TREvoSim Documentation

Release 3.0.0

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

Mar 23, 2024

CONTENTS

The [Tr]ee [Evo]lutionary [Sim]ulator program.

TREvoSim is an individual-based evolutionary model, focussing on the simulation of evolutionary trees and associated character data through a first-principles approach. It shares a number of elements in common its sister package REvoSim (see Garwood et al. 2019, Furness et al. 2023 in relevant references), but it differs in species concept, and eschews both space and sexual reproduction.

TREvoSim has been released with the intention that it can be used as a multi-purpose platform for the study of many evolutionary phenomena, but in particular, acts as one of a limited number of methods for simulating character data and tree topology at the same time and is one of very few individual-based simulations capable of doing so. We note that, as with REvoSim, this package is complementary to the many other approaches of studying evolution on a range of different timescales, and will be continually developed by the core team to expand its capabilities.

A full description of the algorithm can be found in Keating *et al.* (2020), referenced below. In brief, a simulation employs digital organisms comprising binary strings. These are used to calculate fitness through comparison to against

one or several sets of random binary strings (the environment(s)), and ultimately provide the character matrices and trees output by the simulation. Organisms exist in a playing field and compete with each other; one is selected, the probability of which being based on a fitness algorithm, for reproduction each iteration. When this occurs, the organism is mutated, and returned to the playing field, overwriting the least fit organism. A new species is defined based on hamming distance to the parent species, or last species to originate within the lineage. Species are kept in a list, and their genomes are recorded on extinction. The simulation runs until the requested number of species have existed, and then writes data in the requested format.

e: palaeoware@gmail.com

w: https://github.com/palaeoware

CHAPTER

ONE

RELEVANT 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 (describes the initial release of REvoSim, including fitness algorithm).

Keating, J.N., Sansom, R.S., Sutton, M.D., Knight, C.G. and Garwood, R.J., 2020. Morphological phylogenetics evaluated using novel evolutionary simulations. Systematic Biology 69(5):897-912. DOI: 10.1093/sysbio/syaa012 (describes, and utilises, TREvoSim v1.0.0)

Mongiardino Koch, N., Garwood, R.J. & Parry, L.A. 2021. Fossils improve phylogenetic analyses of morphological characters. Proceedings of the Royal Society B 288(1950):20210044. DOI: 10.1098/rspb.2021.0044 (describes, and utilises, TREvoSim v2.0.0)

Mongiardino Koch, N., Garwood, R.J. & Parry, L.A. 2023. Inaccurate fossil placement does not compromise tip-dated divergence times. Palaeontology 66 (6): e12680. DOI: 10.1111/pala.12680 (utilises TREvoSim v2.0.0)

CHAPTER

TWO

TABLE OF CONTENTS

2.1 Introduction

2.1.1 Overview

TREvoSim is an individual-based evo-evolutionary model that lacks complex spatial structure (if you want space, check out sister package REvoSim). It is abstracted, and particularly well-suited to simulating discrete binary (~phenomic) data for phylogenetic research, and the outcomes of other evolutionary and environmental processes in the absence of space. The papers on the page *Welcome to TREvoSim's User Manual* provide useful information regarding the software, its approach, and its limitations.

In brief, individuals comprise binary strings, referred to throughout this documentation as their genome, which can be output as a phylogenetic matrix at the end of the run (i.e. these form the character data in the simulation). Genomes are compared with one or several environments, also comprise a series of binary strings (masks). Fitness of the individuals is calculated based on the hamming (i.e. character) distance between their genome and the masks. For a full description of the fitness algorithm, see Garwood et al. (2019) and Keating et al. (2020), cited on the *Welcome to TREvoSim's User Manual* page. Individuals are placed in a list structure called the playing field (of which there can be one, or many, with the same – or different – environments), and compete with other individuals within their own playing field. An individual's fitness dictates its probability of duplication, which happens for one individual per playing field per iteration. On duplication there is a user-defined chance of a mutation being applied. TREvoSim employs a lineage-based species concept: a new species is defined based on genomic distance to either its parent species, or to that parent's last daughter species to appear. As a simulation runs, a tree representing the evolutionary relationships between the species is recorded, and the character data for each species is recorded at species extinction. A phylogeny and character data can be output after a requested number of taxa has evolved, or after a set number of iterations.

2.1.2 Simulation setup

Variables can be defined by clicking on the Settings button in the toolbar. This creates a pop up window, which has two tabs:

Organisms and simulation
: This tab contains a series of settings that defines the parameters for the simulation setup, and properties of the organisms. See *Settings - Organisms and simulation*.

Environment and extinction
: Variables defined within this tab control extrinsic factors that impact on the organisms. See *Settings - Environment and Extinction*.

2.1.3 Output setup

TREvoSim provides functionality for two types of logs. All logs are placed within a folder called TREvoSim_output in the save path defined in the toolbar. To change this click on change in the toolbar. The options associated with these can be accessed by clicking on the Output button in the toolbar, which creates a pop up window with the following tabs:

End run log
On this tab you can configure outputs at the end of a log, including:

- Two custom output files – the options allow you to define file name, content (replacing keywords in ||double pipes|| with the requested information – see *Logging the Simulation*) and extension.
- A standard nex tree file.
- A working log which includes all operations the software performs across a run if you want to inspect its workings (this file can be very large).

Running log
This tab allows a running log to be defined that can record the state of the simulation at user-requested frequencies (see *Logging the Simulation*). The tab includes options for:

- Header text
- Body text
- An option to write the running log (if not requested this is not output)
- A tick box to output for a specialised Ecosystem Engineering log
- A spin box that dictates the frequency with which which the running log is written

2.1.4 Quick start

The TREvoSim defaults will allow you to create exemplar data in TREvoSim by hitting the Run button on the toolbar. This will output a nexus file with the final character matrix in it that can be loaded in e.g. R for analysis, placed in a folder called *TREvoSim_output*, the default location of which is your desktop. It will also output a nexus formatted tree file in this folder.

2.1.5 Default settings

TREvoSim's default simulation parameters are intended to be a good general place to start when thinking about phylogenetic questions. Their outputs (trees, characters) are benchmarked against twelve total evidence empirical datasets. Details of the measures used, the script used to generate them, the source of the empirical data, and the output graphs from the script, are available in the folder *Benchmarking* in the source code repository.

2.2 Compiling, Installation, and Requirements

2.2.1 Compiling from Source

Windows 64-bit

For v1.X,2.X - QT Creator + QT v5.x 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 "MSYS2-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 MSYS2 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 MSYS2 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.

For v3.X - QT Creator + QT v6.x using Qt installer and MinGW (64-bit) For the TREvoSim v3.0.0 release a build was created using an stand install of Qt6 using the QT online installer available from the Qt website. If following this approach, ensuring the installation includes the latest Qt v6 release, and Qt Creator will allow build of the software. To achieve this, open CMakeLists.txt in Qt creator following installation, select to build a release following the Qt creator documentation, and then initiate a build (ctrl + B / Build, then Build Project). This will create an executable that can be launched by double clicking. To create a build that includes the tests, it will be necessary to modify the cmake file as per the instructions provided with the Google Test framework.

Note: At the time of release of v3.0.0 the authors temporarily have limited access to windows machines to allow us to make the above changes to cmake, and provide more explicit instructions. We expect to make a patch release with those changes, and featuring improved documentation for Qt v6.x builds, in May 2024.

Ubuntu 22.04 64-bit - QT Creator + QT v6.x 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 qt6-base-dev libqt6core5compat6-dev
```

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

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

3. Within TREvoSim folder, run the following command to build the software:

```
cmake --build .
```

4. Launch the software by double clicking on the TREvoSim binary that has been created in this folder.

Using Qt creator.

1. Install Qt 6.X on your system by running the installer from Qt: https://www.qt.io/download
2. Download source code, launch Qt Creator, and open the CMakeLists.txt file. Configure build and follow standard debug/release procedure.

MacOS - QT Creator + QT v6.x using Clang and xcode

1. Xcode can be downloaded from the Apple Store or Apple's developer website (including older versions of Xcode): https://developer.apple.com/download/. We recommend always using the latest Xcode available from Apple that has been tested with the Qt version you are using.
2. Install Qt 6.X and Qt Creator on your system by running the installer from Qt: https://www.qt.io/download. An alternative may be to install via homebrew:

```
$ brew install qt
```

3. To build the software and test suite, download source code, launch Qt Creator, and open the .cmake file.
4. Configure build and follow standard debug/release procedure.

- See the Qt Mac Deployment tool: https://doc.qt.io/qt-6/macos-deployment.html#the-mac-deployment-tool
- Also see the volume package information to make a .dmg: https://doc.qt.io/qt-6/macos-deployment.html#volume-name

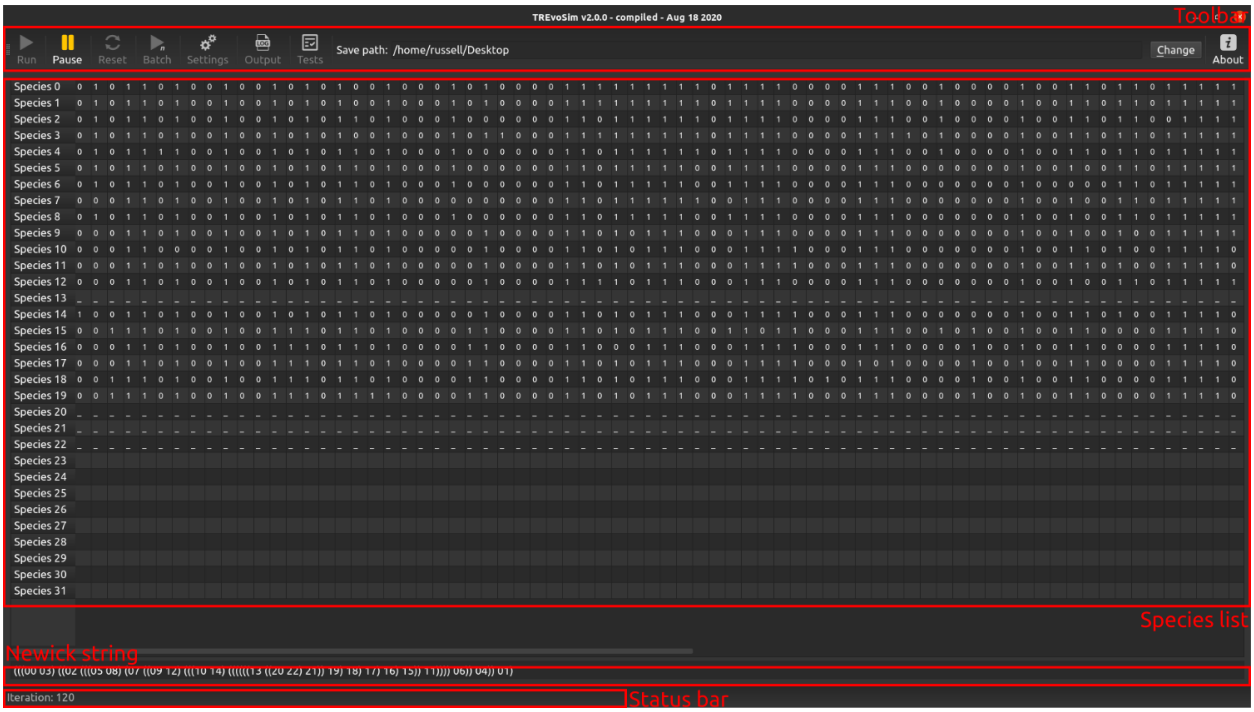
2.2.2 Installation

From the TREvoSim GitHub repository pre-compiled binary releases and packaged installers can be downloaded. For Windows users we provide both a portable binary release (.zip; v1-3) - which just needs extracting to a convenient location as per the instructions on the release - and a self contained installer (currently v1-2 only). For Mac we provide a zip containing the TREvoSim program that can be downloaded from the TREvoSim GitHub repository. To install the software, double click on the .dmg and follow the on screen instructions. You may be required to the approve the software in security and privacy settings before it will launch. For Linux users, the above instructions 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.

2.2.3 Requirements

TREvoSim has no minimum requirements as such, and will run on most standard systems (Windows/Linux/Mac); it however has not been tested on versions of Windows older than Windows 10, before Ubuntu 22.04, and below 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.

2.3 Window Layout



The main window comprises a number of elements, outlined below.

2.3.1 Toolbar

The main toolbar appears as follows when the simulation is not running:

And when it is running, as below:

The buttons control the simulation, as well as launching dialogues, and defining program output location.

Run
: This button launches a simulation, and then runs it until the requested number of species has evolved (see *Settings - Organisms and simulation*), the simulation is paused, or cancelled (escape key).

Pause
: Pauses simulation, which can be resumed when requested by pressing pause again.

Reset
: Resets the simulation by removing all digital organisms from the playing field and species list.

Batch
: For repeated runs using the same settings, TREvoSim provides a batch mode. The number of runs is requested on launching batch mode, and output files are labelled accordingly. On launching a batch run, TREvoSim will complete the first run in serial, updating the GUI, allowing it to provide warnings

at the end of the run (and provide the opportunity to cancel). After this, all remaining batches will be completed in parallel (note that this speeds up each simulation significantly: the duration of the first run is not a particularly good indicator of how long you should expect the batch to run for in total).

Settings
Launches *Settings - Organisms and simulation* dialogue.

Output
Launches *Logging the Simulation* dialogue.

Tests
Runs the TREvoSim test suite, reporting test results in a new panel that appears on the right of the main window.

Save path
All files created by the program are written to a folder titled *TREvoSim_output*, created in this location.

Change
Launches a file explorer to change the save path.

About
Opens a dialogue with information about TREvoSim, and links to the code and for bug reporting.

2.3.2 Species list

This panel updates as a run of the simulation progresses: it shows the binary string for the species in a simulation (up to the first 128 characters are displayed), once available. The panel starts as blank; once speciation occurs and any given species remains extant (i.e. there is a living representative in the playing field), the species is represented by a series of dashes. Once the species is extinct, and characters have been recorded (see algorithm description in Keating *et al.* 2019 for more details), these are printed to the panel.

2.3.3 Newick string

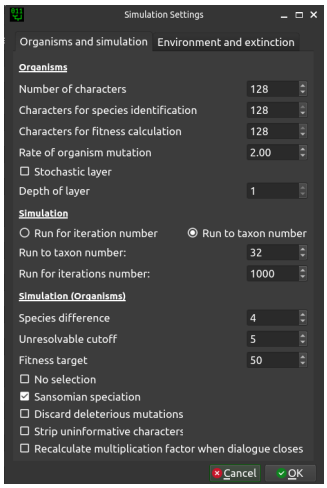
This text box updates with a Newick string showing the tree of the current simulation run, which updates as speciation events occur. This provides an immediate picture of, for example, the tree asymmetry within a run.

2.3.4 Status bar

The status bar is updated with messages during and after a run: for example, during a run it displays the iteration number, and afterwards it can provide messages about the data (e.g. number of uninformative characters, if these are not set to be stripped out, number of identical terminals). It also displays progress bars where one is required.

2.4 Settings - Organisms and simulation

Clicking on the settings button of the toolbar will launch a settings dialogue which has two tabs - one with settings for the organisms in the simulation, and one controlling the environment for the simulation.



2.4.1 Organisms

Number of characters/genome size
: This defines the number of bits in character string for each organism, and ultimately the character number of the matrix output by the software.

Characters for species identification
: This defines the number of characters used to identify species, between zero and the stated limit. By default this will update to be the same value as the genome size, but changing it to a smaller value allows a portion of characters that are not tied to species identity.

Characters for fitness calculation
: This defines the number of characters used in the fitness algorithm, between zero and the stated limit. By default this will update to be the same value as the genome size, but changing it to a smaller value allows for a 'non-coding' portion of the genome, with a more drift-like/stochastic behaviour (although this portion will hitch-hike and thus could reflect patterns driven by the selection of organisms).

Rate of organism mutation
: This is the rate of mutation for the organisms in the simulation, in units of mutations per hundred characters per iteration.

Stochastic layer
: This tick box dictates whether a stochastic layer is used. The stochastic layer is a secondary string of bits that can be used to abstract the fitness calculation from the genome and its mutations, through a mechanism based on many-to-one mapping. When this box is ticked, on settings close, another dialogue is launched that allows you to - for every possible combination of 4 bits - define an output, i.e. 0 or 1, referred to here as the map. When the setting is enabled, each organism stores an internal genome four times the length of the organism's gnome. During the simulation, mutations are applied to the stochastic layer (at the user defined rate per 100 characters), but all other operations employ the standard genome. When mutations are applied, the map is then used to convert bits from the stochastic layer into the genome that is used elsewhere (i.e. for every four stochastic bits, the output is 1 standard bit, based on the map). This abstraction process could be considered, for instance, a highly simplified parallel to e.g. ontogeny: it removes the fitness of the organism from direct control of its genome, and thus facilitates silent mutations and neutral theory of evolution type / less strongly adaptationist dynamics.

2.4.2 Simulations

Run mode radio
This has two options, run for iteration number or run to taxon number, which switches between the two options below.

Run to taxon number
If run mode is taxon number, a simulation will run until this number of species has evolved, and then terminate.

Run for iterations number
If run mode is iteration number, a simulation will run until this number of iterations, and then terminate.

Species difference
The hamming distance between a selected organism, post mutation, and either the character string when the species originated, or the last species to originate within the lineage (see paper for rationale) is used within the TREvoSim species concept. This setting defines the required hamming distance for a speciation to have occurred.

Unresolvable cut off
With low character numbers, especially when TREvoSim is set to strip out uninformative characters, terminals can have the same character string within a matrix, and thus be unresolvable. This setting defines upper limit for identical terminals. If there are more unresolvable taxa than this number in a single run, output files are not written and a warning is provided. If batch mode is underway, the current run is discarded and started again.

Fitness target
This is the target value for the count of 1s in the fitness calculation. See Garwood et al (2019) or Keating et al (2020) for details of the fitness algorithm. At low and high values (max being mask number multiplied by character number) fewer genomes will have peak fitness, at half max value, a larger number of peak fitness organisms exist. This can be quantified using fitness histogram menu command.

No selection
If this option is enabled, fitnesses are not calculated for digital organisms in the playing field, and instead every iteration and random individual is selected for replication.

Sansomian speciation
This option dictates when the character string of a species is recorded: at speciation, or at extinction. Sansomian speciation, the default, is to record the this when a species goes extinct. This ensures if it is a long-surviving species that the recorded characters string best reflects that closest to its sister group. When the option is not selected, genome is recorded at speciation.

Discard deleterious mutations
Optionally, TREvoSim can accept only mutations which are neutral, or improve the fitness of an organism.

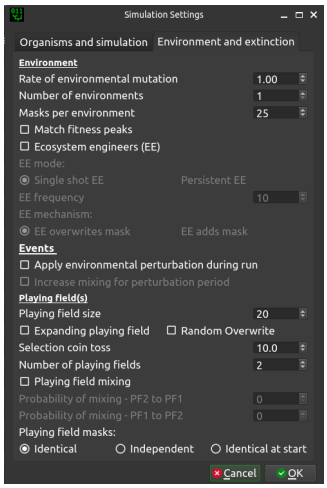
Strip uninformative characters via subsampling
TREvoSim gives the option of writing matrices of only parsimony informative characters. When this option is checked, the software attempts to provide the requested character number of only informative characters. It achieves this by multiplying the number of characters and species difference by an empirically calculated factor (see below) at the start of a run. After a run has completed, informative characters are randomly subsampled to the requested number of characters; if there are not enough characters to achieve this, in batch mode the run is discarded and restarted, in single run mode an error message is provided.

Recalculate multiplication factor on dialogue close
The multiplication factor required to achieve a set number of informative characters via subsam-

pling depends on the settings of any given run. Selecting this option calculates and sets this factor empirically after the settings dialogue is closed by conducting a ten-run batch and working out the proportion of informative characters within those runs. This needs to be recalculated after any settings are changed. If it is not, or has not been set, the software will run with a large factor, and thus be slower than necessary.

2.5 Settings - Environment and Extinction

Clicking on the settings button of the toolbar will launch a settings dialogue which has two tabs - one with settings for the organisms in the simulation, and one controlling the environment for the simulation.



2.5.1 Environment

Rate of environmental mutation
The environment in TREvoSim - which comprises a series of random numbers, or masks - also mutates, allowing lineages to track fitness peaks. This setting dictates the rate of mutation for the environment, in units of mutations per hundred characters per iteration.

Number of environments
From v2.0.0 TREvoSim allows multiple environments (i.e. numerous sets of masks). During a simulation the fitness of each organism is calculated for every environment, and the organism's overall fitness is defined as that from the environment to which it is best suited (i.e. for which it has the highest fitness).

Masks per environment
This dictates the number of masks (random numbers) in the environment. The more there are the flatter the fitness landscape is likely to be. This can be explored using the count peaks functionality.

Match peaks
When TREvoSim has multiple environments associated with a playing field, enabling this option ensures that at initialisation each environment has the same maximal fitness (i.e. fitness peak height). Note, that this is currently applied within, not across, playing fields, if the latter have different masks. When this setting is enabled, as well as achieving fitness peaks of the same height, TREvoSim will use a heuristic search to seed the simulation with the fittest organism it can that also has the same fitness for all environments (it achieves this in more than 99% of runs; if playing fields have different environments the algorithm will search for organism with the best mean fitness across playing fields).

As currently formulated, once a simulation is running, mutations will cause peak fitnesses to diverge between environments (add mask / overwrite mask).

Ecosystem Engineers
From v3.0.0 TREvoSim allows organism-environment feedback through an ecosystem engineering functionality. When this option is enabled, ecosystem engineering starts at either half the requested iteration count, or after half the requested species have appeared. The first time EE occurs, a single, randomly-chosen individual is assigned "EE status" (as are any other individuals with identical genomes), and its genome is used as the basis for organism-environment feedback, as outlined below.This status is passed on to all offspring of this individual for the remainder of the run.

EE mode
EE can either be applied once or persistently, defined by this pair of radio buttons. In single shot EE, engineers are applied once, half way through a simulation, as outlined above. In contrast, if this radio button is selected, then EE is applied repeatedly after its first application. The frequency at which this occurs is user-defined (see below). On reapplication, a random individual is chosen from within those individuals with EE status every reapplication as the basis for organism-environment feedback. If EEs have gone extinct, no action is taken.

EE Frequency
this is the frequency with which persistent EE are applied after the first application at half way through a run, in iterations.

EE mechanism
EE-based organism-environment feedback can occur in two ways in TREvoSim if EE overwrites mask is selected, then the genome of the selected individual is used to overwrite the last mask overwrite for every environment on every playing field (i.e. TREvoSim selects a single individual from across all playing fields with EE status and then uses this to modify the environment across all, even those in which the organism does not reside). In contrast, if EE adds mask is selected, on the first application of EE only, a new mask is added to all environments across all playing fields, which matches the genome of the selected engineer (this modifies the adaptive landscape in different ways to overwriting a mask). All subsequent applications of EE, if they are persistent, revert to overwriting the mask that was added at first application.

2.5.2 Events

Apply perturbation during run
TREvoSim offers the option of applying an environmental perturbation during a run. When this is selected, at halfway through a run (after the speciation of the n/2 the requested species, or at half the requested iteration count), all masks for all environments are overwritten with new, random, masks. If playing field masks are independent this results in new random masks for all environments on all playing fields. If masks are set to be identical across playing fields, then new masks are written to all environments, but these are identical across the playing fields. This perturbation then lasts for 10% of the iteration number at which it began, during which the masks gradually return to their pre-perturbation state (specifically, over the course of the perturbation period, 90% of bits across all masks and environments are copied back over the new masks from a copy created at perturbation initiation). As currently coded, during this period, only mutations occurring to the new, perturbation, rather than the previous environment will be applied, and overall the environmental change due to mutation will be lower than the remainder of the run (although rate of change will likely be higher as the perturbation environment is reverting to the pre-perturbation one).

Increase mixing for perturbation period
When mixing is enabled between playing fields, this can be accelerated during a perturbation. When this option is selected, mixing between playing fields increases by a factor of ten when a perturbation is occurring, then returns to background rates once it is complete.

2.5.3 Playing field(s)

Playing field size
This dictates the size of the playing field within the simulation, i.e. the number of individuals alive at any given time. Small sizes will lend themselves to asymmetrical trees with short tips.

Expanding playing field
When this option is enabled, each species only has a single entry in the playing field. This is overwritten by an individual's progeny every time that species is selected for duplication, and at speciation a new species is added to the end of the playing field (which thus expands to accommodate new specis - hence the name). This removes intraspecific competition.

Random overwrite
When this is checked, when a new organism is returned to the playing field (see Keating et al. 2020 for algorithm details), it will overwrite an individual at random. When it is not checked it overwrites the organism with the lowest fitness (or one of these at random if multiple individuals share the lowest fitness).

Selection coin toss
This dictates the probability of choosing any given individual when moving down the playing field in the coin toss to select an individual to duplicate. The probability of selecting an individual is the reciprocal of this (i.e. 1 / this number). If, e.g., this is 2.0 there is a 50% chance of selecting the first organism in the playing field, then 50% selecting the next, and so on.

Number of playing fields
From v2.0.0 TREvoSim allows multiple playing fields. These are initialised with the same individual, and then operate independently (each playing field will thus form a clade). When playing fields have different masks, a heuristic search is used to initialise the simulation with an organism that has the best mean fitness across all playing fields.

Playing field mixing
When there is more than one playing field, it is possible from v3 of TREvoSim to allow mixing between playing fields. When this option is selected, mixing occurs through the overwriting of a random individual in one playing field with a copy of a random organism from another. The chance of mixing - defined below, per iteration - applies to a playing field as a whole. When there are more than two playing fields, each playing field has an equal probability of mixing occurring (defined by the Probability of playing field mixing - PF2 to PF1 option below). When it does, a random individual from the chosen playing field is used to overwrite a random individual in one of the other playing fields (also randomly chosen). When there are two playing fields, it is possible to have asymmetrical mixing rates, using both the below options, and otherwise mixing occurs in the same way.

Probability of playing field mixing - PF2 to PF1
If a random number bounded to 100 is smaller than this value, then mixing will occur any given iteration. As such, as value of 10 here equates to a 10% chance of mixing occurring every iteration. When there are more than two playing fields, this value defines the probability of mixing for all playing fields per iteration (the label updates to reflect this when this is the case). When there are just two, it defines the probability of mixing from playing field 2 to playing field 1.

Probability of playing field mixing - PF1 to PF2
As above, but when there are two playing fields, this defines the probability of mixing from playing field 1 to playing field 2. When there are more than two masks, this is not used.

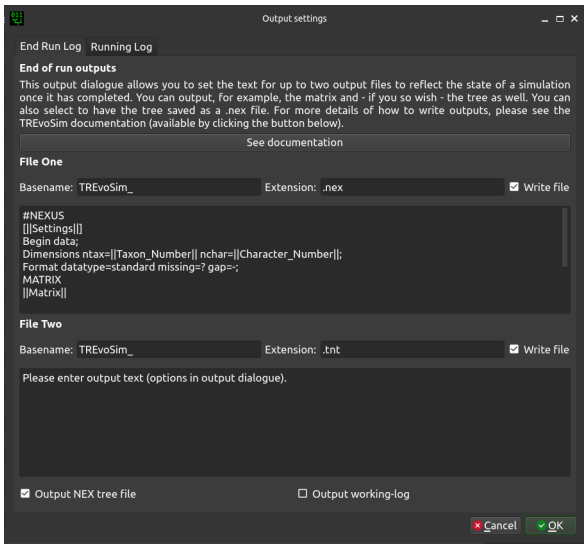
Playing field masks
The options here define the behaviour of masks across playing fields. They can be identical, start identical and then evolve away from each other, or be independent. If the last is selected the playing fields are initialised with the individual that has the highest mean fitness across all playing fields after 5000 initialisations (with a single playing field the initialising organism is within the top 10% of possible fitnesses for the starting masks).

2.6 Logging the Simulation

2.6.1 Logging system

End run log

TREvoSim has a versatile logging system which allows the user to define outputs in a range of formats required for phylogenetic inference packages, or e.g. R, provided these allow plain text inputs. Clicking on the output button of the toolbar will launch the output dialogue, which has two tabs. The first is a log that is output at the end of the run:



A typical use case for this is when you would like to use the outputs of a finished simulation to do downstream analyses for e.g. phylogenetic methods. This tab provides options for two custom log files, which are placed, after a run, within the *TREvoSim_output* folder created on the save path. Basename defines the start of the filename, which then either includes a number, which iterates with subsequent runs when output is not set to append, or *_batch* if the outputs are set to append and a batch is being run. The file extension is also defined here. The two text boxes then allow custom file content to be written: text is written as provided to the file – for example with run instructions or program commands – and any of the keywords shown below included within two vertical bar (|) symbols are replaced as a file is written. The default outputs are shown in the figure above – they create a vanilla nexus file which allows you to, for example, load the character matrix into R.

There are two further options at the base of the dialogue:

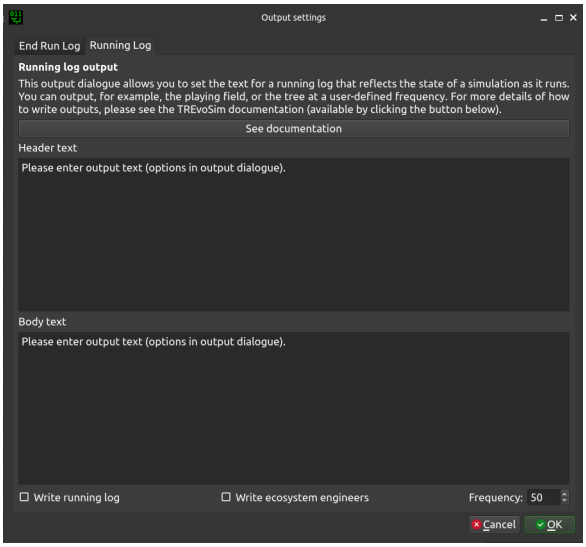
Output NEX tree file
: This outputs a tree for each run in a standard nexus format with the tree and translate block, as well as a comment with the settings of the run written to it. These do not append, but each file has the run number at the end of the file name.

Output working log
: When this is checked, TREvoSim outputs a text file outlining many of the steps each iteration, such as the state of the playing field, the environment, and the processes the software is going through. This helps understand and fact check any given run, but for significant playing field sizes, taxon numbers, or character numbers, it creates a significant (10s – 100s of MB) text file.

Running log

The other tab is a running log which is output during a run:

This is of utility for studying processes as they occur during a run, as it is output on a regular basis as the simulation progresses. The text entered in the header text region is output once, when the file is created – so if, for example, a



CSV output is required and a series of strings are comma separated, these will form the column headers. Note that in order to write the first data output on the next line, a newline is required at the end of this string.

The next box (body text) is output at the requested frequency and can be used to record anything in the state of the simulation required to study a process of interest. It uses the same keywords (shown below) as the end run log, outputting data in the current state for the iteration at which the log is written. Again, this requires a line break at the end of the string of interest if you would like outputs for each iteration to appear on subsequent lines.

There are three further options at the base of the dialogue:

Write running log
: By default this log is not output when a simulation runs. In order to write the files this option needs to be enabled.

Write ecosystem engineers
: There is a custom log for simulations in which ecosystem engineers are enabled (the nature of this functionality does not allow all required information to be easily output using the running log: this is primarily a convenience function).

Frequency
: This dictates the frequency, in iterations, with which the running log is written.

2.6.2 Keywords

Keywords within two vertical bars (e.g. ||Matrix||) are replaced as a file is written as follows:

Character_Number
: This outputs the character number.

Count
: This is replaced with a counter for batch runs; incrementing by one using C++ numbering (i.e. starting from zero).

Ecosystem_Engineers
: This prints a list of species, and their ecosystem engineering status (i.e. whether any members of this species are ecosystem engineers).

Iteration
: This outputs the current iteration number.

Matrix
This is replaced with the matrix from the run.

MrBayes_Tree
This writes a tree in standard Newick format, including branch lengths (these are based on iteration number throughout), e.g.

```
(S_01:13,(S_02:8,(S_03:13,(S_04:32,((S_06:4,(S_07:10,(S_08:6,(S_09:5,(S_10:10,(S_11:6,(S_
↪12:6,(S_13:2,(((S_22:11,(S_23:12,(S_24:9,((S_26:28,((S_28:37,(S_29:2,(S_30:2,(S_31:1,S_
↪27:1):1):38):23):20,S_25:12):31):41,S_21:31):24):27):1):70,S_14:13):42,(S_15:21,(S_
↪16:2,(S_17:11,(S_18:2,(S_19:10,(S_20:14,S_
↪05:25):7):1):4):2):1):2):1):13):58):6):2):2):5):2):36,S_00:20):3):1):3):7):85
```

Note The mechanism used for tree writing differs between TNT and MrBayes outputs – the tree topology is the same, but the taxon order differs.

Settings
Writes the settings to the file (this is provided as a useful way to record, with any output data, the set up for TREvoSim for any given run).

TNT_Tree
This writes a tree, if required, in TNT format (i.e. only brackets and terminal labels), e.g.

```
(((((00 ((((((((((((((05 20) 19) 18) 17) 16) 15) (14 ((((21 ((25 ((((27 31) 30) 29)␣
↪28)) 26)) 24) 23) 22))) 13) 12) 11) 10) 09) 08) 07) 06)) 04) 03) 02) 01)
```

Note The mechanism used for tree writing differs between TNT and MrBayes outputs – the tree topology is the same, but the taxon order differs.

Time
Adds a timestamp.

Taxon_Number
Writes taxon number.

Unresolvable
This prints a list of unresolvable taxa (or a notice that there are none if required).

Uninformative
Writes the number of uninformative characters.

As an example, the following would output a block of text that could be run as a macro in tnt:

```
NSTATES nogaps;
xread
'Written on ||Time|| Variables: ||Settings||'
||Character_Number|| ||Taxon_Number||
||Matrix||
;
piwe-;
keep 0; hold 100000;
rseed *;
xmult = level 10; bbreak;
export - ||Count||_POUT.nex;
xwipe;
```

Should any other output options be required, please file a feature request. Keywords are not case sensitive.

2.7 Menu commands

TREvoSim has a series of options which can be selected using the Commands menu at the top left of the program (above the run button). These are as follows.

2.7.1 Save Current Settings

By default TREvoSim saves the simulation settings between sessions (it writes a settings file to the same folder as the software binary / executable on close). This menu command (or shortcut ctrl + s) rewrites that setting file without closing the program.

2.7.2 Save settings as...

This opens a file save dialogue, and then writes a TREvoSim settings file to the requested location (shortcut ctrl + shift + s). TREvoSim settings files are written in an XML document that can be opened in any text application.

2.7.3 Load settings from file...

Load settings file (ctrl + o) opens a saved settings file and updates the simulation settings accordingly.

2.7.4 Restore default settings

If at any point you require default settings, this menu option (or shortcut ctrl + shift + d) restores all settings to default.

2.7.5 Fitness histogram

The fitness algorithm for TREvoSim is described in Keating et al. (2020). The fitness landscape varies depends on the simulation settings. This menu command (shortcut ctrl + shift + h) provides a basic assessment of that landscape by showing the distribution of fitnesses for every possible genome for the current settings. It provides an option to select how many bits in the test organism's genome as larger genome sizes (e.g. >16) take exponentially longer given the need to try every potential genome rearrangement.

2.7.6 Run tests

TREvoSim includes a test mode, which is launched by clicking the Tests button on the main toolbar, this menu command, or the shortcut ctrl + shift + t (see *Tests*)

2.7.7 Set uninformative factor

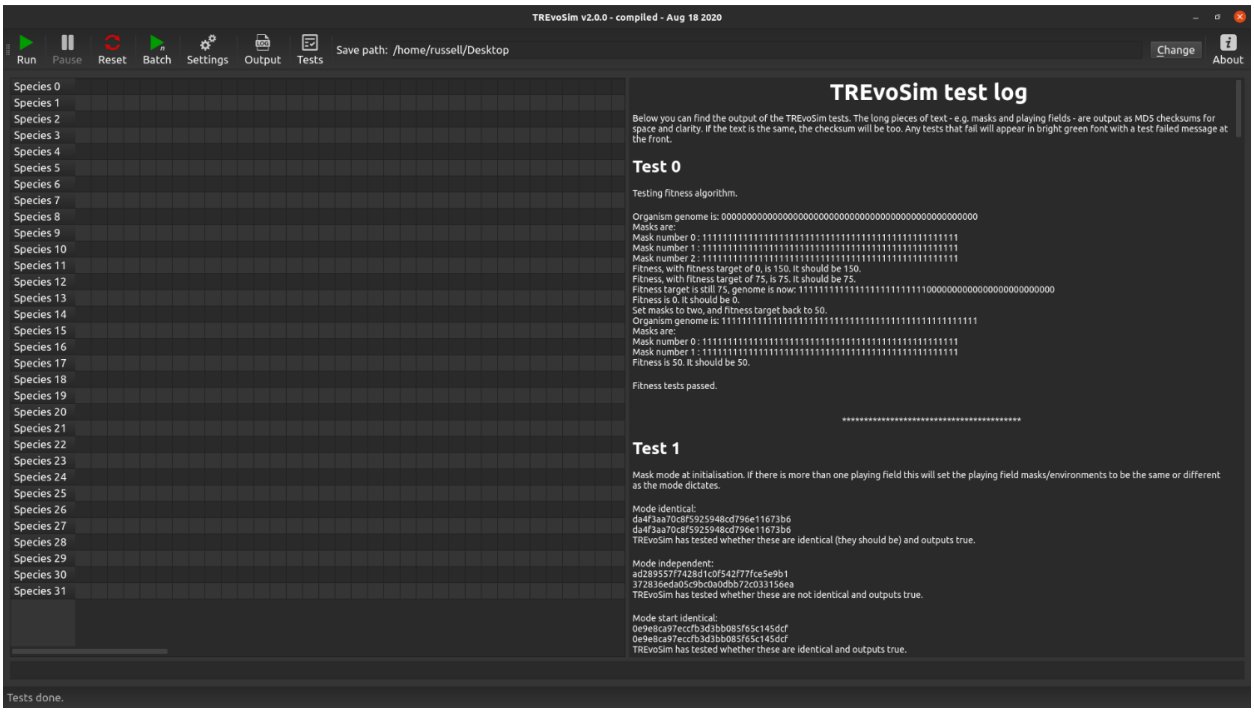
This menu command (or shortcut ctrl + shift + f) opens a dialogue that allows the strip uninformative factor to be set manually.

2.7.8 Random Seed

By default, the organism used to initialise a simulation is one near peak fitness for the starting environment(s), to prevent the resulting tree from documenting a lineage adapting to a fitness peak (this results in a highly asymmetrical tree). This menu command (shortcut ctrl + shift + r) toggles between this approach, and one in which a random individual is used to initialise the simulation.

2.8 Tests

TREvoSim has tests covering every function used in the simulation process. If you are working on the TREvoSim code, be aware that the tests are run as part of the build chain – when a test fails, so will the build. However, if you are a user, and would like to see the outputs of the tests, then this is possible by hitting the Test button on the toolbar menu, or Tests in the command menu. When this is selected, the main window is first split into two, and the software runs its test suite. The output of the results is then displayed on the panel on the right, shown below.



As such, it is possible to check TREvoSim is running as expected and described in the publications introducing the model (you can alternatively/additionally output a working log to check the internal workings of the software - see *Logging the Simulation*).

The output in the test log describes each test, and also the expected output. The system then makes sure the output meets these expectations. Failed tests will be highlighted in green.