A Tool for Visualising Cell Model Results
MInf Project Phase 2

James Hulme

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2014

Abstract

This will be the abstract

Acknowledgements

Acknowledgements go here.

Table of Contents

Chapter 1

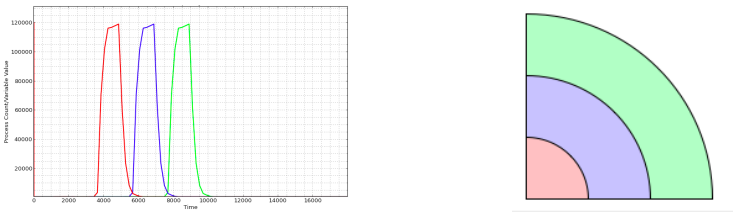
Introduction

Biologists often use computer models to help guide their research as modelling is much cheaper than experimentation. There are a number of tools available for biological modelling. These tools typically require a certain level of numerical confidence to create and interpret. Not all biologists have this numerical confidence. Some researchers find writing and interpreting models a challenge; this can make them less effective in their research. It is therefore necessary for the tools they use to help them relate the data to their field by incorporating domain knowledge.

One such tool that can be used for modelling is Bio-PEPA [1], an extension of the PEPA process algebra. Bio-PEPA is currently implemented as a plugin for the Eclipse IDE. Bio-PEPA visualises the model results as time-series graphs. There is one team of Src researchers in particular, who use Bio-PEPA and do not have the numerical confidence, as described above, to be comfortable using Bio-PEPA. This team was the original focus for the project. The purpose of this project was to extend Bio-PEPA's visualisation capabilities to allow the previously mentioned team, and other similar users of Bio-PEPA, to more effectively analyse their results.

A significant problem with Bio-PEPA's visualisation capability is that it is difficult to represent spatial change on a time-series graph (as can be seen in Figure 1.1). In Bio-PEPA you can have a species at different locations in the cell, for example, next to the nucleus, next to the cell membrane and throughout the cytoplasm. The movement of this species through the cell can by modelled by seeing the population of it in each location over time. This is difficult to visualise on a time series plot as three lines are too abstract. It requires the use of a biological metaphor for spatial information to be easily interpreted. In this case using a visualisation based on a cell can more intuitively show how the species moves through the cell. It is this type of inference that the Src researchers find challenging to do with Bio-PEPA currently.

Over the course of the project the scope has been expanded. The original objective was to assess which forms of visual representation are most helpful and informative to laboratory science. At the end of the first project phase the objective changed (to reflect that the project was about delivering a finished program and not about the results of many experiments) to be to develop a tool to visualise the results of dynamic



(a) Time Series Representation (b) Spatial Representation

Figure 1.1: One species at three locations in the cell represented traditionally on a time series graph and also spatially in a cell

time-series models of intra-cellular behaviour based on biochemical reactions. This objective was focused on visualisation to aid those researchers who are not numerically confident. The second phase of the project has added to this objective to also aid interpretation and collaboration. This change in objective is to make the tool better for all users. To reflect these changes the focus on the original Src team has been generalised. Evaluations are performed with other potential users who are also biologists.

1.1 Where Does This Tool Fit In?

In the first stage of the project a review was performed of the features of a number of modelling and visualisation tools. This review included specialised software aimed at biologists and general software for anybody doing data visualisation.

As the project scope expanded to include goals not specifically related to visualisation it was prudent to perform another software review covering the new features, in particular software that allowed for collaborative editing. It is important to see what features are commonplace, which features are not commonplace but are useful, and which features are not useful. This analysis would then be used to guide development of the collaborative features of the new tool.

Four products were chosen for review: Google Drive [2], Pidoco [3], Lucid Chart [4] and SynBiowave [5]. Analysis of these software can be found below.

1.1.1 Previously Reviewed Software

The software packages reviewed at the start of the project were: Bio-PEPA, Uppaal [6], V-Cell [7], Cell-O [8], Copasi [9], Cell Designer [10] and WEAVE [11]. Bio-PEPA, V-Cell, Cell-O, Copasi and Cell Designer have been written for biological modelling. Uppaal is modelling software for general use, and WEAVE is a general data visualisation tool.

All offered some level of visualisation, some simply graphs, others more complex visualisations. Bio-PEPA offered only line graphs. Uppaal had visualisations that

highlighted where in a finite state machine view of the model the current state is. V-Cell had visualisation of the model in a hierarchical set of circles. It could also display spatial elements of the results data by displaying a heat model view of the cell. Cell-O was aimed more at multi-cell models and was able to show them moving and splitting; it also had visualisation of the model as a finite state machine. Copasi only had graphs, although the user had more control over the display of the plots. WEAVE had the largest visualisation capacity being able to display a variety of standard graph types along with more interesting ones, such as geographical maps, but it did not appear to have anything specialised for biological models. WEAVE is also the tool that gave the user the most control over the visualisation.

The existing biological modelling software seems to be focused more on the ease of modelling. The visualisation features on offer are typically quite basic. They also lack the more innovative features that can be found in the newer general data visualisation software.

1.1.2 Google Drive

Google Drive which was previously Google Docs, is one of the most widely used collaborative pieces of software. Google Drive is used by a variety of user types. The focus of the review was on the word processor. Of the collaborative software reviewed this was felt to be the most user friendly. One of the nicest features was a cursor indicating where every user currently editing the document is typing. Each user is also given a different colour allowing you to know who is doing what in real time. Many people can edit a single document at once. As well as editing documents users can also leave comments on the document. Changes made by users appear near instantly to all users, the speed of editing is very important as it is frustrating as a user to have to wait to see changes others are making. It is also very easy to invite others to edit the document with you. Each document has a unique link and if a user visits that link they are taken to the document and can start editing. Different permissions can be granted to different users allowing some level of collaboration with people who you don't necessarily want to grant full write access, these users can then just look at it in real time and offer comments. Different parts of the editor have different levels of collaboration. All text that is changed is changed for all, but preferences like font choice are only changed for the user, unless another user edits any text. Also importantly from a User Experience (UX) perspective is that any conflicts that arise appear to be resolved without any user intervention. A history of what each user has done to the document is also provided and any unwanted changes can be rolled back.

1.1.3 Pidoco

Pidoco is a collaborative wireframing tool. It was not as user friendly for collaboration as Google Drive. Pidoco is much less instant. Sometimes manual refreshes were required to display the work the other users had done. Pidoco also supported multi user editing, however there was no way of seeing which users were editing a document.

There was also a history of what changes each user had made. Pidoco has no messaging or commenting system which makes asynchronous collaboration more difficult. Sharing the work requires an email to be the sent to the user, they cannot simply be given a link. Again different parts of the workspace have different levels of collaboration. Any User Interface (UI) widgets that are placed are shared, but if one user zooms in on a particular area that other users are not forced to that zoom level.

1.1.4 Lucid Chart

Lucid Chart is a collaborative diagramming tool. Lucid Chart lies between Google Drive and Pidoco in terms of collaboration speed. It is not as instantaneous as Google Drive, but it does not require periodic manual refreshes like Pidoco. It also allows multi user collaboration and documents can be shared by link or email. Users can be granted read or read and write permissions on the document, like Google Drive, so you can collaborate with people you don't want to be able to fully edit. It has a chat system and users can comment on the document making asynchronous collaboration easier. Lucid Chart does not let you see what a user is doing in real time, but it does provide a full revision history so you can see what changes each user has made. Lucid Chart is different in that it appears to be fully collaborative. Even font preferences are synced across users, if one user clicks bold all users will start typing in bold.

1.1.5 SynBioWave

SynBioWave was a biological extension to Google Wave. It is the only example found of a platform for real time collaboration focused at synthetic biology that could be found. Unfortunately it could not be made to work. The instruction page still refers to Google Wave. Google Wave has been discontinued since 2012 (some of its real time features found their way into Google Docs). An attempt was made to follow the instructions using Apache Wave (Google Wave was handed over to the Apache foundation) but the SynBioWave tool did not work. The user guide does seem to give a good indication of how the tool would work. The review is based off this. Wave was used to enable the collaboration. You would create a new wave and add your human collaborators. You would also add the SynBioWave agent as a collaborator. When the SynBioWave agent is added to the wave it adds a menu bar, through this menu the human collaborators can call various functions, the agent then calculates the result and displays the visualisations. The agent and its results can be interacted with in real time. SynBioWave is modular and there are multiple agents available. Custom agents can also be created. The use of Google Wave allowed the biologists to focus on implementing domain functionality without having to worry about the code infrastructure for real time collaboration. Unfortunately by relying on a third party for the backbone of the system SynBioWave appears to have become unusable after Google Wave shut down.

1.1.6 Findings:

From studying the alternative modelling tools available and their visualisation capabilities it was clear that the visualisation capabilities of Bio-PEPA had to be increased. It was also clear that all the modelling tools were limited in their visualisation capability and that increasing the interactivity and customization available would be very useful.

We can see that there are a number of tools now that offer the ability to collaborate in real time. Only one, however, could be found that was focused on synthetic biology. This one tool appeared to now be dead. Given how useful tools like collaborative document editors have been more commonplace it seems like a good idea to try and bring this functionality into the synthetic biology domain. This would give Bio-PEPA a competitive advantage and hopefully encourage more researchers to use it.

1.2 Previous Work

Early on in the first stage of the project it was decided to separate this project from the Eclipse plug-in. It was felt that Eclipse is not the right tool to do data visualisation in.

The initial development stages were focused on getting the new tool from having zero functionality to matching the visualisation features of the Eclipse plug-in. This involved writing an early version of the UI, parsing the Bio-PEPA results data, and plotting it using matplotlib [12].

The next stage of the project was to extend the functionality. The first new feature was intensity plotting where the colour of the line increases in intensity/opaqueness as the population of the species increases. The next feature added was visualisation of the model. Model visualisation used a system of nesting circles and rings to build a hierarchical view of the cell from its model components. Finally the user was given control of the plot, allowing them to alter whether lines are plotted or not, what colour the line is and the thickness of the line. Figure 1.2 shows how the main screen of the program looked at the end of the first stage of development.

Over the course of the first stage of work a number of evaluations were carried out with potential and actual users of Bio-PEPA. The findings from these evaluations were used to improve the tool.

1.3 Results

ORPHAN PARAGRAPH FROM FINDINGS ABOVE - SEEMS TO FIT MORE HERE
By implementing more advanced visualisation features this tool should make Bio-PEPA a more attractive modelling tool. The visualisation features have been brought into line with the features in alternative modelling tools.

Break down by project stage? Anticipating this to be 1.5 - 2 pages

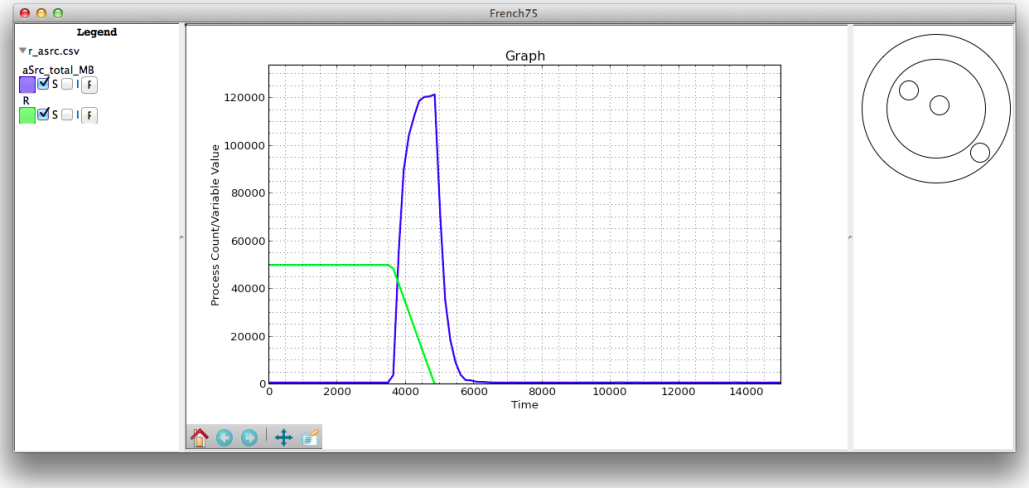


Figure 1.2: Main Screen of the Tool at the End of the First Stage of Development

Chapter 2

Goals

To help guide development during the project a number of goals were identified. Over the course of the project the goals have been refined, new goals have been added and some goals have been dropped.

2.1 Previous Goals

Through research into the problem a number of goals were identified. After performing a review of the existing software the goals were refined. The initial user evaluations also fed into the refinement of the goals of the project. Some of these goals were completed in the first stage of development, the other goals were carried forward into the second phase of development.

The goals that were to be completed in the first stage of development are:

- Improve existing capabilities – The purpose of the goal was to add to Bio-PEPA's visualisation capabilities and bring the more into line with the other modelling software. This goal was completed in the first phase of development. The user was given extra control over the appearance of their graphs. Intensity plotting was also implemented.
- Visualise the model more intuitively – The purpose of this goal was to help the biologists to increase their understanding of what the model and the graph is showing them. This goal was completed in the first phase of development with the implementation of the hierarchical drawing of the model components.

These other goals were to be completed during the second phase of development:

- Provide visualisations that take into account the domain – The purpose of the goal was also to help the biologists further understand what the model and the graph are showing them. This has been implemented in the second phase of the project. This goal was merged with animation of the data.

- Animation of the data – The purpose of this goal was to make it easier for the biologists to interpret spacial data by showing the species in their results moving through the cell. This has been implemented: the user can now visualise species moving through a cell.
- Investigating which visualisation combinations work best – The purpose of this goal was to try and work out whether there were any combinations on visualisations that the program could offer that the biologists found particularly useful. This was originally planned to be performed in the second stage of the project, but due to the project focusing on non visualisation features the goal has been removed.
- Add the ability to annotate the visualisations – The purpose of this goal is to allow the biologists to add extra supporting information directly onto the visualisation. This goal has been implemented: the user can add annotations to the visualisations.
- Allow the ability to save and load the program state – The purpose of this goal was to allow the biologists to complete their work in multiple sessions. This goal has been implemented: users can save and load the program state into files, these can then be emailed to other users who can modify them.
- Provide a full session history to the user – The purpose of this goal allow the biologists to recover from any mistakes they might make. This has been implemented, the user has a full undo and redo history within the session allowing them to easily correct mistakes.
- Data Mining – The purpose of this goal was to have the program provide some level of automatic annotation and setting optimisation to reduce the workload for the biologists. Due to other features being added that reduce the need for this goal the goal has now been dropped. The research for this goal was not lost however as a similar goal was added. This is not a goal that should be abandoned either, instead it would be implemented if the development continued.
- Making meta data accessible – The purpose of this goal was to The plan for this has not changed since the first phase.

2.2 New Goals

During the first half of the second phase of development new goals were identified.

2.2.1 Data Manipulation and Export

This goal was added after a meeting with the maintainer of BioDARE [13]. This web based tool is aimed at results from laboratory experiments, specifically experiments relating to circadian rhythm.

The maintainer explained that he had found that researchers often visualise their data after it has been normalised. This removes any issues where different species have different scales and makes it easier for the users to interpret the data.

BioDARE can also export the raw data allowing the researchers to use it in other applications if they desire.

This seemed like a very useful feature for users and it was added as a project goal.

2.2.2 Time Series Data as a Query

There has been a lot of research in recent years [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26] into using time series data as a query against a database of other time series plots through some similarity measure.

This would be a very useful feature to add to the tool. By allowing researchers to use a plot as a query they would be able to discover other plots exhibiting similar behaviour. This could be the same species reacting the same to different species, or a species that is exhibiting similar behaviour. This could help the biologists discover alternative candidate species for research.

This feature would be most effective if there was an online repository of plots for the biologist to search. This would also allow them to discover potential collaborators. Online repositories of data do exist, BioDARE is one of them. These repositories are unlikely at the moment to store the data in a suitable way to allow for efficient search.

Given the potential usefulness of this feature, and the fact that none of the other modelling software had this capability it has been added as a goal.

2.2.3 Real Time Collaboration

An important area that all the reviewed software were lacking in is collaboration, real time or not. The current work flow using the existing software is: perform your analysis, save it, email it to a colleague with additional files and notes, have them open it, they attempt to work out the visualisation is showing. It is a disjointed conversation.

It is not just the modelling software that did not offer collaboration, none of the visualisation software reviewed offered real time collaboration either.

Adding real time collaboration was added as a goal to offer researchers a better work flow.

Chapter 3

Work Done

This section details the development work that has been performed in the second phrase of the project. This includes conceptual problems faced, effectiveness of the result and any future work that could be performed.

3.1 Animation

Animation was a key goal of the project. The core aim of the project is to help biologists who aren't comfortable with traditional time-series graphs. The goal was to provide them with visualisations closer to what they see in their domain. This has been accomplished by displaying spatial data in the shape of a cross-section of a cell.

The animation is ideally used to display species moving through a cell. This collapses what would be multiple different lines in a graph, that give no indication of their real position in the cell, into a single image. This can be seen in Figure 1.1.

For each line in the visualisation one cell cross section is drawn. Each cross section is split into compartments, these are parsed from the model file. The colours in each compartment reflect the colour of the line on the graph.

Over the course of the animation the colour of the segment changes to reflect the colour in the intensity plot version of the line. This allows the researcher to compare the two visualisations and will hopefully help build their confidence understanding the graph plot.

Animation takes advantage of the data centric model. When the play button is pressed a thread runs whose job it is to increment the clock and refresh all the UI elements. When refreshed the UI elements pick up on the change of clock and update what they display appropriately.

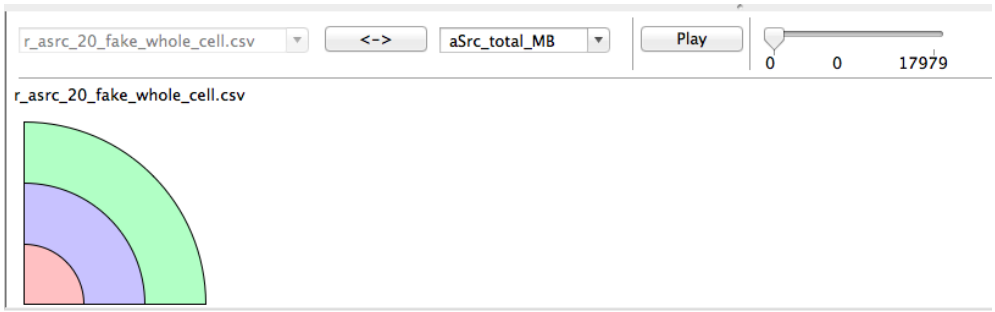
The first step towards animation was calculating at what points the line changes colour. This is done during the interpolation phase that was written in the first stage of development. To do the interpolation the original plot is split into multiple separate plots with the results padded with null values to allow matplotlib to plot them as one. The

number of these null values is counted and that gives the time at which the colour should change. This is stored as an array of time, colour tuples. Each line also has an internal counter to say how far through this array they have gone. When the animation clock is updated and the refresh triggered each cell segment finds the appropriate colour change point in the line and updates the segment colour. This position is then used as the new starting point on the next refresh. This saves looking through past points unnecessarily.

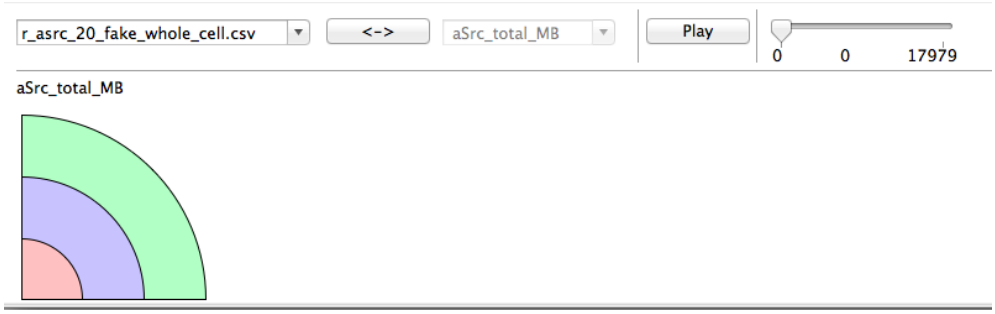
The user has can control the time through the time slider. When the time slider is used each line has its internal clock set to 0. Then when the animation thread triggers a refresh the whole of the lines colour change points are searched to find the appropriate point. This is a much neater solution than keeping track of previous points and slicing the arrays.

A nice UX feature that has been added a line to the main graph panel that indicates the current time. When the animation is running the line moves along the graph. This again helps the user build a correspondence between the two visualisation types.

The user is also given control over what is displayed. The cell segments can be drawn in a file focussed manner or a species focussed manner. In the file focussed manner a cell cross section is drawn for every species in that file. The user can switch between all results files that have been added to the session. In the species focussed manner a cell cross section is drawn for every file that contains that species. The user can select between all species found across all the results files. These are illustrated in Figure 3.1



(a) Animations Drawn in Species Focussed Manner



(b) Animations Drawn in File Focussed Manner

Figure 3.1: The Two Ways of Drawing Animations: File Focussed and Species Focussed.

MOVE THIS INTO WHEREREVER SESSION CREATOR IS TALKED ABOUTThe

requirement of the model file for parsing animation has also led to animation replacing the previous model visualisation. In the set up phase after the the model and the species have been parsed the user is presented with a cell segment, similar to what is seen in the animation panel. The cell segment is split into different regions, one for each region of the cell. These segments are then coloured if the selected species is present in them. The user can select between all species in the selected results files. This has a number of benefits. First, they can sanity check that they have matching results and model files. Second, they can see how the model is structured.

MOVE THIS INTO WHEREREVER SESSION CREATOR IS TALKED ABOUT To make setting up the animation user friendly to control required the model file so that the location hierarchy could be parsed. Before this there was an awkward system where the user had to input where in the cell a species is. This was time consuming, awkward and quite brittle. At the time it assumed that there would be three compartments in the species, which is a terrible assumption to make.

3.2 Annotation

Annotation was an important feature to add. It is one of Grinstein's features that data visualisation software should have [27]. This is because the purpose of a visualisation is to aid analysis. If you have a print out of a visualisation you are able draw on it and highlight areas of interest. Users need to be able to do this on digital versions of their visualisations. As well as helping users analyse their data, being able to annotate also means that images for presentations can be prepared without having to save the visualisation and open it in an external program. If you did this and then wanted to change the visualisation you would have to re-annotate it. This is frustrating for a user and wastes their time. Being able to do it from within the visualisation solves this problem as the raw data and the annotation data are together. Another benefit of being able to annotate is having another way of attaching additional information to the visualisation when sending it to a colleague. Having this information on the visualisation saves them from flicking back and forth from an email or other supporting documentation.

Implementing annotation of the visualisations on offer in this tool had to be done in two parts. This is because there are two parts to the annotation. The graph and the cell level view. One is a static visualisation and one is a dynamic visualisation. Each of these required a different approach. These are detailed below.

3.2.1 Annotation of the Graph

Annotation on the graph was relatively straightforward to implement. matplotlib provides annotation capabilities in two forms. One form is annotations that use arrows with or without text, and the other is arbitrary drawing on the graph. Both of these forms have been utilised for annotations.

Users of the new tool are provided with four annotation types: arrow, text, arrow with text and circles. Buttons for each of these annotation types have been placed on the matplotlib toolbar. This can be seen in Figure 3.2

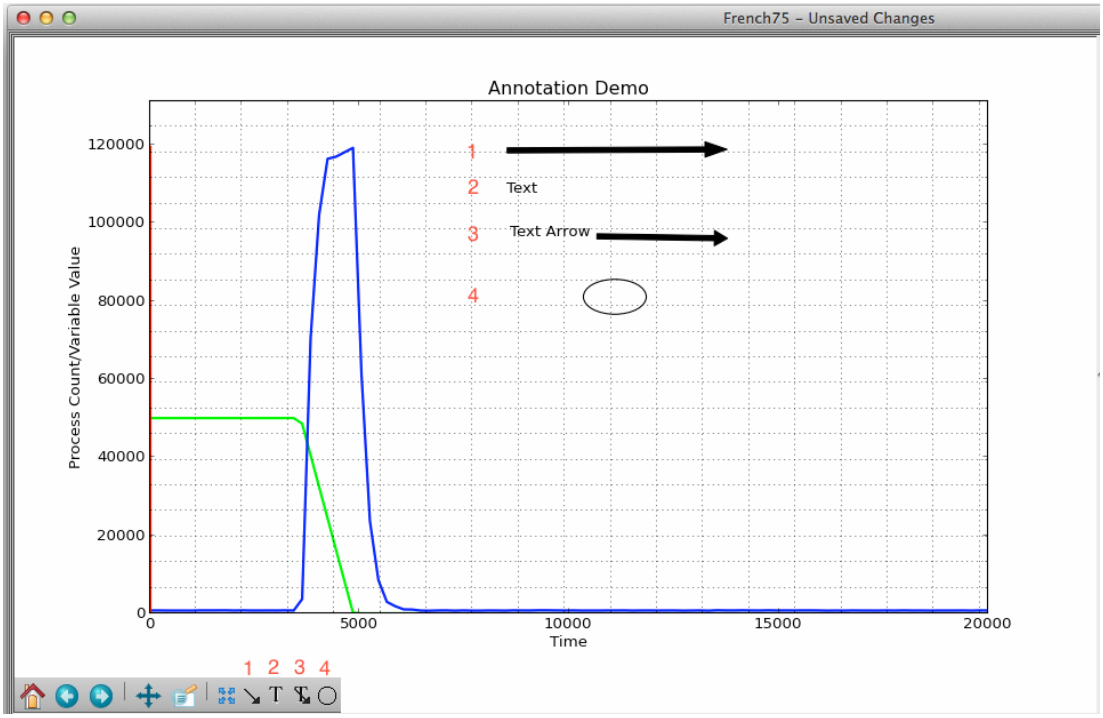


Figure 3.2: Example of Each of the Four Annotation Types and the Toolbar Buttons that Correspond to Them

There were problems in making the annotation creation process user friendly. For all of the annotations the user needs to first press the appropriate button on the on the toolbar. The next actions depend on the annotation type. Text and circle annotations were intuitive as all they require is one click. The user clicks where they want the annotation to be placed and it will be drawn on the graph there. However the two arrow annotation types required two clicks. The first click marks the start point (the tail of the arrow) and the second click is the finish point (the head of the arrow). This was not obvious, when handed over to the users they didn't know that it required two clicks and didn't know whether the arrows would be drawn head to tail or tail to head. The technique for placing arrows was changed so that the first click still fixed the position of the tail of the arrow. However the behaviour after the first click has changed, now a temporary annotation is continuously redrawn that has the head of the arrow wherever the mouse is. This allows the user to see the arrow they are drawing. To indicate to the user that they need to click on the graph the cursor is changed after pressing one of the annotation buttons on the toolbar. The use of different cursors is a common technique to help guide users to perform actions.

It was important that annotations could be edited or deleted. The annotations can not just be clicked as they are not a UI widget like a button. The solution to this was to have an array of annotations. When a user right clicks on the graph it searches through all annotations and selects the annotation that was closest to the click (if that distance

was below a certain threshold). The selected annotation is then highlighted red, and a context menu appears to give feedback to the user that they have successfully selected an annotation. This can be seen in Figure 3.3 The context menu then gives the user the option to edit or delete an annotation. Editing an annotation only allows for editing text. For changing position the annotation has to be deleted and redrawn.

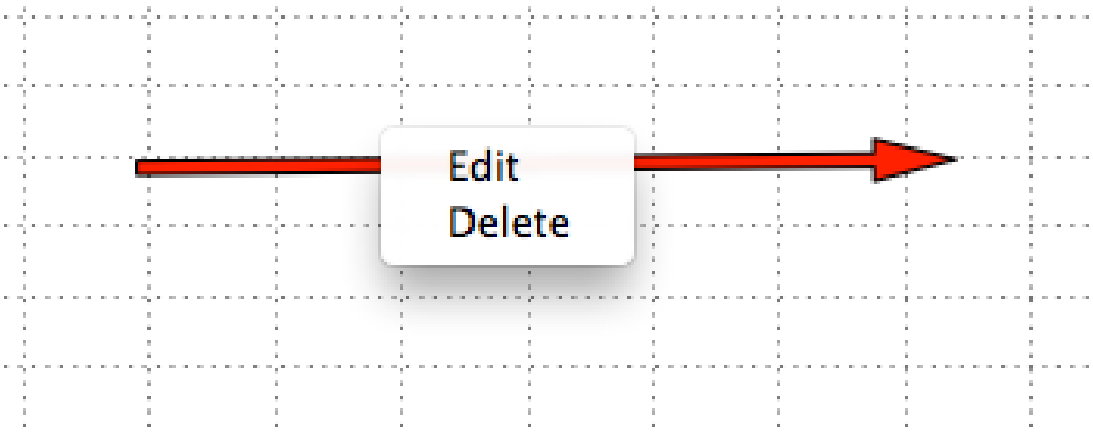
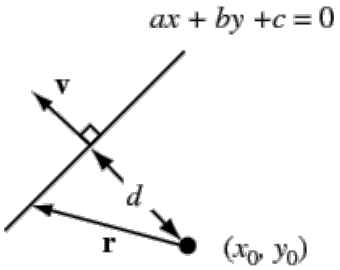


Figure 3.3: Context Menu on Selection of Annotation

To calculate the distance from the location of the mouse click to the annotation two problems had to be solved. The first was how to calculate the distance from a point to a line. An equation for this can be seen in Figure 3.5. Equation 3.1a is the equation for a line in two dimensions. This represents the annotation. The equation can be calculated from the start and end points of the annotation. Equation 3.1b is the position of the mouse click. Equation 3.1c shows the equation for calculating the distance from the point to the line. The second problem that had to be overcome was the difference in scale. In effect we have two coordinate systems. The results coordinate system and the visual coordinate system. These difference are illustrated in Figure 3.7. The annotations in Figure 3.7a are further away from each other in the results coordinate system than the annotations in Figure 3.7b, however they are closer in the visual coordinate system. When selecting an annotation the user is using the visual coordinate system but the distance is calculated using the result coordinate space. This could lead to the wrong annotation being selected. The solution to this was to transfer one coordinate system into the other. When calculating the distance from the point to the line the values are scaled by the size of the graph.

A similar issue to the difference in scale when calculating the distance from the mouse to annotation was encountered when switching the graph to normalised mode. When normalised the graph in the y axis only goes between 0 and 1. The coordinates of the annotation are likely to be very much outside this range and the annotation would not appear on the graph. The solution was to if the graph is normalised then to also normalise the positions of the annotations when drawing them. A side effect of this technique is that the lines are potentially going to rescale themselves. This could leave an annotation point to nothing. This is unavoidable as the annotation has no concept of what it is pointing at. It may be pointing to the intersection of more than one line, in this case it would be impossible to keep the annotation pointing at what it was originally pointing at. Neither of these is optimal. Both approaches lose the information that



(a) Diagram of the Point to Line Distance Problem

$$y = -\frac{a}{b}x - \frac{c}{b} \tag{3.1a}$$

$$(x_0, y_0) \tag{3.1b}$$

$$d = \frac{|\, ax_0 + by_0 + c \,|}{\sqrt{a^2 + b^2}} \tag{3.1c}$$

(b) Equations for Calculating the Distance From a Point to a Line.

Figure 3.5: Equations and Diagram for Calculating the Distance From a Point to a Line (taken from WolframMathWorld [28])

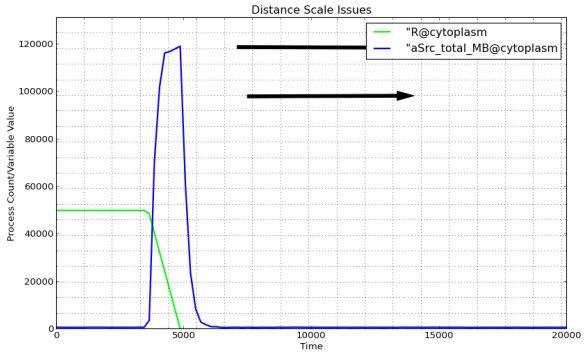
the user originally added. During user evaluations, when the annotation would not be visible during data normalisation, the user was very confused as to why their annotation had disappeared. It was therefore decided to have the annotation be visible but most likely incorrect as it is clearer to the user what has happened.

3.2.2 Annotation of the Animation

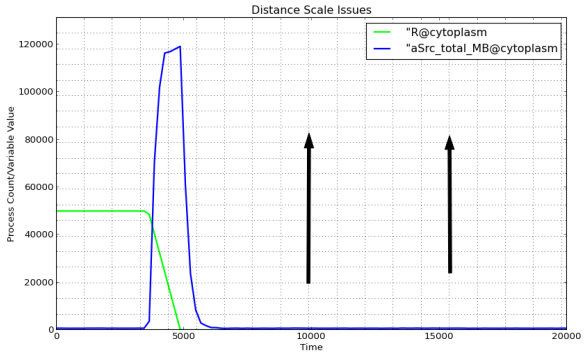
After completing annotation of the graph it was important to expand this to the animation panel, as this is the other visualisation option available to a user. Annotating animations posed more of a challenge than annotation of the graph and there were a number of issues to overcome.

1. How to implement the annotations? For the graph matplotlib has built in annotation support. wxPython does have drawing support but not in built annotation support. Annotating on the animation will need manual handling of the drawing on top of the animation visualisation. Manual drawing means that the automatic layout functionality that wxPython provides cannot be used.

2. When to display the annotations? When an annotation is drawn on the graph it is displayed at all times. The appearance of the graph does not change over time. However the appearance of the animation visualisation does change over time. The problem faced when annotating is whether to have annotations available at only specific times in the animation, or to have them there the whole time, and if they are going to appear and disappear how can it be done without being distracting?



(a) Annotations With Distance of Approximately 20000



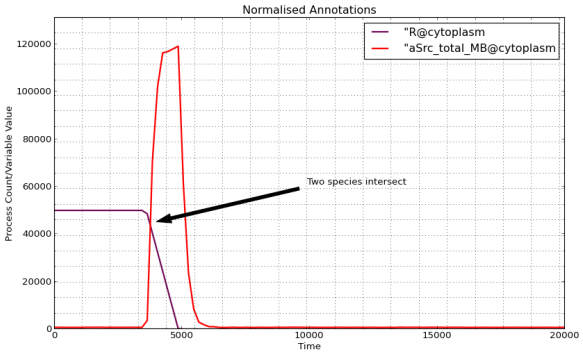
(b) Annotations With Distance of Approximately 5000

Figure 3.6: Two Sets of Annotations Illustrating the Problems When Calculating Distance to an Annotation

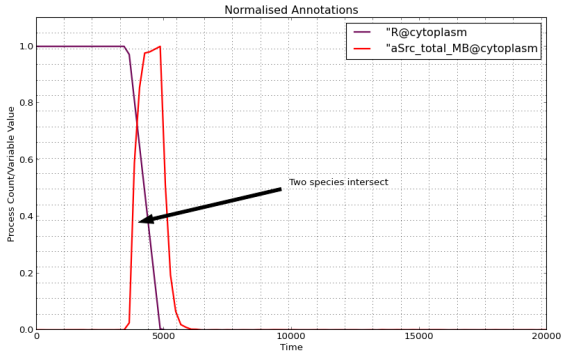
3. How to give the user control over the annotations? When a user wants to edit or delete an annotation on the graph it is always there. However on the animation panel if the annotation is temporal, then it is not always visible for the user to edit or delete and it would be frustrating for a user to constantly have to search through the animation to look for annotations to change them.

A platform where users are able to annotate an 'animation' is YouTube [29]. YouTube's annotation interface can be seen in Figure 3.8. Youtube's approach is to allow the user to set what period of time an annotation is visible for. There is also a management panel where the user can see all of the annotations and edit or delete them. This approach solved issues two and three and was adapted for this tool.

On the right hand side of the tool there is a panel which lists all annotations that have currently been added to the tool, this can be seen in Figure **??**. This provides the user with the persistent view of what has been added. When the user creates an annotation they are shown a dialogue that allows them to enter the annotation text and to choose a duration. This indicates to the user that the annotation will only be visible at certain times. The start and end times are given in model time. The duration displayed to the user is given as the real time that the annotation will be visible for. This can be seen in Figure 3.10.



(a) Annotation Pointing at Correct Intersection of Species



(b) Annotation Pointing at Nothing After Data Normalisation

Figure 3.7: Graphs Illustrating What Happens to Annotations Whilst Data is Normalised

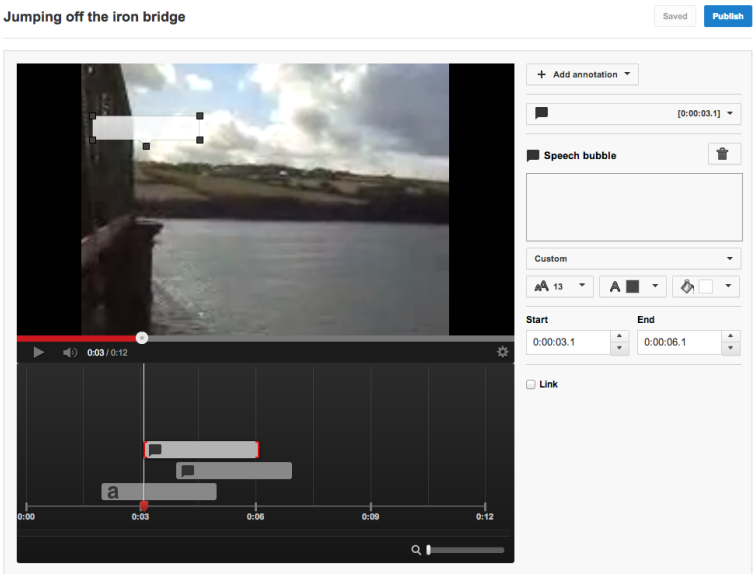


Figure 3.8: YouTube Video Annotation Interface

The first problem – how to display the annotations – has also been solved. Drawing in wxPython takes place on panels. Each cell cross section is placed on its own panel.

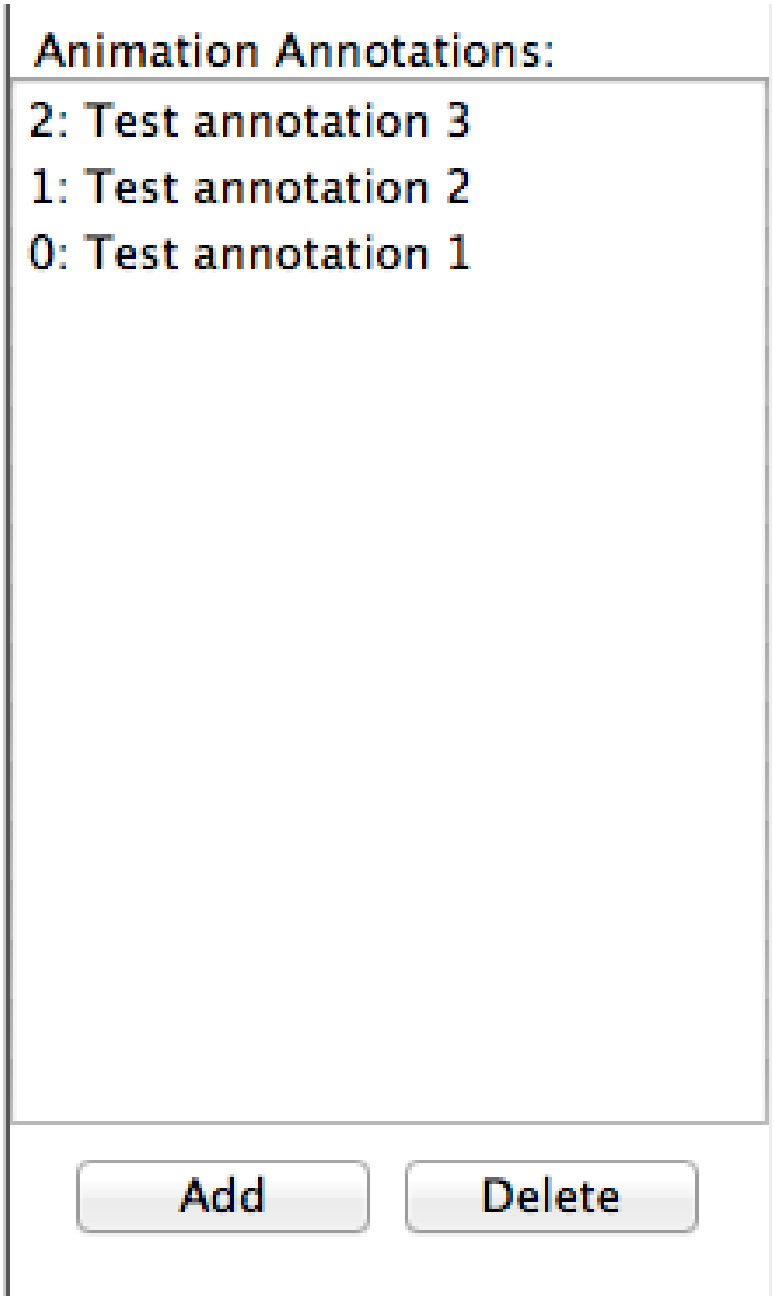


Figure 3.9: Animation Annotation List

This drastically limits the available space. The panels are not wide enough to have the annotation text drawn on. The solution was to assign each annotation a number and that number is what is used to annotate the cell. The number is then displayed in the annotation list box allowing the user to read the appropriate annotation. This can be seen in Figure 3.11

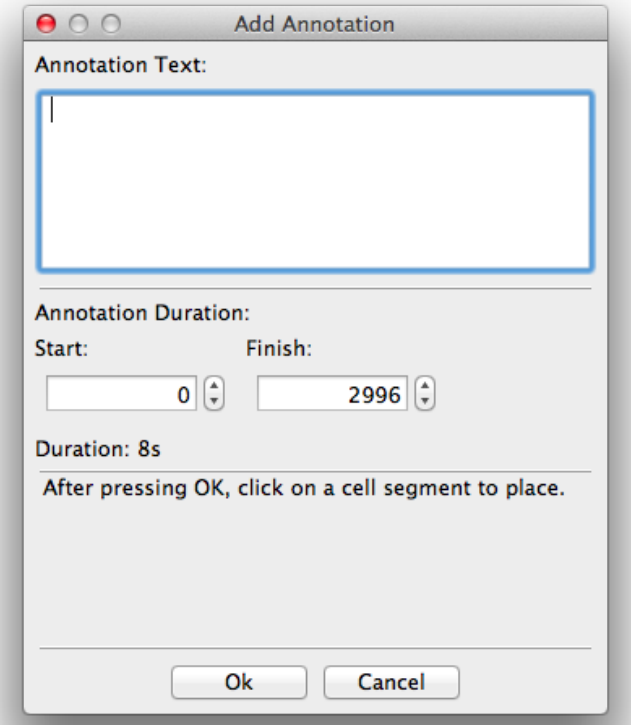
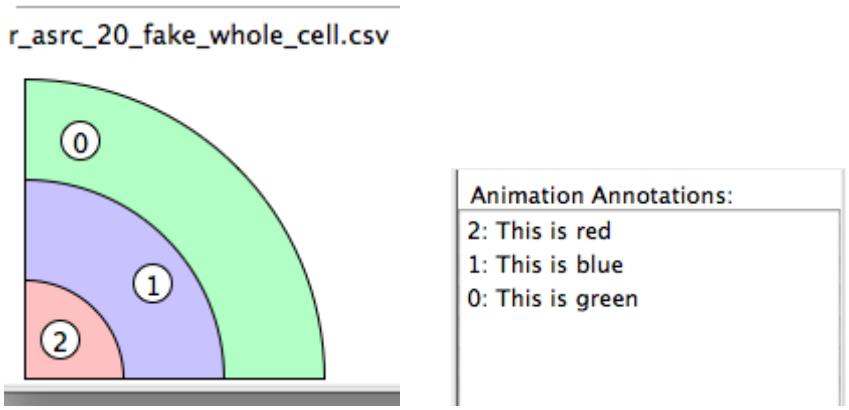


Figure 3.10: Animation Annotation Dialogue



(a) Annotated Cell Section (b) Corresponding Annotation List

Figure 3.11: Animation Annotations

3.3 Data Mining

Gone - WHY?

3.4 Search

Begin able to use a time series data as a query against a database of time series data was one of the new goals added to the project. It was felt it would be of great use to the biologists as it would allow for discoverability within the tool.

Find a citation that backs this as being useful

. It is also a feature in the tool that incorporates cross domain knowledge in an area of active research.

Some problems needed to be overcome before using plots as a query was useful:

- How to cope with different scales?
- How to cope with events happening at different times?
- How to represent the plot to allow for efficient search?
- How to determine similarity between two graphs?

Early techniques used simple similarity measures such as Euclidean distance, but these gave poor results. The reason for the poor results is that Euclidean distance does not take into account scale or time. Figure 3.12 illustrates a case where Euclidean distance would give a poor similarity measure. The two lines are identical, but the green line has been shifted in time. Euclidean distance will not recognise that they are similar. The same problem would happen if one line had been shifted in the y-axis above the other.

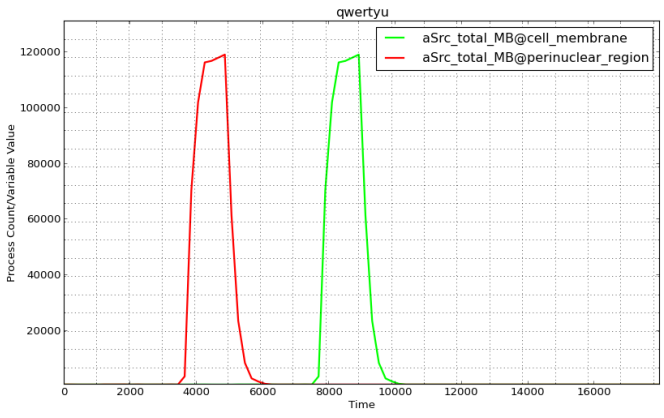
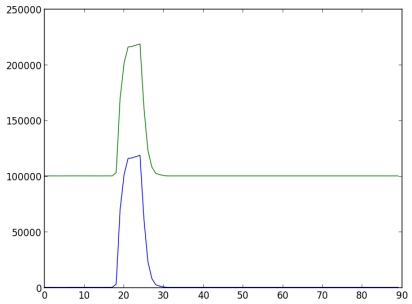


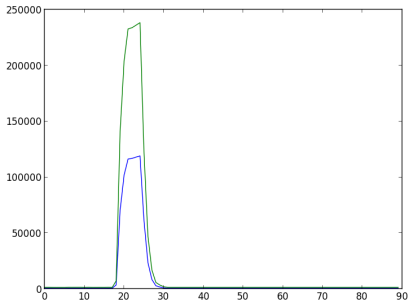
Figure 3.12: Identical Plots Shifted in Time

It is important to define what it means for two lines to be similar. It is obvious that in case such as Figure 3.12 that the two lines are similar. Both lines are the same shape, they are exhibiting the same behaviour. When determining similarity of plots we should take a time invariant approach. Perhaps less obvious is the case where we have offset in the Y-Axis as shown in Figure 3.13. Figure 3.13a shows two identical lines where one line had an initial 100000 population level. Figure **??** shows two lines

where one line is double the other line. Figure 3.13a is effectively the same case as Figure 3.12. Figure 3.13b is different, but it is clear looking at the graph that they are exhibiting the same behaviour. when determining the similarity of plots we should take a scale invariant approach.



(a) Identical Plots Shifted in Population Level



(b) Identical Plots Scaled in Population Level

Figure 3.13: Graphs Illustrating Different Ways Lines can be Offset in the Y-Axis

After researching current techniques an approach was taken that solves all the problems above.

The first step is to convert the input data to be a list of all n length sub lists of the input data. i.e. with input data [1,2,3,4,5] and n = 3 our input data would become [[1,2,3], [2,3,4], [3,4,5]]. The sub lists are our features. We then normalise each feature to be zero mean, unit variance.

The next step is to convert the continuous data into discrete data and reduce the dimensionality of it. By normalising the data to be zero mean and unit variance we have allowed a normal distribution to be easily fitted to our data. Using the normal distribution we convert a data point in our sub list to be one of three possible values. Here we use the characters ‘a’, ‘b’ & ‘c’. This builds a string representation of the plot.

The paper this approach is taken from then calls for reducing local duplicates. This further reduces the size of our fingerprint and increases the efficiency. Removing the local duplicates means any ignoring runs of a certain string can be replaced with just one instance of the string.

This provides us with a finger print of the plot. A string size of 8, with 3 possible

value leads to a vocabulary size of 6561 possible representations. A single plot will have very few of these, so we use a sparse representation. For each line a dictionary is stored. The dictionary keys are the strings in the fingerprints and the value is the count of that string in the plot's fingerprint.

This format of the fingerprint also solves the problem of similar features happening at different times. The plot fingerprint is a bag-of-words, in this case a bag-of-features. There is no order to them, it is as though we are assuming that all features appear at the same instant.

Now that we have a representation of the plot that is scale and time invariant and allows for efficient comparisons we can find a method to calculate the similarity.

We could simply just use Euclidean distance again, but this does not do anything to weight rarer features that can tell us more about a graph. An alternative distance measure would be cosine distance, but this suffers the same drawback.

It was decided to implement tf.idf weighted cosine as the similarity measure. This takes into account term frequencies to provide a higher weight to rarer vectors. tf.idf is very well researched in the field of information retrieval and text mining. Given the representation of the finger print is equivalent to a bag-of-words it seems sensible to apply tf.idf to this domain.

3.4.1 tf.idf

is a way of ranking how important a word is. Taken into account are frequency of a word in the query, frequency of a word in the document, length of the document, average length of documents in the corpus, total number of documents and frequency of the word across all documents.

The formula is: INSERT FORMULA.

These term weights are then used in a tf.idf weighted cosine.

This approach allows for effiecient search. If we have two indexes. One where keys are the documents and the values are the words in the document and their frequencies. Then we have another index, an inverted one. Where the keys are the words and the values are a list of documents that contain them. The individual components of the tf.idf equation can therefore be calculated with very little effort. This means that a plot can be compared to the database of plots in O(n) time.

Initial results are very promising with the tf.idf weighted cosine providing much clearer results than with simple euclidean distance. There is not a large enough truth set of plot similarities to be able to confidently say that it is effective. However these early promising results seem to indicate that further research would certainly be worthwhile, but outside the scope of the project.

DISCUSSION OF THE EARLY RESULTS

3.5 Collaboration

The initial step of allowing collaboration is enabling two instances of the program to talk to each other. There are a number of techniques to allow for this. Data could have been sent directly through sockets, another option would be to create a web server and send get requests and pass parameters. It was decided to use an existing library to handle the communication. The library chose was simplexmlrpc. This allows each instance to run a client and a server. The server has an api and this can be called.

On initial connection the 'master' sends its entire session state the the 'slave' uses this as its initial state. After this whenever either client performs an action it issues a call to the other clients server to perform the same action.

An issue that was encountered is how to ensure that both clients are seeing everything in the same order. Lamport clock proved to be the solution.

3.6 Usability

In all the evaluations of the project users have commented on the difficulty of using parts of the tool. Action has been taken to make it easier to use. Many of the changes have been guided by Shneiderman, Norman & Nielson's guidelines. Specifics are detailed below.

3.6.1 Undo & Redo

Shneiderman calls for easy reversal of actions and Nielson calls for user control and freedom – an emergency exit from an unwanted state. To address this, an undo/redo functionality has been added. This required refactoring of the project code, so that the session data is in one location, inside a singleton. Any changes to this data are picked up during the next UI update and are reflected in the visualisations. The session data is stored as a dictionary. To implement undo and redo, copies of the data dictionary are pushed and popped onto the stack. Copies are pushed onto the undo stack on any atomic change the user makes. This gives the user a full session history to go back through and this was one of the early goals from the first project phase.

A problem was encountered when trying to copy the dictionary onto the stack. When just pushing the dictionary onto the stack it would not put a new copy of it onto the stack, so any changes to the dictionary after it has pushed onto the stack are also there in the stack. Python dictionaries have a copy method. Copy only does a shallow copy – any objects in the original dictionary will have their reference placed in the new dictionary. This was fine for some parts of the session dictionary, but for others it was not. In particular, lines and annotations, which are custom objects presented problems. This was solved by using deep copy. With deep copy a new copy is made of objects as well. Some elements of wxPython and matplotlib were unable to be deep copied, but

this was fixed whilst focusing more around the data – so the UI elements use the data, not the other way around.

3.6.2 Saving

It is important that a user is able to save and load the visualisation session as they may not be able to complete all their analysis in one sitting and may want to come back to their work in the future. Without the ability to save and load the user would have to repeatedly add annotations and change preferences and attach files. It was possible to add Saving and loadingby building on the work done to implement undo & redo, although further work was required. Python has a module called pickle to serialise and deserialise data. When saving, the dictionary containing the session data is pickled and written to a file and when loading the reverse happens. Because the program is now focused on the data model, once a previous session has been loaded, a UI refresh is triggered and the visualisation reflects the loaded data.

Saving the data also enables limited collaboration. The user can customize the appearance and add annotations. They can then save the state to a file and email that file to a colleague. The colleague can then load the file and see the user's work. The colleague can then correct any issues and add their own work. The colleague can then save this and email it back to the user. This is useful and is better than no collaboration, but it is entirely non realtime.

3.6.3 Feedback

Norman and Shneiderman both call for feedback to be given to the user so that the user can be sure that an action has been accomplished. This feedback can come in a number of different forms and was in place in some parts of the project already.

Existing feedback in the project was a natural byproduct of some of the features; for example, when loading a results file the feedback that the load operation has been successful is that a graph appears on the screen. If the graph does not appear then something has gone wrong. Additional feedback has been added to the project:

- When adding annotations the cursor changes to indicate to the user that they can interact with the graph in a different way.
- The title bar text changes to display "unsaved" when the user makes a change and then changes back to "saved" when a successful save has been performed.

3.6.4 Guiding the User

The first evaluation of the second phase of the project unearthed that the users struggled to choose the correct action as there were multiple ways of doing the same action that had slightly different use cases. There was also confusing language in the menu

options. These multiple paths have been removed. For example, now there is only one way to open results files initially. To help guide the user further UI elements are enabled and disabled as appropriate. Now when the program is first loaded the only action a user can perform is to load a session or start a new session. Afterwards other UI elements are enabled to allow the user to start using the tool effectively.

3.7 Data Manipulation and Export

3.8 Finished Product

Overview and walkthrough of tool

Chapter 4

Evaluation

4.1 Evaluation

4.1.1 First Evaluation

The first evaluation in the second phase of the project occurred in November 2014. The user group was made up of two people. One who had taken part in the first evaluation meeting and one person who had no knowledge of the project.

4.1.1.1 User Group

As I did not have a domain expert available I was not able to do insight based evaluation. I took a more traditional approach. Before the evaluation I prepared a typical scenario that a user might encounter. The task was to open a file, annotate it and run the animation visualisation, and attach supporting documentation. The task was prepared at two levels of instruction. The first level was a paragraph of text that described what was to be done. The second level was a step-by-step list of instructions to perform. I observed the user group as they attempted the task and offered assistance when required. Afterwards the user group was given a questionnaire to fill in about their experience. After filling in the questionnaire we went through and discussed their answers and any further thoughts that they had.

The task was prepared at two levels to try and gauge how easy the program is to use. The users were first presented with the textual description and if they had been unable to complete the task with this then they would have been given the step-by-step instructions instead. The users were able to complete the task from the textual description alone. This is a good sign that the new tool is usable.

Some issues were encountered:

- The users were unfamiliar with MacOS – Both users were unable to locate the menu bar as it is not attached to the program as in Windows. Future evaluations will use Windows.

- The users were unclear as to what was going to happen when annotating. When annotating the graph with an arrow the user has to click twice to place it but there is no indication of this, nor was it clear to them which way the arrow would be drawn. This has now been fixed. Different cursors are used to give feedback to the user that they should click, and rather than just relying on two clicks with no information as to where the arrow is going to point, after the first click (which places the tail of the arrow) an arrow will be drawn that follows the cursor until the second click placing the annotation.

- Lack of ability to edit, move, or delete annotations – Once an annotation was placed it was there permanently. The ability to edit annotations was always planned, but had not been implemented in time. But the amount of frustration it gave the users was very high. It was a principle in all three of Norman, Neilson and Schniederman's lists that a user should be able to fix mistakes. Since the evaluation, editing and deleting of annotations have been implemented. This means any mistakes can be corrected.

- Initially they were confused by what all the buttons on the matplotlib toolbar did. After discovering the tooltips and seeing what effect the buttons had they were comfortable with them. If a user were to do something they did not intend they are able to undo it. All the matplotlib built-in buttons on the toolbar can be undone and redone from the toolbar. Any buttons implemented for this tool are covered by the undo and redo functionality implemented across the whole program. Being able to recover from their actions on the toolbar means no hindrance to discovery and so needs no further action. It would be desirable to have the two undo methods unified but a way to do this could not be found.

- The users were confused by some of the terminology. In particular "save graph" and "save model". These items in the menu have now been grouped more carefully to help the user distinguish them. A related issue was worrying that "save graph" was going to override the results file. To rectify this the menu items that create new files have been renamed "export ...".

- The users struggled to start a new session. When asked for a title they did not know what the title was going to be used for. When trying to add files, rather than use the add files button in the dialogue, they tried to use the file menu. Having two routes into the visualisation seemed to be confusing them. Now the file menu open file has been removed. To create a visualisation the user has to go through the new session wizard.

- When placing species in the cell one of the users did not understand what they were being asked to do. One of the users did understand. To fix this user input has been removed from the equation. This has required the model file to also be chosen, but then species locations are parsed automatically.

- They liked the animation feature and thought it would be very useful (One of the users did their PhD in transport and expressed a desire to have had this feature during the PhD). They did feel that it wouldn't be useful directly for papers, but that it would be useful when deciding what to include in a paper.

- One of the users asked if there was a map of the cell. When presented with the model visualisation they thought that it did look nice, but they were unsure of its usefulness. The model viewing has since been merged into the animation visualisation.

- The results from the questionnaire indicated that both users thought the tool's appearance was good. The tool was average in difficulty to use – neither easy nor difficult. The annotation buttons on the toolbar were clear as to what they did. It was obvious how to attach supporting files. Both users thought that it is very useful to attach files to the session so that they can be easily emailed to a colleague. They thought it would be useful to have the graph automatically annotated, but they wanted to the ability to disable any automatic annotations. Both users found the animation useful.

4.1.1.2 Personal

At this stage in the development the program was in a state where some existing functionality had been broken and gone unnoticed during the implementation of the new features. This highlighted architectural flaws in the code. There were multiple paths through the program that data was taking, and duplicated code in places. Since then a majority of these bugs have been ironed out and the duplications removed. The code much better architected. At the time of the evaluation with the users not all the features could be tested with them – mainly the plot preferences dialogue. These features have now been fixed and they will be evaluated by the users at the next meeting.

Having the users use the program has also highlighted a number of usability problems: menus being badly organised and named, features such as annotation rely on assumed knowledge to work them. All this created an unfriendly environment for the user. This was due to losing sight of the need for usability during development and when testing new features not removing the knowledge of the code from my mind. Since this evaluation the three lists of usability have been refocused on and the code gone through and the principles applied.

I am pleased with the positive feedback on animation and annotation – two of the core new features.

4.1.2 Evaluation 2 - Start of Second Semester

4.1.2.1 User Group

4.1.2.2 Personal

4.1.3 Evaluation 3 - End of Second Semester

4.1.3.1 User Group

4.1.3.2 Expert User

4.1.3.3 Personal

4.1.4 Overall Self Evaluation

Scribble over lots of stuff talk about changes

Chapter 5

Conclusion

5.1 Conclusion

5.1.1 Comparison to Objective

5.1.2 Challenges Faced

5.1.3 The Future

5.1.4 Final Remarks

Bibliography

[1] Bio-PEPA. Bio-pepa homepage. http://www.biopepa.org.

[2] Google Drive. homepage. drive.google.com.

[3] Pidoco. homepage. https://pidoco.com/en.

[4] Lucid Chart. homepage. https://www.lucidchart.com.

[5] SynBioWave. homepage. http://www.synbiowave.org.

[6] Uppsala Universitet and Aalborg University. Uppaal. http://www.uppaal.org/.

[7] University of Connecticut Health Center. V-Cell: Virtual Cell Modeling & Analysis Software. http://www.nrcam.uchc.edu/index.html.

[8] Azevedo Lab, University of Houston and MBI, German Cancer Research Center. Cell-O: Biological Simulation Framework. http://www.cell-o.org/.

[9] Mendes Group, Virginia Bioinformatics Institute and Department Modeling of biological Processes, University of Heidelberg and Mendes group, University of Manchester. COPASI: biochemical network simulator. http://www.copasi.org/.

[10] The Systems Biology Institute, Tokyo, Japan. CellDesigner: A modeling tool of biochemical networks. http://www.celldesigner.org/index.html.

[11] Institute for Visualization and Perception Research, University of Massachusetts Lowell and Open Indicators Consortium. WEAVE: Web-based Analysis and Visualization Environment. http://www.oicweave.org/.

[12] matplotlib. Python graphing library. http://matplotlib.org/.

[13] BioDARE. homepage. https://www.biodare.ed.ac.uk/.

[14] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, December 2012.

[15] ChotiratAnn Ratanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, and Gautam Das. Mining time series data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 1049–1077. Springer US, 2010.

[16] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In DavidB. Lomet, editor, *Foundations of Data Organization and Algorithms*, volume 730 of *Lecture Notes in Computer Science*, pages 69–84. Springer Berlin Heidelberg, 1993.

[17] Tetsuya Nakamura, Keishi Taki, Hiroki Nomiya, Kazuhiro Seki, and Kuniaki Uehara. A shape-based similarity measure for time series data with ensemble learning. *Pattern Analysis and Applications*, 16(4):535–548, 2013.

[18] DinaQ. Goldin and ParisC. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In Ugo Montanari and Francesca Rossi, editors, *Principles and Practice of Constraint Programming CP '95*, volume 976 of *Lecture Notes in Computer Science*, pages 137–153. Springer Berlin Heidelberg, 1995.

[19] Jessica Lin and Yuan Li. Finding structural similarity in time series data using bag-of-patterns representation. In Marianne Winslett, editor, *Scientific and Statistical Database Management*, volume 5566 of *Lecture Notes in Computer Science*, pages 461–477. Springer Berlin Heidelberg, 2009.

[20] C. Faloutsos, H. V. Jagadish, A. O. Mendelzon, and T. Milo. A signature technique for similarity-based queries (extended abstract). In *In Proceedings of SEQUENCES97*, pages 11–13. IEEE Press, 1997.

[21] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, June 2002.

[22] I. Popivanov and R.J. Miller. Similarity search over time-series data using wavelets. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 212–221, 2002.

[23] Eamonn Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 406–417. VLDB Endowment, 2002.

[24] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD Rec.*, 23(2):419–429, May 1994.

[25] Béla Bollobás, Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Time-series similarity problems and well-separated geometric sets. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, SCG '97, pages 454–456, New York, NY, USA, 1997. ACM.

[26] T. Kahveci and A. Singh. Variable length queries for time series data. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 273–282, 2001.

[27] Georges Grinstein. Keynote on Integrating Visualization and Analysis. In *Vizbi Conference*, 2012. `http://vizbi.org/2012/Presentations/Georges_Grinstein_Keynote.pdf`.

[28] WorlframMathWorld. Point-Line Distance–2-Dimensional. http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html.

[29] YouTube. Video Sharing Website. www.youtube.com.