Developing a Mathematical Model of Mutation Sequence in Tumour Development

K Atwell A. Berrington R. Woloszczuk L.Hutchinson I. Nazarov

January 17, 2013

Contents

Abstract

This project builds upon the work of Smallbone et al. [5] and subsequent developments by the A team [4]. We model the growth of a tumour using a hybrid cellular automata simulation. Tumour cells can be in a variety of states, reflecting whether they have mutated to become hyperplastic, glycolytic and/or acid resistant. The spread of each cell type is affected by three environmental factors: the availability of glucose and oxygen diffusing out from a blood vessel, and the pH, which is lowered by the production of acid by glycolytic cells. We investigated the potential for antiglycolytic drugs to slow tumour progression, and saw some evidence that they could prevent the establishment of glycolytic cell types. We then extended the model, adding additional cell states representing cells with increased their mutation rate. The conclusion was that the progression of the bulk of tumour cells from hyperplastic to glycolytic to acid resistant phenotypes is stable regardless of the initial mutation that occurs. Glycolytic and non glycolytic cells with the acid resistant and hyperplastic mutations can coexist for long periods. Finally, rapidly mutating cells decrease in frequency once the optimally adapted tumour genotype has been reached.

1 Introduction to the biological problem - Islom Nazarov

In 2008 it was recorded that 7.6 million people were dying each year from cancer related diseases worldwide, in spite of advances in our understanding of the pathophysiology and treatment of the disease. Overall cancer related diseases as a cause of death account for approximately 13% of all deaths each year.[1]

It is hypothesised that in general cancerous growth starts with an adaptation of hyper-plasticity (increased proliferation) in cells. Initially the cell survival and proliferation rate are constrained by level of oxygen, which is governed by oxygen diffusion limit away from blood vessels. As the level of available oxygen to the cells reduces, the microenvironment promotes cells adapted to generate ATP via the inefficient anaerobic pathway (glycolysis - more than an order of magnitude less efficient than its aerobic counterpart, producing only 2 ATP per glucose in comparison to approximately 36 ATP). This will cause cell proliferation towards the glucose diffusion limit.

Furthermore the by-product of glycolysis results in acidification of the extracellular environment (by production of lactic acid). The acidic microenvironment promotes proliferation of cellular acid resistance, this phenotype will have a greater proliferation advantage. This is because by reducing the surrounding pH it will kill its competitors, resulting in tissue damage due to cell death and degradation of the extracellular matrix, without having any toxic effect on itself.

The model we based our work on is a hybrid cellular automata simulation that investigates this progression of mutation.

2 Comments on the project we inherited - Kathryn Atwell

The quality of the work we received from our predecessors was extremely high, both in terms of the code and the report. The code was well organised into a manageable number of functions, with enough commenting that we could understand most files without looking at the documentation. A lot of effort had been made to vectorise wherever possible, to keep simulations from running slow. The report was succinct and covered all the main points, although it tended to focus on the implementation with less to say about the biological interpretation of their results. It was clear to us what license both code and report were covered by.

We had only a couple of minor issues with the project. There was an easily fixable bug in the visualisation function that caused colourbars to rescale misleadingly. It wasn't made clear in the report that some hypoplastic cells are also acid resistant, and units weren't given for the variables and parameters. The simulations are still a little time consuming, and the profiler indicates that the bottleneck is in the files Diffusion.m and Protons.m. It seems that using sparse allocation rather than making a dense matrix and then converting it to a sparse one can speed these functions up considerably, but at the cost of making the code far less readable. Overall, I would say the previous group gave us a good foundation on which to build.

3 Extracting useful data from simulations - Kathryn Atwell

The code we inherited displayed the state of the simulation as a series of subplots in a full screen figure. That figure then refreshes at each iteration. While this is helpful for getting a sense of how the simulation is progressing and for debugging, none of the data generated is stored permanently. When we started, there was no means to allow post processing of data, and no way to compare two runs with different settings. So our first job was to put these systems in place.

Options were added to RunSimulation.m to turn off all outputs to the screen and console, to avoid wasting time while generating data files. Next we created a storage matrix for each of the matrices that make up the simulation state; for example, GlucoseOutput to store Glucose level matrices. Every N iterations the current Glucose matrix is appended to the bottom of GlucoseOutput, forming a list of matrices over time. The same process is applied to all the matrices in the simulation state. When the simulation ends, all the Output variables are saved in a .mat file, and can be reloaded and processed later.

Once data could be stored, we pursued two different approaches to comparing simulation runs. One involved plotting the total number of cells in each state as a function of time. The other involved making two movies and playing them side by side in the same figure.

3.1 Plotting cell numbers

When processing data, we are usually just interested in how the number of cells in each state evolves in time. As such we wrote a function StatePlot.m that extracts states from the StateOutput matrix one iteration at a time, counts the number of cells in each state and stores them for plotting.

Comparing runs can still be difficult, because of the stochastic element to the simulation, which makes it unclear whether a particular result is a fluke. We solved this by running 6 repeats of each simulation and plotting them all with the above function, giving us an idea of what constitutes representative behaviour. We also added an option to allow mutant seeding, where a mutant cell of a particular type is placed in the centre of the basement layer at time 1. This evens out the rate of progression across runs, and avoids running for many iterations before seeing interesting behaviour.

3.2 Playing back movies side by side

Summing up the number of cells in each state means all the spatial information from the simulation is lost. One way to compare two runs in more detail is to play them back side by side. Therefore, we wrote a function, MakeMovie.m, that given a saved output matrix produces a .avi movie file. Working with .avi files is not ideal, especially because MATLAB fails to compress .avi files. However, it does put the struct of images into a form that can be played outside MATLAB and included in presentations. We recommend that anyone reusing our code compresses the movie files for long term storage using the tool ffmpeg, which is freely available under GPL.

The most efficient method to play simulations side by side would be to display frames one after another taken from the structs used to make the two movies. However, incase all you have available are the movie files, a second function, Plot2Movies.m has been written that can read from two .avi files frame by frame and play them next to each other in a figure.

Again, we had to consider how to make two runs more comparable. For making movies to compare side by side, seeding mutations is recommended. It is straightforward to add lines to the seeding mutations lines at the top of RunSimulations.m to set up a more complicated series of seeding events as required. Setting parameter pa to 0 will then switch off all additional mutations and make the simulation deterministic, if you are only interested in the effect of parameter changes on the spread of mutant cells.

4 Antiglycolytic Treatment for Cancer - Adam Berrington

4.1 Theory

The previous model aimed at capturing the progression of a tumour as cancer cells change their metabolic pathway. The cells are hence programmed to mutate randomly into a glycolytic form, which does not respire aerobically as is the case for the normal cells. The glycolytic cells then take over and by the end of the simulation have replaced all normal cells. The rate of glycolysis, k, has a strong influence on which cell types dominate. At a low k, hyperplastic cells are outcompeted by Hyperplastic, Acid Resistant, Glycolytic (HAG) cells, which can use glycolysis as an alternative food source in an oxygen-poor environment. At a high k, hyperplastic cells are able to proliferate more rapidly than HAG, as HAG are unable to satisfy their glucose need.

Glycolytic cells make use of the abundant glucose to metabolise, hence targeting this metabolic pathway via drugs is an option to prevent a tumour from surviving in hypoxic environments.

There are a few suggestions as to possible drugs which could target this pathway. These drugs are termed antiglycolytics and the most common 2-DG (2-deoxyD-glucose) is an analogue to glucose that cannot be metabolized. The effects of 2-DG has been tested and shown to be better in hypoxic environments, since cells rely on anaerobic respiration in this case. Normal cells also use this mode of respiration, therefore a drug which blocks this pathway needs to be designed appropriately.

Table 1: Parameters. For further detail on the model see report by the A team. [4]

k	glycolytic rate	10,100,1000
ht	tumour cell acidity threshold	8600
hn	normal cell acidity threshold	930
pa	adaptation rate	0.001, 0.0001
width	width of the grid	50
height	height of the grid	35
na	Number of ATP molecules produced during complete oxydation	36
a0	Minimum ATP for a cell to survive	0.1
dg	$1/dg^2$, where dg is diffusion coefficient for glucose	$1/130^2$
dc	$1/dc^2$, where dc is the diffusion coefficient for oxygen	1/25

4.2 Model

The inherited metabolic cancer model reproduced the work done by Smallbone et al. We can write the glucose consumption rate per cell of ϕ_g in terms of the extracellular glucose concentration in terms of the normal concentration, g. For a carcinogenic glycolytic cell, glucose consumption is increased significantly by a factor k, which is a ratio of the glycolytic cell consumption rate and normal rate, such that

$$\phi_g = \{ \begin{array}{ll} g & \text{Normal} \\ kg & \text{Glycolytic} \end{array} \tag{1}$$

If an antiglycolytic agent, such as 2DG, is administered, then the glucose pathway for the glycolytic cells is hindered and as such the value for the relative glucose consumption k should be reduced.

We have developed a drug model, in which a diffusive antiglycolytic agent is administered from the bloodstream throughout the simulation from the beginning.

The diffusive agent has a dimensionless concentration c, with a maximum value 1 at the blood vessel and diffuses up the simulation with a diffusion coefficient D_d. We require that where the concentration $c = 0$ (where there is no drug present), the original value of k is returned. The second requirement is that at $c = 1$, k is modified by a factor α – the drugs effectiveness. We assume a linear fit between these two conditions such that for the modified value of glucose uptake, k_{drug}

$$k_{drug} = k(c(\alpha - 1) + 1) \tag{2}$$

The value of k_{drug} replaces the value of k in the finite difference solution to the glucose diffusion equation explained in the previous model. This was added to the file ATPUpdate.m.

4.3 Optimum Drug Efficiency

The purpose of the model is to find by what factor does the glucose uptake constant need to be reduced in order for a drug to prevent glycolytic mutation. Thus we need to optimise the value of α for the simulation. For this, we can set c = 1 over the entire matrix to optimise for the ideal case – the diffusion coefficient of the drug $D_d \to \infty$ and all cells receive the same drug amount.

We ran the simulation for several different drug efficiencies, α with this setup for the standard parameters given in Table 1 . The results can be seen in Figure 1.

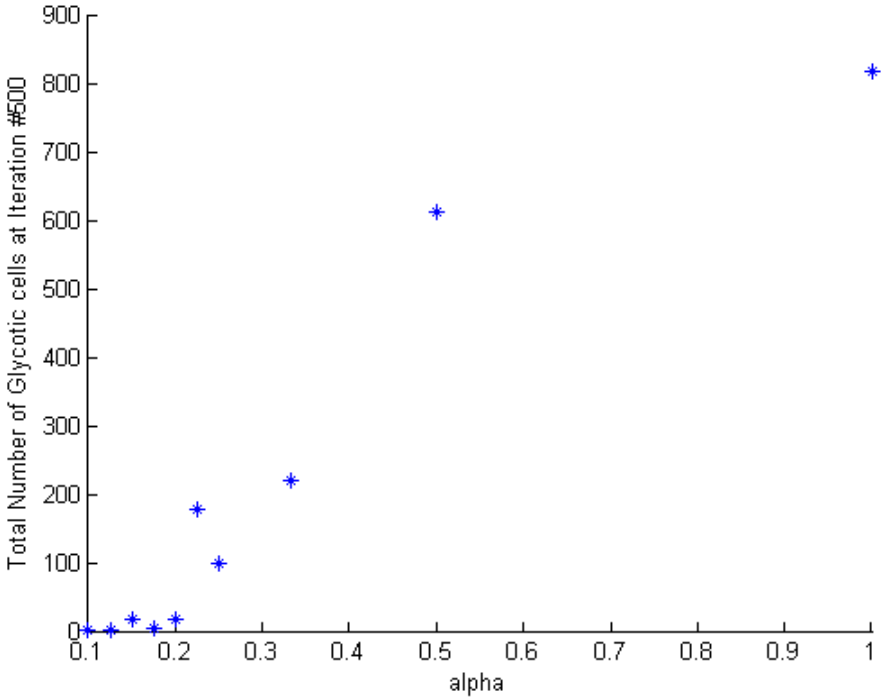

Figure 1: Average number of glycolytic cells present at iteration 500 for values of α ranging from 0.1 to 1. Points averaged over 5 runs and values < 0.1 observed no glycolytic cells. Plotted using the default values Table 1. Can see that α = 0.2 is a critical value for glycolytic tumour prevention – no glycolytic tumours observed.

4.4 Drug Diffusion

In order to gain an insight into the distribution of the drug molecule in the system, a diffusion coefficient of $D_d = 1.86 \times 10^{-6} cm^2/s$ was obtained from [6]. This is the value measured for 2-DG in the body. This value was then calculated using the same diffusion solver in the literature, except the diffusion was assumed not to be affected by presence of cells. The Diffusion.m file was modified to accommodate for this diffusion which was initialised with InitialiseDrug.m.

The results for α = 1 (control), α = 0.1 are shown below in Figure 2.

4.5 Remarks

The results of the simulations clearly show that blocking the glycolytic pathway via a drug which prevents the uptake of glucose by cancer cells is a useful strategy. The optimum value from this work was calculated to be α = 0.2, this corresponds to an 80% reduction in uptake rate of glucose for the cancer cells. For this work k = 10, therefore this value suggests that reducing the glycolytic mutant glucose rate to only twice that of the normal rate via drugs would hinder glycolytic cell proliferation.

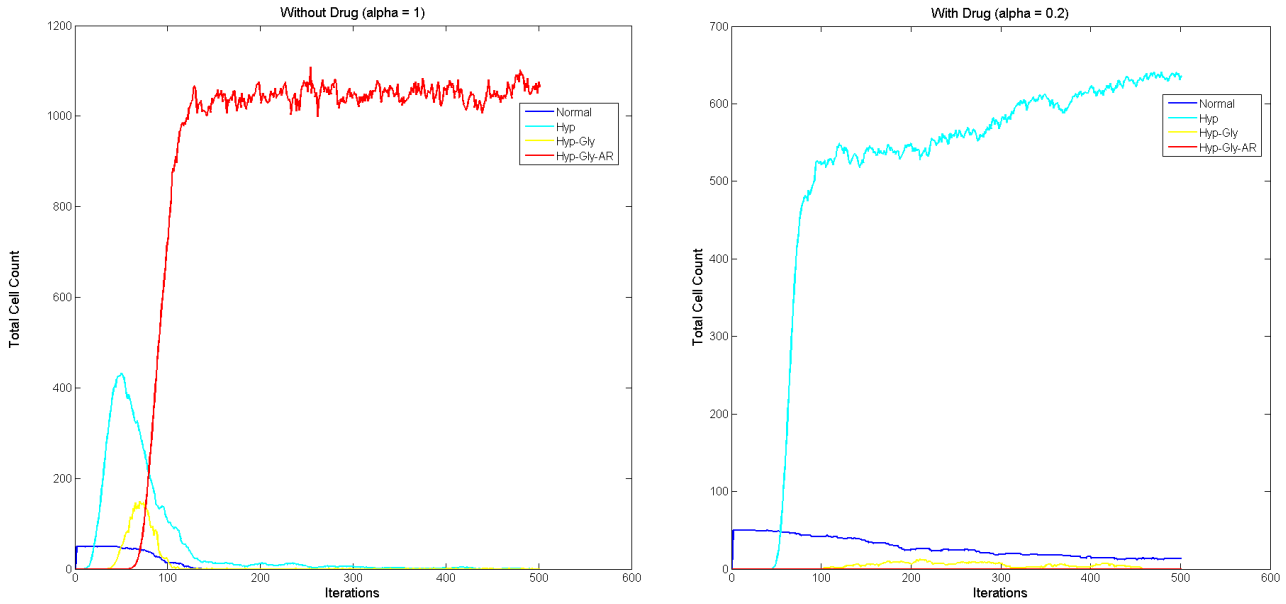

Figure 2: Total cell counts for each population plotted without antiglycolytic treatment (left) and with treatment α = 0.2 (right). The treatment has clearly surpressed the selection advantage for glycolytic cells (Red and Yellow) compared to the control. At 500 iterations the normal cells are all dead in the control, compared to the treated cells, which half still remain.

5 Introduction of "Mutator Gene" - Ronja Woloszczuk and Lucy Hutchinson

The original model allowed for cells to be in a limited number of possible states, depending on which mutations they had acquired. The cells begin as 'normal', with no mutations, and then develop to become either 'Hyperplastic' (H), 'Hyperplastic and Glycolytic' (HG) or 'Hyperplastic, Glycolytic and Acid Resistant' (HGAR). We found that there was room for improvement for two reasons:

- Cells can jump from having no mutations to having several
- Not all combinations of H, G and AR are represented

5.1 Changes to the model

Biologically, once cells become cancerous, they acquire mutations at a faster rate. We built this into our model by introducing a new mutation called 'Mutator' (M). Cells which pick up this mutation can then mutate faster. We implemented this by increasing the parameter for the mutation rate, pa, by a factor of 10 for cells with M. In order to allow cells to pick up and lose one mutation at a time, we used a 4-bit binary system. There are four bits representing H, G, AR and M. Each of these bits contains a 1 or a 0 depending on whether that cell has each mutation or not. On each iteration, one randomly selected bit can flip from 1 to 0 or vice versa, or stay the same. This is depicted in3. The use of the binary system can be seen in the matlab file mutate.m.

Building an extra state into the model, and allowing all combinations of genetic mutations to take place creates a total of 16 (2^4) possible states for cells. It would not be especially informative to be able to see 16 different colours during visualisations of the model. For this reason we visualise only the following states:

- Normal Cells
- Non-hyperplastic but mutated cells
- Hyperplastic only cells
- Hyperplastic and glycolytic (both mutator and non-mutator)

- Hyperplastic and acid resistant (both mutator and non-mutator)
- Hyperplastic, glycolytic and acid resistant (both mutator and non-mutator)

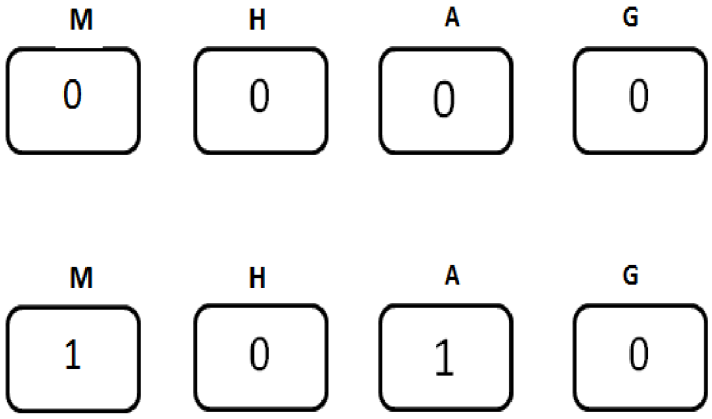

Figure 3: Binary system used for desribing the state of cells. (0000) represents a normal cell, and (1010) represents cells which have aquired the mutator and the acid resistant genes.

5.2 Results

The improved model gave some very interesting and biologically relevant results. We ran several simulations and used the StatePlot function to show the numbers of cells in each state. By planting a 'seed' cell into the original layer of normal cells with particular parameters, we could observe how cancer would develop and spread.

Figure 4 shows the results for a glycolytic seed at the first iteration. This shows that since glycolytic cells do not acquire hyperplasticity, they do not have a growth advantage over only-hyperplastic cells at the beginning. HGA and HGAM cells take over after about 600 iterations, but as can be seen from the plot on the right (showing the fraction of hyperplastic cells which also have M), once they are in this state there is a disadvantage to having M so the fraction of mutator cells decreases.

Figure 5 shows the results for a hyperplastic seed. The outcome is similar to that for a glycolytic seed. The number of hyperplastic cells increases first and is then replaced by acid resistant cells and finally HGA and HGAM cells which later become mostly HGA. Similar results are observed for an acid resistant seed.

Figure 6 shows the results for a hyperplastic seed with the mutator mutation. It is clear that on average the HGAM cells take over earlier in the simulation.

5.2.1 Discussion of Results

We found that hyperplastic cells spread in three distinct phases.

1. In the first phase, hyperplastic cells spread and proliferate. At this stage, no other cell types are present in such large numbers.
2. In the second phase, acid resistant hyperplastic cells accumulate as the concentration of acid increases and non acid resistant cells die off.
3. Finally in the third phase, cells acquire the glycolytic mutation and are able to survive in oxygen depleted environments. Cells without the glycolytic mutation are restricted to the oxygen rich region of the plot and are gradually outcompeted by HGA cells. After the cells become HGA cells, those which also have the mutator mutation have the disadvantage that they may readily lose it. Therefore the fraction of HGA cells without the mutator mutation increases rapidly.

Figure 7 shows the three distinct stages. These results appear realistic and logical, suggesting out model might be able to reproduce results typical of early stage cancer.

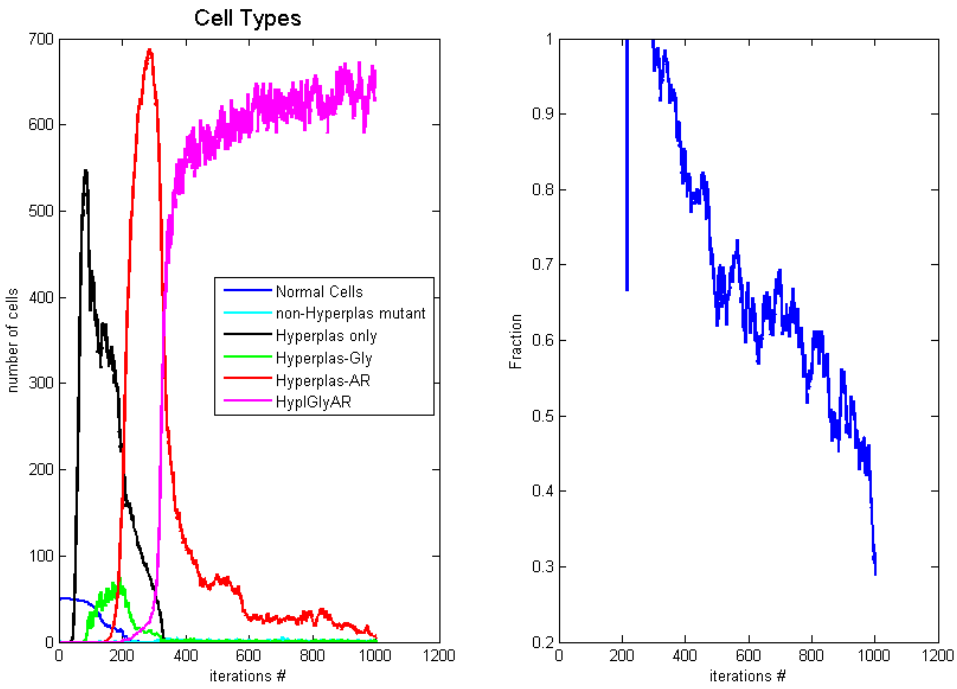

Figure 4: Plots showing the number of each cell type, and the fraction of hyperplastic cells which have the mutator mutation with glycotic seed. k=100

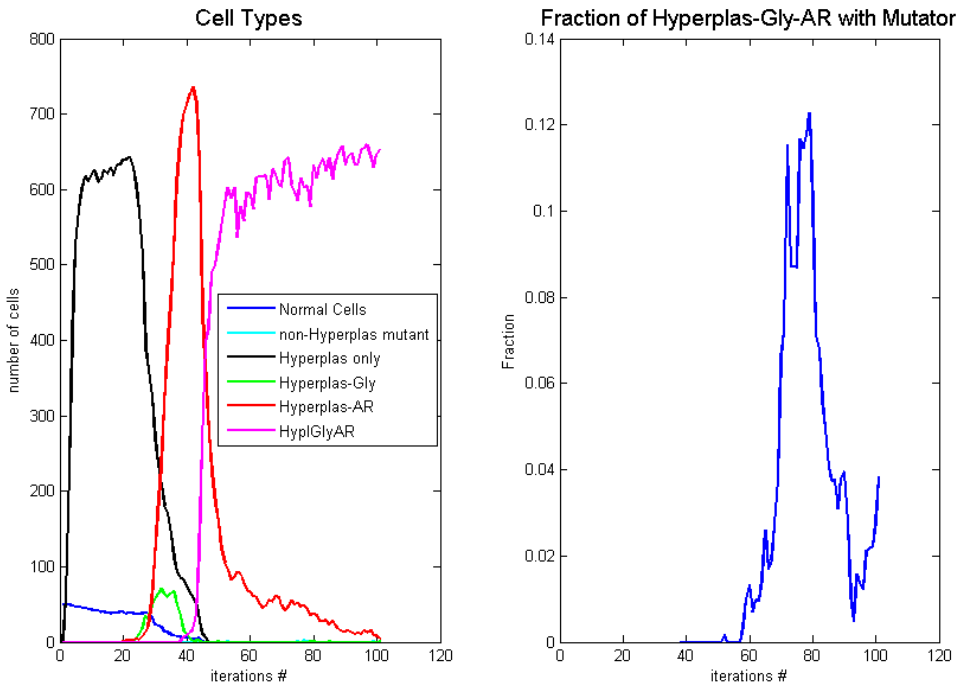

Figure 5: Plots showing the number of each cell type, and the fraction of hyperplastic cells which have the mutator mutation with hyperplastic seed. k=100

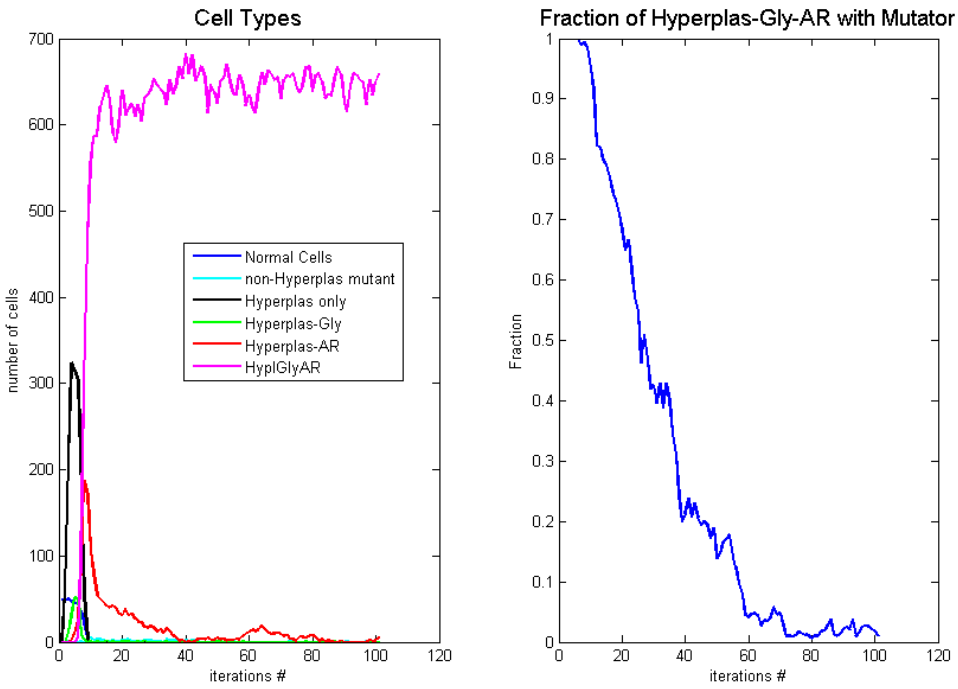

Figure 6: Plots showing the number of each cell type, and the fraction of hyperplastic cells which have the mutator mutation with hyperplastic seed. k=100

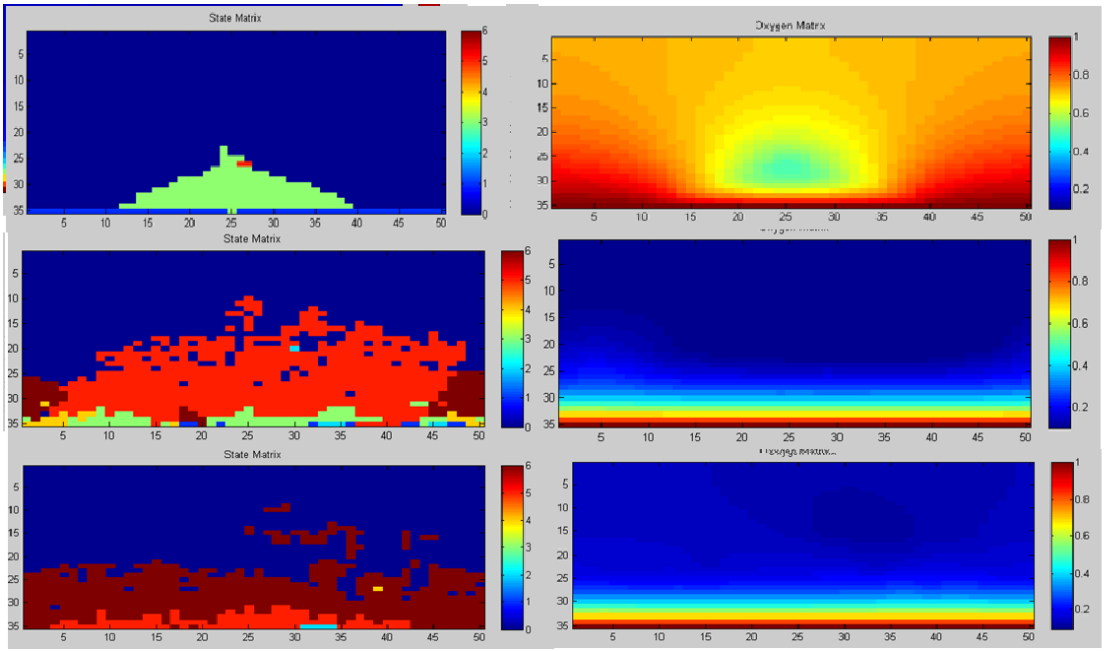

Figure 7: Three stages of the state matrix and the corresponding oxygen matrix for hyperplastic seed. Green shows hyperplastic cells, red shows acid resistant hyperplastic cells, brown shows hyperplastic, glycolytic and acid resistant cells.

6 Future work - Ronja Woloszczuk and Lucy Hutchinson

Further avenues for exploration are the inclusion of the motility of cancer cells representing a later stage of the disease, a function to demonstrate the decay of necrotic cells, and a way to represent angiogenesis in the cancer model. It would also be informative to analyse how glucose, ATP and proton concentrations vary with cell number.

Comment on our code

We'd like to draw attention to the fact that our code now consists of two versions. Since our model extension that adds extra cell states changes many files in the original model, it was decided to make that a separate version of the code. This allowed Adam to continue running drug simulations using the original model, while the rest of the group worked on the version with extra mutant states. The two versions are stored in separate folders in the GitHub repository managed by KathrynA.

References

[1] Jemal A, Bray, F, Center, MM, Ferlay, J, Ward, E, Forman, D (February 2011). "Global cancer statistics". CA: a cancer journal for clinicians 61 (2): 6990

[2] Rohzin, J., Sameni, M., Ziegler, G., Sloane, B.F., 1994. Pericellular pH affects distribution and secretion of cathepsin B in malignant cells. Cancer Res. 54, 65176525

[3] Gatenby, R.A., Gillies, R.J., 2004. Why do cancers have high aerobic glycolysis? Nat. Rev. Cancer 4, 891- 899

[4] Frot, B., Cox, O., Esperanca, P., Meiburg, A., Demharter, S., 2013. DTC-OSTI-MatlabCancerModelling-A Report. GitHub repository of the same name, admin benjaminfrot.

[5] Smallbone, K., Gatenby, R.A., Gillies, R.J, Maini, P.K., Gavaghan D.J., 2007. Metabolic changes during carcinogenesis: potential impact on invasiveness. J. Theor. Biol. 244, 703-713

[6] Cole, W.C., Garfield, R.E., Kirkaldy, J.S., 1985. Gap junctions and direct intercellular communication between rat uterine smooth muscle cells. Am J Physiol Cell Physiol. 249:C20-C31

[7] El Mjiyad, N., Caro-Maldonado, A., Ram? rez-Peinado, S., Mun? oz-Pinedo, C., 2011. Sugar-free approaches to cancer cell killing. Oncogene. 30, 253264