Process book

Date of entry: 12.05.
What I've worked on:
Today, I loaded the data into my project notebook and also looked at
different libraries for creating plots. To try something new, I think I will do plots using Vega-Altair plots instead of seaborn. The plots on their website seem very nice.

What problems I encountered:
The challenge was to code the data loading in such a way that I can use the
same code for different projects/data as well. Avoiding hard-coding as much
as possible.

What I learned:
Useful for file loading is the glob library in python, since you can use the * unix wild card to generalize code: eg using '*.csv' to get all csv files.

Which resources did I use:
- Vega-Altair website:
https://altair-viz.github.io/
- glob documentation:
https://docs.python.org/3/library/glob.html

Date of entry: 15.05
What I've worked on:
I want to first visualize the data to see if there are any striking irregularities. I created some bar charts in altair to visualize how many data-points (neurons, or rings around neurons) I have per mouse and per image. I deleted the data of one image that seemed to be an outlier.

What problems I encountered:
I was fighting a bit with altair syntax. It seems to be similar to JSONs somehow and a bit different to seaborn.

What I learned:
Altair seems to have a lot of interactive plot options.

Which resources did I use:
- Vega-Altair website:
https://altair-viz.github.io/

Date of entry: 16.05
What I've worked on:
Today was the first try of visualizing the data I want to analyze. I made bar graphs of the average number of synapses in a typical wild-type (non-genetically modified) vs knock-out (genetically modified) mouse.

What problems I encountered:
-The altair library apparently has a problem with slightly larger data sets such as mine (over 5000 rows). I switched to seaborn again...
-Creating functions that plot data was a bit tricky. Especially if you want to loop over functions.

What I learned:
-Altair is good for interactive plotting but better seaborn for large data sets.
-If you create a plotting function, make it return the axes object it created, so you can later modify it.

Which resources did I use:
-A stackoverflow entry was useful to understand a bit more about axes in images.
-seaborn documentation:
https://seaborn.pydata.org/

Date of entry: 18.05
What I've worked on:
I finished the descriptive-plots part of my project. And moved on to doing some statistics. I did a simple t-test comparing the average number of synapses around neurons in wild-type mice vs knock-out mice.

What problems I encountered:
-What mainly still bothers me is that it is difficult to visualize as well as to statistically test the data I have because it is nested in two levels: Each mouse has multiple images. Each image has hundreds of neurons. Just taking an average per mouse is like throwing statistical power out of the window.
-I struggled a bit with plotting individual data values in the descriptive plots. Usually you have individual dots over bars, showing individual data points. It gave an error due to the type of data I gave it. I solved it by creating a groupby data frame where I show the mean values of images.

What I learned:
Important arguments in seaborn plots eg. dodge -> controls whether bars are plotted over or besides the xtick-labels. The function 'stripplot' is a way to plot a scatter plot over a regular bar plot in seaborn.

Which resources did I use:
-seaborn documentation.
-wikipedia entry on student's t-test.

Date of entry: 20.05
What I've worked on:
I did not code anything today. Just performing research on how to account for the nested structure of my data. In graphpad (a program to do scientific plots and statistics) there is a nested t-test, however, this does not seem to be the correct term. I found some possible tests I could perform.

What problems I encountered:
Wikipedia entries of statistical tests are quite heavy.

What I learned:
ANCOVAS can be used to control for variables that also vary with your independent variable. Also mixed effects models can be used (but I am unsure whether categorical data can be used).

Which resources did I use:
-The statistics version of stack overflow:
https://stats.stackexchange.com/
-Wikipedia

Date of entry: 22.05
What I've worked on:
I implemented a version of, what I think, is the correct statistical model to account for the nested structure of my data. I decided to implement a
linear mixed effects model.

What problems I encountered:
-I keep getting a "convergence" error from the test. Apparently it first uses its default method for fitting the model, and when that does not work it uses one alternative method. If both fail --> you fail. This may be due to outliers or failed assumptions. I'll have to implement assumption checks.

What I learned:
- In a linear mixed effects model "fixed effects" denote the effect we are interested in (eg what is the genotype of the mouse to which this neuron belongs) and "random effects" are the effects of variables we are not interested in such as, which specific mouse does this neuron belong to.
- It is good to test the assumptions of a model BEFORE implementing it.

Which resources did I use:
-Useful blog post on linear mixed effects models in python:
https://www.pythonfordatascience.org/mixed-effects-regression-python/
- Statsmodels documentation. More specifically for the linear mixed effects model:
https://www.statsmodels.org/devel/generated/statsmodels.regression.mixed_linear_model.MixedLM.html

Date of entry: 23.05
What I've worked on:
-I wrote code that tests the assumptions of a linear mixed effects model: 1) normal data (actually normally distributed errors for each predicted value) 2) equal variance in errors for every predicted value. I implemented each in the function that performs the linear mixed effects model. To test the normality of errors the function returns qq_plots and also another plot for the equal variance of errors. You apparently can visually guestimate if the data is normal enough.
- Added code that removes outliers and plots box plots to show if we have outliers.

What problems I encountered:
- large data means some tests do not work well: The only test I knew for testing normality of data, the shapiro wilk test, apparently gives incorrect answers when you give it very large sample sizes. I had to switch to the Anderson darling test.

-Non-normal data (*sigh*): The data is not normally distributed. However luckily only normality of errors for each predicted value of the dependent variable is important (wuhuu!)

-When outliers determine outliers: I started off using standard deviations to identify outliers. The problem is: extreme outliers will influence the standard deviation. I found on stats.stackexchange another measure known as the median absolute deviation, that identifies outliers based on the median.

What I learned:
-Use anderson darling test to determine normality for large sample sizes.
-linear mixed effects models only assume normality of erros for each predicted value of the dependent variable not the normality of data itself.
-Use the median absolute deviation for detecting outliers and not standard deviations.

Which resources did I use:
-Again the blog post mentioned before was very useful
-Again the statistics version of stack overflow.

Date of entry: 24.05
What I've worked on:
I took a break from statistics today to do research on how to perform image processing. I looked up what the steps are in a typical image segmentation process.

What problems I encountered:
There is no 100% 'right way' to analyze an image. You'll read a lot of 'Try this and see how it works with your images'.

What I learned:
- A typical image segmentation pipeline may look something like this:
1) Equalize the histogram of the image (greater variability between pixel intensities) 2) Smoothen image 3) apply a threshold 4) erode the results (delete small individual pixels not connected to others, avoids a scattered image segmentation) 5) dilate the segmentation a bit to make it larger again after the erosion.

Which resources did I use:
-ImageJ (an image processing program) has a nice documentation that explains concepts in image processing:
https://imagej.net/imaging/segmentation
-There is also a forum where I looked how people suggest you segment images
https://forum.image.sc/
- This blog post was gold. It helped me understand concepts such as gray-scale images and other techniques:
https://www.geeksforgeeks.org/image-segmentation-using-pythons-scikit-image-module/

Date of entry: 26.05
What I've worked on:
I continued working on the implementation of the linear mixed effects model. After removing outliers, all rounds of fitting the model converge with either the default method of the alternative 'plan B' method to converge.

What problems I encountered:
When trying to implement two levels in the hierarchy (neurons nested within images nested within mice) I receive the 'Singular matrix' error. This error likely results from a too complex model. However, nesting images within mice is frequently enough, because you commonly apply a mean to one image. Additionally, this data that I have now does not seem to show a lot of variation in between different images within mice. Almost all the variation seems to happen on the level of different mice, thus it is also okay to ignore the clustering level of the image and cluster neurons directly on mice. I did both and observed no difference.

What I learned:

-On a more meta level: There is no rule written in stone what statistical method is the one perfect one in a given situation. It depends a lot on the specifics of the data and what type of interpretation one wants to be able to make.
-A model that takes into account more variables can sometimes be worse than a model that takes into account less.

Which resources did I use:
-statsmodels documentation
-statistics stack exchange

Date of entry: 29.05
What I've worked on:
I started implementing the workflow I decided on a couple of days ago. Starting with how to correctly load images and display them in a jupyter notebook. I also started implementing a smoothing with the scipy.signal.convolve2d function.

What problems I encountered:
-switching images from a numpy array back to a PIL Image object, somehow took a lot of time. The trick was to convert the entries in the array to the correct data type (numpy unsigned 8-bit integer uint8)
-My smoothing also does not look very nice. It is just completely smeared.
-I am also unsure how to apply erosion and dilation with convolution. I read that you can implement the two with convolution, but I could not come up with any kernel that would fit.

What I learned:
-The length of the output of a convolution is the sum of the individual components.
-Gray-scale images are best represented with uint8 data types.

Which resources did I use:
- I watched two great videos on what convolution is and how it works on youtube:
https://www.youtube.com/watch?v=8rrHTtUzyZA
and
https://www.youtube.com/watch?v=KuXjwB4LzSA

-scipy documentation, specifically the convolve2d function:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html

-PIL library:
https://pillow.readthedocs.io/en/stable/reference/Image.html

Date of entry: 30.05
What I've worked on:
-I changed to implement the blurring of images with the .filter method for PIL Image objects. This way you can smooth using a gaussian, which looks nicer. I used a standard-deviation based threshold to segment images now.
-The OpenCV library also has a lot of nice functions I used to perform erosion, and dilation.

What problems I encountered:

What I learned:
- How to properly smooth an image, erode it or dilate it.

Which resources did I use:
- openCV documentation:
https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html
- PIL library

Date of entry: 31.05
What I've worked on:
I implemented the functions I wrote yesterday into the notebook. I applied the previously written functions for plotting and performing statistical tests.

What problems I encountered:
-Again the greatest challenge was 'how to code this so I can reuse it in the future'. I decided to create a Mouse class for this, since this time, loading the data was more complicated, as it was not only csv files but images, which had to be processed.

- Another problem was that the functions that I wrote previously could not be applied to the new data without problems. I had to go back and refine them to make them more generalizable.

What I learned:
- write functions as modular as possible in the beginning of a programming project. This will save you a lot of time and hassle later on.

Date of entry: 01.06
What I've worked on:
I cleaned up the code. Added more comments and wrote some more descriptions in markdown in the jupyter notebook. No very large changes, just appearance mostly and I changed the order of the notebook a bit.

What I learned:

-Creating doc strings for functions takes more time than expected.

Which resources did I use:
- mostly caffeine…