CSC8391 Research Project for Bioinformatics

Stochastic, agent-based modelling of mtDNA dynamics in skeletal muscle fibres

William Whinn

Newcastle University

w.w.whinn2@newcastle.ac.uk

Abstract

Mitochondria are cellular organelles which provide the energy cells need to function. Mitochondrial DNA (mtDNA) diseases are caused by mutations within mtDNA and may affect any organ within the body with varying degrees of severity. They are currently untreatable. Recent research in this area focusses on understanding the expansion of mtDNA mutations within cells and the distribution of mutant mtDNA through replication, which may help to reveal potential avenues for treatment. Creating simple computer models of mtDNA population dynamics which include spatial effects will help to investigate ideas about mutant expansion such as the perinuclear niche effect. Implementing models including dynamic control of copy number will allow us to investigate the bottleneck effect for example. The agent based model outlined within this report implements spatial dynamics and population control, showing the effects that mutant mtDNA populations may have on a human skeletal muscle cell over an 80 year period. Written in the Julia programming language and using the Agents.jl package, the efficiency and simplicity of the Agent Julia model leans on the fast processing capability of the language to make it outperform models written using more traditional languages such as R or MATLAB. Results mirror those from the well-established Gillespie algorithm.

1 Introduction

Mitochondrial diseases are a diverse group of complex conditions characterised by a dysfunction in the transfer of energy to the cell caused by mutations in either mitochondrial DNA (mtDNA) or nuclear DNA (nDNA) which code for proteins (Steele et al., 2017; Alston et al., 2021). Mitochondria are responsible for a number of essential biological functions, most notably the transfer of energy in the form of adenosine triphosphate (ATP) to the cell through oxidative phosphorylation. Mitochondrial proteins are largely encoded in the nuclear genome and as such mutations occurring in mtDNA or in nuclear genes responsible for encoding mitochondrial proteins can lead to impaired function, particularly in systems which require large amounts of energy such as the brain, muscle tissues and nervous system (Finsterer, 2007; Saneto, 2017; Alston et al., 2021) The high degree of complexity and pathological heterogeneity that characterises mitochondrial diseases as a group is accounted for by the polyploid nature of mtDNA and its role in coding for mitochondrial proteins and high rate of mtDNA turnover (Alston et al., 2021). Onset of mitochondrial disease may occur at any point in the patient's life and may affect any organ, with multiple systems often affected (Saneto, 2017). Furthermore, phenotypic expression is highly variable, with mutations in the same protein complex resulting in different conditions in different patients (Nunnari and Suomalainen, 2012; Smith 2020).

The onset and progression of mitochondrial disease is determined by the increase, or clonal expansion, of mutated mtDNA molecules within the cell (Kowald and Kirkwood, 2018). A single cell may contain as many as several thousand mitochondrial genomes, although this varies depending upon the energy demands of the tissue (Krishnan et al., 2007). Where the cell contains mutated mtDNA (due to either point mutation in one base or deletion of entire sections of the genome), it exists in a state of heteroplasmy (Kowald and Kirkwood, 2018; Lawless et al., 2020). Once the cell reaches a high load of mutated molecules, mitochondrial dysfunction may occur, causing mitochondrial disease at any point in the patient’s lifespan as the molecules replicate and concentrations of mutated molecules evolve (Lawless et al., 2020). Studies of tissues taken from patients with mitochondrial disease show a distinctive ‘mosaic’ pattern in which levels of deficiency vary evenly across adjacent cells indicating that this process is a discrete stochastic one, occurring randomly at the level of individual cells (Piotrowicz, 2020).

2 Background

Current research focusses on understanding the clonal expansion of mutated mtDNA over wild-type in order to guide identification and development of more effective therapies and thus improve patient outcomes (Wallace and Chialkia, 2013; Kowald and Kirkwood, 2018; Lawless et al., 2020). This process is complex and as such poorly understood, particularly as evolution varies depending upon whether the mutation is inherited or not, the nature of the mutation (whether point or deletion) and cell type (Lawless et al., 2020). There is evidence to suggest that random genetic drift, whereby diversity is eliminated from the population through random sampling (Kimura, 1968), may account for clonal expansion in point mutations (Elson et al, 2001). Random genetic drift is, however, insufficient to explain clonal expansion in the case of mutations caused by mtDNA deletion and a range of ‘selective pressures’ that may influence this process have been proposed (Kowald and Kirkwood, 2013) ranging from smaller (i.e. mutated) genomes replicating faster or having a longer half-life than healthy molecules (Wallace, 1992; de Grey, 1997) to transcription processes that may actively select mutant molecules for replication and which may vary across cell types (Kowald and Kirkwood, 2018; Trifunov, 2018).

Significant progress in understanding the causes and progression of mitochondrial disease has been achieved through mathematical and computational modelling of clonal expansion. In particular, these methods have proven to be promising in overcoming many of the difficulties posed by observing a dynamic, stochastic process that occurs heterogeneously across cells (Lawless et al., 2020).

Successful examples of modelling this process include the model by Elson et al. (2001), which was tailored toward exploring relaxed replication within non-dividing cells over a human lifespan of 120 years. This study also included the potential for mutant mtDNA molecules being introduced at random throughout a patient’s lifetime. An earlier paper (Chinnery and Samuels, 1999) explores the variety in heteroplasmy with regards to the level of mutants and how their population crossing a critical threshold will trigger a cell to ‘express’ a defect. In exploring the heteroplasmy and limits of population copy levels, a theoretical model created by Johnston et al. (2015) explores the mtDNA ‘bottleneck effect’, which is the mechanism by which cell heteroplasmy is maintained. Comparing the results against actual genomic data from a mouse, Johnston et al. demonstrate that this bottleneck effect does in fact exist and is flexible in how it controls population copy numbers. These phenomena and observations will act as a central pillar for the development of the simulation engines presented in this paper.

Although these models contain many required elements such as temporal dynamics and rudimentary population control, there is a common factor lacking within each of them up to this point: the concept of ‘space’. Spatial dynamics are important because molecules are able to move through three-dimensional space in nature. To address this limitation, more modern implementations of these types of models will typically be based on agents. Agent-based modelling allows a series of 'agents' to interact with one another in zero, one, two, or three-dimensional space, an agent in this case being an mtDNA molecule. A study by Dalmasso et al. (2017) uses an agent-based model to propose that the effect of an environment on its constituent mtDNA molecules is a factor in changes in homeostasis, meaning that spatial dynamics are relevant when exploring mtDNA population dynamics. Because mutation levels affect cell function, particularly during ageing, an agent-based model by Hoffman et al. (2017) was developed to explore this problem in detail. The outcome of this research was a model which could accurately predict the effects of ageing on mtDNA which were comparable to real-world data. In 2019, a paper by Maldonado et al. explains that there are many functions available to combine for the creation of a model such as temporal and spatial dynamics; the boom in omics and big data are partially driving these advances. Although this paper briefly explains the advantages of agent-based modelling as a factor in these advances, it also explains that an interdisciplinary approach, combining the fields of mathematics, programming, and biology will be responsible for creating more and more accurate models. This may, for example, render obsolete the need to surgically cut tissue from a patient for analysis and provide more avenues for developing potential treatments. An example of this combination of functions may be demonstrated by the PhysiBoSS application, which explores the reaction of cells in response to TNF treatment using an agent-based model for multicellular behaviour and boolean modelling for each individual cell. This presents the possibility of creating more complex models, although this model demonstrates the creation of cells as an agent, it is theoretically possible to also model mtDNA within those cells, which would be able to move at their own rate, independent of the cells outside depending on the type and location of said cell. This concept is inferred throughout the Kaul and Ventikos paper (2015), stating that these systems are typically non-linear in nature and agent-based modelling is arguably the most elegant way of simulating a system and its varying layers of complexity.

This level of complexity comes with a high computational price however, as more complex behaviour has a direct hit on performance for even simple simulations. To address this situation, a programming language called Julia was developed. Although it is a universal programming language, Julia is popular in scientific fields because of its efficiency and relative simplicity. Languages such as R and Python have one advantage over Julia at the time of writing, and that is the number of packages available for data science and machine learning. A report by Gao et al. (2020) breaks these considerations down in great detail. The performance of Julia is also explored, comparing its performance with well-known machine learning algorithms such as the Support Vector Machine (SVM) and Deep Learning and in all categories explored, Julia proved to perform quicker than languages such as MATLAB, R, Python, Go, and Fortran. Julia is steadily improving over time, becoming faster and more efficient, but there are examples of Fortran performing better than Julia which have since been addressed (Gevorkyan et al, 2019).

This project will explore the implementation of a simple agent-based model written in Julia, exploring both temporal and spatial dynamics of mtDNA populations over the lifespan of 80 years. This model will simulate a human patient with an inherited mutation within a skeletal muscle cell. To confirm the validity of the agent-based model, a gillespie model will form the basis for implementation and comparison.

3 Methods

In this project I created a model to explore the dynamics of mtDNA populations in single cells over a period of 80 years, corresponding to a human lifespan. The first implementation of the model is based on the Gillespie stochastic simulation algorithm (SSA) and is intended to serve as a reference point in the creation of the Agent Julia implementation. By carrying out simulations based on an established and proven algorithm, the results can be compared to verify their accuracy. Both models have been written using the high-performance Julia scientific programming language. Julia is free-to-use and open-source, capable of being used on platforms such as Microsoft Windows, Apple MacOS, and Linux. The full source code, documentation, and example results for this project may be found at: https://github.com/ultraviolet-1986/agent_julia.

3.1 Simulating Patients

Both the Gillespie and Agent Julia simulation engines simulate a human patient with a pre-existing mtDNA mutation. The area of the body being modelled is a small, cylindrical section of a skeletal muscle cell. Both the Gillespie SSA and Agent Julia will record the number of each species of molecule at each recorded time point, this is referred to as the mtDNA copy number. This count lists the number of copies of a species at work within a cell at a given time. This is especially important as these counts form the basis of each dataset generated per-simulation and per-program and will help to show the simulation results overall. Without a known copy-count, it would be impossible to know the changes within heteroplasmy throughout the patient's lifespan. These copy levels are set within both models to not surpass 200 molecules initially, but throughout the simulation, a maximum of 250 molecules are supported. Both models contain a dynamic population control mechanism to ensure the population will not overpopulate or go extinct, maintaining stable copy levels throughout the simulation runtime.

3.2 Simulation Initial Conditions

At the beginning of any simulation, the patient will have an inherited 50 mutant molecules and 150 wild-type molecules, creating an initial mutation load of 25%. The initial conditions for the simulation are recorded toward the top of each model file and they can be redefined to increase the initial copy levels of molecules (section 3.1), for example, it is possible to set the initial mutation load to 50% by defining the initial mutation count to 100 and lowering the wild-type population to 100 molecules.

3.3 Modelling Biochemical Reactions

The Gillespie SSA and Agent Julia instances simulate biochemical reactions and both models perform replication and degradation for both wild-type and mutant mtDNA by using the following formulae below.

Reaction	Formula	Propensity
Replicate wild-type mtDNA	$W \xrightarrow{rW} 2W$	`[1, 0]`
Degrade wild-type mtDNA	$W \xrightarrow{dW} \varnothing$	`[-1, 0]`
Replicate mutant mtDNA	$M \xrightarrow{rM} 2M$	`[0, 1]`
Degrade mutant mtDNA	$M \xrightarrow{dM} \varnothing$	`[0, -1]`

Table 1: Propensity table for calculating reactions.

The propensity of each reaction is a small vector containing an update for wild-type and mutant molecule copy numbers respectively. The wild-type replication formula for example shows a propensity of [1, 0]. This means that, during this reaction, one molecule of wild-type mtDNA is being added to the population and no additional molecules of mutant mtDNA are added. In terms of removing a molecule from the copy count, this propensity is described as [0, -1] for the reaction degrading mutant mtDNA for example. This shows that no wild-type molecules are added or removed from the population and one mutant molecule is removed.

3.4 Gillespie SSA Model

This model contains a series of parameters which dictate how the simulation will proceed. These parameters include rates for replication, degradation and mutation, along with the initial levels of wild-type and mutant mtDNA molecules. These values are defined in-code and may be adjusted to suit a multitude of patient types (section 3.1).

3.5 Agent Julia Model

The Agent Julia model makes use of the Agents.jl (Datseris et al., 2021) package to create a model where agents are free to move organically around a two-dimensional grid-space. An agent in this case refers to a single mtDNA molecule and the grid-space represents the cell in which the mtDNA exists.

This continuous-space has been sized to resemble a muscle cell and agents are free to move within. This model may be configured in the same way as the Gillespie model, using the same definitions for temporal units, kinetic rates, and initial mtDNA levels.

3.6 Simulation Runtime and Data Collection

Both models have a target simulation runtime of ~80 years to mimic a human lifespan, and all temporal units and kinetic rates have been adjusted to match previous work (Morgan, 2020). By default, each of the models will make use of the same kinetic rates for mtDNA replication and degradation, but these variables are defined independently to ensure some readability within code and to allow changes to the model should new knowledge emerge regarding these rates. Data are collected every month by default, this means that there are a total of 960 epochs per simulation (80 years × 12 months = 960 steps). This is achieved by the introduction of a variable ‘δ’ (delta). By changing the value attached to this variable, it is possible to collect more data. Note that this is the only change required by the user if they wish to change the amount of data recorded, both models will dynamically adapt to and use new rates and data-collection steps and functionality will not be impaired in any way.

An example of capturing more data is to adjust the value of delta to a fortnight instead of a month (for example), this would double the time-points, creating 1,920 points of data as opposed to 960. Although this can be adjusted to any time value to create more detailed results, it may also introduce issues relating to memory and drastically increasing overall simulation runtime. It is not feasible to set the value of delta to a second or a minute because the simulation will record a vast amount of data-points, making the simulation take much longer or crash outright. This memory management factor is magnified by the inclusion of simulation loops; by default, both models will perform a thousand stochastic simulations, which are used to create statistical summaries of stochastic behaviour. If the delta variable is set to record a larger amount of data, the length of time a simulation is performed will be multiplied by a factor of 1,000. The functions within the model files themselves are set to perform only a single simulation, but providing an integer in the form of an argument will allow multiple simulations to be performed, each set of results will be returned to the user and written to CSV file(s).

3.7 Capturing Ensemble Results

The stochastic nature of each simulation ensures that results from one simulation to another will look radically different, regardless of which model performed the simulation. Because of this, it is difficult to capture a big-picture overview of observations and potential trends. It has been possible to overcome this limitation by implementing the concept of simulation repetition. Both models return the results of their simulation and so by placing the initiation code within another function, it was possible to set the number of repeats and where to save each simulation’s individual results. This creates an ensemble of results, from which it is possible to calculate results across any number of simulations (section 4).

The Gillespie SSA model will save all of its results to a 3-dimensional matrix whilst the Agent Julia model makes use of the DataFrames.jl package. By using data frames in the Agent Julia model, it is possible to record data in a format more suitable for exporting to comma-delimited files (CSV). In a data frame, a field of `mutant_count` at step 100 may be accessed by using the command `data.mutant_count[100]` as opposed to the language of a matrix, which requires the specific axis, range, and step, making it difficult to read for large amounts of data.

4 Results

The stochastic nature of each simulation means that it is impractical to compare results from a single simulation, both per-model or per-simulation from the same model. By making use of the ‘simulation looping’ concept outlined in section 3.7, it is possible to record an ensemble of results, allowing for the calculation of the overall copy levels (section

4.1) for 1,000 simulations, or the proportion of mutant molecules (section 4.2) for 1,000 simulations, both taking place over 80 years.

4.1 Ensemble mtDNA Copy Levels

In both the Gillespie SSA and Agent Julia models, it is possible to see that in all 1,000 simulations, there is a large amount of variety but the median copy level stays roughly within the pre-established boundaries explained earlier. Note that both models initial conditions are set not to rise above, or fall below, the specified boundaries of between 250 and 150 molecules. This means that the total initial count is the level which the population remains at on average throughout the full lifespan of the simulations.

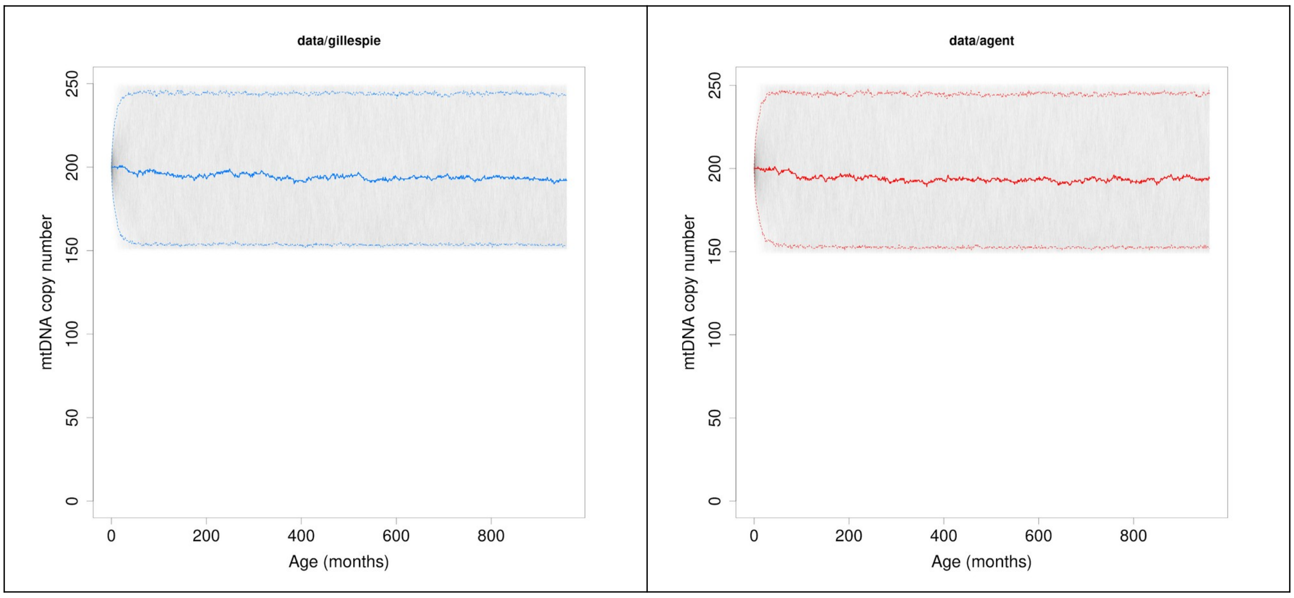

Figure 1: Statistical summaries for ensembles of 1,000 stochastic simulations, Gillespie SSA (left) and Agent Julia (right). Median (50th percentile) counts are shown by a thick line with upper and lower quantile counts shown by dotted lines.

The total copy count does not reflect the individual populations of wild-type and mutant mtDNA and section 4.2 and 4.3 will cover this. These results show that technically speaking, the kinetic rates and dynamic population control are fully functional and actively affect the total population of molecules within the modelled biological system. Biologically speaking, these results are consistent with other phenomena in nature such as the bottleneck effect, ensuring that there are never too many or too few molecules active within the cell at any one time, maintaining a natural equilibrium which the cell requires for continued function.

4.2 The Effects of Mutation Load Variability

By summarising the mutation load across all 1,000 simulations, it is possible to discover a consistent trend in the overall proportion of mutant mtDNA molecules. As in section 4.3, results between models (figure 2) are remarkably similar and show a trend toward a declining mutant population over the simulation runtimes for both models. This is likely due to the patient's inherited mutation being only a quarter of the total population. This being the case, the mutant population steadily declines over time as the wild-type load steadily rises to take its place. There is a large amount of variety however, and there are simulations recorded which demonstrate that the mutant population has, in cases, reached a 100% mutation load before the end of the simulation(s).

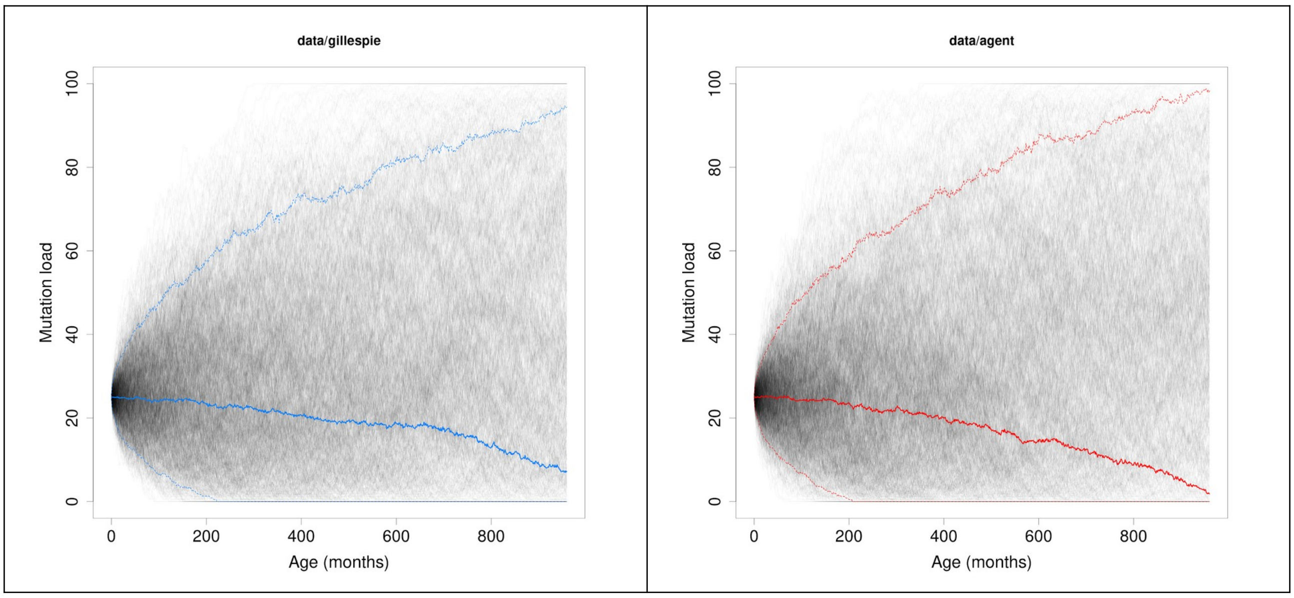

Figure 2: Statistical summaries for ensembles of 1,000 stochastic simulations, Gillespie SSA (left) and Agent Julia (right). Median (50th percentile) proportions are shown by a thick line with upper and lower quantile proportions shown by dotted lines. Mutation load set to 25%.

In technical terms, both models display the same results, giving them an air of credibility in spite of their radically different implementations in-code. Biologically, this is consistent with clonal expansion replicating mutant mtDNA molecules throughout the lifetime of the simulation, distributing mutants at the same rate as the wild-type molecules, helping the population to survive as long as possible, but ultimately trending toward extinction. Patients with a larger than 50% mutant inheritance will see a reverse in this trend as it is the dominant population (figure 3). In figure 3, the initial counts of molecules have been reversed, setting the initial mutation load to 75%, showing a steady increase at the same rate as its decline in figure 2. Conversely, a mutant population of exactly 50% shows both species surviving in equilibrium (figure 4) as both populations have an equal chance at survival and extinction, this is more likely due to de novo mutation being deactivated within the Gillespie model and unimplemented within the Agent Julia model.

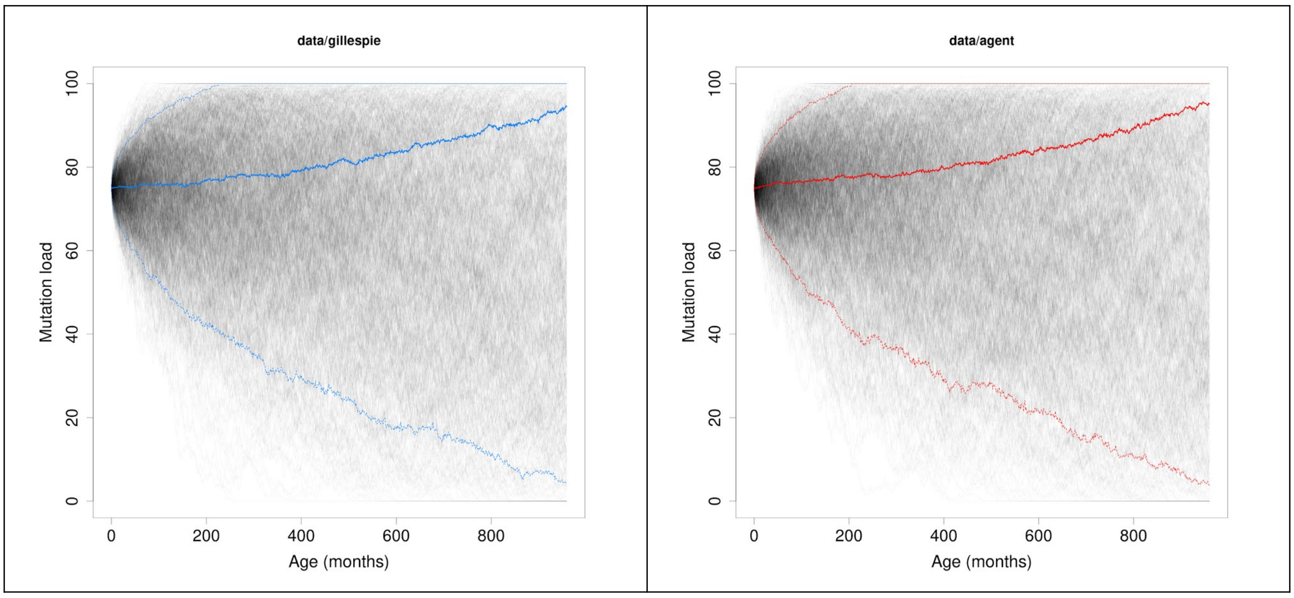

Figure 3: Statistical summaries for ensembles of 1,000 stochastic simulations, Gillespie SSA (left) and Agent Julia (right). Median (50th percentile) proportions are shown by a thick line with upper and lower quantile proportions shown by dotted lines. Mutation load set to 75%.

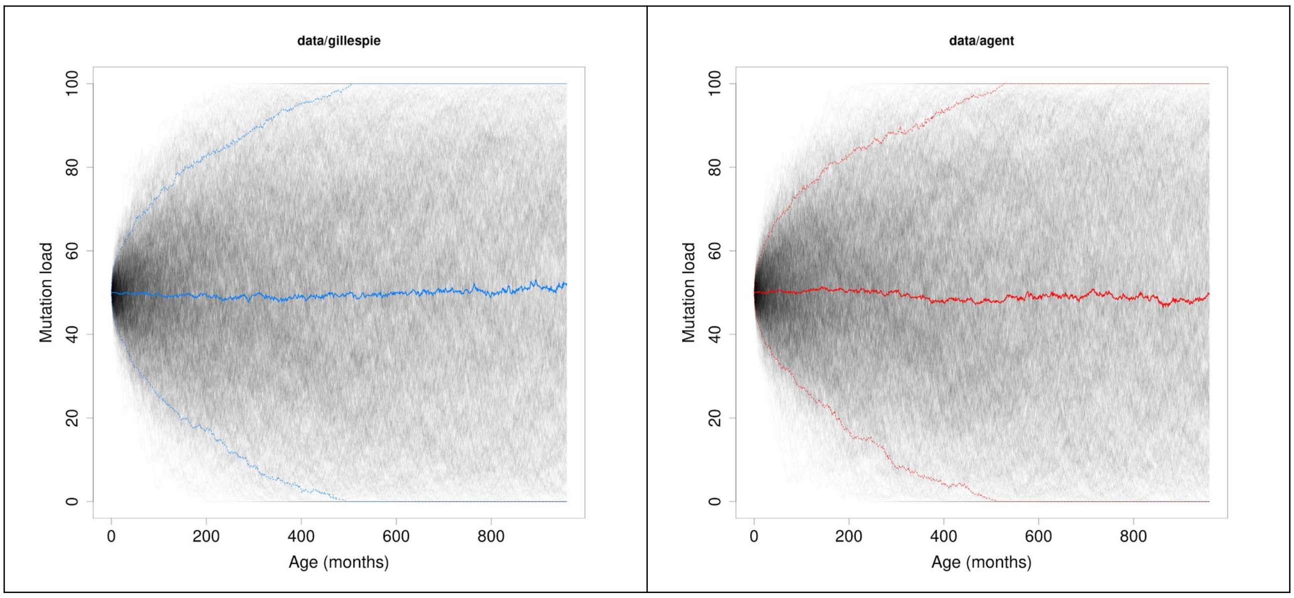

Figure 4: Statistical summaries for ensembles of 1,000 stochastic simulations, Gillespie SSA (left) and Agent Julia (right). Median (50th percentile) proportions are shown by a thick line with upper and lower quantile proportions shown by dotted lines. Mutation load set to 50%.

4.3 Agent Julia: Video Simulation

It is possible to generate a video simulation of molecules interacting over the length of the simulation. The importance of implementing space within these models may help to show the behaviour of mutant colonies and how they react to the presence of other organelles should they form. Phenomena such as the perinuclear niche effect (Vincent et al. 2018) may be explored with further modification of the Agent Julia model because it is possible to include other organelles as 'structs' within model code. This may increase the complexity and runtime of a simulation, but may also provide valuable insights into the behaviours of organelles and molecules within space. The shape of the space has been adjusted so that the spatial component of the model can be seen visually and not only the results the model generates. Note that this is the result of a single Agent Julia simulation and it is not currently possible to create an ensemble of results because of the magnitude of data generated for this simulation including molecule movement and velocity, this is akin to a game of Chess and how many different games are possible. It is possible to see however that the simulation does show the full target runtime of 80 years and during that time, the population of wild-type and mutant can rise and fall in a controlled yet stochastic manner (section 3.7). This simulation also demonstrates that each molecule is placed evenly, yet randomly throughout the cell. This shows that the model assumes ‘perfect’ mixing conditions and the simulations will not contain colonies or groups toward the beginning of the simulation and the model will show how populations can replicate and degrade within this context over the lifetime of the simulation and the patient.

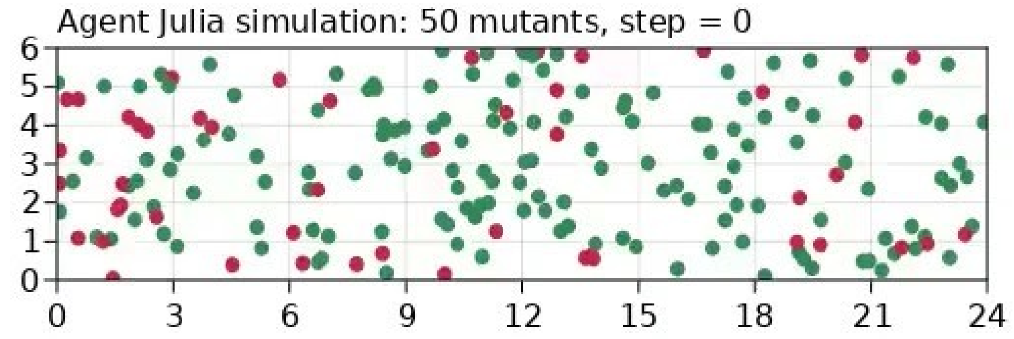

Figure 5: Initial molecule placement per-species for individual Agent Julia simulation render. Green molecules indicate wild-type mtDNA and red indicate mutants.

Figure 5 shows molecules of two differing colours, the green represents wild-type mtDNA whilst red molecules indicate mutant mtDNA. Because the screenshot shows step zero, it is possible to see that there are a total of 50 red molecules at the beginning of this simulation, showing that the model is populated correctly. This top-down view also shows a reason why the simulations take so much time for this particular model (section 4.4).

4.4 Model Performance

By 'piping' the model scripts through the BASH command 'time', it is possible to get exact simulation runtimes for both models. At present, the Gillespie SSA model performs 1,000 simulations in ~20 seconds, including time to write each simulation to an individual CSV file. The Agent Julia model however, makes use of many different libraries, including Agents.jl, meaning that this model leans on existing work which may not be as optimised as a from-scratch implementation; for example, the Gillespie model requires no additional libraries beyond basic mathematical utilities, and the Agent Julia model requires Agents.jl and its dependencies such as 'CairoMakie', which is used to render a simulation to video. This brings the total runtime of the Agent Julia model (excluding rendering to video) to ~4:20 for 1,000 simulations, and writing their respective CSV file. This is true of Julia 1.6.3 on a six-core Intel i5-8400 CPU with 16Gb of RAM under Linux. Whilst Julia is a language with extensive parallelisation capabilities, allowing the use of multiple cores, this functionality has not been implemented in the models at the time of writing.

4.5 Gillespie SSA or Agent Julia?

As shown throughout this section, the Gillespie SSA and Agent Julia models both show remarkably similar results and it is therefore difficult to decide which of the two displays the most accurate results in terms of copy numbers and mutant mtDNA proportion. Although the Gillespie SSA model is built on proven mathematical concepts, the Agent Julia model is able to take into account the concept of space and actual population dynamics, making it functionally more organic when compared to its counterpart. This helps to investigate a greater variety of phenomena such as the bottleneck effect, and the perinuclear niche effect for example; all of which require the concept of space to explore. By doing so, the mechanisms of clonal expansion may be better understood.

Instead of considering which model is most accurate, it is possible to use whichever model suits an area of study best. It is possible to update both models' code to introduce new functionality or update old functionality in the event that more accurate replication, mutation, and degradation rates become known over time. In the context of mtDNA population dynamics, Agent Julia shows promise in that it is capable of showing these dynamics and phenomena visually within plots and videos thanks to the Agents.jl package, helping to visualise what may happen to either population over an 80 year period, showing each individual molecule's movements and lineage. It is possible to query an individual molecule's movements and their species and mass at any given time-point. At present, only the raw copy numbers are used as part of a simulation's results and in terms of generating raw mtDNA copy counts, the Gillespie SSA model out-performs Agent Julia by a factor of 10, making it more appropriate for data which is not spatial or does not require tracing a specific molecule's lineage. Because the Gillespie model was built to serve as a foundation for the Agent Julia model, and Agent Julia displaying similar behaviour and showing almost identical results, the viability of an agent-based model for this research has been proven successful and the use of Julia as a base programming language is shown to be viable and even practical for complex scientific problems.

5 Discussion

Both population models contain mechanisms to allow for replication and degradation of wild-type and mutant mtDNA molecules. Each model's data output will resemble each other in this regard, providing a set of discrete molecule counts per-step and per-species according to the delta variable ('δ'), which dictates the number of counts (rows) saved per-simulation. The model's ability to replicate and degrade molecules shows that these processes can maintain both a wild-type and mutant population simultaneously for the lifetime of a patient. Both replication and degradation occur at the rate of 'λ' (lambda), ensuring that no one molecule exists for an unrealistic amount of time. The dynamic population control mechanism keeps an equilibrium between both populations and their respective copy counts throughout any number of simulations, this also helps to validate the temporal units and kinetic rates found within the Morgan (2020) Gillespie model. The results and trends shown were arrived at primarily due to the simulation repeating mechanism, the outcome of a thousand simulations and their calculated median values provided a bird's eye view to a vast amount of data. Because each simulation was repeated 1,000 times, each model was able to generate a thousand similar-to-identical cells, recording 80 years of data in potentially a matter of seconds, an approach which would be impractical and almost impossible in the real world.

The original purpose of this project was to create a Gillespie SSA model for exploring mtDNA population dynamics within a section of a skeletal muscle cell using the Julia programming language, and then use that model as a stepping stone for the creation of an agent-based model. This model allows the implementation of more complex operations which more closely mirror how these reactions occur within nature. At present, both models are set to use exactly the same kinetic rates for reactions and temporal units for representing the passage of time within the simulation. These kinetic rates and temporal units were taken from the Gillespie model created by Morgan (2020) in MATLAB. This has the benefit of introducing variables within the Julia-based models which have been thoroughly researched and are considered reliable at the time of writing. This being the case, each kinetic rate has been set to match their temporal unit counterparts and so changing one or other will likely introduce instability to whichever model has been modified within this regard.

Although the Agent Julia project has been successful in creating an agent-based counterpart to the Gillespie SSA model, there have been issues with regards to speed and memory-usage. For example, one simulation was left to run for a total of five hours before it was forced to terminate. The Julia debugger, though verbose served to only show the line in-code where an error was detected, but proved frequently unreliable in explaining why or how an error occurred, and this output would often include a trace back through a dependency package's source code, making the process of debugging difficult and time-consuming when compared to languages such as Python or R. Deciphering this output is possible with practice, but requires the perception of an experienced developer in spite of the overall simplicity of the language, making it a potential hurdle for researchers who focus more on the mathematical or biological considerations of a system.

Memory usage was optimised in Agent Julia by reducing the amount of functions required to loop a simulation, overwriting variables to adjust their structure instead of working with copies, and resetting the model definition per-simulation. Other previous attempts at memory reduction included reducing all kinetic rates and temporal units by a factor of one million and forcing the use of e-notation numbering as opposed to decimal. This reduction was called ‘α’ (alpha) and could only be found within the Agent Julia model. The approach was to initialise it with a value of ‘1e6’ (one million) and then divide the first temporal unit, the definition of ‘day’, by alpha. This means that Julia’s memory held a variable of ‘1e6’ instead of ‘1000000’. Because all other temporal units were calculated based on this variable, all values remained equivalent but their overall size was reduced by alpha. This produced the same results as before, and comparable with the Gillespie SSA model, but with a smaller memory footprint and a slightly quicker runtime than was possible before. All kinetic rates and temporal units were still equivalent to one another, but further corrections and adjustments removed the necessity for this reduction and so ‘α’ was ultimately removed. Removing ‘α’ kept Agent Julia’s results exactly inline with the Gillespie SSA, but this concept presents data compression as a potential area of research should the model need to record more data in future.

Whilst the Gillespie SSA model performs much quicker than the Agent Julia model, the Agent Julia model supports a much broader range of operations and concepts that allow for project expansion and reduction depending on the scope and focus of research. Both models perform their tasks reliably, but a subtle improvement for each model could be to introduce concurrency functionality into the source code. As Julia is a language tailored toward this functionality, improvements to simulation runtimes are possible and feasible to those with a deeper understanding of the language. Julia’s script-based design also lowers the bar to entry for future researchers and it is easy to see at a glance what action is being performed and when and why. All code has also been commented where appropriate to facilitate this. The Agent Julia project and its full source code are also GPLv3 licensed, requiring only a citation from the new user to use legally, meaning the project, like the Julia language, is also fully open-source and free-to-use.

In terms of future research, the Agent Julia model presents the most interesting starting point. The model already includes the concept of space, and although it is currently possible to convert the two-dimensional space into fully three-dimensional, this functionality is currently in testing for the Agents.jl package and so was not further pursued at the time of writing. Implementing this would require some fundamental changes to how ‘space’ is defined and how molecules are to move throughout this new environment. An example of this change could be to add a new property to the molecule definition such as ‘attitude’; like spacecraft or a submarine, each molecule must have a position in space and a direction to travel at that trajectory, this means that the ‘velocity’ property can be adapted or replaced depending on the use-case to fit this new paradigm.

By making use of other functionality within the Agents.jl package, it is also possible to render the act of replication by binary fission explicitly. This would be mostly an aesthetic change, and will likely not influence the results of the simulation but could serve as a great visual aid when rendered in three dimensions as a video for example. At present, this concept is implemented in that a daughter molecule will overlap with the parent and then both will travel in their own stochastically-determined velocity. This is to imply that there is some effect of three-dimensional space within the limitations of the model. Another factor in this regard is that molecules may travel across the edges of the grid space only to appear on the direct opposite side. This is not an error or a bug, but an attempt to simulate a cylindrical space within the confines of the current 2D environment.

Because of the possibility of introducing new properties to the molecule object, another subject of future research could be to apply a unique identifier to each new molecule which is generated randomly, with an additional field for storing which molecule has which parent when replication occurs. This would enable the possibility of recording the lineage of each molecule throughout the lifetime of the simulation. This would increase the amount of data generated by several orders of magnitude and so a separate function may need to be included to run this specific simulation scenario only once as opposed to running the simulation multiple times.

References

Alston, C., Stenton, S., Hudson, G., Prokisch, H., Taylor, R. (2021) ‘The genetics of mitochondrial disease: dissecting mitochondrial pathology using multi-omic pipelines’, *The Journal of Pathology,* 254(4), pp. 430-442.

Chinnery, P., Samuels, D. (1999) ‘Relaxed replication of mtDNA: A model with implications for the expression of disease’, *American Journal of Human Genetics,* 64(4), pp. 1158-1165.

Dalmasso, G., Zapata, P., Brady, N., Hamacher-Brady, A. (2017) ‘Agent-based modelling of mitochondria links sub-cellular dynamics to cellular homeostasis and heterogeneity’, *PloS ONE*, 12(1).

Datseris, G., Vahdati, A.R., and DuBois, T.C. (2021) 'Agents.jl: A performant and feature-full agent based modelling software of minimal code complexity'. [Preprint]. Available at: https://arxiv.org/abs/2101.10072 (Accessed: 26/04/2021).

de Grey, A. (1997) ‘A proposed refinement of the mitochondrial free radical theory of aging’, *BioEssays* 19, pp. 161-166.

Elson, J., Samuels, D., Turnbull, D., Chinnery, P., (2001) ‘Random intracellular drift explains the clonal expansion of mitochondrial DNA mutations with age’, *Am J Hum Genet.* 68(3), pp. 802-806.

Finsterer, J. (2007) ‘Hematological Manifestations of Primary Mitochondrial Disorders’, *Acta Haematol.* 118 pp. 88-98.

Gao, K. *et al.* (2020) ‘Julia language in machine learning: Algorithms, applications, and open issues’, *Computer Science Review,* 37.

Gervorkyan, M., Demidova, A., Korolkova, A., Kulyabov, D. (2019) ‘Statistically significant performance testing of Julia scientific programming language’, *Journal of Physics Conference Series,* 1205.

Hoffman, T., Barnett, K., Wallis, L., Hanneman, W. (2017) ‘A multimethod computational simulation approach for investigating mitochondrial dynamics and dysfunction in degenerative aging’, *Aging Cell,* 16, pp. 1244-1255.

Johnston, I. *et al.* (2015) ‘Stochastic modelling, Bayesian inference, and new in vivo measurements elucidate the debated mtDNA bottleneck mechanism’, *eLife,* 4.

Johnston, I., Jones, N. (2016) ‘Evolution of cell-to-cell variability in stochastic, controlled, heteroplasmic mtDNA populations’, 99(5), pp. 1150-1162.

Kaul, H., Ventikos, Y. (2013) ‘Investigating biocomplexity through the agent-based paradigm’, *Briefings in Bioinformatics,* 16(1), pp.137-152.

Kimura, M., (1968) ‘Evolutionary rate at the molecular level’, *Nature,* 217, pp. 624-626.

Kowald A, Kirkwood T. (2013) ‘Mitochondrial mutations and aging: random drift is insufficient to explain the accumulation of mitochondrial deletion mutants in short-lived animals’, *Aging Cell,* 12(4), pp. 728-731.

Kowald A, Kirkwood T. (2018) ‘Resolving the enigma of the clonal expansion of mtDNA deletions’. *Genes* 9(3).

Krishnan, K., Greaves, L., Reeve, A., Turnbull, D. (2007) ‘The ageing mitochondrial genome’, *Nucleic Acids Research,* 35(22), pp. 7399-7405.

Lawless, C., Greaves, L., Reeve, A., Turnbull, D., Vincent, A. (2020) ‘The rise and rise of mitochondrial mutations’ *Open Biology,* 10(5).

Letort, G. *et al.* (2018) ‘PhysiBoSS: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling’, *Bioinformatics*, 35(7), pp. 1188-1196.

Maldonado, E., Taha, F., Rahman, J., Rahman, S. (2019) ‘Systems biology approaches toward understanding primary mitochondrial diseases’, *Front. Genet*.

Morgan, J. (2020) ‘Mitochondrial DNA copy number control in single human cells is asymmetric’ [Unpublished Master's Thesis] Newcastle University.

Nunnari, J., and Suomalainen, A., (2012) ‘Mitochondria: In Sickness and in Health’ *Cell,* 148:6 pp. 1145-1159.

Piotrowicz, M. (2020) *Mitochondria and clonal expansion.* Available at: http://mito.ncl.ac.uk/clonexp/clonal_expansion/ (Accessed: 22rd July 2021).

Saneto, R. (2017) ‘Genetics of Mitochondrial Disease’ *Advances in Genetics,* 98, pp. 63-116.

Smith, M. (2020) *Gene Environment Interactions: Nature and Nurture in the Twenty-First Century*, London: Academic Press.

Steele, H., Horvath, R, Lyon, J., Chinnery, P. (2017) ‘Monitoring clinical progression with mitochondrial disease biomarkers’, *Brain*, 140(10), pp. 2530-2540.

Trifunov, S. *et al.*, (2018) ‘Clonal expansion of mtDNA deletions: different disease models assessed by digital droplet PCR in single muscle cells’, *Scientific Reports,* 8.

Vincent, A. (2018) ‘Subcellular origin of mitochondrial DNA deletion in human skeletal muscle’, *Ann Neurol.* 84(2), pp. 289-301.

Wallace, D. (1992) ‘Diseases of the mitochondrial DNA’, *Annual Review of Biochemistry,* 61, pp. 1175-1212.

Wallace, D., Chialka, D. (2013) ‘Mitochondrial DNA genetics and the heteroplasmy conundrum in evolution and disease’, *Cold Spring Harb Perspect Biol.,* 5(11).