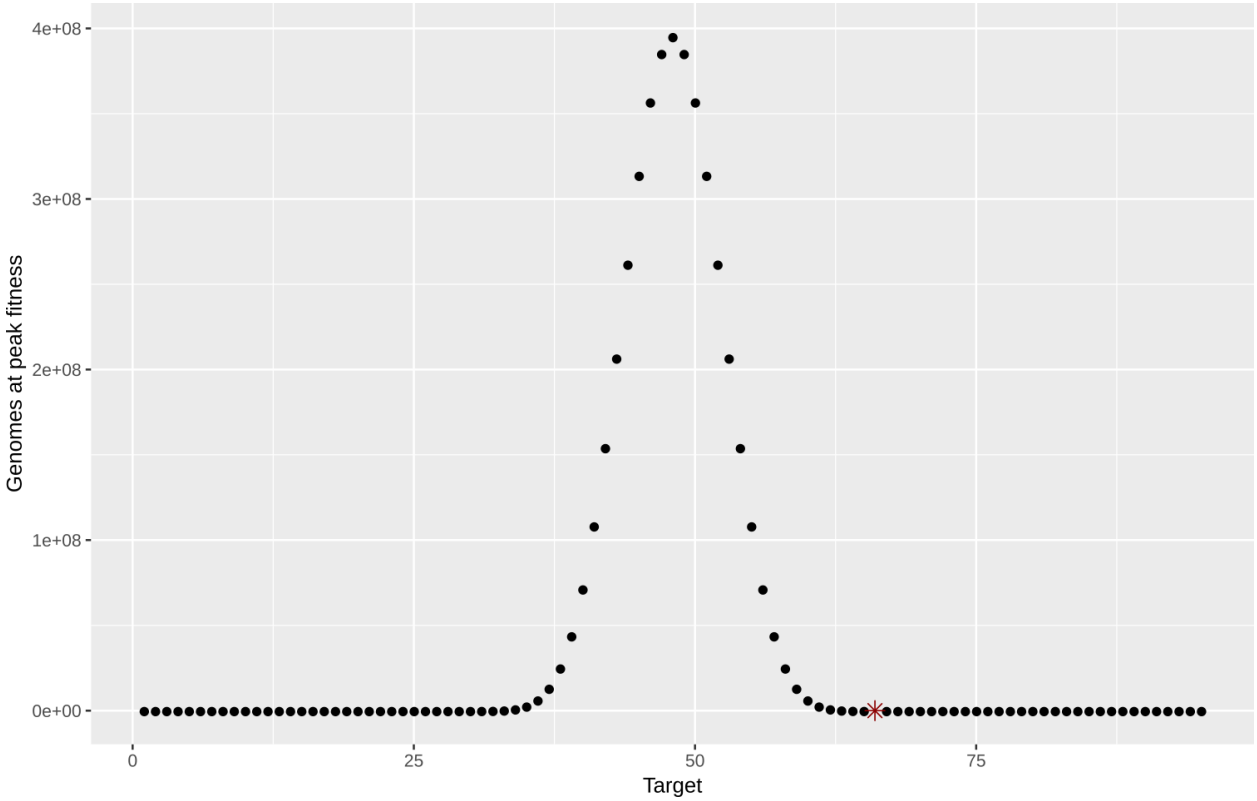
Grid Y This dictates the size of the simulation grid in the Y direction. If that is smaller than the environmental image it selects the top of the image.

Slots Every pixel in the simulation grid can have multiple digital organisms - each lives within a slot (this is equivalent to a Z axis). This dictates the number of slots (how these are filled during reproduction is outlined in the 2019 REvoSim paper).

2.6.3 Simulation settings

Fitness target This is the target value used by the fitness algorithm as described in the REvoSim paper. In brief, it is the value that the hamming distance between the gemome words in the fitness system and the environmental masks (numbers dictated by the colour) is optimised towards. Changing it modifies the fitness landscape: a value of 48 - with one genome word - has the greatest number of genomes with peak fitness, and either side of this value, fewer genomes display peak fitness. This is shown in the graph below, where black points represent the number of genomes with peak fitness for all fitness targets. The red star is the default value for one word, 66; in this particular set of runs, there there are ~3000 genomes capable of peak fitness with the default target, but this can vary given differing masks. Note that there are some values for which no organisms will be capable of survival: if this occurs, REvoSim will provide a warning. Also note that the graph below can be created and distribution of fitnesses quantified using the count peaks option in the commands menu of REvoSim. When changing the number of genome words used by the fitness system from the organism tab, REvoSim will provide the option of updating the fitness target to a commensurate value for the

new number of words. If you would like to further discuss the nature of fitness landscapes within REvoSim please contact the authors, as this is something we have thought about a lot.



Energy input This is the amount of energy provided per pixel of the environment grid, which is then split between the digital organisms living in that pixel on the basis of their fitness. This is then used as part of the breeding system in REvoSim.

Settle tolerance This it controls the distance from the fitness target at which organisms are no longer viable (i.e. have zero fitness and, absent energy-based interactions, die). It also provides the maximum attainable fitness within the simulation.

Recalculate fitness For efficiency REvoSim only calculates organism fitness for any individual on initial settling: subsequent changes in the environment will not modify an individual's fitness. For some settings (e.g. rapidly changing environments or long-lived organisms) this may not be desirable. When checked, this option recalculates the fitness for every organism in the simulation every iteration, overcoming this limitation but also significantly slowing the simulation.

No selection This option turns off the fitness algorithm so energy is split equally between organisms in any cell, irrespective of their fitness.

2.6.4 Phylogeny settings

These radio buttons dictate the mode which by REvoSim tracks phylogeny.

Off Does not track phylogenies and is thus the fastest mode (this could be useful for - as an example - studies focussing on changes in fitness).

Basic phylogeny Identifies species in time slices to allow species to be coloured in the population view, and species diversity to be recorded.

Phylogeny Identifies species, and then records their phylogeny, allowing a tree to be created at the end of a run.

Phylogeny and metrics Does the same as *Phylogeny*, and also records a number of other metrics for each species, also output (when requested) at the end of a run.

Note: Moving between off and any form of tracking has a significant performance cost: there is little computational overhead moving between the different tracking options. Moving from basic to phylogeny to metrics does, however, come with an increasing memory overhead, as the trees and metrics are by necessity stored in RAM during a run, and written when the run completes. This could have implications for runs with a significant number of organisms and large numbers of iterations. See *Running Log* and *Configuring Outputs and Run End Log* for more details REvoSim outputs.

2.6.5 Linkages

This is a capability introduced in v3.0.0 that links REvoSim variables to image maps (or each other, a functionality that has yet to be added, but is easily implemented on request). As default, REvoSim offers limited linked variables, however the system is designed so more can be added with ease. Please contact the authors if you would like one to be added - we will happily do so.

As currently coded, the image stack selected is used to spatially control a variable: specifically, for integer variables, the value of the R channel is multiplied by the value of the selected variable as defined in the settings docker. So, for example, if energy is set to 5 overall, for a pixel with an R value of 0 in the image map the energy will be set to 0 for that iteration. If the R value is set to 255, the energy level for that pixel will be 1275 (5x225). *Note that this means that when using a linked integer variable it is likely that the default values will need to be changed.* For boolean variables, an R channel value of zero is off, all other zero values (1-255) is on.

Create new linkage

To create a new linkage, use the push button labelled "Link a variable to an image mask". This creates a pop up window with the following options:

Variable This is the variable to which you would like to control using an image stack.

Load Images Clicking this image loads a dialogue allowing you to load an image stack that controls the variable defined above, as described in the introduction to linkages.

Mode This radio button defines the image mode for the stack controlling the variable. The modes are the same as used for the environment - see *Environment settings*.

Interpolate images As with the environment settings, this defines whether REvoSim interpolates between images.

Refresh rate The update rate of the image stacks controlling variables does not need to be the same as e.g. the environmental refresh rate: this spin box defines how often (in iterations) the next image in the stack is loaded.

Edit linkage

Once added, a linkage appears as a list item in the *Current linkages* text box on the bottom right of the simulation docker. To edit one double click on the linkage. Doing so will load a pre-filled dialogue with the same options as the add new linkage dialogue outlined above, all of which can be changed as required. The linkage can also be deleted using this dialogue box.

2.7 Configuring Interactions

2.7.1 Interaction settings

Interactions were added to REvoSim for the release of v3.0.0. These are tools that allow individuals (or segments of individuals' genomes) in REvoSim simulations to interact with other individuals in the same cell. This occurs when offspring are settling, or every iteration, as described below. Interactions can increase or decrease the fitness of an individual, or its energy. In general, interaction systems compare the genomes (or parts thereof) of two individuals - individual 1, to which any fitness modification is applied, and individual 2. When interactions are applied, individual 1 typically interacts with multiple other individuals (i.e. the program can loop through a random selection of individual 2s), and this process is repeated for every individual in a cell (except in instances where interactions are only applied on settling, i.e. fitness interactions when recalculate fitness is not selected).

Genome blocks The genomes of both individuals are split into two-bit blocks: each two bits can be a number from 0-3 (i.e. 00,01,10 or 11). For every two bit block, the numbers in each genome are used to look up a value from the a-priori interaction table (this is a 4x4 grid of user-set interaction numbers that can be modified by clicking *Edit the interacions grid*). The values are summed across all two bit blocks of the genome. For fitness interactions, this total is added to the fitness of organism 1. See below for energy interactions.

Genome XOR Here a bit shift is applied to the genome of individual 1, which is then compared to individual 2. More specifically, the bits in the genome of individual 1 are shifted one to the right, and the results of an exclusive or between the resulting binary string and the genome of individual two are subject to a bit count, providing a number. The process is then repeated with a bit shift in the other direction for individual 1, to provide a second number. For fitness interactions, this number is subtracted from the first, and the resulting number is used to modify the fitness of individual 1. See below for energy interactions.

Interactions This sets the number of interactions attempted by each organism. In fitness interactions this occurs when fitness is calculated - this means that, by default, this only occurs during settling, unless the simulation is set to recalculate fitness every iteration. When energy interactions are selected this is is the number of interactions that each individual attempts every iteration.

Interactions Change Fitness Interactions modify the fitness of organisms (see above for a note on when this is applied). Note that partners for each are chosen at random, and if an empty slot is selected, no interaction is performed. The total change to the fitness of an individual is the sum of all successful interactions.

Interactions Change Energy In energy interactions the absolute value of either the interactions lookup (genome blocks) or the difference between bit shifts (genome XOR) is subtracted from a predation target, currently hard coded as 12 * the number of genome words in use in the system. The absolute value of the result is then subtracted from the predation target, and that number n is used to determine what proportion of organism 2's energy is lost by that organism. n/(predation target) of organism 2's energy is lost.

Interactions Do Nothing Do not apply interactions.

Edit the interactions grid Edit the 4x4 grid of user-set interaction numbers.

Min. Predator Score (Energy Only) It may occasionally be useful to prevent interactions from resulting in any transfer of energy between organisms unless the predator organism is sufficiently well adapted for predation. In this case, this setting can be used to prevent organisms from obtaining energy through interactions when their predation score is below some threshold.

Predation Efficiency (Energy Only) When interactions result in the direct transfer of energy from one organism to another, only a fraction of the energy lost by organism 2 is gained by organism 1 (as

is realistic). This fraction is 10 * the value for this setting (e.g. a value of 3 would represent 30% energy transfer efficiency).

Restrict interactions This setting only functions when interactions change energy, and after 100 iterations to allow trophic levels to become established. After this point, it prevents organisms from gaining energy through interactions unless their target organism has a current trophic level (prior to the interaction) at least 0.5 lower than their own trophic level. It thus prevents primary consumers from acting as facultative secondary consumers in addition to being primary consumers. Trophic level is defined over the lifetime of an organism, based on an average of the sources from which that organism gains its energy, weighted by the amount of energy gained from each source.

2.7.2 Pathogen settings

REvoSim v3.0.0 also includes a newly added pathogens system. Given the abstract nature of the software, these could equally be considered equivalent to predators: in a basic sense they act as antagonists. At present there are two options, drift and evolve:

Drift With drifting pathogens, there is one pathogen per cell. Each has pathogen comprises a genome of n words. Pathogens mutate with a user-defined probability/generation, and are applied with a user-defined frequency. This system works by performing an exclusive or between the pathogen and each living creature, and then summing the ones. It then uses this sum to define a probability of that individual dying as a result: when the result of the XOR is zero (i.e. pathogen and organism genome are identical), the probability of the organism dying from the interaction is 0.5. Where the result of the XOR is maximised, this is 0. There is a linear relationship between these extremes (other probability distributions can be provided on request). An organism is killed by setting its age to 1, resulting in its removal in the next iteration.

Evolve Pathogens evolve for virulence when this setting is selected. With this setting, there are 5 pathogens per cell, whose fitness is defined by how many organisms they kill in an iteration. Every iteration, the pathogens for a given cell, and those orthogonal to it (i.e. a 3 x 3 square of pixels, minus the diagonals), are applied to the individuals in a given cell. The pathogen that kills the most organisms living in that cell (or the first one to reach this number if there is more than one) is applied to the organisms in the cell. It is then replicated, mutated following the pathogen mutate probability (currently this is set to n/256 chance of mutation irrespective of number of words the system applies to), and then placed back in the pathogens layer overwriting a previous pathogen at random prior to the for next pathogen iteration. Including the cells orthogonal to that which pathogens are being applied allows the pathogens to move across generations, and thus follow e.g. species and environments.

For both forms of pathogen, the follow options are available:

Pathogen mutation This defines the probability of a mutation occurring any iteration that pathogens are applied. Smaller numbers define a smaller probability. This works by generating an 8bit random number (0-255), and applying a mutation if that number is *less than* the integer in the checkbox so, for example, if this is 1, the probability of a mutation occurring is 1/256.

Pathogen frequency This is the frequency, in iterations, with which pathogens are applied.

There are a great many other mechanisms by which antagonists could operate, and if you are interested in modifications, please contact the authors.

2.8 Configuring Outputs and Run End Log

From v3.0.0, REvoSim has a versatile logging system, that can be accessed by clicking the logging button on the main toolbar. This launches an Output Dock, which appears in the left dock area and has two tabs, placed at the bottom. The

Output settings tab has a number of options that dictate the behaviour of the REvoSim logging system, and buttons to output two predefined logs at any point in a run. In addition to these, the Running log tab provides a flexible system for writing logs as a simulation run progresses.

2.8.1 Output options

Options in the Output settings tab are as follows:

Output save path This is the folder into which all outputs from REvoSim (logs, images, any other files) are saved. For consistency these are all placed within a newly created folder called *REvoSim_output*. Text files are placed within the root of this folder, and images are placed in their own folder within *REvoSim_output*.

Refresh/polling rate This is the frequency (in iterations) that the simulation is polled. Polling includes running the full clustering analysis intergral to the thorough species-identification algorithm (see RevoSim 2019 paper), writing the in-simulation log, writing any requested image files to disk, and updating the GUI (unless disabled). Because the species identification system has a significant computational overhead, if species tracking is on, frequent polling will significantly slow the simulation.

Image logging Any of the simulation visualisations during a run can be saved as an image (png stacks labelled by iteration number). The tick boxes in this part of the tab dictate which images are saved. See *Population Scene* for full descriptions of each option.

2.8.2 Run end log

REvoSim provides a detailed log file at the end of runs - or on request during a run - that features the information below. This can be automatically output at the end of every run in batch mode (see option *Automatically create detailed log on batch runs* below), or dumped as required using the *Write data for current run* button highlighted below. The log is written to the current output folder, and is placed in a file called REvoSim_end_run_log.txt. This file is structured as follows:

Timestamp The first line is a time stamp highlighting when the run was written, in the following format: 2018-12-30T11:57:51

Settings A printout of all REvoSim settings for this run then follows, divided into integers and then bools. This means that at any point it is possible to revisit and check all settings for that run.

Legend There is then an explanation of the contents of this log file: "This log features the tree from a finished run, in Newick format, and then data for all the species that have existed with more individuals than minimum species size. The exact data provided depends on the phylogeny tracking mode selected in the GUI."

Tree There then follows a tree in Newick format for the run to the point at which the log is written, excluding species below minimum species as requested, and also excluding species without descendents if requested. This can then be loaded into, e.g. FigTree to be rendered, or into e.g. R, for analysis. An example tree is shown below - note species labels are prefaced with id for clarity, and also include the maximum size of that species as part of their species name, after a hyphen:

```
((((((((((((((((((((((((((id27-81050:50,id28-2:50,id29-1:50,id30-1:50,id31-1:50,id32-
↪5:50,id33-2:50,id34-8:50,id35-1:50)id26-81050:50,id36-3:50,id37-2:50,id38-4:50,id39-
↪3:50,id40-2:50,id41-3:100,id42-4:100,id43-5:100)id25-81050:50,id44-17:150,id45-
↪4:100,(id47-23311:100,id48-2:50)id46-23311:50,id49-2:50,id50-2:100)id24-81050:50,
↪id51-4:50,id52-1:50,id53-5:100,id54-2:50,id55-10:100,id56-11:50,id57-61:200,id58-
↪49:200)id23-81050:50,id59-2:100,id60-2:50,id61-1:50,id62-4:50)id22-81050:50,id63-
↪13:250,id64-2:50,id65-8:50,id66-1:50,id67-1:50,id68-4:50,((id71-24648:50,id72-
↪1:50)id70-24648:50,id73-2:50,id74-3:100)id69-24648:200)id21-81050:50,id75-3:50,id76-
↪14:150,id77-3:50)id20-81050:50,id78-2:150,id79-8:50)id19-81050:50,id80-2:50)id18-
↪81050:50,id81-3:50,id82-2:50,id83-1:50)id17-81050:50,id84-1:50,id85-7:50,id86-2:50,
↪id87-9:150)id16-81050:51,id88-1:1)id15-81050:18,id89-18:69)id14-81050:16,id90-
↪8:50,id91-2:50)id13-81050:8,id92-2:42)id11-81050:3,id93-2:16)id10-
↪81050:14,id94-1:13)id9-81050:8,id95-2:11)id8-81050:1,id96-1:9)id7-81050:14,id97-
↪1:12)id6-81050:22,id98-1:12)id5-81050:87,id99-2:73)id4-81050:2,id100-1:12)id3-
↪81050:4,id101-2:141)id2-81050:64,id102-1:10)id1-81050:11,id103-3:82)id0-81050:237
```

(continues on next page)

(continued from previous page)

Log - detailed species data The rest of the log file comprises detailed data for each species in csv format, in the same order they appear in the tree. For each species, for every polling iteration, REvoSim provides the following data:

- Species ID
- Species ID of Parent
- Iteration number (i.e. the polling iteration for which this was recorded)
- Number of individuals at polling iteration (size)
- A sample genome for the species, selected as for running log (i.e. randomly), presented as a 32-bit number
- The above genome as a binary string
- Genomic diversity - i.e. the number of different genomes in the species
- The number of pixels occupied by this species, subtracted 1 (i.e. real range is 1-65536, but -1 allows C++ style numbering: 0-65535)
- The geographic range in the form of the maximum distance between outliers
- The centroid of range in X - the mean of all X positions
- The centroid of range in Y - the mean of all Y positions
- Mean fitness of all members of species, stored multiplied by 1000 to allow small changes to be easily identified
- Minimum R, G, then B - the log then reports the minimum R, G and B values the species is found in
- Maximum R, G, then B - as above, but maximum values
- Mean R, G, then B - the final three numbers are the mean R, G, and B values the species inhabits

An example log thus appears:

```
id,ParentID,iteration,size,sampleGenome,sampleGenome_binary,diversity,cellsOccupied,
↪geog_range,centroid_x,centroid_y,mean_fit,min_env_red,min_env_green,min_env_blue,
↪max_env_red,max_env_green,max_env_blue,mean_env_red,mean_env_green,mean_env_blue
27,26,1073,34539,17476623570733825285,
↪11110010100010010111000010001010110111101100111011111100100000101,6780,4199,41,20,49,
↪9566,30,41,88,35,54,112,30,44,90
28,26,1073,1,18017055526017752864,
↪11111010000010010111000010001010110111101100011011011111001000000,1,1,0,29,44,10000,
↪30,44,90,30,44,90,30,44,90
29,26,1073,1,17441298461089501447,
↪11110010000010111111000010001010110110101100111010010001000000111,1,1,0,31,79,10000,
↪30,44,90,30,44,90,30,44,90
30,26,1073,1,17312242184062138796,
↪11110000010000010111000010001010110011101000111001111001101011100,1,1,0,5,6,8000,30,
↪44,90,30,44,90,30,44,90
31,26,1073,1,17726573350043672487,
↪11110110000000010111000010001010110111100100001001011111101001111,1,1,0,41,22,9000,
↪30,44,90,30,44,90,30,44,90
32,26,1073,5,18021559125636701700,
↪11111010000110010111000010001010110111100100011001011110000000100,1,4,2,3,83,9000,30,
↪44,90,30,44,90,30,44,90
33,26,1073,2,17439188498342378892,
↪11110010000001000111000110001010110111101100011001111101100001100,1,2,0,3,9,9000,30,
↪44,90,30,44,90,30,44,90
```

(continues on next page)

(continued from previous page)

```
34,26,1073,3,17440594842165369120,
↪1111001000001001011100001001101011001110110001101101100100100000,1,2,1,36,49,9000,
↪30,44,90,30,44,90,30,44,90
35,26,1073,1,16358041978649091335,
↪1110001100000011011100001000001011001110010011101001110100000111,1,1,0,36,97,9000,
↪30,44,90,30,44,90,30,44,90
26,25,1023,34348,17476623570733825285,
↪1111001010001001011100001000101011011110110011101111110100000101,6582,4201,51,20,49,
↪9357,30,41,88,70,105,209,30,44,90
...
```

2.8.3 Run end log options

REvoSim provides the following options for the run end log:

Automatically create detailed log on batch runs This option outputs the detailed log at the end of each run when REvoSim is operating in batch mode.

Write data for current run This option outputs the detailed log for the currently running simulation at the point at which the button is pressed.

Exclude species without descendents Under most settings a significant number of small, short-lived species may appear regularly within a REvoSim run. Given the significant amount of data REvoSim can generate, and the fact that these short lived species will be unimportant for many studies (potentially masking important obvservations), this option rationalises REvoSim detailed logs by only including species with descendents in the end run log and tree.

Minimum species size It is also possible to filter the species data in the log files so that only species above a certain number of individuals are included in the logs. This spin box dictates what that minimum cut-off is.

2.8.4 Other options

Don't update GUI This option allows runs to proceed without updating the GUI (although note that this prevents images being saved during a run). Checking this allow REvoSim to run marginally faster, and may be of utility for very long runs.

2.8.5 Custom logs

In addition to the Run end log, and running log (see *Running Log*), there are a series of custom logs that can be selected in the tools option of the main menu (see *Main Menu*).

2.9 Running Log

In addition to the Run end log and image logging (see *Configuring Outputs and Run End Log*), plus the custom logs (see *Main Menu*), REvoSim has a versatile system to write customised logs at every polling iteration during a run. When enabled, this writes a file called REvoSim_log.txt to the output folder, then updates this by appending more data every polling iteration, giving an overview of the current run. The v3.0.0 logging systems functions by outputting text that has been entered in this dock (including line breaks). When doing so, it replaces key words/phrases surrounded by *stars* with values that document the state of the simulation. The three text areas/boxes in this dock allow different kinds of text for logs to be entered, as outlined below.

2.9.1 Buttons

Write to file When this option is checked a log file is written during the course of every run. See *Running Log* for more details of REvoSim logs.

Instructions This button creates a pop up window with simplified instructions for writing the running log.

Validate logs This button will highlight, in red, any star-bounded *key words* that are not recognised by the logging system.

Command line log file This option outputs an XML file for the current log settings that can be loaded from the command line for batch runs.

v2.0.0 log This populates the text areas with the default running log text from v2.0.0.

v2.0.0 CSV log This populates the text areas with the default running log text from v2.0.0, formatted as a CSV.

2.9.2 Header text

This text area can be used for text that should appear at the start of a log file, but is only written when the file is created. This could include the settings for any given run, comments relevant to the experiment, or a header for the remaining columns, for example.

2.9.3 Iteration text

This text is written to the log file every polling iteration. This is likely to include statistics about the grid as a whole, for example, number of living organisms, iteration number, or mean fitness. A full list of options is provided below.

Write header from iteration log This writes a header in the header text area based on the iteration text. Unrecognised keywords will be coloured in red.

2.9.4 Species text

This text is written to the log file for each species, for every polling iteration. This data might include species ID, its origin time, the ID of its parents, and the number of organisms in that species. A full list of options is provided below.

Write header from species log This writes a header in the header text area based on the species text. Unrecognised keywords will be coloured in red.

2.9.5 v3.0.0 log options

In v3.0.0 and later, custom logs can be created by entering outputs in the text areas as described above, including *keywords*. These keywords are replaced with outputs for further analysis when the log is written (e.g. *iteration* is replaced with the current iteration number).

Keywords are show below, first per iteration keywords, then per species keywords (although some can be used in either context). The escape sequence for a star (*) is two stars (**).

Iteration keywords

dumpGenomes This writes the genomes of every living digital organism, and the X then Y coordinate in which they are found, separated by commas.

gridBreedEntries The number of organisms in the grid which are attempting to breed at the polling iteration.

gridBreedFails The number of failed breeding attempts in the polling iteration for all organisms in the grid.

gridBreedSuccess The number of successful breeds in the polling iteration for all organisms in the grid.

gridMeanFitness The mean fitness of all the organisms in the grid at polling iteration.

gridNumberAlive The number of organisms alive at polling iteration.

gridGeneration This calculates the average age (thus generation time) of all successful parents in the polling iteration. Note that it calculates this based on the assumption (which is true for the majority of settings), that organisms will only successfully breed once in their lifetime. This assumption can be checked using the *gridNumberAlive* and *gridBreedSuccess* outputs to calculate the mean number of breeding individuals at any generation.

gridTrophicHistograms This outputs a histogram of the trophic levels of all the organisms in the grid, at 0.1 intervals between 0 and 3.

gridSpeciesRange This reports the average range of the species alive at the polling iteration: it is a mean of the number of pixels each species is found in, across all species.

iteration The current iteration.

printSettings This prints a string of all REvoSim settings at the polling iteration, or start of the run if placed in the header.

printTime A string showing the time.

speciesCount The number of species alive at polling iteration.

Species keywords

Ca This is the sum of the gene frequency differences from the origin of a species for the word(s) included in the fitness algorithm.

completeSpeciesData This writes the complete species data for any given species, of the form described for the end run log (see *Configuring Outputs and Run End Log*).

Cr This is the sum of the gene frequency differences from the last polling iteration for the word(s) included in the fitness algorithm.

currentGeneFrequencies This writes the mean number of on bits for every position of the genome across a species.

originTime This is the polling iteration at which the species was first identified as reproductively isolated (see 2019 paper for a description of the species algorithm).

originalGeneFrequencies This writes the mean number of on bits for every position of the genome across a species at its origination.

NCa This is the sum of the gene frequency differences from the origin of a species for the word(s) *not* included in the fitness algorithm.

NCr This is the sum of the gene frequency differences from the last polling iteration for the word(s) *not* included in the fitness algorithm.

speciesGenomeDiversity This is the number of distinct genomes included within a species.

speciesID REvoSim's ID for a species, which is useful for correlating statistics with the tree output by the software.

speciesMeanEnvironmentalFitness The mean fitness of the species from the environmental fitness algorithm.

speciesMeanFitness The mean fitness of the species from the environmental fitness allgorithm plus any interactions that impact on fitness.

speciesMeanRunningEnergy This is the mean total lifetime energy of the organisms within a species (see *Interactions*).

speciesMeanRunningStolenEnergy This is the mean total lifetime stolen energy of the organisms within a species (see *Interactions*).

speciesModalGenome This outputs the modal genome of the species.

speciesParent This outputs the REvoSim species ID for the parent species.

speciesSize The number of individuals within the species.

speciesTrophicLevel The mean trophic level of the organisms in a species.

2.9.6 v2.0.0 log

The v2.0.0 log is structured as follows:

Timestamp The first line is a time stamp highlighting when the run was written, in the following format: 2018-12-30T11:57:51

Settings A printout of all REvoSim settings for this run then follows, divided into integers and then bools. This means that at any point it is possible to revisit and check all settings for that run.

Legend There is then an explanation of the structure of the log files. Every iteration, the log records data about the simulation to file in a format designed to be easy to parse into a range of analytical environments (e.g. R, Python). This structure is as follows for each iteration:

```
- [I] Iteration Number
- [P] Population Grid Data:
  - Number of living digital organisms
  - Mean fitness of living digital organisms
  - Number of entries on the breed list
  - Number of failed breed attempts
  - Number of species
  - Trophic histograms
- [S] Species Data:
  - Species ID
  - Species origin (iterations)
  - Species parent
  - Species current size (number of individuals)
  - Species mean Environmental Fitness
  - Species current genome (the modal genome of the species, the genome that occurs␣
↪most frequently)
  - Species trophic level (the mean trophic level of individuals in the species)
  - Species genome diversity
```

Log data The log then begins. Iterations are separated by new line breaks. Every iteration has a single [I] line, one [P] line, and then an [S] line for every species above the minimum species size. We note that it does not exlude species without descendents because it is written during the simulation, appending to the file for speed. To filter out those species without descendents would introduce the need to store and then regularly filter the log data, and thus would come with a notable computational overhead.

CSV format If the 'Log file formatted as CSV' option is checked in output settings, the log file has a different and simpler format using the 'comma separated value' system. This may be easier to parse in some software (e.g. spreadsheets). A single header row is generated at the start of the file, providing titles for columms. Subsequent rows are generated for each [S] record described above, but these also include columns with the [I] and [P] records in each row. All fields described above are included in the output.

This logging system is designed to allow as many potential elements of a RevoSim run to be quantified as possible. Should any further measures or statistics be required, please file a feature request.

2.10 Genome comparison dock

The Genome Comparison Dock allows genomes to be inspected and compared within RevoSim, and can be launched from the main menu or the toolbar. By default it appears landscape on the bottom of the main window. Its landscape orientation is intended to allow whole genomes to be viewed without scrolling on most operating systems, however we note that on operating systems with low resolution displays, the REvoSim GUI may struggle to accommodate the docker in its default position: the environment and population views may be forced to be small, or the genome comparison docker may not have the vertical extent to show more than one genome. If this is the case, the docker can be floated, or alternatively docked to the right or left of the main window.

A genome can be added to the comparison dock by right clicking on the population scene. By default, this will add the genomes of all of the digital organisms living within that pixel to the comparison dock, rendering them as zeros and ones, as well as displaying, from left to right in the row, the colour of that organism's environment, that organism's species ID, the colour assigned to that species ID, that organism's environmental fitness (EF) in its environment, that organism's current fitness (F) (after interactions), the lifetime total energy accumulated by that organism (Le), and the total energy acquired by that organism through direct energy theft in interactions (Se). A colour is also displayed for each genome word, to highlight differences between genome words among organisms. Alternatively, as shown in the video above, when Modal genome only is selected on the right of the dock, this adds just the model genome for any clicked pixel. By default, auto compare is also enabled: this compares and highlights the differences between each entry on the genome list and that before it.

The genomes can be deleted individually from the table by selecting a genome and using the Delete button. Alternatively all genomes in the table can be removed from Genome Comparison Dock the using the Reset All button.

The first ten genomes from the Genome Comparison Dock are pulled through to the reseed dialogue (shown in figure below): this allows one to be selected as the starting genome for a new run. This allows multiple repeats of simulations using the same genome, which is known to be capable of surviving within a given environment using the masks loaded at launch (note that these will not persist between sessions unless through loading a saved simulation).

2.11 Advanced Options

REvoSim has the following options available for experienced users.

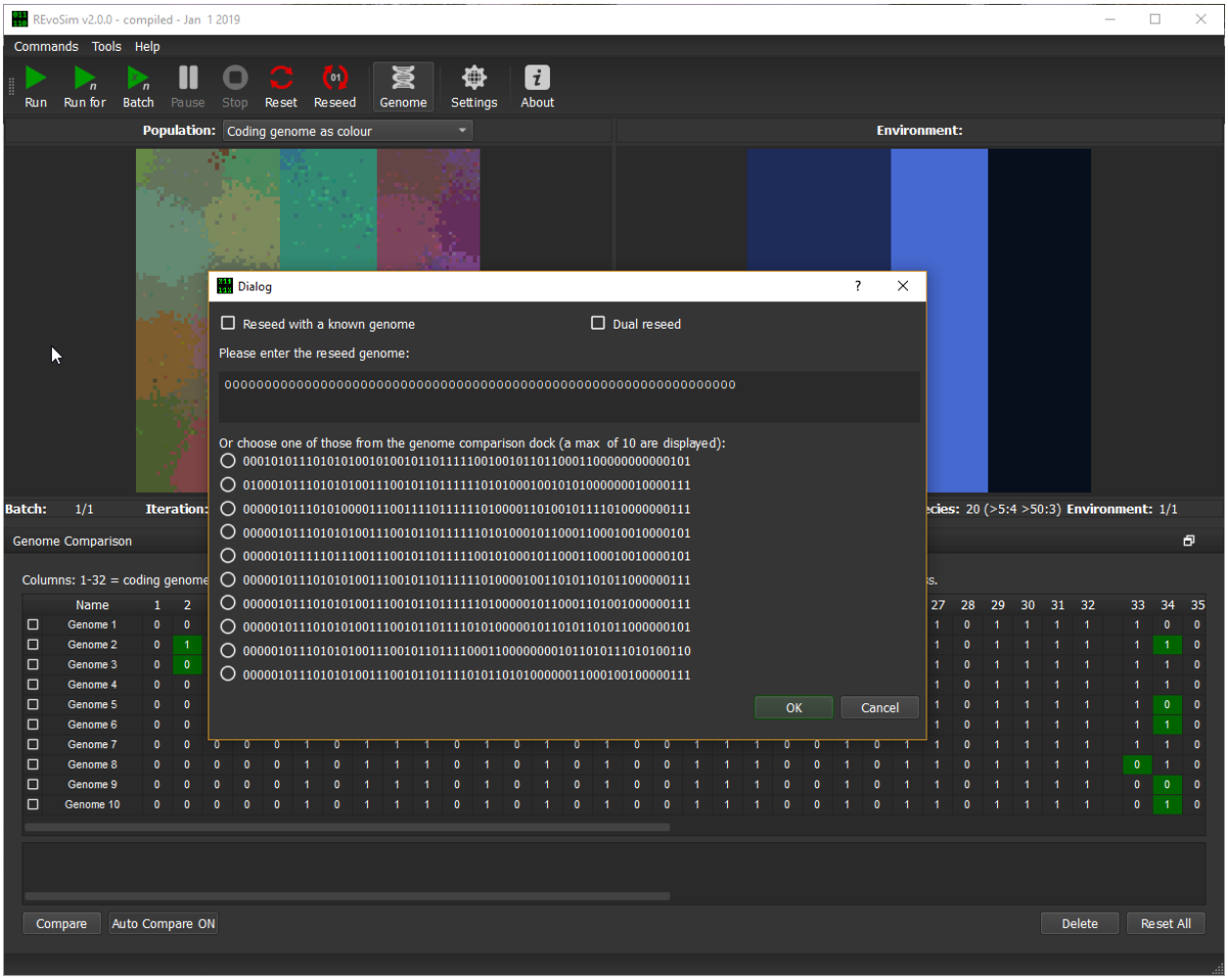


Fig. 8: Reseed dialog showing first ten genomes as selected in the Genome Comparison Dock (created with v2.0.0).

2.11.1 Count peaks

A key element of REvoSim is its fitness algorithm, described in depth in the REvoSim paper. The countpeaks command in the REvoSim main menu is included to provide a simple quantification of the fitness landscape given the current masks. These change when the software is restarted, and thus in any given run there will be shifts in the fitness lanscape.

To provide quantification, the count peaks command cycles through every possible 32-bit number, calculating its fitness for the current masks, selected fitness target, and user-defined RGB values (these are requested from the user on initiation of the count peaks command). When this is complete, the software outputs these in a text file written to the output folder (as defined in the Output tab of the Settings dock) that lists fitness *vs* genome count. An example is shown below.

```
REvoSim Peak Counting 2018-12-17T16:56:57
==================

Below is a histogram showing the different fitnesses for all potential 32-bit␣
↪organisms in REvoSim under the user-defined RGB levels.

==================

Fitness counts for red=128, green=128, blue=128

0,3400873943
1,261711486
2,206660322
3,154146136
4,108257580
5,71295660
6,43807340
7,24949080
8,13052205
9,6196135
10,2627235
11,976050
12,310646
13,82512
14,18036
15,2930
```

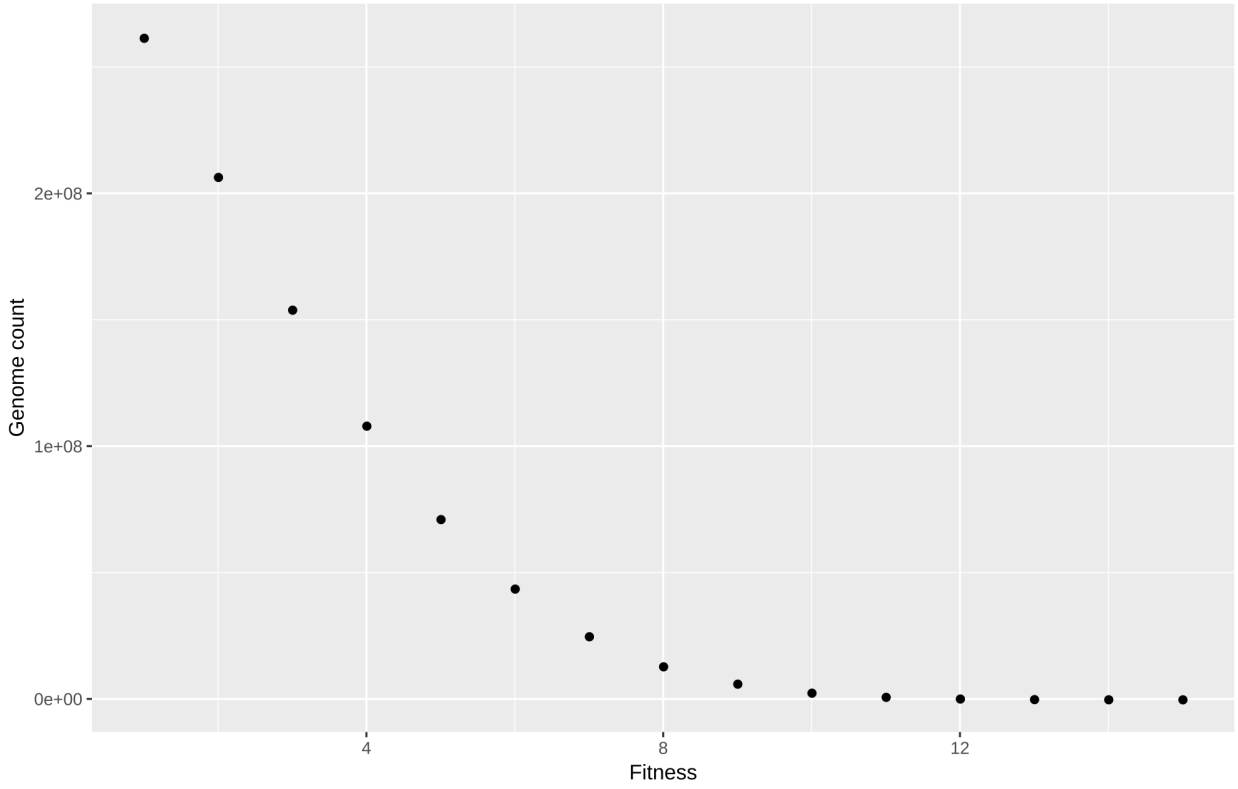
The above result is for a typical run, which provides a distribution within those organisms capable of surviving as follows, for red=128, green=128, blue=128:

2.11.2 Custom Random Numbers

REvoSim employs a pre-generated table of 65,536 random numbers 0–255 which it uses by default during the simulation, both for speed and to avoid potential biases from pseudo-random number generators. 10Mb of quantum-generated random numbers from randomnumbers.info are packaged into the executable and used to generate this table on load (i.e. different runs will have different random numbers, based on the quantum-generated random numbers); these can be replaced with any other random number file preferred by the user.

To load a custom file of random numbers use the 'Commands > Load Random Numbers…' command from the main menu to open a file selection dialog.

The random number file should be encoded as a random binary string, and should be a minimum of 65536 bytes. Once the desired file is selected press the 'Open' button to import the new random numbers. REvoSim will then ask for a byte offset to read the file from (thus allowing runs to be repeated with the same random numbers, if desired).



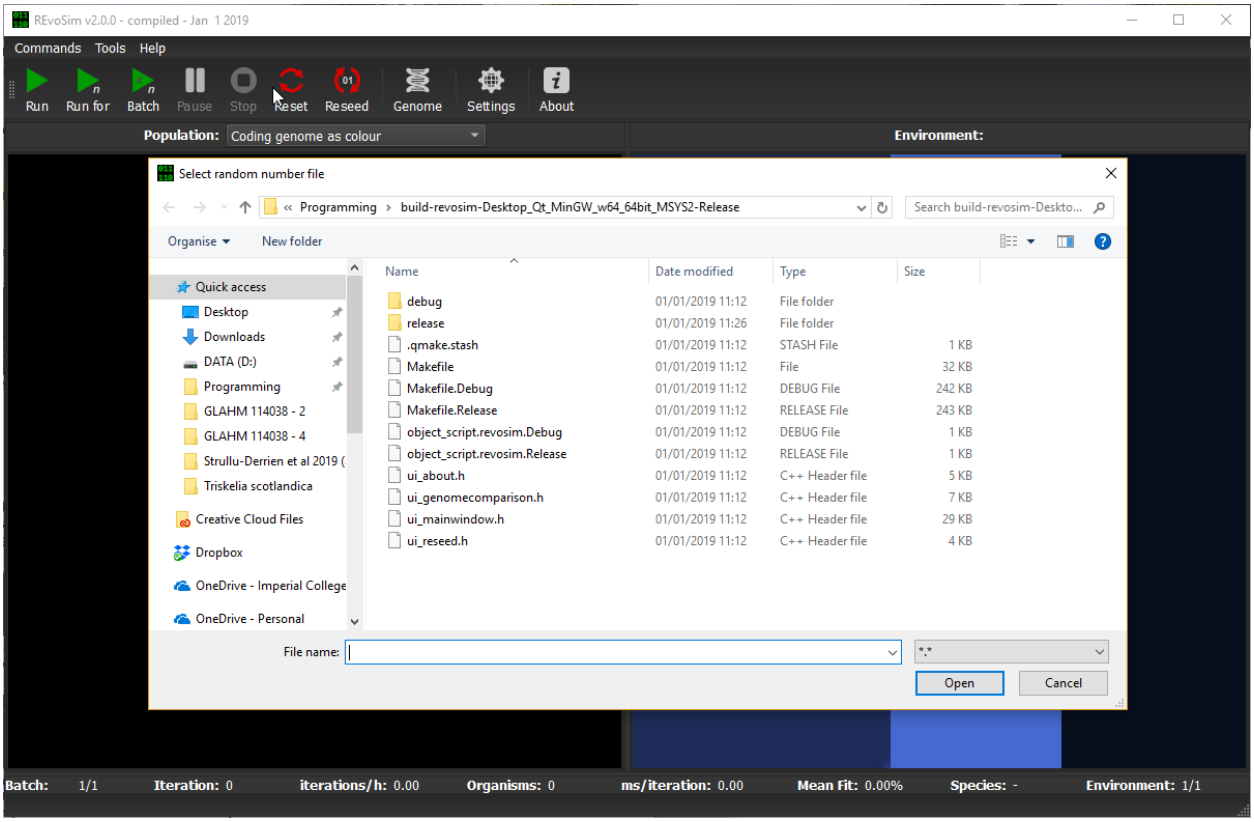


Fig. 9: Custom Random Number file open dialog.

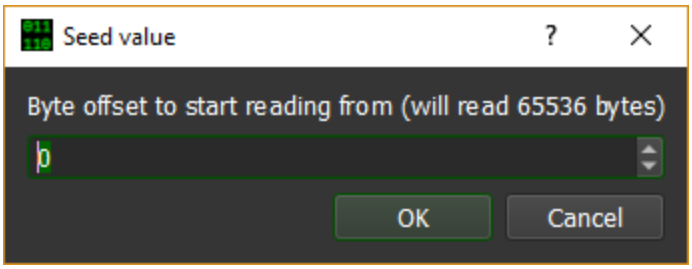


Fig. 10: Custom Random Number byte offset form.

Note that REVOSIM will always read 65536 bytes; and will throw an error message if it cannot.

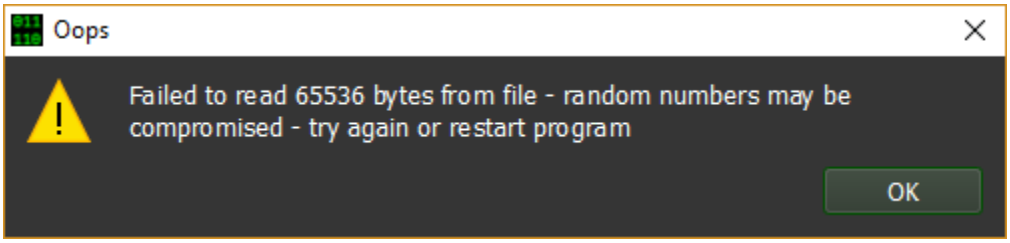


Fig. 11: Custom Random Number error on load message.

On success a pop-up message reading "New random numbers read successfully" will appear.

2.12 Command Line Options

REvoSim provides command line options to enable full control of simulations from command line or batch file environments. Every parameter configurable in the GUI can be set from the command line, and the program can be instructed to execute a simulation and then exit.

This document does not provide instructions on how to write a batch file, or determine correct paths (which will differ between operating systems).

Command line switches are documented below, and examples are given of their usage. Any combination of switches in any order is allowed. Note that on some systems, path names with spaces may cause problems: these are best avoided, but alternatively paths can be encased in double quotes if they contain a space.

Note in particular the –auto command, which automatically starts a simulation, and exits the program at its end. This will be required in most batch scenarios, as will the -k switch to turn on one or both types of logging.

e.g. a command line to run a simulation on the environment files in "c:\revosim\data\env1" for 50000 iterations, using normal (running) logs, and with settings at defaults except for 'toroidal' (on):

```
c:\\revosim\\revosim.exe -e c:\\revosim\\data\\env1 -t=On -k=Normal \--auto=50000
```

On Linux this command (assuming you are in the same folder as the REvoSim binary):

```
./revosim -e /home/user/environment/ -t=On -k=Normal \--auto=50000
```

The easiest way of modifying multiple variables for runs - and of setting custom log text, which was added in REvoSim v3.0.0 - is to use the software to output a settings files, and then to load this from the command line (–settings).

2.12.1 Running via SSH

In many instances it may be beneficial to run REvoSim on a remote machine. With remote desktopping solutions that provide a GUI, this can be achieved very easily following the normal operating procedures. However, due to the development of REvoSim, launching the software via the command line launches the GUI. This means that it is not possible to launch the command line version via SSH unless X-forwarding or another such solution is implemented. However, if no interaction with the GUI is required, there are a number of solutions that can bypass this requirement. The simplest, that will work on the majority of systems with Qt installed, is to include the flag -platform offscreen:

```
/revosim \--auto 1000 -platform offscreen
```

This co-opts a Qt platform plugin to simply render to an offscreen buffer. Another option is to install XVFB - this is an "X server that can run on machines with no display hardware and no physical input device" (More information). Once installed, you can launch revosim as follows:

```
xvfb-run -a  ./revosim \--auto 1000
```

In any case, once a run is complete, you can then use e.g. SFTP to collect output files as required.

2.12.2 Single-letter switches

These can be used as either in single letter form (-a) or long form (–startage - note the double -). All require a value, which can be separated from the switch using a space or an equals - or in single character mode, without a separator at all (this doesn't work in – long form, where a separator IS required). Where no parameter is noted in square brackets this is a boolean option, which is either On or Off (you can also use 1/0, true/false, yes/no, y/n - which are not case sensitive). e.g.

- -a30
- -a=30
- -a 30
- –startage=30
- –startage 30

For a boolean:

- -q On
- -q 1
- -q=truE
- -q Y
- –recalcfitness True
- –recalcfitness = Y

Full list of single-letter switches (also available using switch -h, help):

- **-a, --startage <age (integer)>** Starting age for organisms.
- **-b, --breedthreshold <threshold (integer)>** Breed threshold.
- **-c, --breedcost <cost (integer)>** Breed cost.
- **-d, --maxdifftobreed <maxdifftobreed (integer)>** Maximum difference to breed.
- **-e, --environment <directory>** Directory containing environment images.

- **-f, --usemaxdifftobreed <On/Off>** Use maximum difference to breed criterion.
- **-g, --breedwithinspecies <On/Off>** Only allow breeding within a species.
- **-i, --disperal <distance (integer)>** maximum dispersal distance.
- **-j, --outputpath <path>** path for output logs.
- **-k, --logtype <Phylogeny/Normal/Both>** logs to generate (phylogeny is end run log, normal is running log).
- **-l, --excludenodescendents <On/Off>** Exclude species without descendents from phylogeny logs.
- **-m, --environmentmode <mode (Static|Once|Loop|Bounce)>** Environment file cycling mode.
- **-n, --energy <energy (integer)>** Energy input.
- **-o, --tolerance <tolerance (integer)>** Settle tolerance.
- **-p, --phylogeny <Off|Basic|Phylogeny|Metrics>** Phylogeny logging mode.
- **-q, --recalcfitness <On/Off>** recalculate fitness each iteration.
- **-r, --refreshrate <rate (integer)>** environment refresh rate.
- **-s, --slots <slots (integer)>** Slots per pixel.
- **-t, --toroidal <On/Off>** Toroidal environment.
- **-u, --mutation <chance (integer)>** Chance of mutation (0-255).
- **-v, --csv <On/Off>** Use CSV format for normal log.
- **-w, --interpolate <On/Off>** Interpolate environmental images
- **-x, --gridx <size (integer)>** Grid (image) size, x.
- **-y, --gridy <size (integer)>** Grid (image) size, y.
- **-z, --genomesize <size (integer)>** Number of words in genome.

2.12.3 Long option only switches

We ran out of letters! These require the long format, with –. Otherwise they work as above.

- **--polling <rate [integer]>** Set polling rate for logging and screen refresh.
- **--auto <iterations [integer]>** Automatically start simulation and exit program after completion of specified number of iterations.
- **--nonspatial <On/Off>** Use non-spatial simulation mode.
- **--minspeciessize <size [integer]>** Minimum species size to appear in logs.
- **--fitnesstarget <target [integer]>** Fitness target.
- **--breed <Obligate/Facultative/Variable/Asexual>** Breeding mode.
- **--variablemutate <On/Off>** Variable mutation rates.
- **--nogui <On/Off>** Don't update GUI.
- **--pathogens <On/Off>** Turn pathogens on or off.
- **--pathogenmutate <chance (integer)>** Chance of mutation (0-255).
- **--pathogenfrequency <frequency (integer)>** Frequency pathogens are applied.

- **--customlogging <On/Off>** Record all custom logs.
- **--disparityLogging <On/Off>** Record disparity log.
- **--interactblocks <On/Off>** Turn block interactions on/off.
- **--multibreedlist <On/Off>** Turn multiple breed lists on/off.
- **--interactrate <frequency (integer)>** Frequency at which interactions occur.
- **--pathogenevolve <On/Off>** Set pathogens to evolve (on or off). If this is not set, the default is drift.
- **--minpredatorscore <threshold (integer)>** Minimum predator score required for direct energy theft.
- **--predationefficiency <integer>** Trophic efficiency of direct energy theft predation.
- **--interactXOR <On/Off>** Turns XOR interactions mechanism on/off.
- **--log, --logFile <file>** XML File containing the log outputs.
- **--v2log <On/Off>** Initiates v2.0.0 logging style.
- **--interactfitness <On/Off>** Interactions modify fitness.
- **--interactenergy <On/Off>** Interactions modify energy.
- **--li_population <On/Off>** Log images for population.
- **--li_fitness <On/Off>** Log images for mean fitness.
- **--li_sys_visualisation <On/Off>** Log images for visualisation system 1.
- **--li_sys_visualisation2 <On/Off>** Log images for visualisation system 2.
- **--li_species <On/Off>** Log images for species.
- **--li_settles <On/Off>** Log images for settles.
- **--li_fails <On/Off>** Log images for breed/settle fails.
- **--li_environment <On/Off>** Log images for environment.
- **--sys_fitness <Word string>** Fitness system.
- **--sys_breed <Word string>** Breed system.
- **--sys_mutate <Word string>** Mutate system.
- **--sys_var_mutate <Word string>** Variable mutate system.
- **--sys_var_breed <Word string>** Variable breed system.
- **--sys_pathogens <Word string>** Pathogens system.
- **--sys_species_ID <Word string>** Species ID system.
- **--sys_interactions <Word string>** Interactions system.
- **--sys_visualisation <Word string>** Visualisation system.
- **--sys_visualisation2 <Word string>** visualisation2 system.
- **--settings <file>** Load a REvoSim settings file.
- **--maxthreads <thread count (integer)>** Specify maximum threads to use.
- **--L1_variable <Energy/No_selection/Mutation_rate>** Variable to be linked (required).
- **--L1_imageSequence <directory>** Directory containing linkage mask images (required).

- **--L1_mode <mode (Static|Once|Loop|Bounce)>** Image file cycling mode (defaults to static).
- **--L1_interpolate <On/Off>** Image interpolation (defaults to true).
- **--L1_change_rate <rate (integer)>** Image refresh rate (defaults to 100).
- **--L2_variable <Energy/No_selection/Mutation_rate>** Second variable to be linked (required).
- **--L2_imageSequence <directory>** Directory containing second linkage mask images (required).
- **--L2_mode <mode (Static|Once|Loop|Bounce)>** Image file cycling mode (defaults to static).
- **--L2_interpolate <On/Off>** Image interpolation (defaults to true).
- **--L2_change_rate <rate (integer)>** Image refresh rate.

2.13 Tests

REvoSim has a test mode which is toggled by hitting the Tests button on the menu bar. This runs a series of software tests and then displays the output within the REvoSim main window (the Test Log). The GUI of the software when in test mode is shown below.

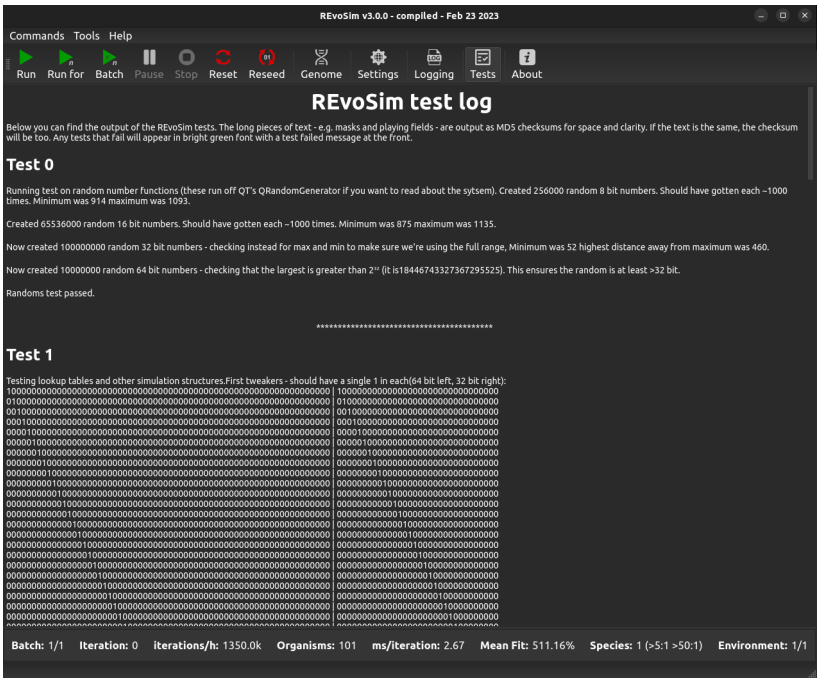


Fig. 12: REvoSim main window in test mode.

2.13.1 REvoSim test log

Our chosen approach allows all users to visually inpsect the outputs of the tests if they so wish, even if they are not able to build the software themselves. Tests that fail will appear in bright green font with a test failed message at the front. If you are using the software for your research, please do feel free to inspect the test outputs and contact the authors with any queries.

The expectations of each test, as well as the results, are written to a test log. Outputs are generally either numbers, or text strings which should be identical. The latter are output as MD5 checksums for space and clarity (if the text is the same, the checksum will be too). Each test generally comprises multiple components testing the different elements of a logically connected element of the software. An annotated exmaple of one of these is shown below.

Test zero - Annotated output

As an example of one of REvoSim's test, we use here test zero - which ensures REvoSim's random numbers are working as expected. The output is as follows:

> Running test on random number functions (these run off QT's QRandomGenerator if you want to read about the sytsem).

Many tests will provide useful information regarding the functioning of the underlying elements – in this case, REvoSim's random numbers rely upon the QRandomGenerator class of the Qt framework.

> Created 256000 random 8 bit numbers. Should have gotten each ~1000 times. Minimum was 914 maximum was 1093.

The first element of the test ensures an even spread of random 8 bit numbers, and reports the outputs. The outputs will vary, but will numbers within sensible bounds will allow the test to pass.

> Created 65536000 random 16 bit numbers. Should have gotten each ~1000 times. Minimum was 875 maximum was 1135.

The second element of the test ensures an even spread of random 16 bit numbers, and reports the outputs.

> Now created 100000000 random 32 bit numbers - checking instead for max and min to make sure we're using the full range, Minimum was 52 highest distance away from maximum was 460.
>
> Now created 10000000 random 64 bit numbers - checking that the largest is greater than 2^{32} (it is18446743327367295525). This ensures the random is at least >32 bit.

REvoSim also relies on longer random numbers. In the interests of speed, the tests change from checking for an even spread, to ensuring that in a large number of randoms, the minimum and maximum are close to the bounds of the number.

> Randoms test passed.

As long as all of the individual parts of a test pass, the test as a whole does.

2.13.2 Failed Tests

If a test fails, the outputs appear in green with a failure message at the top, and message at the bottom highlighting explaining what has failed.

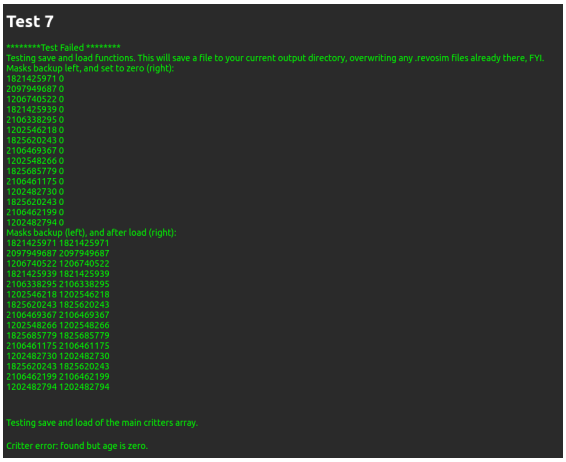


Fig. 13: A failed REvoSim test. Don't panic, it's in the development branch.