Protocol: Adding NMDA Receptor Antagonists, cortisol, and carbenoxolone

1. Always prepare or add compounds inside the biohood, following all cleaning and sterilization procedures.
2. We need to calculate the volumes we need to prepare. Looking at the groups:

Groups

- **Control/Cortisol (10 MEAs)**
- **Memantine/ Cortisol (10 MEAs)**
- **Lanicemine/ Cortisol (10 MEAs)**
- **Traxoprodil/ Cortisol (10 MEAs)**

Calculations: **Mass = Concentration x Volume x Molecular Weight, C1*V1 = C2*V2**

Cortisol: 5 different concentrations of cortisol will be used ranging from a very low dose to a very high dose. They are 0.138, 0.2756, 0.5518, 1.1, 2.2 uM. MW: 362.46 g/mol

Lowest Dose: 8 dishes * 5 ug/dl = 40 ug/dl * 1g/10^6ug * 10dl/L * 1mol/362.46 g *10^6uM/1M = 1.1 uM

Low Dose: 8 dishes * 10 ug/dl = 80 ug/dl * 1g/10^6ug * 10dl/L * 1mol/362.46 g *10^6uM/1M = 2.2 uM

Medium Dose: 8 dishes * 20 ug/dl = 160 ug/dl * 1g/10^6ug * 10dl/L * 1mol/362.46 g *10^6uM/1M = 4.4 uM

High Dose: 8 dishes * 40 ug/dl = 320 ug/dl * 1g/10^6ug * 10dl/L * 1mol/362.46 g *10^6uM/1M = 8.8 uM

Highest Dose: 8 dishes * 80 ug/dl = 640 ug/dl * 1g/10^6ug * 10dl/L * 1mol/362.46 g *10^6uM/1M = 17.6 uM

Total: 1.1 + 2.2 + 4.4 + 8.8 + 17.6 = 34.1 uM

Volume: 40 dishes * 0.5 mL = 20 mL + 1 = 21 mL

Make whole stock solution:

Take 1 mg of cortisol and mix with 20 mL of absolute ethanol.

1 g/20L * 1 mol/362.46 g * 1000000 uM/M = 138 uM

Make final solutions:

Lowest dose:

138 uM * V1 = 0.138 uM * 4.2 mL

V = 4.2 uL of main solution + 4195.8 uL of DMEM+

Low dose:

138 uM * V1 = 0.2756 uM * 4.2 mL

V = 8.4 uL of main solution + 4191.6 uL of DMEM+

Med dose:

138 uM * V1 = 0.5518 uM * 4.2 mL

V = 16.8 uL of main solution + 4183.2 uL of DMEM+

High dose:

138 uM * V1 = 1.1 uM * 4.2 mL

V = 33.6 uL of main solution + 4166.4 uL of DMEM+

Highest dose:

138 uM * V1 = 2.2 uM * 4.2 mL

V = 67.2 uL of main solution + 4132.8 uL of DMEM+

Carbenoxolone: Concentration of 50 uM. MW: 614.72 g/mol

Total: 30 dishes * 50 uM = 1.5 mM

Volume: 30 dishes * 0.4 mL = 12 mL + 1 = 13 mL

Make stock solution:

Take 1 mg of carbenoxolone and mix with 15 mL of sterilized deionized water.

1 g/15L * 1 mol/614.72 g * 1000000 uM/M = 108.45 uM

Make final solution:

108.45 uM * V1 = 50 uM * 13 mL

V1 = 6 mL of stock solution and 7 mL of DMEM+

Memantine: Concentration of 10 uM. MW: 215.76 g/mol

Total: 8 dishes * 10 uM = 80 uM

Volume: 8 dishes * 0.1 mL = 0.8 mL + 1 = 1.8 mL

Make stock solution:

Take 1 mg of memantine and mix with 1 mL of sterilized deionized water.

1 g/L * 1 mol/215.76 g * 1000000 uM/M = 4635 uM

Make final solution:

4635 uM * V1 = 10 uM * 1.8 mL

V = 3.9 uL of main solution + 1796.1 uL of DMEM+

Lanicemine: Concentration of 10 uM. MW: 271.19 g/mol

Total: 8 dishes * 10 uM = 80 uM

Volume: 8 dishes * 0.1 mL = 0.8 mL + 1 = 1.8 mL

Make stock solution:

Take 1 mg of Lanicemine and mix with 10 mL of sterilized deionized water.

1 g/10L * 1 mol/271.19g * 1000000 uM/M = 369 uM

Make final solution:

369 uM * V1 = 10 uM * 1.8 mL

V = 48.8 uL of main solution + 1751.2 uL of DMEM+

Traxoprodil: Concentration of 10 uM**.** MW: 327.42 g/mol

Total: 8 dishes * 10 uM = 80 uM

Volume: 8 dishes * 0.1 mL = 0.8 mL + 1 = 1.8 mL

Make stock solution:

Take 1 mg of Traxoprodil and mix with 10 mL of DMSO.

1 g/10L * 1 mol/327.42 g * 1000000 uM/M = 305 uM stock solution.

Make final solution:

305uM * V1 = 10uM * 1.8 mL

V = 59 uL of main solution + 1741 uL of DMEM+

3. Prepare the stock solutions using the above calculations. Put the solutions in a sterile mL falcon tube (remember to mix the solution before and after preparation).
4. After the solutions are prepared, the electrical recordings can begin.
5. Administer 0.4 mL of the Carbenoxolone solution to the experimental groups through the wall of the plate.
6. After one hour, record the electrical activity for 10 minutes of the MEAs for the control and experimental groups.
7. Combine the appropriate 0.5 mL of cortisol concentration/ 0.1 mL of NMDA receptor antagonist in a separate falcon tube. Make sure to mix the solution.
8. Administer the combined cortisol concentration/NMDA receptor antagonist solution to the experimental groups and just cortisol to the control groups. Both should be applied directly to the center of the dish.
9. After 90 minutes, record the electrical activity for 10 minutes.
10. Record the electrical activity of the MEAs every 24 hours for an additional two days.

11. If there is extra time, repeat experiment using same steps. This would be too emulate “injections” of ketamine (in this case 3 other NMDA receptor antagonists) and see if the results agree with the literature on how multiple injections of ketamine lead to longer antidepressant effects.
12. Otherwise, discard the medium and clean the MEAs

Notes:

i) In the case of the control, Carbenoxolone is replaced with 0.5 mL of sterile deionized water.

ii) The naming convention of file names is as follows: Group_LL_Dish#_Day#_BFT

- Ex. Control_L_19_Day1_BFT
- Ex2. Mem_H_14_Day1_AFT
- Legend
- LL = Lowest Dose of cortisol
- L = Low dose of cortisol
- M = Medium dose of cortisol
- H = High dose of cortisol
- HH = Highest dose of cortisol
- BFT = Before treatment of NMDA receptor antagonist and cortisol
- AFT = After treatment of NMDA receptor antagonist and cortisol
- Control
- Mem = Memantine
- Lan = Lanicemine
- Tra = Traxoprodil

Analysis of electrical activity

1. Data processing and analysis will be done using the Python programming language.
2. The data received from NeuroExplorer will be in .mat format which can be read in Python.
3. The code below reads in the .mat data files in the working directory and retrieves: the mean number of spikes in each plate (sum of spikes / 30 channels), the sem of spikes between each unit, and the number of units in each dish and assigns the data into a dictionary.

```
from scipy.io import loadmat

import os

from scipy.stats import sem
```

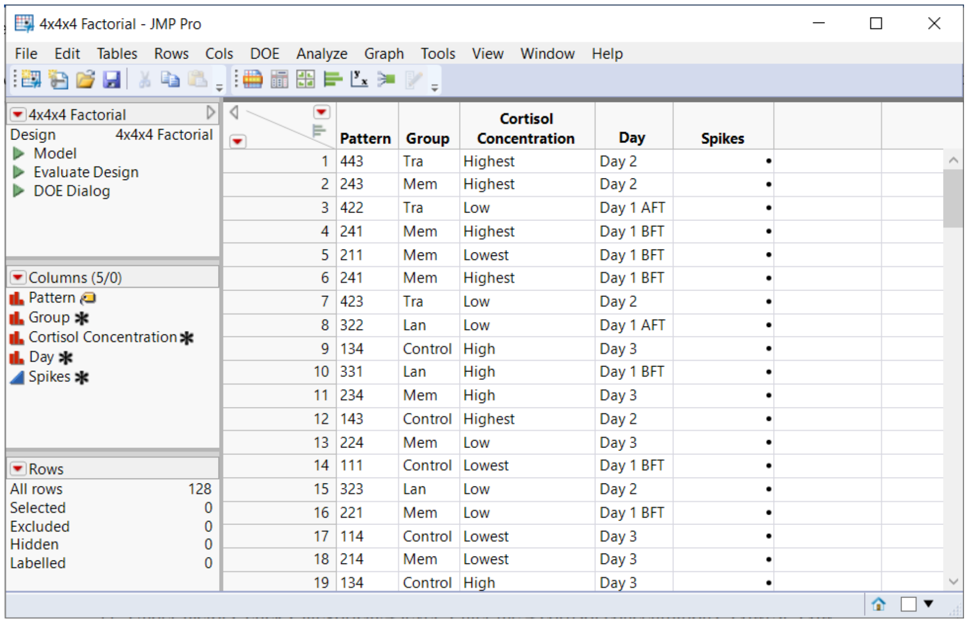
```
# %% Load in data in directory
data_dict = {}
for item in os.listdir():
    if item != 'OSCAR_Project.py':
        raw = loadmat(item)
        raw.pop('__header__')
        raw.pop('__globals__')
        raw.pop('__version__')



        num_spikes = []

        for key in raw.keys():
                num_spikes.append(len(raw[key]))

    # Puts in mean number of spikes in dish, sem of spikes, and num of units
        data_dict[item] = [sum(num_spikes)/30,sem(num_spikes),len(raw)]
```

4. The two dishes of each treatment will be combined and plotted with the code below:

```
import re
combined_dict = {}
combined_datframe = pd.DataFrame()
for key in data_dict.keys():
    list_terms = key.split('_')
    pattern = '(?=.*%s)(?=.*%s)(?=.*%s)(?=.*%s).*' %
(list_terms[0],list_terms[1],list_terms[3],list_terms[4])
    for key2 in data_dict.keys():
            if re.findall(pattern,key2) != []:
                temp_key = '_'.join([list_terms[0],list_terms[1],list_terms[3],list_terms[4]])
```

```
        if temp_key in combined_dict:
            if data_dict[key][0] not in combined_dict[temp_key]:
                combined_dict[temp_key].append(data_dict[key][0])
                combined_datframe = combined_datframe.append({'Group':list_terms[0],'Cortisol_Concentration':list_terms[1],'Day':list_terms[3]+'_'+list_terms[4],'Spike1':combined_dict[temp_key][0],'Spike2':combined_dict[temp_key][1]},ignore_index=True)
        else:
            combined_dict[temp_key] = [data_dict[key][0]]



dfm = combined_datframe.melt(id_vars = ['Group','Cortisol_Concentration','Day'],value_name=('Spike'))



sns.set_theme()
sns.catplot(data=dfm, kind="bar", x="Day", y="Spike",hue='Day',row='Cortisol_Concentration',capsize=0.2,col='Group',row_order=['LL','L','H','HH']).set_xticklabels(['Day1 BFT','Day1 AFT','Day2','Day3'])
```

5. From here, the analysis of the factorial experiment will be performed using JMP.
6. Open JMP
7. Click DOE/Classical/Full Factorial Design
8. Start to put in the factors
9. The response variable can be renamed to Spikes
10. Under factors, click Categorical/4 level. Enter the 4 groups: Control, Mem, Lan, Tra.
11. Under factors, click Categorical/4 level. Enter the 4 cortisol concentrations: Lowest, Low, High, Highest.
12. Under factors, click Categorical/4 level. Enter the 4 days: Day 1 BFT, Day 1 AFT, Day 2, Day 3
13. Click Continue
14. For Number of Replicates, put 1 and then click Make Table. The table should look like:


4x4x4 Factorial - JMP Pro
File Edit Tables Rows Cols DOE Analyze Graph Tools View Window Help

4x4x4 Factorial
Design 4x4x4 Factorial
Model
Evaluate Design
DOE Dialog

Columns (5/0)
Pattern
Group
Cortisol Concentration
Day
Spikes

Rows	
All rows	128
Selected	0
Excluded	0
Hidden	0
Labelled	0

	Pattern	Group	Cortisol Concentration	Day	Spikes
1	443	Tra	Highest	Day 2	•
2	243	Mem	Highest	Day 2	•
3	422	Tra	Low	Day 1 AFT	•
4	241	Mem	Highest	Day 1 BFT	•
5	211	Mem	Lowest	Day 1 BFT	•
6	241	Mem	Highest	Day 1 BFT	•
7	423	Tra	Low	Day 2	•
8	322	Lan	Low	Day 1 AFT	•
9	134	Control	High	Day 3	•
10	331	Lan	High	Day 1 BFT	•
11	234	Mem	High	Day 3	•
12	143	Control	Highest	Day 2	•
13	224	Mem	Low	Day 3	•
14	111	Control	Lowest	Day 1 BFT	•
15	323	Lan	Low	Day 2	•
16	221	Mem	Low	Day 1 BFT	•
17	114	Control	Lowest	Day 3	•
18	214	Mem	Lowest	Day 3	•
19	134	Control	High	Day 3	•

15. After the data is collected, put in the data in the necessary locations. Note: It's randomized on purpose.
16. After that, click on Analyze/Fit Model
17. The Y variable should be the Spikes column.
18. Highlight the categorical variables. Then, go to the Macros button on the bottom and click Full Factorial
19. Scroll down to the Effect Summary and Parameter/Scaled Estimates tabs and open. It will list the individual and interactions and the p-values.

Statistical Notes/Background

1. The following section describes in detail the statistical methods that will be used in this experiment.

2. The analysis of the electrical activity obtained in this experiment is to determine how the concentration of cortisol and type of NMDA receptor antagonist affects the sustained increase in hippocampal neural firing rate. Day is also a variable of interest.
3. This project will conduct a 2^3 factorial experiment analysis since there are three variables of interest (cortisol concentration, NMDA receptor antagonist, day). (If "injection" was included there would be four variables and a 2^4 factorial experiment would be performed.)

The data table would look like:

Group	Cortisol Concentration	Day 1 BFT	Day 1 AFT	Day 2	Day 3
Control	Lowest				
	Low				
	High				
	Highest				
Mem	Lowest				
	Low				
	High				
	Highest				
Lan	Lowest				
	Low				
	High				
	Highest				
Tra	Lowest				
	Low				
	High				
	Highest				

a) Note: The Med Concentration of cortisol will be used for immunofluorescence

b) Note: If one variable has no significant effect on hippocampal neuron firing rate, a two-way ANOVA will be conducted.

c) Note: The table used for graphing purposes will have one column containing the names of the days and the last column of spiking.

d) Note: A sign table is normally used for a factorial experiment where + is where the factor is at its high level and – where the factor is at its low level.

4. A table of the immunofluorescence image analysis is shown below. This is a two-way ANOVA that is going to be performed twice: one where the response variable is maximal dendrite length and the other where the response variable is dendrite number.

	Maximal Dendrite Length (um)		Dendrite Number	
	Before Treatment	After Treatment	Before Treatment	After Treatment
Control				

Mem				
Lan				
Tra				

5. One of the requirements for a two-way ANOVA to hold is that the additive model has to hold. The additive model applies when the interactions all equal to zero. To determine this on the two-way ANOVA table, look at the p-value for the interaction between the group and treatment. If the p-value is above 0.05, then the additive model holds and the two-way ANOVA can be properly used.
6. If the additive model does not hold, then a table of the cell means will be made including the row and column means and compare.
7. If the additive model holds and if there is a significant difference in groups and/or treatments, then a post-hoc test will be conducted to determine which ones are different
8. The post-hoc test that will be used will be Tukey's method for multiple comparisons.

Immunofluorescence Protocol

1. The immunofluorescence protocol has been copied from the one in Microsoft Teams and pasted here with a few additions.
2. The "Medium Concentration of cortisol" group will be used for immunofluorescence

<u>Plate Preparation</u>

1. Spray ethanol 70% on the biohood work surface;
2. Place the sterile materials inside the biohood (remember: spray ethanol 70% before):
    - Pipette tip 1000;
    - Pipette tip 200;
    - Pipette tip 10;
    - Sterile circular coverslips;

- 24 well plate;
- Tweezers;
- Beaker for disposal.

3. Turn-on the UV-light for 15 minutes;
4. Thaw the poly-D-lysine (100 uL per well);
5. Put gloves and spray ethanol 70%;
6. Inside the biohood open the 24 well plate;
7. Using the tweezer take a coverslip (they are very thin, be careful to don't take more than one at a time);
8. Put the coverslip into the well;
9. Repeat the steps 7 and 8 until a coverslip is placed in each well (as needed);
10. With the pipette tip 1000 add 100 uL of poly-D-lysine to the center of each coverslip;
11. Close the plate and place it in the incubator (at least 20 minutes, but you can keep for a few days if needed);
12. Carefully remove and discard the poly-D-lysine (do not touch the cover slip surface);
13. Place the plate in the incubator and keep it until the time of plating.

<u>Plating (see the protocol: Primary Culture)</u>

1. Calculate the aliquot volume of cell suspension to plate 4-5 x 10^4 cells in each well (for example: after counting, the number of cells found was 30.7 x 10^5 per mL. Then:

   30.7 x 10^5 – 1 mL

   0.4 x 10^5 – x

   X = 0.013 mL OR 13 uL per well

2. Add the volume found in the center of each coverslip;
3. Place the plate in the incubator for 1-4 hours;
4. After this period carefully add (by the wall) 1 mL of DMEM 5/5 or Neurobasal + in each well;
5. Place plates in incubator;
6. After 4 days discard 50% of the medium and add 50% of DMEM +;
7. Take pictures of the plate under the microscope to follow the development of the culture and observe possible contaminations;
8. Repeat the step 6 and 7 every 4 days for 30 days (or for the necessary time determined in the experimental design).

<u>Experiment</u>

1. Prepare the solutions of the substances to be tested with the same type of medium used during the culture maintenance (DMEM + or Neurobasal +). Use warmed medium and consider the volume needed in each well: 700 - 1000 uL;
2. Completely remove the culture medium from each well (slightly tilt the plate and aspirate the medium through the wall);

3. Carefully add (by the wall) the solutions prepared in step 1;
4. Incubate for the necessary time determined in the experimental design.

<u>Fixation with metanol</u>

1. After completing the experiment, completely remove the solution from each well;
2. Carefully add (by the wall) 700-1000 uL of PBS 1x in each well;
3. Completely remove the PBS 1x from each well (slightly tilt the plate and aspirate through the wall);
4. Repeat the steps 2 and 3 two more times;
5. Carefully add 800 uL of methanol to each well;
6. After 3 minutes remove and discard the metanol;
7. Carefully add (by the wall) 700-1000 uL of PBS 1x in each well;
8. Completely remove the PBS 1x from each well (slightly tilt the plate and aspirate through the wall);
9. Repeat the steps 2 and 3 two more times;
10. Proceed with the protocol or add 1 mL of antifreeze solution to each well and keep the plate in the freezer until it is possible to continue the protocol.

<u>Immunofluorescence</u> (see instructions for preparing the antibodies and solutions used in this protocol at: solutions and antibodies – protocol) - can be performed outside of biohood

1. Carefully add (by the wall) 700-1000 uL of PBS 1x in each well (if the plate is with antifreeze solution, completely remove it and then add the PBS 1x. Wash it with the PBS 1x two more times);
2. Completely remove the PBS 1x from each well (slightly tilt the plate and aspirate through the wall);
3. Add 700 uL of BSA (bovine serum albumin) 5% in each well;
4. Place the plate on the shaker (light shaking – 20 rpm) for 1 hour;
5. Completely remove the BSA from each well (slightly tilt the plate and aspirate through the wall);
6. Carefully add (by the wall) 700-1000 uL of PBS 1x in each well;
7. Completely remove the PBS 1x from each well;
8. Repeat the steps 6 and 7 two more times (do not remove the PBS after the third wash);
9. Cover the plate lid with parafilm, taking care to not create rips or wrinkles;
10. Add 40 ul of primary antibody to the plate lid, at the position corresponding to the center of each well (be careful to not make air bubbles);
11. With a watchmaker tweezers carefully pick up the coverslip and place on top of the lid, with the upper side facing the antibody drop (the side containing the cells must be in direct contact with the antibody), in the position corresponding to the well where the coverslip was taken;
12. Repeat the step 11 for each coverslip;
13. Cover a tray with a paper towel dampened with water;
14. Carefully place the plate inside the tray and cover the tray with plastic wrap;
15. Place the tray in the fridge overnight;

16. The following day, remove the plate from the refrigerator and, using watchmaker tweezers, carefully return each coverslip to its respective well, remembering to leave the side that was in contact with the antibody facing up;
17. Carefully add (by the wall) 700-1000 uL of PBS 1x in each well;
18. Completely remove the PBS 1x from each well;
19. Repeat the steps 17 and 18 two more times (do not remove the PBS after the third wash);
20. With a kimwipe and alcohol, clean the surface of the lid covered with parafilm;
21. **IN THE DARK**, Add 40 ul of secondary antibody to the plate lid, at the position corresponding to the center of each well (be careful to not make air bubbles);
22. With a watchmaker tweezers carefully pick up the coverslip and place on top of the lid, with the upper side facing the antibody drop (the side containing the cells must be in direct contact with the antibody), in the position corresponding to the well where the coverslip was taken;
23. Repeat the step 11 for each coverslip;
24. Carefully place the plate inside the tray and cover the tray with foil paper;
25. Keep at room temperature, in the dark, for 2 hours;
26. Then, still in the dark, using watchmaker tweezers, carefully return each coverslip to its respective well, remembering to leave the side that was in contact with the antibody facing up;
27. Carefully add (by the wall) 700-1000 uL of PBS 1x in each well;
28. Completely remove the PBS 1x from each well;
29. Repeat the steps 27 and 28 two more times;
30. Add 700 uL of the dapi solution to each well;
31. After 3 minutes remove the dapi;
32. Carefully add (by the wall) 700-1000 uL of PBS 1x in each well;
33. Completely remove the PBS 1x from each well;
34. Repeat the steps 27 and 28 two more times (do not remove the PBS after the third wash);
35. Clean the slides with alcohol 70% and identify with name, date, experimental group;
36. Add 5-10 uL of Fluoromount on each slide (you can place 2 coverslips on each slide, then add two drops of the fluoromount and leave enough distance between them);
37. With a watchmaker tweezers carefully pick up the coverslip and tap lightly on a paper towel to remove the excess of PBS;
38. Place the coverslip on the slide, with the upper side facing the fluoromount drop (the side containing the cells must be in direct contact with the fluoromount);
39. Repeat the steps 37 and 38 for each coverslip;
40. Place the slides in a horizontal position in an appropriate tray or box, in a dry and dark place, for drying.
41. After drying, take to the fluorescence microscope for analysis and photo capture (focus cells and take pictures of approximately 20 different fields on each coverslip, prioritizing single cells, without overlapping).

Naming Convention:

Group_Dish#_Pic#_BFT or AFT

Ex. Control_14_Pic1_BFT

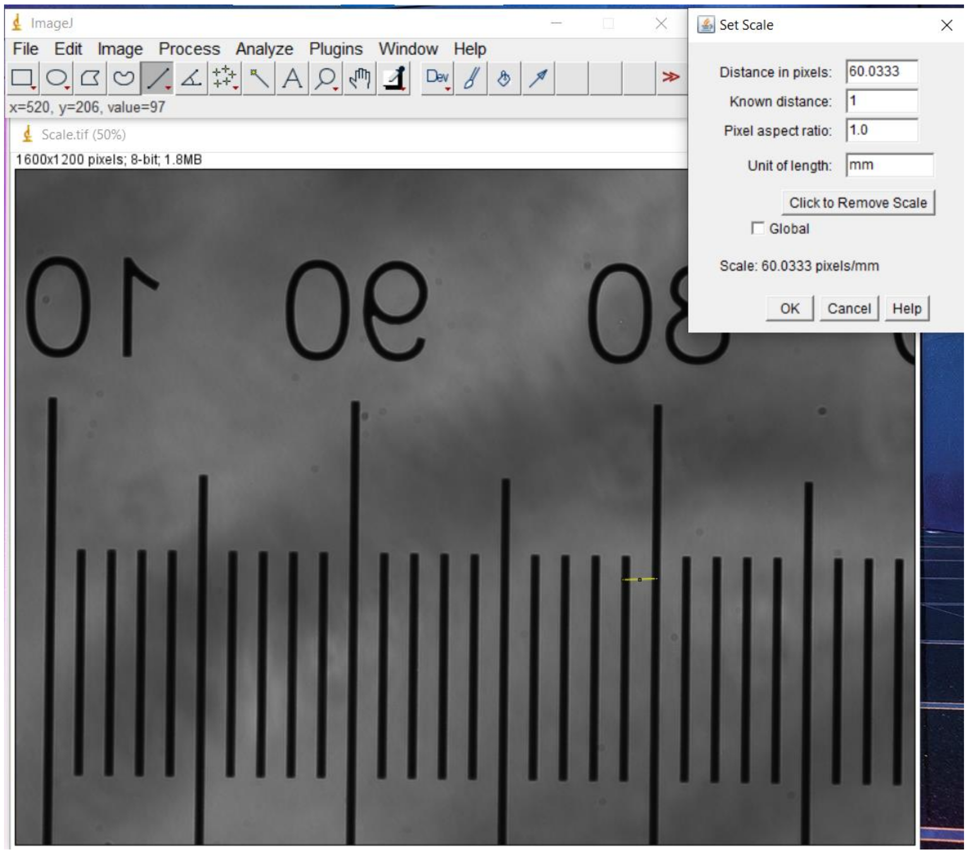
Immunofluorescence Analysis

1. Analysis of the immunofluorescence images will be done using ImageJ
2. From the protocol, 20 images of individual neurons will be obtained from each group (control, Mem, Lac, Tra of the Medium Cortisol Concentration treatment) before and after treatment for a total of 160 images.

a) Note: If the immunofluorescence images are too blurry for analysis, it is possible that deconvolution could be performed. (More information below)

3. The analysis will determine if there is any statistical difference in dendritic length and dendrite number between the treatments.

4. Open ImageJ. Using the line tool, first set the scale. The units will be in pixels. To convert units, go to Analyze/Set Scale/Click Global/Click OK. Use an image of known distance. Example below: The distance between two adjacent tick marks is 0.1 um. It would be a good idea to take multiple measurements and average them. Note: The unit of length is incorrect in the picture. It should be um. Also, note the dimensions of the image below (1600 x 1200 pixels) which should be the same dimensions as the immunofluorescence images.

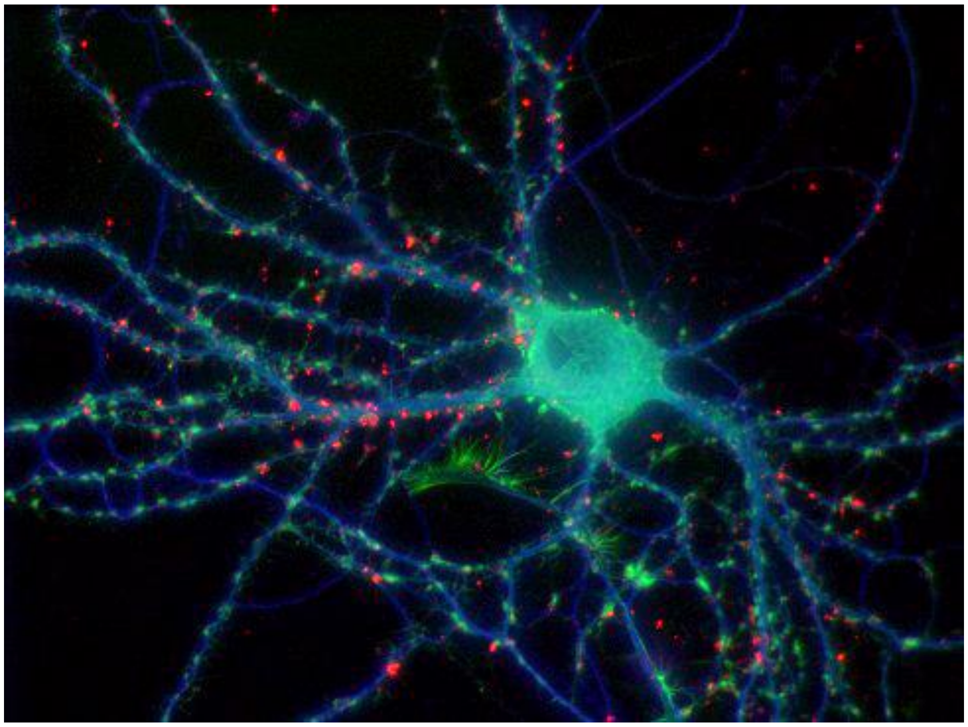


x=520, y=206, value=97

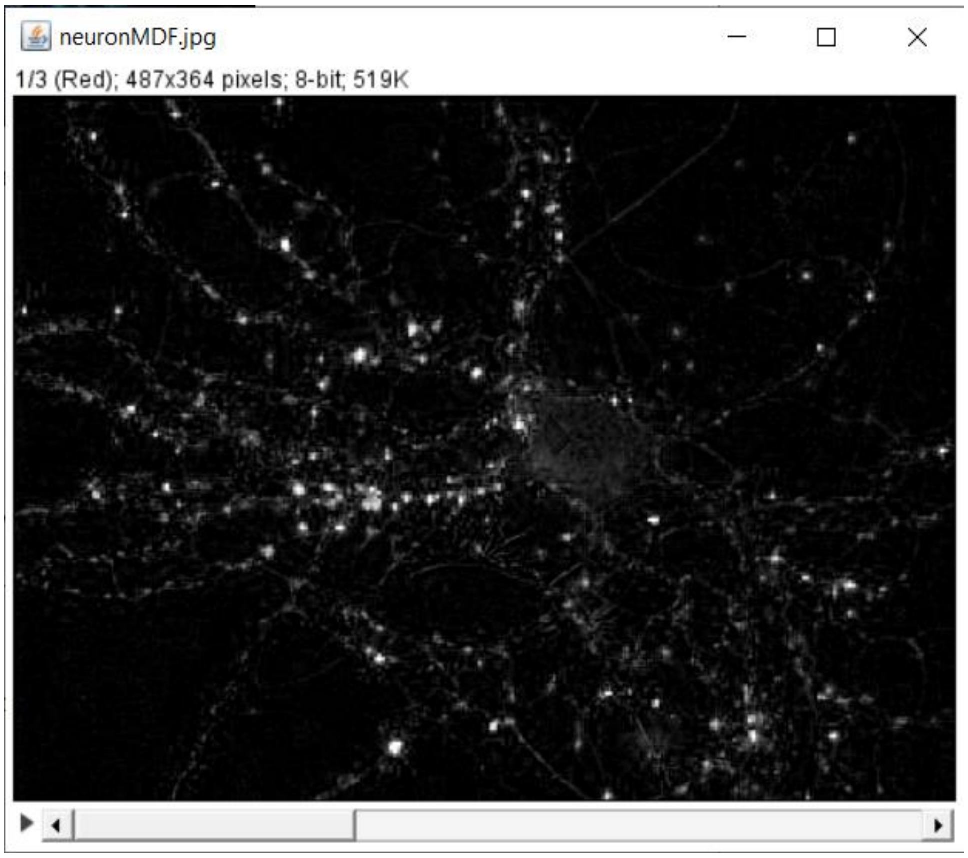
1600x1200 pixels; 8-bit; 1.8MB

Distance in pixels: 60.0333
Known distance: 1
Pixel aspect ratio: 1.0
Unit of length: mm
☐ Global
Scale: 60.0333 pixels/mm

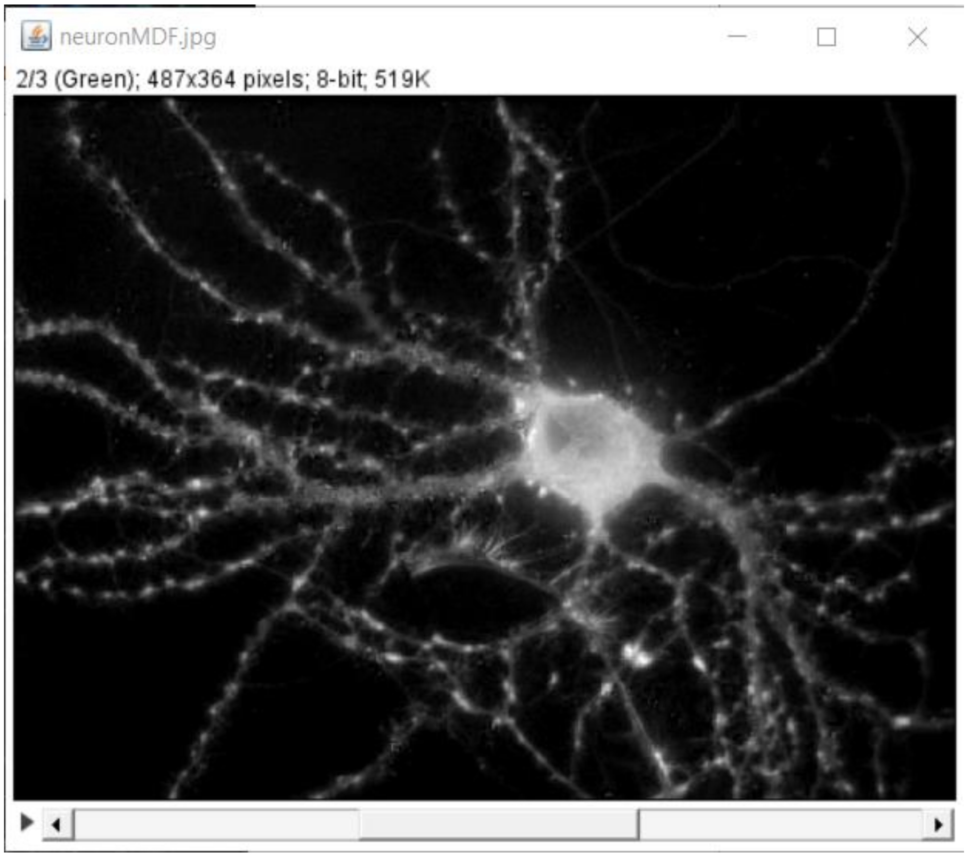
5. Next, open an immunofluorescence image. Example below:



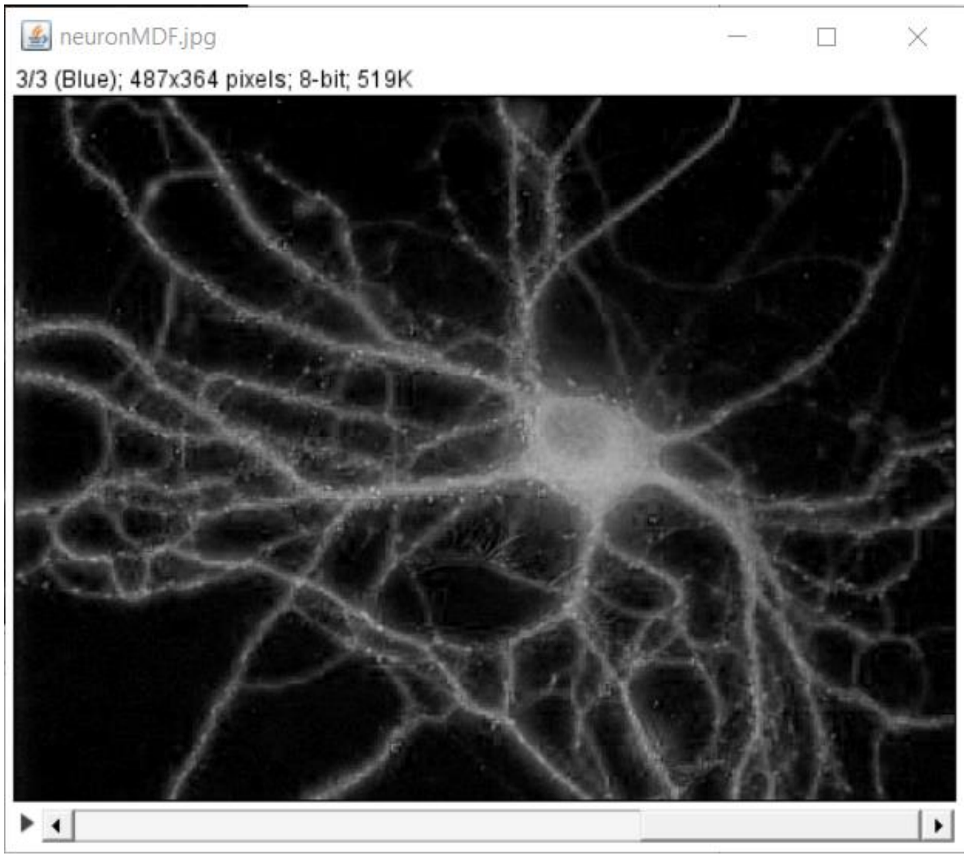
6. Next, go click on Image/Type/RGB Stack. 3 images will be produced:



1/3 (Red); 487x364 pixels; 8-bit; 519K

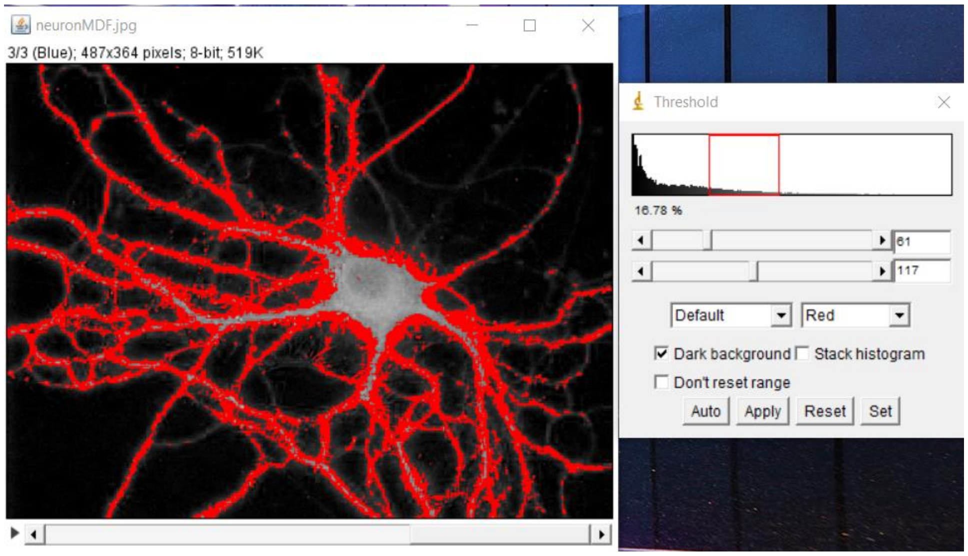


2/3 (Green); 487x364 pixels; 8-bit; 519K

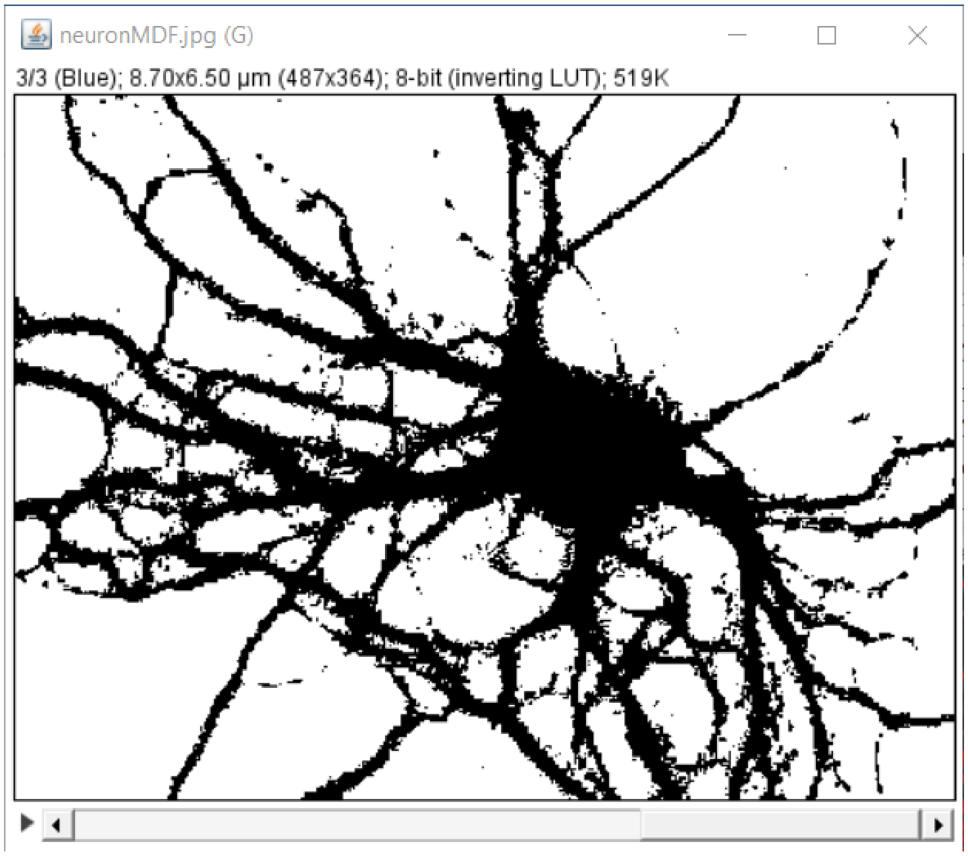


3/3 (Blue); 487x364 pixels; 8-bit; 519K

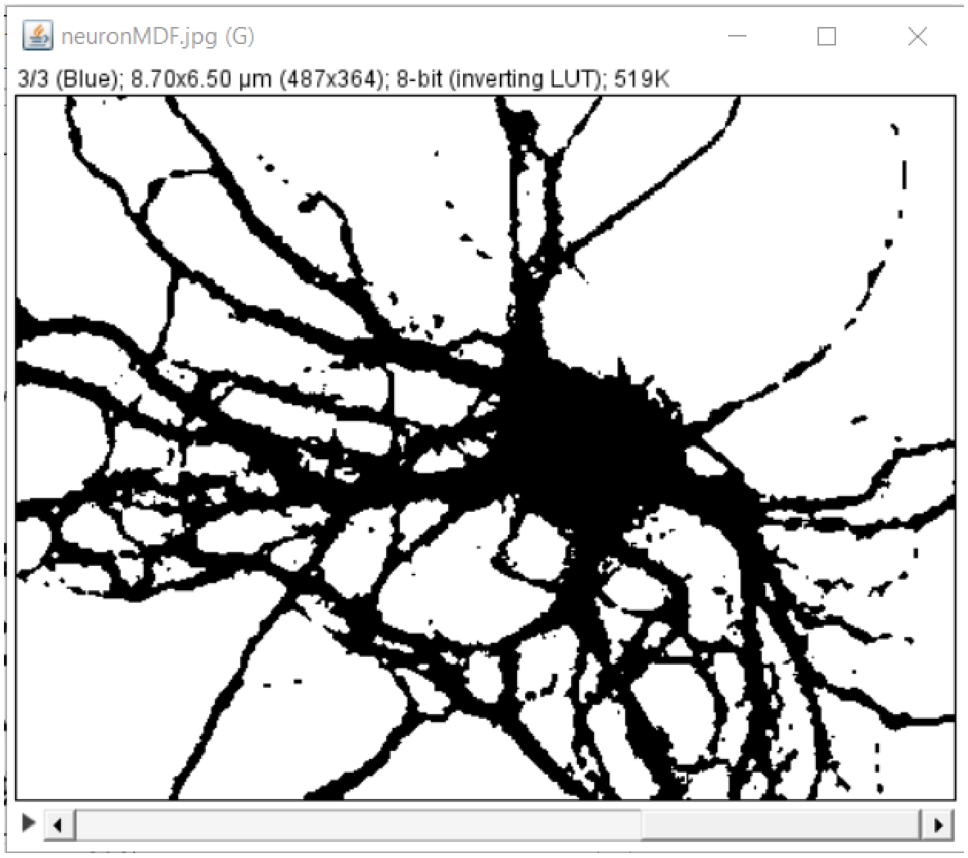
7. Choose the image where the dendrites are seen the best. In this case, it looks like the third image shows the dendrites the best.
8. Click Image/Adjust/Threshold. Move the cursors while maximizing the number of dendrites while minimizing the noise. Example below:



9. Click Set to confirm the threshold and then click Apply. The result is:

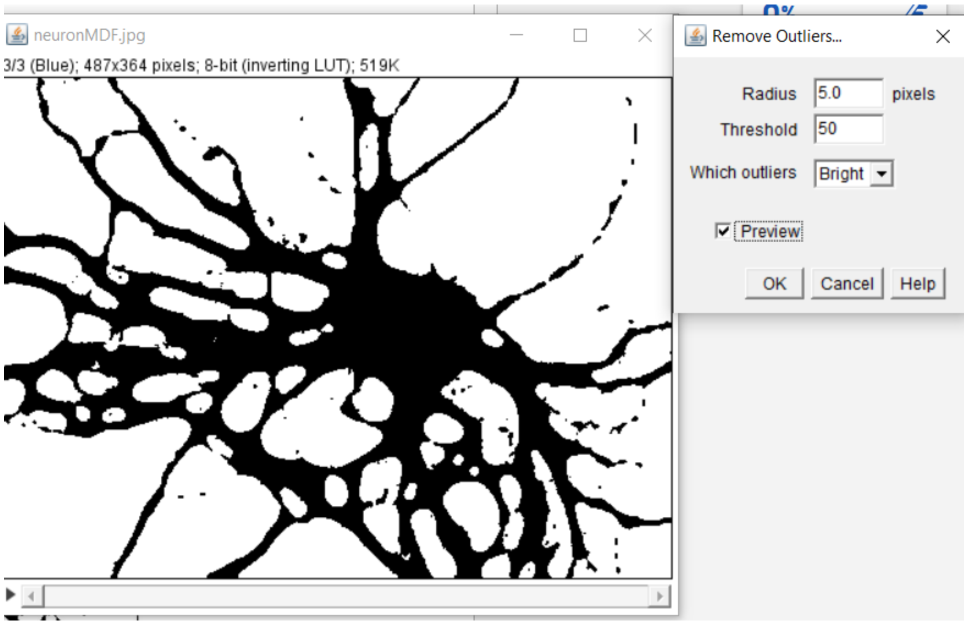

neuronMDF.jpg (G)
3/3 (Blue); 8.70x6.50 µm (487x364); 8-bit (inverting LUT); 519K

10. The picture is quite noisy. Go to Process/Noise/Despeckle. The result is:



3/3 (Blue); 8.70x6.50 µm (487x364); 8-bit (inverting LUT); 519K

11. The next step is to make the picture easier to analyze. Go to Process/Noise/Remove Outliers. Adjust the radius until it looks like most of the noisy white space is removed. The result is:


neuronMDF.jpg

3/3 (Blue); 487x364 pixels; 8-bit (inverting LUT); 519K

Remove Outliers...

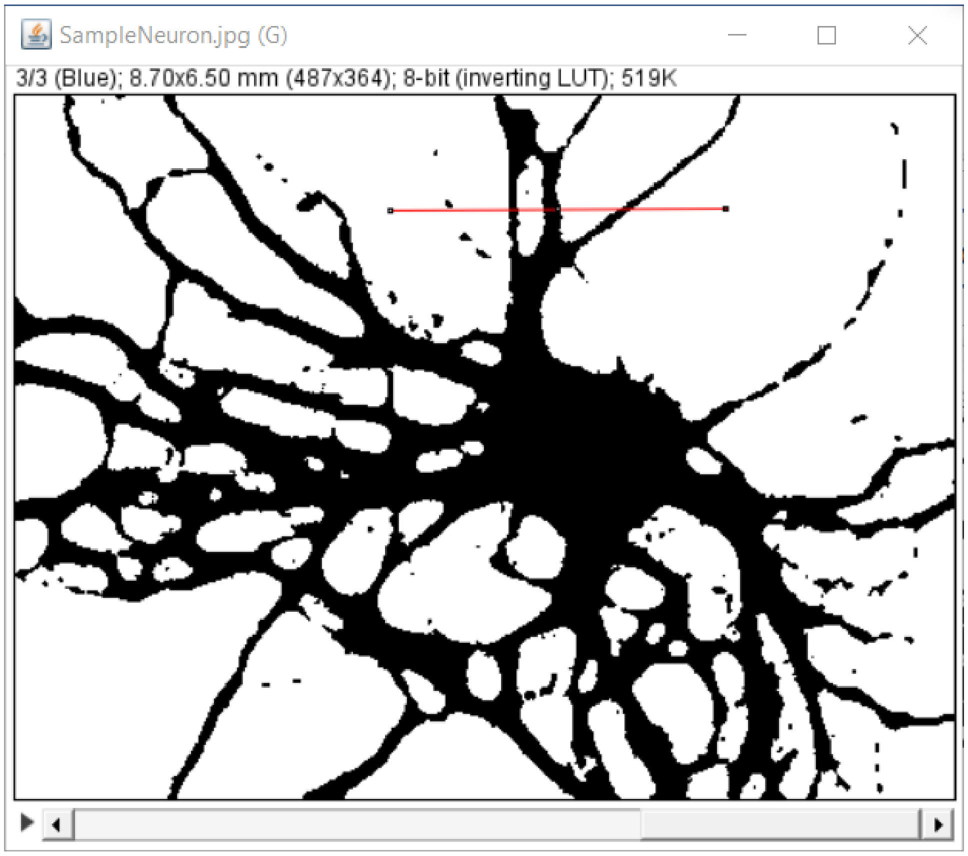
Radius 5.0 pixels

Threshold 50

Which outliers Bright

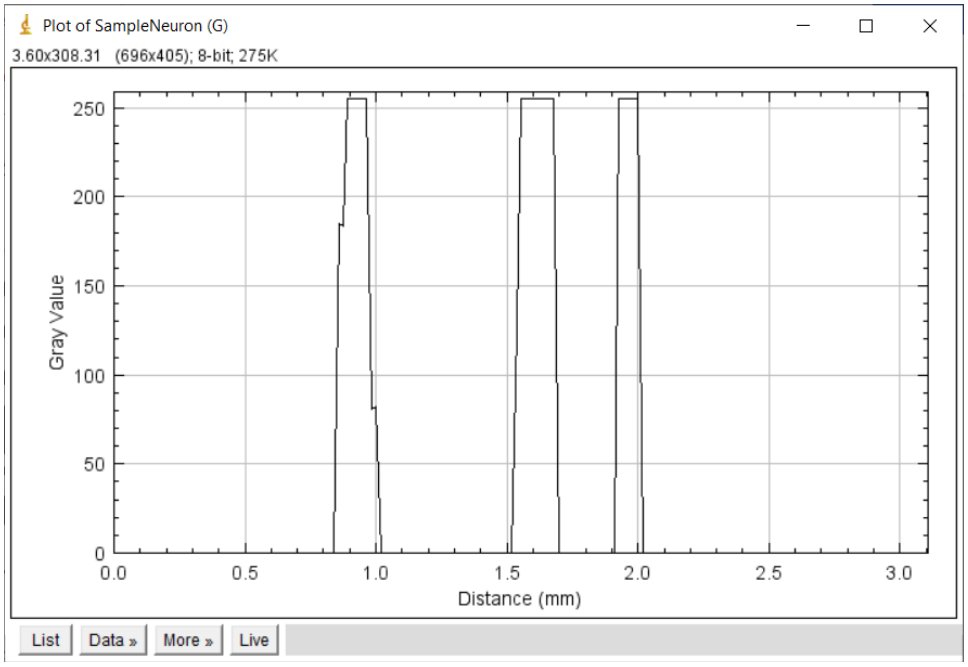
[✓] Preview

12. The next step is to start counting the dendrites. Select the “Straight” line tool. Draw a straight line across a section. Example below:

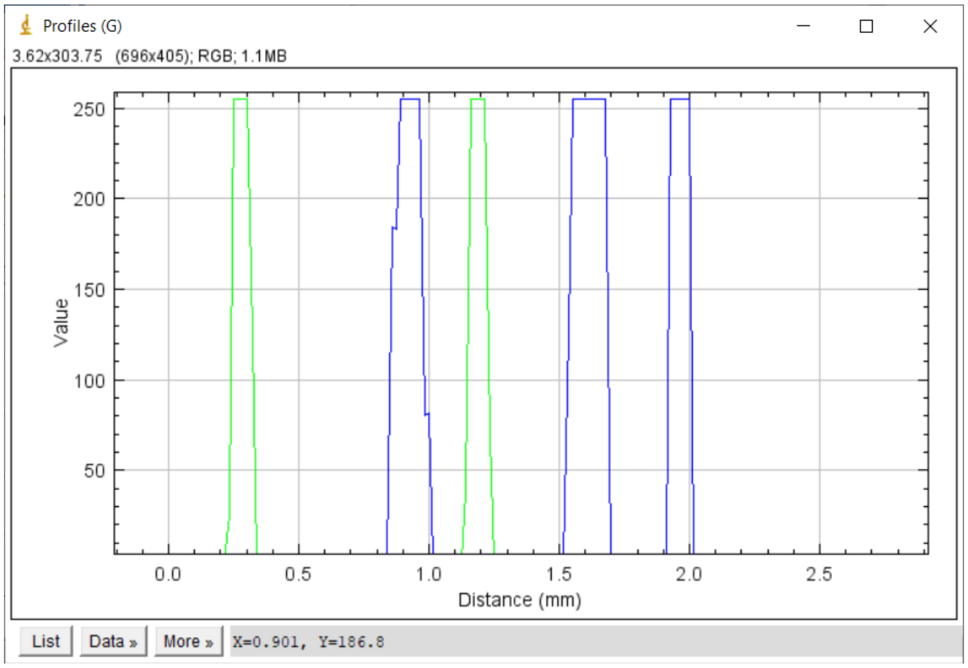

SampleNeuron.jpg (G)

3/3 (Blue); 8.70x6.50 mm (487x364); 8-bit (inverting LUT); 519K

13. The next step is to go to Analyze/Plot Profile. The result is:



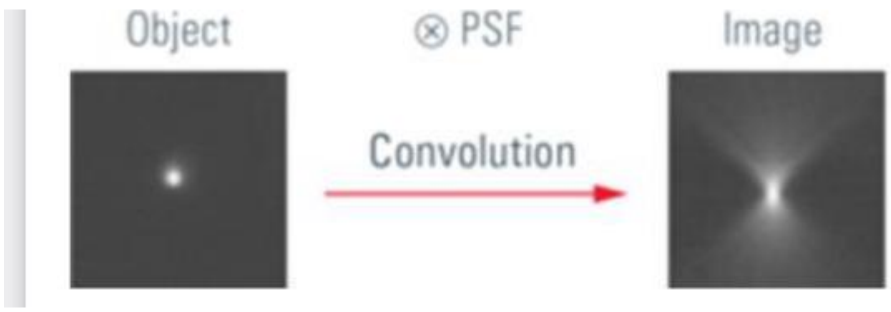
14. The peaks represent the dendrites. This method isn't perfect. For example, the section where the line was drawn appears to have 6 dendrites and not 3. One dendrite branches and gives rise to other dendrites.
15. If you make another line, the current line will disappear. To save it, push T. The ROI window will appear. Multiple lines can be added and saved here. To view all lines, check the "Show all" box. To view all plot profiles in one place, click on More/Multi Plot. The result is:



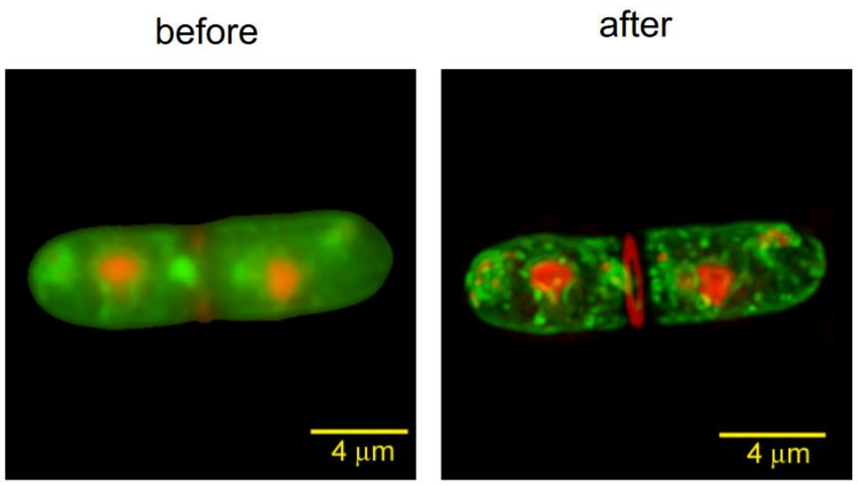
16. Continue this until all dendrites are counted. (Note: It may be easier to count manually and have lines keep track of progress. Using Multi Plot on many lines will be messy. Also might be easier to start at cell body and go outwards.)
17. The dendrites I counted in this sample picture was 53.
18. The last measurement is to measure the longest dendrite. The best way is to look for the longest dendrites and measure them using the "Straight" line tool. Like the same method above, push T to save the line. After creating all lines along the dendrite, click "Measure" on the ROI Manager window. Add up the lengths (which should be in um).
19. The longest dendrite I measured was 9.552 um.
20. To set a scale bar, go to Analyze/Tools/Scale bar.
21. Record the data on an excel spreadsheet where the data will be analyzed in Python.
22. The data will be plotted.
23. After that, the data will be analyzed in JMP
24. Open JMP
25. Go to File/New/Data Table
26. The column names will be Group, Treatment, and either Maximal Dendrite Length or Dendrite Number. The Groups are "Control, Mem, Lan, and Tra" and the Treatment will be "BFT or AFT" meaning before and after treatment. Input the data into the correct cell
27. After that, go to Analyze/Fit Model
28. The response variable will be either Maximal Dendrite Length or Dendrite Number and the categorical variables will be Group and Treatment.
29. Then go to Macros/Full Factorial and click Run.
30. The Effects Tests will show the two-way ANOVA table
31. If the p-value for Group and/or Treatment is below 0.05 and the interaction is above 0.05, then a Tukey test will be performed.
32. Go back to the data table

Deconvolution of Immunofluorescence Images

1. This section will give some background information regarding deconvolution and its role in immunofluorescence imaging.
2. First, starting with the definition of convolution. A simple definition of convolution is that convolution is the "blending" of two functions. The symbol is $f * g$.
3. In this experiment, there are three variables of interest: $y = x * h$ where x is the light entering the lens, y is the light exiting the lens, and h is the lens of the microscope.
4. The light entering the lens is a point of light. While the light travels through the lens of the microscope, the light will be diffracted resulting in an Airy disk. A visual is shown below:



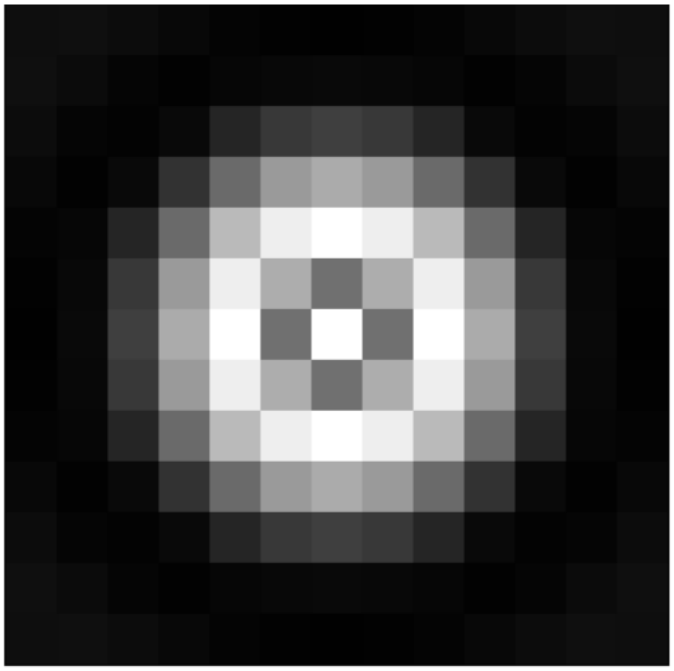
5. Going back to cells, the object is the original object and it is being convolved, or overlapping, with the Point Spread Function which results in a blurry image. What we want is deconvolution which is going from a blurry image to a clearer image:



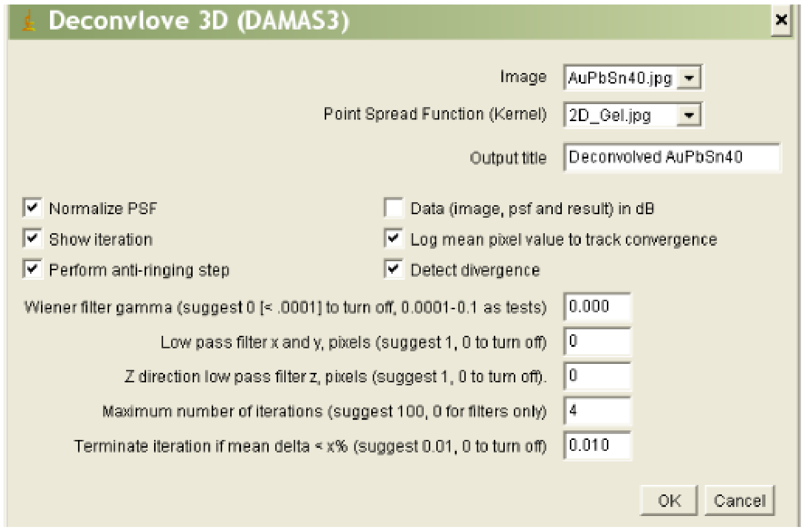
6. A Point Spread Function can be measured or computed theoretically. We will use a computed PSF.
7. To start, download the "Iterative Deconvolution 3D" from: https://www.optinav.info/Iterative-Deconvolve-3D.htm

For windows users, move Iterative_Deconvolve_3D.class and Iterative_Deconvolve_3D.java to plug-ins folder of the ImageJ application
○ For mac users, right-click the ImageJ icon in the applications folder and choose "show package contents." This will reveal the location of your plug-ins folder.

8. Open ImageJ and open a cell image. Crop the image (Image/Crop) so that the size of the picture is no bigger than 256x256. Otherwise, the plug-in will run out of memory.
9. Open the Point Spread Function that will be used. It would look something like:



10. Go to Plugins/Iterative Deconvolution 3D. The settings should look like except the maximum number of iterations should be 10. Also, the image should be in black and white. To do this, go to Image/Type/8 bit:



Deconvlove 3D (DAMAS3)

Image: AuPbSn40.jpg

Point Spread Function (Kernel): 2D_Gel.jpg

Output title: Deconvolved AuPbSn40

[✓] Normalize PSF

[✓] Show iteration

[✓] Perform anti-ringing step

[ ] Data (image, psf and result) in dB

[✓] Log mean pixel value to track convergence

[✓] Detect divergence

Wiener filter gamma (suggest 0 [< .0001] to turn off, 0.0001-0.1 as tests): 0.000

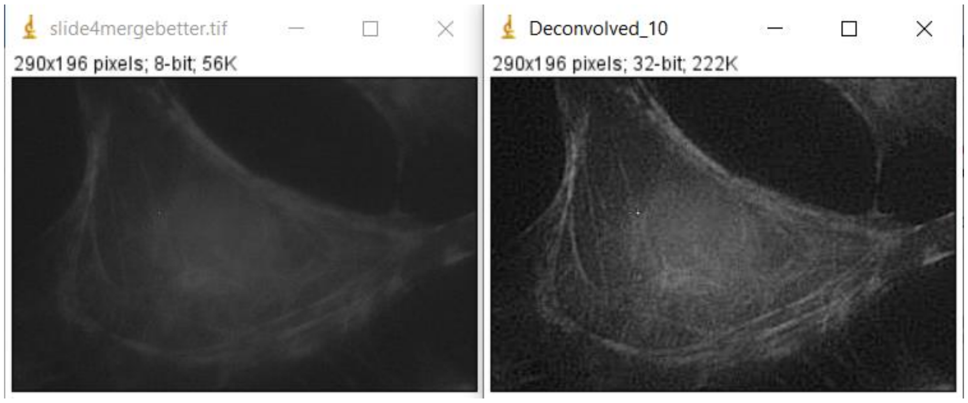
Low pass filter x and y, pixels (suggest 1, 0 to turn off): 0

Z direction low pass filter z, pixels (suggest 1, 0 to turn off): 0

Maximum number of iterations (suggest 100, 0 for filters only): 4

Terminate iteration if mean delta < x% (suggest 0.01, 0 to turn off): 0.010

11. A before and after convolution picture is shown below:



Experimental Model/Simulation

1. The model/simulation that will be developed will describe and predict how the concentration of cortisol and type of NMDA receptor affects the sustained increase in hippocampal neuron firing rate.
2. The model/simulation will be created in Python using Object Oriented Programming (OOP).

Space to plan for model

Make a grid (nxn) of neurons

Go through grid and determine whether or not it spiked

At the end of the grid, add up spikes for that time point.

Variables that determine whether or not neuron spikes: cortisol concentration, NMDA receptor antagonist, voltage.

Maybe two types of neurons: normal and depressed. Normal could have a resting membrane potential of – 70mv while a depressed neuron has a resting membrane potential of – 90mv. If the voltage reaches – 55mV, the neuron will spike.

A patient will have the grid before and after application of NMDA receptor antagonist.

After I finish, will have to use methods to determine how it holds up to data collected.

72 hours so 72 time points

After 1 time point, reset neuron grid.

Include diffusion equation solution and three diffusion constants: 6, 12, and 24 um^2/s

Include toxicity and efficacy components of drug. ED50: 0.57 mg/kg. TD50: 2.56 mg/kg

70 kg patient: ED50: 39.9 mg. TD50: 179.2 mg.

Smaller diffusion coefficient leads to more toxicity. (1 and 6) e,t

Use values drawn from a log-normal distribution to add randomness to neurons spiking

Will be in the form of a sigmoidal function.

$$\frac{[A]}{ED50 + [A]}$$

Efficacy should include depressed neurons

Toxicity should include all neurons

Net Benefit = Efficacy * %depressed – Toxicity * 1

Might need to modify sigmoidal function such as (Generalized logistic function):

$$\frac{1 - Ae^{-\alpha t}}{1 + Ae^{-\alpha t}} + B$$

$$A + \frac{K - A}{(C + Qe^{-Bt})^{\frac{1}{\mu}}}$$

Diffusivity values from paper:

Diffusion Coefficient um^2/s	Time to reach 50% of brain tissue	Time to reach 100 % of brain tissue
6	15 min	90 min

12	10 min	70 min
24	5 min	40 min
48	1 min	15 min
96	0.1 min	1 min

Determine efficacy and toxicity for diffusion coefficient.

Parameters for generalized logistic function: diffusion coefficient, dose, cortisol concentration, fraction with how many depressed neurons on the grid, 1 for all neurons.

Multivariate graphing showing diffusion coefficient on x-axis, dose on y-axis, and net benefit on z-axis. Sigmoidal curve

Probability of neuron spiking:

$$V = V_m e^{-\left(cortcon + NMDAconcentration + \log normal(0,25)\right)}$$

$$V = V_m e^{-\left(\frac{cortcon \cdot NMDAconcentration}{\log normal 0,0.25}\right)}$$

Want exponential to be bigger

np.random.seed(1)

popt, pcov = curve_fit(exp_func,np.arange(0,3,0.01),y,p0=[54,300,-60,0.20],bounds=([0,0,-np.inf,0],(1000,1000,0,100)))

Spikes as a function of time

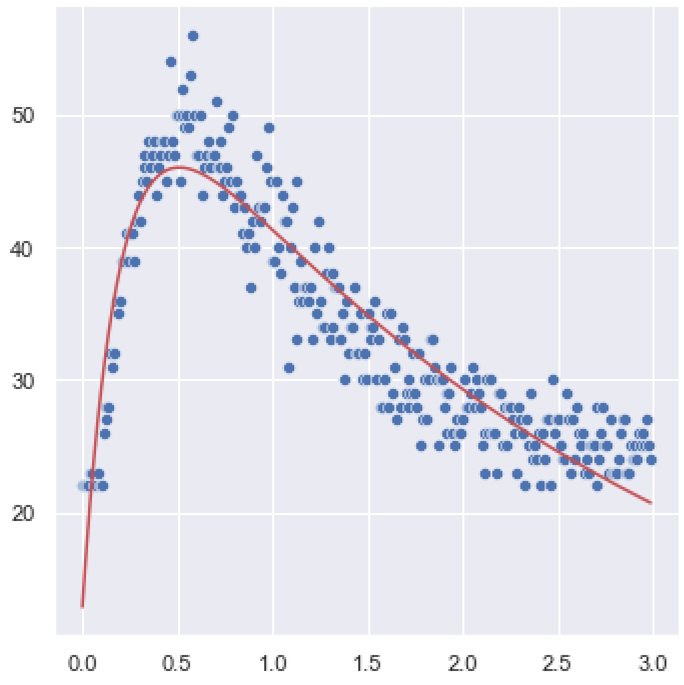
$$S(t) = A_1 e^{-\frac{t}{\tau_1}} + A_2 e^{-\frac{t}{\tau_2}}$$

A_1 = 59.0153

τ_1 = 2.8549

A_2 = -46.1736

τ_2 = 0.194489



11x11 grid, prob_normal = 0.2, probDepressed = 0.8, stepsize = 0.01, mean of 15 simulations

Dose	Diffusivity	Cort_con	A1	theta1	A2	theta2
75	6	1.2	64	3.15	-48.1	0.205
75	6	1.4	72	2.97	-58.2	0.242
75	6	1.6	80.3	2.96	-66.4	0.275
75	6	1.8	86.6	3.00	-72.9	0.300
75	6	2.0	92.1	3.11	-77.6	0.318

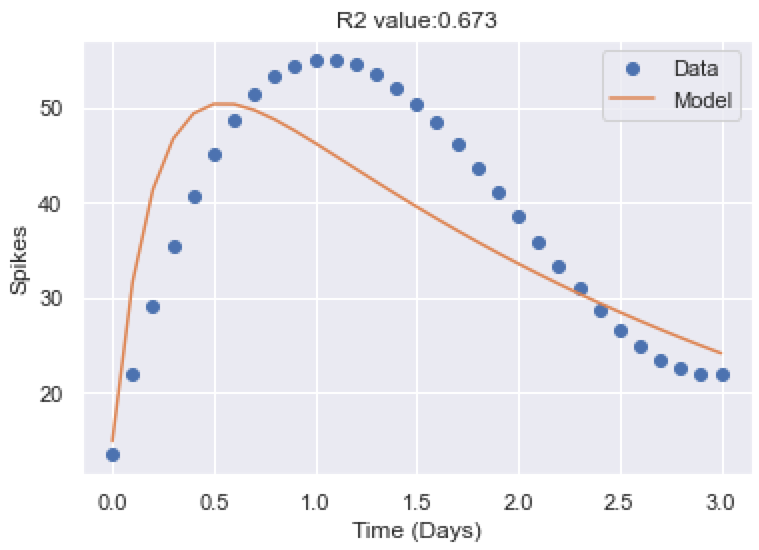
A1 = (35.4*Cort_con+22.36)

Theta1 = 3.04

A2 = -36.85*Cort_con – 5.68

Theta2 = 0.142*Cort_con + 0.0408

Example graph using above parameters on sample data:



$$S = \sqrt{\frac{SSE}{N - P}}$$

Simulation

Patient Time points

Neuron Grid On medication Cortisol levels

Neuron Depressed Neuron

Spike Probability of spiking Voltage

```
# Neuron spiking model/simulation
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

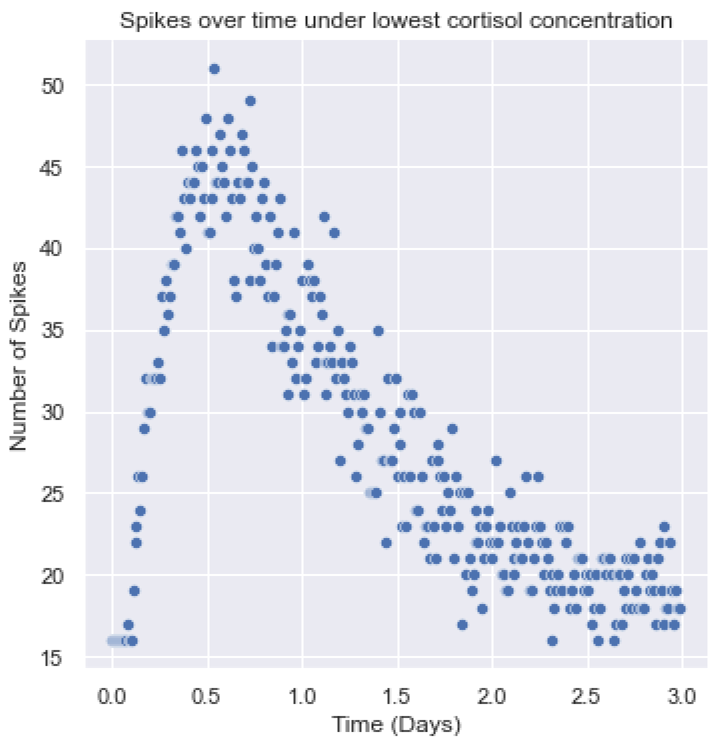
```
from scipy.optimize import curve_fit
from tkinter import *
from sklearn.metrics import r2_score
from scipy import interpolate


class Neuron(object):
    """A neuron that will fire at a "normal" rate"""

    def __init__(self):
        """ Resting Membrane Potential (in mV)"""

        self.voltage = -70

    def has_spiked(self,cort_con):
        """
        Determines whether or not this neuron has spiked

        Arg: cortical concentration (float) 1.2, 1.4, 1.6, 1.8, 2.0 representing lowest to
highest cortisol concentration

        Returns: Boolean (True or False) whether or not neuron spiked depending on
voltage
        If the voltage is greater than the threshold of -55 mV, the neuron will fire.
        Otherwise, it will not
        """

        new_volt = self.voltage*np.exp(-
(((1+np.random.lognormal(0,0.2))**cort_con)/((3+cort_con))))

        if new_volt > -55:
            return True
        else:
            return False


    def __repr__(self):
        return 'Neuron'



class DepressedNeuron(object):
    """ A "depressed" neuron that will spike less by having a lower resting membrane
potential"""

    def __init__(self):
        """Resting Membrane Potential (in mV)"""

        self.voltage = -90

    def has_spiked(self,diff_con,cort_con):
        """
        Determines whether or not this neuron has spiked

        Args: diffusion concentration (float): Diffusion concentration at a specific position
and time
              cort_con (float): 1.2, 1.4, 1.6, 1.8, 2.0 representing lowest to highest cortisol
concentration

        Returns: Boolean (True or False) whether or not neuron spiked depending on
voltage
        If the voltage is greater than the threshold of -55 mV, the neuron will fire.
        Otherwise, it will not
        """
        new_volt = self.voltage*np.exp(-
(((1+diff_con)**cort_con)*((np.random.lognormal(0,0.2))**cort_con)/((3+cort_con))))

        if new_volt > -55:
            return True
        else:
            return False


    def __repr__(self):
        return 'DepNeuron'

class Patient(object):
    """A patient with Major Depressive Disorder"""

    def __init__(self,neurons,cort_con):
        """
        Args: neurons (list of Neuron and DepressedNeuron patient has)
              cort_con (float representing cortisol concentration)
        """
        self.neurons = neurons
        self.cort_con = cort_con
        self.on_medication = False

    def set_on_medication(self):
        """Put patient on antidepressant"""
        self.on_medication = True

    def check_if_on_medication(self):
        """
        Checks if patient is on antidepressant

        Returns: Boolean (True or False)
        """
        return self.on_medication

    def get_neurons(self):
        """ Returns neuron grid in patient"""

        return self.neurons


def createNeuronGrid(gridsize,prob_Normal,prob_Depressed):
    """Creates an nxn grid full of neurons

      Args: gridsize (tuple): gives size of nxn grid
            prob_normal (float): gives probability of placing a normal neuron on grid
            prob_Depressed (float): gives probability of placing a depressed neuron on grid

      Returns: tuple, grid of neurons and depressed neurons
    """

    return
np.random.choice([Neuron(),DepressedNeuron()],size=gridsize,p=[prob_Normal,prob_D
epressed])

def visualizeGrid(grid):
    """Visualize the nxn grid of neurons

      Arg: grid (tuple): grid of Neurons and DepNeurons
    """
    root = Tk()
    for r in range(len(grid)):
        for c in range(len(grid)):
            if isinstance(grid[r][c],DepressedNeuron):
                btn_ipady = Button(root, text="DepNeuron"+str(r)+str(c),
bg="blue",fg="white")
                btn_ipady.grid(column=c,row=r,padx=10,pady=10)
            else:
                btn_rowspan = Button(root, text="Neuron"+str(r)+str(c),bg="red",fg="white")
                btn_rowspan.grid(column=c,row=r,padx=10,pady=10)
    root.mainloop()

def visualizeDrugDiffusion(grid,diffrate,dose):
    """
       Visualize the diffusion of drug across grid over time with a heatmap
       The point (0,0) is located at the center of the grid where the drug is
       administered

       Args: grid (tuple): grid of Neurons and DepNeurons
             diffrate (int): diffusion coefficient of drug in um^2/s
             dose (int): dose of drug in mg

    """
    for t in np.linspace(0.1,3,30):
        total_dose = 0
        dif_grid = np.empty([len(grid),len(grid)])
        midpoint = int(np.floor(len(grid)/2))
        Diff_x,Diff_y = np.meshgrid([np.arange(-midpoint,midpoint+1)],-np.arange(-
midpoint,midpoint+1))
        for row in range(len(grid)):
            for col in range(len(grid)):
                dif_grid[row][col] =
calcDiffusion(Diff_x[row][col],Diff_y[row][col],diffrate,t,dose)
                total_dose += calcDiffusion(Diff_x[row][col],Diff_y[row][col],diffrate,t,dose)
        sns.heatmap(dif_grid,annot=True,xticklabels=np.arange(-
midpoint,midpoint+1),yticklabels=-np.arange(-midpoint,midpoint+1))
        plt.title('Heatmap when t = '+str(np.round(t,2))+' days, '+str(round(total_dose,2))+'
mg')
        plt.pause(0.5)


def calcDiffusion(x,y,D,t,dose):
    """Gives the value from the diffusion equation analytical solution

       Args: x (int): reprents x position on neuron grid
             y (int): represent y position on neuron grid
            D (int): represents diffusion coefficient of drug in um^2/s
            t (float): represents time passed since drug delivery
            dose (int): represents dose of drug in mg

        Returns: diff_val (float): value of concentration of drug at a specific
        time and position

    """
    diff_val = dose*np.exp(-(x**2+y**2)/(4*D*t))*1/(4*np.pi*D*t)
    return diff_val

def exp_func(t,A1,tau1,A2,tau2):
    """ Bi-exponential decay function used for scipy.optimize.curve_fit. Other
        args besides t are calculated by the curve_fit function

        Arg: only t (float): time

        Returns: bi-exponential decay function
    """
    return A1*np.exp(-(t/tau1)) + A2*np.exp(-(t/tau2))

def model_func_values(t,cort_con):
    """ Function that gives values from model developed from simulation

        Args: t (float): time
        cort_con (float) 1.2, 1.4, 1.6, 1.8, 2.0 representing lowest to highest cortisol
concentration

        Returns: value of model at a given time and cortisol concentration
    """
    return (35.4*cort_con+22.36)*np.exp(-(t/3.04)) - (36.85*cort_con+5.68)*np.exp(-
(t/(0.142*cort_con+0.0408)))




def spiking_simulation(gridsize,prob_Normal,prob_Depressed,diffrate,dose,cort_con):
    """
    Runs the simulation of neurons spiking. Calculates drug concentration at a
    specific position and time to help a depressed neuron spike. Also plots
    time and number of spikes

    Args: gridsize (tuple): gives size of nxn grid
```

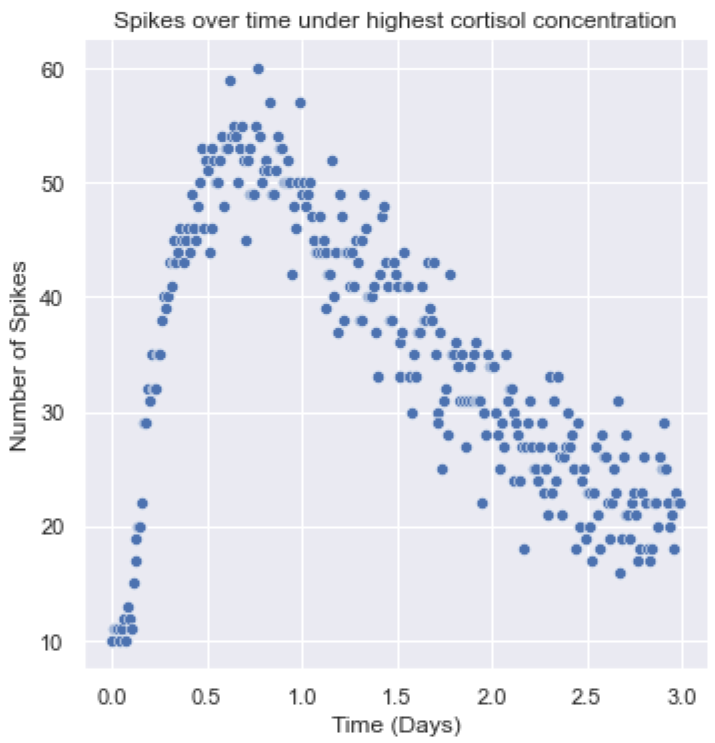
```
         prob_normal (float): gives probability of placing a normal neuron on grid
         prob_Depressed (float): gives probability of placing a depressed neuron on grid
         diffrate (int): diffusion coefficient of drug in um^2/s
         dose (int): dose of drug in mg
         cort_con (float) 1.2, 1.4, 1.6, 1.8, 2.0 representing lowest to highest cortisol
concentration

    Returns: total_spikes (list): a list of total spikes over time
    """
    total_spikes = []
    patient = Patient(createNeuronGrid(gridsize, prob_Normal, prob_Depressed),cort_con)
    for t in np.linspace(0.1,3,100):
        spikes = []
        grid = patient.get_neurons()
        midpoint = int(np.floor(len(grid)/2))
        Diff_x,Diff_y = np.meshgrid([np.arange(-midpoint,midpoint+1)],-np.arange(-
midpoint,midpoint+1))
        for row in range(len(grid)):
            for col in range(len(grid)):
                if isinstance(grid[row][col],Neuron):
                    spikes.append(grid[row][col].has_spiked(cort_con))
                else:
                    diff_val = calcDiffusion(Diff_x[row][col],Diff_y[row][col],diffrate,t,dose)
                    spikes.append(grid[row][col].has_spiked(diff_val,cort_con))
        total_spikes.append(sum(spikes))
    sns.set_theme()
    sns.relplot(x=np.linspace(0.1,3,100),y=total_spikes)
    return total_spikes

def
spiking_simulation_with_delay(gridsize,prob_Normal,prob_Depressed,diffrate,dose,cort
_con,stepsize):
    """
    Runs the simulation of neurons spiking with a time delay before medication
    Calculates drug concentration at a specific position and time to help a
    depressed neuron spike. Also plots time and number of spikes

    Args: gridsize (tuple): gives size of nxn grid
          prob_normal (float): gives probability of placing a normal neuron on grid
          prob_Depressed (float): gives probability of placing a depressed neuron on grid
          diffrate (int): diffusion coefficient of drug in um^2/s
          dose (int): dose of drug in mg
        cort_con (float) 1.2, 1.4, 1.6, 1.8, 2.0 representing lowest to highest cortisol
concentration
        step_size (float): represents step size of time data points

    Returns: total_spikes (list): a list of total spikes over time
    """

    total_spikes = []
    patient = Patient(createNeuronGrid(gridsize, prob_Normal, prob_Depressed),cort_con)
    time = np.arange(0,3,stepsize)
    time_counter = 1
    for t in np.arange(0,3,stepsize):
        if t > 0.1:
            patient.set_on_medication()
        spikes = []
        grid = patient.get_neurons()
        midpoint = int(np.floor(len(grid)/2))
        Diff_x,Diff_y = np.meshgrid([np.arange(-midpoint,midpoint+1)],-np.arange(-
midpoint,midpoint+1))
        for row in range(len(grid)):
            for col in range(len(grid)):
                if isinstance(grid[row][col],Neuron):
                    spikes.append(grid[row][col].has_spiked(cort_con))
                else:
                    if patient.check_if_on_medication():
                        diff_val =
calcDiffusion(Diff_x[row][col],Diff_y[row][col],diffrate,time[time_counter],dose)
                        spikes.append(grid[row][col].has_spiked(diff_val,cort_con))

                    else:
                        spikes.append(grid[row][col].has_spiked(0,cort_con))
        if patient.check_if_on_medication():
            time_counter += 1
        total_spikes.append(sum(spikes))
    sns.set_theme()
    sns.relplot(x=time,y=total_spikes)
    return total_spikes

def
spiking_simulation_with_multiple_doses(gridsize,prob_Normal,prob_Depressed,diffrate,
dose,cort_con,stepsize,num_days,next_dose):
    """
    Runs the simulation of neurons spiking with additional doses taken after the first one
    Calculates drug concentration at a specific position and time to help a
    depressed neuron spike. Also plots time and number of spikes

    Args: gridsize (tuple): gives size of nxn grid
          prob_normal (float): gives probability of placing a normal neuron on grid
          prob_Depressed (float): gives probability of placing a depressed neuron on grid
          diffrate (int): diffusion coefficient of drug in um^2/s
          dose (int): dose of drug in mg
          cort_con (float) 1.2, 1.4, 1.6, 1.8, 2.0 representing lowest to highest cortisol
concentration
          step_size (float): represents step size of time data points
          num_days (int): represents how many days are shown on graph
          next_dose (int): represents how many days before another dose

    Returns: total_spikes (list): a list of total spikes over time

    """

    total_spikes = []
    patient = Patient(createNeuronGrid(gridsize, prob_Normal, prob_Depressed),cort_con)
    time = np.arange(0,num_days,stepsize)
    dose_time_array = np.zeros(int(num_days/next_dose+1))
    dose_counter = 0
    for t in np.arange(0,num_days,stepsize):
        if t > 0.1:
            patient.set_on_medication()
        if t % next_dose < 1e-6:
            dose_time_array[dose_counter] = 1
            dose_counter += 1
        spikes = []
        grid = patient.get_neurons()
        midpoint = int(np.floor(len(grid)/2))
        Diff_x,Diff_y = np.meshgrid([np.arange(-midpoint,midpoint+1)],-np.arange(-
midpoint,midpoint+1))
        for row in range(len(grid)):
            for col in range(len(grid)):
                if isinstance(grid[row][col],Neuron):
                    spikes.append(grid[row][col].has_spiked(cort_con))
                else:
                    if patient.check_if_on_medication():
                        diff_val = []
                        for num in range(np.count_nonzero(dose_time_array)):
                            time_counter = int(dose_time_array[num])
```

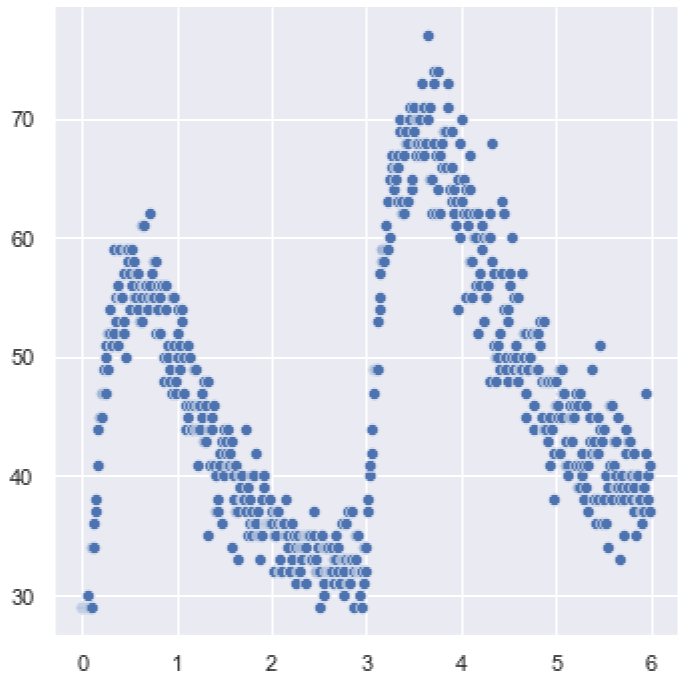
```
diff_val.append(calcDiffusion(Diff_x[row][col],Diff_y[row][col],diffrate,time[time_coun
ter],dose))
                        spikes.append(grid[row][col].has_spiked(np.sum(diff_val),cort_con))
                    else:
                        spikes.append(grid[row][col].has_spiked(0,cort_con))
        if patient.check_if_on_medication():
            for num_counter in range(np.count_nonzero(dose_time_array)):
                dose_time_array[num_counter] += 1
        total_spikes.append(sum(spikes))
    sns.set_theme()
    sns.relplot(x=time,y=total_spikes)
    return total_spikes


#x = spiking_simulation((9,9), 0.2, 0.8, 6, 75, 1.2)


# y = spiking_simulation_with_delay((11,11), 0.2, 0.8, 6, 75, 1.2, 0.01)
# popt, pcov = curve_fit(exp_func,np.arange(0,3,0.01),y,p0=[54,30,-
60,0.5],bounds=([0,0,-100,0],(500,1000,0,100)))
# plt.plot(np.arange(0,3,0.01),exp_func(np.arange(0,3,0.01),*popt),'r-')
# plt.xlabel('Time (Days)')
# plt.ylabel('Spikes')
# plt.legend(['Model','Data'])
# plt.title('R2 value: ' + str(r2_score(y,exp_func(np.arange(0,3,0.01),*popt))))

z = spiking_simulation_with_multiple_doses((11,11), 0.2, 0.8, 6, 75, 1.2, 0.01,6,3)


# %% See how well model fits data
combined_dict =
{'Control_LL_Day1_BFT':[12,15],'Control_LL_Day1_AFT':[50,60],'Control_LL_Day2_
AFT':[35,42],'Control_LL_Day3_AFT':[20,24]}
cort_con_num = 1.0
x = [0,1,2,3]
for cort_con in ['LL_', 'L_', 'H_', 'HH_']:
    if cort_con == 'H_':
        cort_con_num += 0.4
    else:
        cort_con_num += 0.2
    for group in ['Control_', 'Mem_', 'Lac_', 'Tra_']:
        temp_spikes = []
```

```
          for day in ['Day1_BFT', 'Day1_AFT', 'Day2_AFT', 'Day3_AFT']:
                    temp_spikes.append(np.mean(combined_dict[group+cort_con+day]))
          interp_func = interpolate.interp1d(x, temp_spikes,kind='cubic')
          xnew = np.arange(0,3.1,0.1)
          ynew = interp_func(xnew)
          plt.plot(xnew,ynew,'o',xnew,model_func_values(xnew,cort_con_num),'-')
          plt.xlabel('Time (Days)')
          plt.ylabel('Spikes')
          plt.legend(['Data','Model'])
          plt.title('Cort_con: ' + cort_con[0:-1] + '| Group:' + group[0:-1] + ' | R2 value:' +
str(round(r2_score(ynew,model_func_values(xnew, cort_con_num)),3)))
```

Sample Graphs:


Spikes over time under lowest cortisol concentration
Number of Spikes
Time (Days)
15
20
25
30
35
40
45
50
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Spikes over time under highest cortisol concentration
Number of Spikes
Time (Days)
10
20
30
40
50
60
0.0
0.5
1.0
1.5
2.0
2.5
3.0



Space to plan for presentation

- At beginning, say name and start to read formal definition of depression on notecard. Slow down and rip up card and say depression is bad.
- Incorporate depressive thoughts into presentation as a thought voice. Will need to edit it good
- Make graphs more interactive than just flat by describing it over time

- Another thing, get right. Like know anything (garbage, if it was good it would have had the other parts)

- Take a “field trip” to the lab. Show microelectrode array and microscope reader. Might also show immunofluorescence image of neurons. Talk about other two parts of project. (Might put extra here)

- Facts to include: ketamine is fast (works as fast as in a few hours). Use graph to illustrate this. Effect doesn’t last long. Patients with depression, the hippocampus is less active. Can be represented by neurons spiking (or turning on). This project set out to investigate the relationship between the antidepressant effect of ketamine and cortisol concentration (the chemical responsible for stress). Trying to extend antidepressant effect

- Show neuron grid with Neuron (normal neuron) and depressed Neurons (which fire less)
- Show heat map of diffusion for drug. A higher concentration will lead to a depressed neuron firing more
- Simulation produces random values. A model equation represented by this red line will be compared with the experimental data
- Today not only can the dose of the drug can be changed, how fast it diffuses can also be changed. The last part of the simulation is that given a certain cortisol concentration and level of depression, can calculate the optimal dose and diffusion coefficient to give the greatest net benefit. Drugs have efficacy and toxicity which both dose and diffusivity affect. Show demonstration

- At end, address thoughts and give rise to why this is important to me
- Didn't get to do other parts yet, but will do them. Still learned a lot
- Show name card. Represents past, present, future. Reminds me what I'm working towards.

(Only way to extend is through multiple doses which is not ideal given the harsh side effects)

- (Start showing myself at Red room)
- Hi, I'm Andrew and my project is called prolonging the rapid antidepressant effect of ketamine
- So lets jump right into it
- Depression is a psychological disorder characterized by symptoms such as difficulty concentrating and low self-esteem
- Although many medications for depression exist, many of them are either ineffective or have a delayed effect
- Ketamine, a drug, is commonly used as an anesthetic, but it can also be used as an antidepressant.

- What makes it better than other medications for depression is that it is more effective and works in a matter of hours instead of weeks or months
- The downsides are that the antidepressant effect typically lasts only a few days, it has harsh side effects, and it has the potential to be abused which makes it inaccessible to many patients.
- Much research has been done on understanding how ketamine produces its rapid antidepressant effect, but not much has been done on extending it
- The purpose of this project is to determine the relationship between the antidepressant effect of ketamine and cortisol concentration, the chemical responsible for stress.
- The hypothesis is that a higher stress level could lead to a longer antidepressant effect since people can gain resilience after overcoming difficult experiences
- In this project, depression will be represented by a decrease in the number of hippocampus neurons spiking, or turning on, since patients with depression typically have less neuronal activity in the hippocampus, the part of the brain responsible for learning and memory.
- After applying ketamine, we would expect to see an increase in the number of neurons spiking since antidepressants can increase neuronal activity in the hippocampus for a period of time until the firing rate returns to baseline
- This project consists of three parts, but because of many problems some parts were not able to be completed in time
- (thinking) Just another thing I can’t do right
- But I will talk about the model and simulation I developed to predict how different cortisol levels affect the duration of the increased neural spiking caused by ketamine

- (Go to powerpoint slides)
- So for this project there is a grid with two types of neurons: a normal neuron that will spike at a normal rate with a resting potential of - 70mV and a depressed neuron which will spike less often with a resting potential of – 90 mV. If the resting membrane potential reaches –55 mV, the neuron will spike. Otherwise, it will not.
- Alongside this, ketamine will also be on a grid where it is administered in the center
- As time passes, it will diffuse outward from the center until it's completely gone
- Ketamine will help the depressed neurons fire more while the normal neurons will be unaffected

- (Go to lab)
- Hi again. I thought we'd take a field trip to show how we can measure neuron firing rates in the lab
- We can use one of these micro electrode arrays
- We can plate hippocampus neurons on these, stimulate them with this machine and the micro electrodes will be able to detect neurons spiking
- Now, in order to simulate depression, a chemical called carbenoxolone will be applied to the experimental groups since it is known to cause neurons to fire less like that in depression
- After that, ketamine, alongside one of five different concentrations of cortisol, will be applied to each dish in the experimental group
- We'll then record the neuronal activity for an additional two days
- That's the first part of this project
- The second part has to do with this.
- (Show immunofluorescence image)
- This colorful image was produced using immunofluorescence, a lab technique used to image neurons

- It will be performed on a group of neurons to see how ketamine affects dendrite length and number which are these branches
- These are the two parts of the project that were not able to be completed in time due to complications, but they will be completed soon.
- (thinking) This project was a failure and this presentation isn't going well either
- Now that we have covered all of that, let's get back to the simulation

- (Back to powerpoint slides)
- In the actual experiment, the only variable that will change is cortisol concentration. In the simulation, alongside cortisol concentration, grid size the drug dose and diffusion rate, and the level of depression, represented by what percentage of neurons on the neuron grid are normal or depressed can be changed
- Also, since neurons spiking is somewhat random, there is an element of randomness in this simulation
- Let's do the simulation with a 15x15 grid where 80% of neurons are depressed and 20% are normal with a 75 mg dose diffusing at a rate of 6 um^2/ second starting with the lowest cortisol concentration.
- We'll see how the firing rate looks like before the drug is applied
- It looks likes there are about 20 spikes at each time point
- Now, let's apply the drug
- We see a sharp increase in the neuron firing rate which is expected since ketamine works fast with a max of around 60 spikes.
- After that, we see a general decline back to baseline
- One of the advantages of this simulation is that we can change the parameters to see how the firing rate changes
- If we increase the dose to 150 mg, the effect lasts longer since there's more ketamine available
- If we increase the diffusion rate to 12 um^2/s, the effect is shorter since ketamine is removed faster
- If we increase cortisol concentration to the highest level, the effect is also longer which is the hypothesis for this project
- We can also change the number of doses
- If we give a patient a 75 mg dose every 3 days, the simulation graph looks like this

- It looks like the effect lasts longer with each additional dose.
- This result agrees with the literature that with each additional dose of ketamine, the antidepressant effect is usually more effective
- Now going back to one dose, we can use the simulation to develop a model equation depending on the cortisol concentration.
- After collecting the experimental data, we want to compare how well the model fits it. We only get four data points from each microelectrode array and we need more to compare them so we can use interpolation to generate more data points like this
- Now we can overlay the model and data graphs for that specific cortisol concentration and calculate S the standard error of the regression value which will tell us how well the model fits the data
- For this sample data and model, we get an S value of 8 which means that on average the model and data differ by about 8 spikes
- And now, the last part of this simulation has to do with efficacy and toxicity of ketamine and we'll be looking at two variables: dose and diffusion rate
- Starting with dose, too small of a dose will have little effect and a large dose will be more effective.
- However, like any drug, there is also toxicity we must consider which follows the same general pattern
- Similar curves can be produced for the diffusion rate. A lower diffusion rate will be more effective and toxic, and a high diffusion rate will have little effect and toxicity
- We want to find the dose and diffusion rate that maximizes efficacy and minimizes toxicity
- Thus, we want to find the maximum of this net benefit equation shown on this graph. However, since many optimization algorithms are designed to find a minimum, we'll have to flip this graph and find the minimum
- From the graph, we can see the minimum occurs at a dose of 89.4 mg and a diffusion rate of 7.7 um^2/s and if the graph is flipped back, we would see that this dose and diffusion rate maximizes net benefit

- (Go outside)
- And that's all for my project
- But before I go, I want to address one final thing
- Throughout this presentation, you have heard some of my depressive thoughts that I've had in the past
- I'd be lying if I said that I still didn't have them especially during this project
- Is that a bad thing?
- I don't think so.

- It just reminds me of what I’m working towards and just in case I forget, I always carry this id badge with me.
- It represents three things: the past filled with many enduring hardships, the present filled with its ups and downs, and the future filled with unlimited opportunities.
- So, what’s next? The same thing that’s always next. I'm going to keep learning more and working hard so that I can work towards developing better treatments for depression
- (thinking) Yeah, this project was a sucess and the presentation went well. I’m proud of it

Using Resolve 18

- First shot: Fade in, patch up two things, brighten up picture, fix sound. Add effect downsides (Usually 3 days, like hallucinations and confusion, Can be abused) Put images of hippocampus neurons spiking and hippocampus. Show drop of ketamine, increase neuron spiking
- Second shot: Brighten up picture, fix sound
- Third shot: Brighten up picture, fix sound, birds chirping maybe, fix shaking