Data science and machine learning assessment – Iwan Williams

Automating the interpretation of calcium assay data.

Introduction/motivation

My lab has been sitting on data from a series of experiments for years (since around 2018) that proved too laborious to analyse. The output of the experiments analysed the amount of calcium in the cell over a period of time during which cells were treated with ionomycin. Ionomycin releases calcium from the endoplasmic reticulum (one of the two main calcium storage centres in the cell). By measuring how quickly calcium levels are returning to the baseline after the initial release we can assess the cell’s ability to clear excess calcium. Dysregulation in calcium levels is a common feature of neurodegenerative diseases. Because of this this experiment was conducted with 25 different neurodegenerative diseases and 5 different wild type cells (varying age of patients to account for range of diseased patient ages). We would like to assess how calcium clearance varies in between the wild type and diseased cells.

Upon visualising the data from this series of experiments I noticed a general pattern (as seen in the figure below). Calcium levels began low but raised quickly upon ionomycin treatment. The rise ended in a peak and began to return to normal. A lag phase preceded the primary decline (likely where the cellular response is taking place, signalling mechanisms etc). While returning to levels near the baseline the rate of decline slows and eventually plateaus into a trough. Finally, the calcium levels slowly rise due to lysosomal (the other major calcium storage centre in the cell) calcium release (a delayed response to the application of ionomycin).

Chart

Description automatically generated

Methods.

The data gathered during the experiments takes the form of a number of .csv files. Each csv file has the data for a number of different cells, which all correspond to the same time (sec) column. Because the number of cells in each .csv varies I had to ensure that each line of code could apply to any number of cells (between 1 and 50 per .csv which I deemed appropriate after searching through our data. I did this by using loops that ran through the data from cells 1 through 50 and ignored any missing values without outputting errors. Each cell had three columns of data. These consisted of two wavelengths (as the experiment was carried out using microscopy), and the ratio between the two. I am interested in the ratio values and as this was already calculated I could simply delete the wavelength data my deleting columns that met certain name criteria. The data for some cells was missing or included many NAN values. To account for this I removed these columns from the data frame before beginning my analysis, again I used a function that deletes columns based on their contents (in this case containing only missing or NAN values).

The area of the data that is of interest to us is the consistent decline after the lag period, before the slow plateau/trough. I tried a range of methods to automatically identify this region of the graph but settled on a function that identifies the longest straight(ish) line with a negative gradient. It compares each point on the graph to the next to attempt to build a line, the gradient of each point is compared to the last to ensure that the addition of this new point does not sway the average gradient of the line by more than a certain tolerance percentage. If it does not the line grows, if it does, then the function moves on to the next point to attempt to build a new line. This finds the longest ‘straight’ line (as defined by the tolerance). I ensured that the line identified was negative to avoid the positioning the line on the initial incline or the slow incline that results from lysosomal calcium release. I could avoid inclusion of the lag period or slow plateau by fine tuning changing the tolerance of the function. I found that a general tolerance of 0.3 consistently identified the primary decline, but where required this can be fine tuned for each cell (a plot of the line against the graph is created to visually check the fit of the lines).

A second function then calculated the average gradient of the lines plotted in each cell (each data frame contains numerous cells of the same type) in the data frame, which can then be compared to other diseases.

Results.

I found that this code consistently identified the region of interest in our data and therefore calculated accurate gradients. This will allow us to interpret the mass amounts of data generated by these experiments in a vastly shorter time frame.

Summary/Conclusion.

While I am pleased with the outcome of this project there are some shortcomings that I feel could be improved with further development. As there is no way to verify how well the line fits the primary decline the code must be supervised, I would like to develop either a better method to identify the primary decline, or a way to verify the accuracy of the line once generated with the method stated above. This would reduce the supervision requirement and further decrease the amount of time required to conduct the analysis. A further problem is that while the plots are consistent enough to generate a gradient for the primary decline, they do not always span the whole length of the decline. If the line more accurately spanned the whole of the primary decline it would be useful to calculate the area under the line.