Fluorescence Microscopy Computational Model

Kwame Asiedu

Advised by
Professor Vidya Ganapati

May 11, 2018
## Contents

Abstract .......................................................... 3

Acknowledgements ............................................. 4

Background .......................................................... 5

I. What is Cancer ................................................. 5

II. Fluorescence Microscopy and Its Use in Cancer Detection............................................. 5

III. Autofluorescence ............................................. 8

My Project ......................................................... 13

I. Goals ................................................................ 13

II. Creating a Tissue Sample .................................. 13

III. Statistical Analysis for Characterizing Sample ......................................................... 15

A. 1st Intuition: Distribution Fitting ................................. 15

1. Maximum Likelihood Estimation (MLE) ......................... 18

B. 2nd Intuition: Curve Fitting .................................. 19

IV. Results ............................................................ 20

Future Work and Extensions .................................. 24

I. Constructing Optimization Algorithm for Scipy.stats.rv_continuous.fit ................. 24

II. Reducing Noise of the Sample Data ........................................... 24

III. Finding a Better Function to Fit to the Sample Data ........................................... 24

IV. Locating Actual Excitation and Emission Spectra for Lignin ................................ 25
Appendix A

I. A1: Code for Reading in Emission Spectra Text Files ........................................ 26
II. A2: Code for Making the Number of Emission Spectra Data Points the Same .......... 27
III. A3: Code for Final Creation of Sample Emission Spectrum .................................. 28
IV. A4: Code for Creating Alexa 488 and Lignin Emission Spectra Histograms ............ 28
V. A5: Code for Fitting Continuous Distributions to the Alexa 488 and Lignin Emission Spectra Histograms ................................................................. 29
VI. A6: Code for Creating Curve Fit Function for Curve Fit Method and Returning Optimal A and B parameters to Minimize Square of Summed Residuals ......................... 31

Appendix B

I. B1: Full Commented Code ................................................................................. 32

Bibliography
**Abstract**

Fluorescence microscopy is an extremely useful biological tool used to detect specific molecules within a specimen. Fluorescence microscopy is widely used for the purpose of cancer detection. When trying to locate cancerous regions of a specimen marked by a fluorophore, it is impossible to know whether the fluorescent signal one is seeing is from the fluorophore or an autofluorescent biological molecule. Using Python, I was able to create a computer program that uses a curve fitting method to classify a tissue sample as cancerous or non-cancerous. The program was pretty accurate for data with relatively little noise and got less accurate as data noise increased.
Acknowledgments

I would like to thank Professor Vidya Ganapati for her help throughout this entire project. When certain aspects of the project idea seemed infeasible at times to implement, she was willing to provide ideas that would make the project significant towards cancer research and feasible to implement within one school semester. I am forever grateful for her patience with me through this entire project design process.

I would also like to thank four Swarthmore College undergraduate students: Becky Tang, Daniel Belkin, Caleb Ho, and Alex Robey. Each of these students are highly knowledgeable in computer science and statistics. Without their help, I would not have known much about the statistical methods used in my project or how they worked. Additionally, it would have been difficult for me to able to implement my project using Python. For this, I am truly thankful for each one of them.

Finally, I would like to thank Professor Everbach for his continuous support throughout my whole project. Like Professor Vidya Ganapati, he also took the time to guide me in finding a feasible project idea. I am very thankful for his willingness to work with me even with his incredibly busy schedule.
Background

I. What is Cancer?

In short, cancer is a collection of diseases that arises from an overabundance of cells in the human body. If cells do not die like they are supposed to or new cells arise when they are not needed, an overaccumulation of cells occurs and this leads to the formation of masses of tissues called tumors. These tumors are capable of spreading to other tissues and distributing cancer cells to parts of the body that are far from where the original tumor was created. Cancer is responsible for one of every four deaths in the United States. It is estimated that in 2018, approximately 600,000 people in the United States will die from cancer. Quick and early detection of tumors is crucial in decreasing the number of people that suffer from this terrible collection of diseases.

II. Fluorescence Microscopy and Its Use in Cancer Detection

One means by which researchers detect cancer is through a technique called fluorescence microscopy. Fluorescence microscopy is a significant biological tool that can image specific features of a small specimen. In order to image these features, one has to place a fluorescent chemical marker within the specimen called a fluorophore. A fluorophore will selectively bind to a specific target molecule and emit light, or fluoresce, when excited by light of a particular wavelength. This kind of useful attribute is non-existent in other traditional optical microscopy methods. The use of fluorophores makes it fairly simple to identify certain biological molecules.

1“What is Cancer?”, National Cancer Institute, last modified February 9, 2015, https://www.cancer.gov/about-cancer/understanding/what-is-cancer
as it is easy to visually distinguish fluorescent material from non-fluorescent material. Figure 1 shows a schematic of a fluorescence microscope.

The three components of the microscope that I will explain for the purpose of having a better understanding of the project are as follows:

**Light Source:** Light sources for fluorescence microscopy usually emit light of wavelengths that cover the entire visible spectrum. These light sources are usually in the form of an array of LEDs in the hopes that this array will mimic white light sources. LEDs have uniform and controllable light intensity which ensures that in the case of multiple fluorescence labeling, one fluorophore does not emit light at a higher intensity than another fluorophore. Additionally,

---


LEDs exhibit light intensities at relatively higher powers to ensure that the image of the specimen is clear. It is for all these reasons that LEDs are preferred over other light sources with non-uniform intensities and/or lower powers of light such as mercury arc lamps, xenon arc lamps, and metal halide lamps.  

**Excitation Filter**: The excitation filter is an optical-glass bandpass filter that transmits a specific range of light wavelengths. When choosing an excitation filter, one has to make sure that the filter will only transmit light that will excite the fluorophore of interest. Most excitation filters will transmit light of relatively short wavelengths to ensure that the fluorophore receives light that's high enough in energy to cause it to fluoresce. This makes sense when considering Planck's Equation: $$E \propto \frac{hc}{\lambda^4}.$$  

**Dichroic Mirror**: The dichroic mirror is a thin piece of coated glass that reflects the light filtered by the excitation filter and transmits the fluorescent light of the fluorophore of interest. This mirror is placed at a $45^\circ$ angle to the area where the specimen is placed so that the excitation light can be reflected towards the specimen while allowing the fluorescent light of the fluorophore to pass through the mirror. Dichroic mirrors are longpass filters because the emitted light of the fluorophore is usually of a longer wavelength than the light coming out of the excitation filter.  

**Emission Filter**: Like the excitation filter, the emission filter is a bandpass filter that will transmit a certain range of wavelengths of light while attenuating other wavelengths of light. For

---

this reason, the builder of the microscope has to choose an emission filter that will only transmit
the light emitted by the fluorophore. It is very important that the light filtered by the excitation
filter is attenuated by this emission filter. If this does not happen, the user of the microscope will
see the excitation light and not the emission light.\textsuperscript{10}

What does this all mean for cancer detection? As was stated above, a fluorophore is a
fluorescent chemical marker that will selectively bind to a specific target molecule and fluoresce
when excited with light of a particular wavelength. Conveniently, there are fluorophores that can
indirectly bind to tumors in the body. Gliomas are a type of brain tumor that arise from the glial
cells of the brain. These gliomas take up large amounts of a protein called albumin. Researchers
studying these tumors found that a fluorophore called 5-aminofluorescein selectively attaches to
albumin.\textsuperscript{11} More recently, a fluorophore known as Alexa 488 is used in place of
5-aminofluorescein. Because it is known that Alexa 488 selectively binds to albumin and
gliomas take up albumin, Alexa 488 can be used to image gliomas in a specimen. This one
fluorophore can aid one in the detection of cancerous tissue.

III. \textit{Autofluorescence}

It seems like the process of imaging cancerous tissue in the body is fairly simple. Place
the fluorophore within the specimen, allow it to attach to the target molecule of interest, in this
case the gliomas, and use the fluorescence microscope to figure out where in the specimen these

\textsuperscript{10} Wymke Ockenga, “Fluorescence in Microscopy”, \textit{Leica Microsystems}, last modified April 27, 2011,
\textsuperscript{11} Helmholtz Association of German Research Centres, "Fluorescent Cancer Cells To Guide Brain Surgeons",
fluorophore-covered gliomas are. However, there is an issue that arises when using the microscope to look for these tumors. There are certain biological structures in the body that will emit light naturally when excited by light of a particular wavelength. These biological structures are said to be exhibiting autofluorescence.\(^\text{12}\) Autofluorescence is a significant problem when conducting fluorescence microscopy. A researcher trying to detect certain tumors in the body might mistake an autofluorescent signal for a fluorophore signal or vice versa. Either the removal of an essential biological structure mistaken for cancerous tissue or the negligence of cancerous tissue mistaken for a biological structure could lead to a fatal outcome.

Lignin is a biological molecule that exhibits autofluorescence when excited with light of a wavelength that it is similar to the excitation wavelength of Alexa 488 (488 nm).\(^\text{13}\) Figure 2 shows that the peak excitation wavelength of Alexa 488 is virtually the same as that of lignin (Note: I was not able to locate the actual excitation spectrum for lignin. However it is indeed known that the peak excitation wavelength of lignin is the same as that of Alexa 488 (488 nm). The lignin excitation spectrum that is shown in Figure 2 is the emission spectrum of another fluorophore called Oregon 488. I am going to assume that the Oregon 488 excitation spectrum corresponds to the excitation spectrum of lignin. For the purpose of my computational model, which is to distinguish between a fluorescent signal of interest and an undesired autofluorescent signal, making this assumption is not a problem. I am confident that my model would work had I been able to find the excitation spectrum for lignin). The spectra are almost indistinguishable. This means that when trying to excite Alexa 488, lignin will inevitably emit light. This would not


be a problem if the light that the lignin emits is distinguishable from that of Alexa 488. But this is not the case. Figure 3 depicting the emission spectra of Alexa 488 and lignin overlaid on top of one another shows that Alexa 488 and lignin emit identical colors of light when excited (Note: I was not able to locate the actual emission spectrum for lignin. However it is indeed known that the peak emission wavelength of lignin is similar to that of Alexa 488 (~530 nm). The lignin emission spectrum that is shown in Figure 3 is the emission spectrum of another fluorophore called Oregon 488. I am going to assume that the Oregon 488 emission spectrum corresponds to the emission spectrum of lignin. For the purpose of my computational model, which is to distinguish between a fluorescent signal of interest and an undesired autofluorescent signal, making this assumption is not a problem. I am confident that my model would work had I been able to find the emission spectrum for lignin).

---

Figure 2: Excitation spectrum of Alexa 488 overlaid with the excitation spectrum of lignin (Note: I was not able to locate the actual excitation spectrum of lignin. The excitation spectrum for lignin shown is actually the excitation spectrum of a fluorophore called Oregon 488. For the purposes of my computational model however, this does not create an issue).

Figure 3: Emission spectrum of Alexa 488 overlaid with the emission spectrum of lignin (Note: I was not able to locate the actual emission spectrum of lignin. The emission spectrum for lignin
shown is actually that of a fluorophore called Oregon 488. For the purposes of my computational model however, this does not create an issue).
My Project

I. Goals
I wanted to create a fluorescence microscopy computational model using Python that: 1) depicts realistic emission spectra data of a tissue sample and 2) uses statistical methods to identify whether said tissue sample is cancerous or non-cancerous.

II. Creating a Tissue Sample

In order to create a hypothetical tissue sample using Python, I had to acquire text files that contained wavelengths of light and their intensities of emitted light for the Alexa 488 fluorophore and lignin. The text files were set up very strangely in that the wavelength data contained letters and that the intensities were not relative. For this reason I created a function in my program called read_spectrum, which fixed the format of the wavelength values and made the intensity values relative (see Appendix A1 for commented code). After reading in the files, I had to make sure that the amount of data points for the emission spectra were the same (they were not the same in the original text file)(see Appendix A2 for commented code). After creating the emission spectra of Alexa 488 and lignin in Python (created plots shown in Figure 4), I formulated an equation to create a realistic tissue sample that was a mixture of the Alexa 488 and lignin emission spectra (because in reality, a tissue sample is going to have a mixture of cancerous and non-cancerous cells). I created the equation such that I also could have control over whether the sample I created was cancerous or not. This would enable me to verify that the statistical methods I would later use were successful in characterizing the sample. This equation was: \((A \times \text{Relative Intensities of Alexa 488}) + (B \times \text{Relative Intensities of Lignin}) + \varepsilon\), where \(A\) and \(B\) were the weights associated with the relative intensities of Alexa 488 and lignin.
respectively and \( \varepsilon \) was some noise in order to make the sample data more representative of actual tissue sample data (real data will always have some associated noise). In choosing the values for \( A \) and \( B \), I assumed that the tissue sample of interest was 1 cm\(^3\). Because this was the case, that meant that if at least half of the cells from the sample were covered with Alexa 488, then the sample could be deemed cancerous.\(^{15}\) The noise (\( \varepsilon \)) was a normal continuous random variable with a standard deviation of 0.05 and size of 191 (there were 191 wavelengths in both the Alexa 488 and lignin data)(see Appendix A3 for commented code). Figure 5 shows the final product of the sample’s emission spectrum.

Figure 4: Python Generated Emission Spectra for Alexa 488 and Lignin.

\(^{15}\) “How Many Cancer Cells are in a Tumor?”, Dr. Farrah Cancer Center, accessed May 11, 2018, http://drfarrahcancercenter.com/how-many-cancer-cells-are-in-a-tumor/
Figure 5: Python Generated Emission Spectrum for Tissue Sample.

III. Statistical Analysis for Characterizing Sample

A. 1st Intuition: Distribution Fitting

When looking at the emission spectra for Alexa 488 and lignin, I noticed that they took the shape of some sort of probability distribution. Knowing that certain probability distributions can be parameterized, my first intuition for characterizing this sample was to fit distributions to both the Alexa 488 and lignin emission spectra. If it turned out that there was a parameterized distribution that fit both the Alexa 488 and lignin emission spectra, then I could just find the parameters associated with that distribution for the Alexa 488 and lignin emission spectra and compare them to the parameters associated with that distribution for the sample emission spectra. Whichever emission spectrum (Alexa 488 or lignin) had distribution parameters closer to those of the sample emission spectrum would be the spectrum from which most of the sample (at least
half of it) came from. Having this intuition, the first thing I had to do in order to fit various probability distributions to the Alexa 488 and lignin plots was to create histogram representations of the emission spectra data. This histogram representations are shown in Figure 6 (see Appendix A4 for commented code).

Figure 6: Python Generated Histogram of Alexa 488 and Lignin Emission Spectra.

In Python, there is a library called SciPy, which contains eighty-two different continuous distributions that one can fit to histograms. I went through each distribution and fit them to the histograms that I created (see Appendix A5 for commented code). After looking at how each distribution fit the histograms, I chose the ones the distributions that fit the best. The best-fitting distributions are shown in Figures 7 and 8.

---

Figure 7: Best Fitting Continuous Distributions for Alexa 488 Emission Spectrum Histogram.
1) Maximum Likelihood Estimation (MLE)

After finding the best fitting continuous distributions for the Alexa 488 and lignin emission spectra histograms, I chose to find the parameters that corresponded to a distribution that fit both the Alexa 488 and lignin data very well. I chose the Burr distribution. In order to
find the Burr distribution parameters associated with the Alexa 488, lignin, and the sample data, I
decided to use a statistical method called maximum likelihood estimation (MLE). Using MLE,
one can find the parameter values for a model (in this case, a distribution). MLE returns
parameters that maximize the likelihood that the process described by the model (distribution)
produced the observed data.\textsuperscript{17} Therefore by using MLE, I would be able to compare the Burr
distribution parameters of the sample data to those of the Alexa 488 and lignin data. This could
be done using a function in the SciPy library called scipy.stats.rv_continuous.fit. This function
takes as one of its parameters an optimization algorithm.\textsuperscript{18} Unfortunately, due to time constraints
and the difficulty that came with trying to devise my own optimization algorithm from scratch, I
had to abandon the MLE approach. While this was disappointing, I ended up forming another
intuition, one that would help me achieve my goal in a more direct manner.

\textbf{B. 2nd Intuition: Curve Fitting}

Instead of fitting distributions to histogram representations of the spectra data, I decided
to create a function that I would try to fit to the original spectra data. My intuition was to create a
function that contained the A and B weights associated with the original equation used to create
the sample data. By doing this, I could try to find some way in Python to have the program return
A and B weights that minimized the distance between the fitted curve and the sample data.
Conveniently, there is a function in the SciPy library called scipy.optimize.curve_fit that would
return optimal A and B weights so that the sum of the squared residuals, i.e., the sum of the

squared distances between the fitted curve and the sample data, is minimized. Therefore, by using this scipy.optimize.curve_fit function, I should get back A and B values that are very close to the ones that I inputted when creating the sample data, or at least values that give the same characterization of the sample that I gave it when inputting the A and B values. For example, if I inputted $A = 0.2$ and $B = 0.8$ when creating the sample data, making the sample non-cancerous, the scipy.optimize.curve_fit function should return an $A$ value close to 0.2 and a $B$ value close to 0.8. Alternatively, it should return an $A$ value that is less than 0.5 and a $B$ value greater than 0.5, indicating that the tissue sample is indeed non-cancerous (see Appendix A6 for commented code).

IV. Results

The results for the curve fit method are shown in Figures 9-12

Figure 9: Typical Curve Fit Output for $A = 0.1$ and $B = 0.9$.

Figure 10: Typical Curve Fit Output for $A = 0.3$ and $B = 0.7$.

As one can see in Figures 9-11, it is very clear whether the sample is cancerous or not, i.e., A is significantly less heavily weighted than B and so I should expect an output of the scipy.optimize.curve_fit function to have similar A and B values or have A and B values that indicate that the sample is non-cancerous. I ran into some difficulty with the accuracy of the scipy.optimize.curve_fit function when the A to B ratio got closer to 50/50. When this happened, specifically, when A was equal to 0.48 and B was equal to 0.52, the output parameters were correct 69% of the time (I ran the program 100 times and 69 times out of those 100 times, the program correctly classified the sample). While it is not perfect, this method is still a relatively
efficient method considering it is a computational model. Approximately 70% of the time, one can correctly classify whether a tissue sample is cancerous or non-cancerous.

Considering the possibility that there could be a noisier sample dataset than the one in Figure 5, I tested the curve fitting method with noisier data (noisier data shown in Figure 13).

Setting A equal to 0.48 and B equal to 0.52, the output parameters were correct 52% of the time (I ran the program 100 times and 52 times out of those 100 times, the program correctly classified the sample). This decrease in accuracy is to be expected with noisier data. Again, this method is not perfect by any means, but it does not seem to be terrible. I expect that the accuracy of this program will decrease as the data gets noisier than the data shown in Figure 13, but I
would also argue that a good quality fluorescence microscope would not produce data noisier than what is shown in Figure 13.
Future Work and Extensions

I. Constructing Optimization Algorithm for Scipy.stats.rv_continuous.fit

While I am confident that my second intuition, the curve fitting method, is a more direct approach than my first intuition, the distribution fitting method, I would still like to see how well the distribution fitting method works. I was not able to carry out the distribution fitting method because the function needed to calculate the distribution parameters for the emission spectra data, scipy.stats.rv_continuous.fit, required an optimization algorithm as one of its parameters, an algorithm I was not able to construct. Given more time, I would like to construct this algorithm in order to determine how well the distribution fitting method works.

II. Reducing Noise of the Sample Data

If I had known of a way to reduce the noise of the sample data after creating it, I believe that the curve fitting method would have been more accurate. I would like to investigate a machine learning technique called Principal Component Analysis that can be used for noise reduction and see if this technique could make my model more accurate.20

III. Finding a Better Function to Fit to the Sample Data

The function that I fit to the sample data was a function based on the original equation of the sample data. While this seems like a relatively good function to fit to the data, I’m curious as to whether there is an even better function to fit that would make the model more accurate.

---

IV. Locating Actual Excitation and Emission Spectra for Lignin

It would have been nice to obtain the actual excitation and emission spectra for lignin, but again, for the purpose of my project which was to characterize a sample as being cancerous or non-cancerous, this did not matter. I am confident that had I found the actual spectra for lignin, my model would have still worked.
Appendix A

A1: Code for Reading in Emission Spectra Text Files

```python
def read_spectrum(fname):
    
    Purpose: Organizes the Alexa 488 and Lignin emission spectra data
    Parameters: fname (text file)
    Returns: List of wavelengths and their relative intensities

    with open(fname, 'r') as f:
        data = f.readline().split() #splits text file into individual string
        #elements in list

        x_data, y_data = [], [] #empty lists for x and y values

        for pt in data[2:]: #first two points we'll throw away because
            if pt[0].isalpha():
                x_data.append(int(pt[1:])) #There are letters next to each
                #wavelength data point. Each time we come across
                #a letter, we want to append the numerical part of the element to
                #the x_data
            else:
                y_data.append(float(pt)) #otherwise append the value as a float to
                #the y_data_list; these correspond the intensities

        #we want relative intensities
        max_y = max(y_data) #take max y value
        y_data = [1 - y / float(max_y) for y in y_data] #and divide each value in
        #list by this max y value to get relative intensity

    return x_data, y_data #return list of x and y values
```

26
A2: Code for Making the Number of Emission Spectra Data Points the Same

```python
alexa = 'alexa_488.txt' #Alexa 488 text file with wavelengths and relative intensities
lignin = 'oregon_488.txt' #Oregon 488 text file, representing lignin, with wavelengths and relative intensities

x, y = read_spectrum(alexa) #obtains x and y values from Alexa 488 text file used to plot graph
x2, y2 = read_spectrum(lignin) #obtains x and y values from Oregon 488 text file used to plot graph

alexa_xy = dict(zip(x, y)) #gets rid of duplicate x values; also maps x #value in x list to corresponding y value in y list
lignin_xy = dict(zip(x2, y2)) #gets rid of duplicate x values; also maps #x value in x list to corresponding y value in y list
fst = lambda pair: pair[0] #function called first; returns x coordinate of #pairs
alexa_xy = sorted(alexa_xy.items(), key=fst) #fst returns x coordinate #((pair[0]) and so we're sorting by increasing x elements
lignin_xy = sorted(lignin_xy.items(), key=fst) #fst returns x coordinate #((pair[0]) and so we're sorting by increasing x elements
alexa_x = map(fst, alexa_xy) #takes list of pairs, applies fst to each #element and returns the result; output is a list of increasing x values #for Alexa 488
lignin_x = map(fst, lignin_xy) #takes list of pairs, applies fst to each #element and returns the result; output is a list of increasing x values for Lignin

print("Minimum Alexa 488 wavelength: %d nm" %min(alexa_x)) #475 nm
print("Maximum Alexa 488 wavelength: %d nm" %max(alexa_x)) #675 nm
print("Minimum Lignin wavelength: %d nm" %min(lignin_x)) #485 nm
print("Maximum Lignin 488 wavelength: %d nm" %max(lignin_x)) #699 nm

#Here, I was trying to see whether the data points that were going to be #plotted matched up.
#They didn't, so this needs to be fixed.

alexa_x = alexa_x[10:] #cut off first 10 wavelength values for Alexa 488

print("Length of alexa_x list: %d" %len(alexa_x))) #Length of this list #is 191

alexa_y = y
print("Length of alexa_y list: %d" %len(alexa_y))) #Length of this list #is 203.

alexa_y = y[10:201] #So in addition to cutting off first 10 relative #intensity values, we're also going to cut off last #two relative intensity values in order to make the length of the alexa_y
```
#list also 191.

print("Length of lignin_x list: \%d\" % (len(lignin_x))) #Length of this list
#is 215

lignin_x = lignin_x[:len(lignin_x)-24] #cut off last 24 wavelength values
#for Lignin plot

lignin_y = y2

print("Length of lignin_y list: \%d\" % (len(lignin_y))) #Length of this list
#is 217

lignin_y = lignin_y[:len(lignin_y)-26] #So cut off last 26 relative
#intensity values for Lignin plot

---

**A3: Code for Final Creation of Sample Emission Spectrum**

A = 0.48
B = 0.52

_alexa_y = np.array(alexay) #y values of Alexa 488 emission spectrum in
#array form

_lignin_y = np.array(lignin_y) #y values of Lignin emission spectrum in
#array form

epsilon = stats.norm.rvs(scale=0.05, size=191) #epsilon = array of normally
#distributed numbers with mean = 0 and standard deviation being small

sample = (A*np.array(_alexa_y))+(B*np.array(_lignin_y)) + epsilon
#equation to mix Alexa 488 and Lignin emission spectra; add epsilon to make
#data more realistic

plt.plot(alexa_x, sample, 'go', label='Sample') #plot sample
plt.ylabel('Relative Intensity (%)')
plt.xlabel('Wavelength (nm)')

plt.legend()
plt.show()

---

**A4: Code for Creating Alexa 488 and Lignin Emission Spectra Histograms**

alexa_frequency_list = [] #list that will hold the number of times
#certain wavelengths of the Alexa 488 are illuminated when
#it is excited. This correlates to the relative intensity of
#each wavelength.

for i in range(len(alexay)): #Go through wavelength, relative intensity
#pairs for the Alexa 488 emission spectrum

alexa_frequency = (alexay[i][0], int(round(((alexay[i][1])*100),0)))
#Calculate the frequency of a particular wavelength based on the relative
#intensity. Multiply the relative intensity by 100 and round to the
#nearest whole number
for i in range(alexa_frequency[1]): #For the number of times a certain wavelength appears
    alexa_frequency_list.append(alexa_frequency[0]) #Append that wavelength to alexa_frequency_list
plt.xlim((alexa_xy[0][0]), (alexa_xy[len(alexa_xy)-1][0])) #sets x limits for plot
alexa_hist = plt.hist(alexa_frequency_list, len(alexa_xy), density = True, range = (alexa_xy[0][0], alexa_xy[len(alexa_xy)-1][0]), facecolor = 'red', label = 'Alexa 488') #plots histogram

lignin_frequency_list = [] #list that will hold the number of times certain wavelengths of the lignin are illuminated when it is excited. This correlates to the relative intensity of each wavelength
for i in range(len(lignin_xy)): #Go through wavelength, relative intensity pairs for the lignin emission spectrum
    lignin_frequency = (lignin_xy[i][0]), int(round(((lignin_xy[i][1])*100),0)) #Calculate the frequency of a particular wavelength based on the relative intensity. Multiply the relative intensity by 100 and round to the nearest whole number
for i in range(lignin_frequency[1]): #For the number of times a certain wavelength appears
    lignin_frequency_list.append(lignin_frequency[0]) #Append that wavelength to lignin_frequency_list
plt.xlim(lignin_xy[0][0]), (lignin_xy[len(lignin_xy)-1][0]) #sets x limits for plot
lignin_hist = plt.hist(lignin_frequency_list, len(lignin_xy), density = True, range = (lignin_xy[0][0], lignin_xy[len(lignin_xy)-1][0]), facecolor = 'blue', label = 'Lignin') #plots histogram
plt.legend() plt.xlabel('Wavelength (nm)') plt.ylabel('Relative Intensity') plt.show()

A5: Code for Fitting Continuous Distributions to the Alexa 488 and Lignin Emission Spectra Histograms:

#List of continuous distributions from scipy.stats

for dist_name in dist_names:
    #dist_name = dist_name.replace('"', '') #make the dist name a non-string item
    print("Alexa 488; Distribution: %s" %dist_name)
    x = np.linspace(alexa_xy[0][0], alexical_y[(len(alexa_xy)-1)][0], 1000)
    distribution = getattr(stats, dist_name)
    param = distribution.fit(alexa_frequency_list, floc=0) #
    pdf_fitted = distribution.pdf(x, *param) #Creates pdf
    alexical_hist = plt.hist(alexa_frequency_list, len(alexa_xy), density = True, range = (alexa_xy[0][0], alexical_xy[len(alexa_xy)-1][0]), facecolor = 'red')
    #Plot Alexa 488 emission spectrum histogram
    plt.plot(x, pdf_fitted, color = 'green') #Plots pdf over histogram
    plt.xlabel('Wavelength (nm)')
    plt.ylabel('Relative Intensity')
    plt.show()
    raw_input() #Go to next dist_name when you press Enter

for dist_name in dist_names:
    print("Lignin; Distribution: %s" %dist_name)
    x = np.linspace(lignin_xy[0][0], lignin_xy[(len(lignin_xy)-1)][0], 1000)
    distribution = getattr(stats, dist_name)
    param = distribution.fit(lignin_frequency_list, floc=0)
    pdf_fitted = distribution.pdf(x, *param) #Creates pdf
    lignin_hist = plt.hist(lignin_frequency_list, len(lignin_xy), density = True, range = (lignin_xy[0][0], lignin_xy[len(alexa_xy)-1][0]), facecolor = 'blue')
    #Plot Lignin emission spectrum histogram
    plt.plot(x, pdf_fitted, color = 'green') #Plots pdf over histogram
    plt.xlabel('Wavelength (nm)')
    plt.ylabel('Relative Intensity')
    plt.show()
    raw_input() #Go to next dist_name when you press Enter
A6: Code for Creating Curve Fit Function for Curve Fit Method and Returning Optimal A and B Parameters to Minimize Square of Summed Residuals

```python
x = range(len(alexa_x))
A = 0.48
B = 0.52

_alexa_y = np.array(alexa_y)  # y values of Alexa 488 emission spectrum in array form
_lignin_y = np.array(lignin_y)  # y values of Lignin emission spectrum in array form
func = lambda x, A, B: A * _alexa_y + B * _lignin_y  # function to fit sample data to
print optimization.curve_fit(func, range(len(alexa_x)), sample)  # Prints optimal A and B weights for func to minimize the sum of squared residuals
```
Appendix B
B1: Full Commented Code:

#!/usr/bin/env python2
# -*- coding: utf-8 -*-

Created on Tue Mar 20 15:39:38 2018

@author: kwameasiedu
E90 Project: Fluorescence Microscopy Computational Model
Kwame Asiedu, 2018

import matplotlib.pyplot as plt
from scipy import stats
import scipy.optimize as optimization
import numpy as np

def main():
    #ORGANIZING ALEXA 488 AND LIGNIN EMISSION SPECTRA DATA
    alexa = 'alexa_488.txt' #Alexa 488 text file with wavelengths and intensities
    lignin = 'oregon_488.txt' #Oregon 488 text file, representing lignin, with wavelengths and intensities
    x, y = read_spectrum(alexa) #obtains x and y values from Alexa 488 text file used to plot graph
    x2, y2 = read_spectrum(lignin) #obtains x and y values from Oregon 488 text file used to plot graph

    alexa_xy = dict(zip(x, y)) #gets rid of duplicate x values; also maps x value in x list to corresponding y value in y list
    lignin_xy = dict(zip(x2, y2)) #gets rid of duplicate x values; also maps x value in x list to corresponding y value in y list

    fst = lambda pair: pair[0] #function called first; returns x coordinate of pairs
    alexa_xy = sorted(alexa_xy.items(), key=fst) #fst returns x coordinate of pairs
    lignin_xy = sorted(lignin_xy.items(), key=fst) #fst returns x coordinate of pairs

    alexa_x = map(fst, alexa_xy) #takes list of pairs, applies fst to each element and returns the result; output is a list of increasing x values for Alexa 488
    lignin_x = map(fst, lignin_xy) #takes list of pairs, applies fst to each element
# element and returns the result; output is a list of increasing x values for Lignin
print("Minimum Alexa 488 wavelength: %d nm \%min(alexa_x)\) #475 nm
print("Maximum Alexa 488 wavelength: %d nm \%max(alexa_x)\) # 675 nm
print("Minimum Lignin wavelength: %d nm \%min(lignin_x)\) #485 nm
print("Maximum Lignin 488 wavelength: %d nm \%max(lignin_x)\) #699 nm

#Here, I was trying to see whether the data points that were going to be
#plotted matched up.
#They didn't, so this needs to be fixed.
alexa_x = alexa_x[10:] #cut off first 10 wavelength values for Alexa 488
#plot
print("Length of alexa_x list: %d \%len(alexa_x)\) #Length of this list
#is 191
alexay = y
print("Length of alexa_y list: %d \%len(alexay)\) #Length of this list
#is 203.
alexay = y[10:201] #So in addition to cutting off first 10 relative
#intensity values, we're also going to cut off last
#two relative intensity values in order to make the length of the alexa_y
#list also 191.
print("Length of lignin_x list: %d \%len(lignin_x)\) #Length of this list
#is 215
lignin_x = lignin_x[:\{(len(lignin_x)-24)\}] #cut off last 24 wavelength values
#for Lignin plot
lignin_y = y2
print("Length of lignin_y list: %d \%len(lignin_y)\) #Length of this list
#is 217
lignin_y = lignin_y[:\{(len(lignin_y)-26)\}] #So cut off last 26 relative
#intensity values for Lignin plot

# PLOTTING ALEXA 488 AND LIGNIN EMISSION SPECTRA DATA

plt.figure()
plt.plot(alexa_x, alexa_y, 'ro', label='Alexa 488')
plt.plot(lignin_x, lignin_y, 'bo', label='Lignin')
plt.ylabel('Relative Intensity (%)')
plt.legend()
plt.show()

# CREATING SAMPLE DATA/USING CURVE FITTING METHOD
x = range(len(alexa_x))
A = 0.48 #Weight associated with Alexa 488
B = 0.52 #Weight associated with Lignin

_alexa_y = np.array(alexa_y) #y values of Alexa 488 emission spectrum in
#array form
_lignin_y = np.array(lignin_y)  #y values of Lignin emission spectrum in #array form
func = lambda x, A, B: A * _alexa_y + B * _lignin_y  #function to fit sample #data to
epsilon = stats.norm.rvs(scale=0.05, size=191)  #epsilon = array of normally #distributed numbers with mean = 0 and standard deviation being small
sample = (A*np.array(_alexa_y))+(B*(np.array(_lignin_y))) + epsilon  #equation to mix Alexa 488 and Lignin emission spectra; add epsilon to make #data more realistic (more noisy)
plt.plot(alexa_x, sample, 'go', label='Sample')  #plot sample
plt.ylabel('Relative Intensity (%)')
plt.xlabel('Wavelength (nm)')
plt.legend()
plt.show()

print optimization.curve_fit(func, range(len(alexa_x)), sample)  #Returns optimal A and B parameters to minimize sum of squared residuals

#USING DISTRIBUTION FITTING METHOD

alexa_frequency_list = []  #list that will hold the number of times #certain wavelengths of the Alexa 488 are illuminated when #it is excited. This correlates to the relative intensity of #each wavelength.
for i in range(len(alexa_xy)):  #Go through wavelength, relative intensity #pairs for the Alexa 488 emission spectrum
    alexa_frequency = (alexa_xy[i][0], int(round(((alexa_xy[i][1])*100),0)))
    #Calculate the frequency of a particular wavelength based on the relative #intensity. Multiply the relative intensity by 100 and round to the #nearest whole number
    for i in range(alexa_frequency[1]):  #For the number of times a certain #wavelength appears
        alexa_frequency_list.append(alexa_frequency[0])  #Append that #wavelength to alexa_frequency_list
plt.xlim((alexa_xy[0][0]), (alexa_xy[len(alexa_xy)-1][0]))  #sets x limits #for plot
alexa_hist = plt.hist(alexa_frequency_list, len(alexa_xy), density = True, range = (alexa_xy[0][0], alexa_xy[len(alexa_xy)-1][0]), facecolor = 'red', label = 'Alexa 488')  #plots histogram
lignin_frequency_list = []  # list that will hold the number of times
# certain wavelengths of the lignin are illuminated when
# it is excited. This correlates to the relative intensity of
# each wavelength

for i in range(len(lignin_xy)):  # Go through wavelength, relative intensity
    # pairs for the lignin emission spectrum
    lignin_frequency = (lignin_xy[i][0], int(round(((lignin_xy[i][1]*100),0))
    # Calculate the frequency of a particular wavelength based on the relative
    # intensity. Multiply the relative intensity by 100 and round to the
    # nearest whole number

for i in range(lignin_frequency[1]):  # For the number of times a certain
    # wavelength appears
    lignin_frequency_list.append(lignin_frequency[0])  # Append that
    # wavelength to lignin_frequency_list

plt.xlim(lignin_xy[0][0], (lignin_xy[len(lignin_xy)-1][0]))  # sets x limits

for dist_name in dist_names:
    dist_name = dist_name.replace("", ")  # make the dist name a
    # non-string item

    x = np.linspace(alexa_xy[0][0], alexa_xy[(len(alexa_xy)-1)], 1000)

distribution = getattr(stats, dist_name)
param = distribution.fit(alexa_frequency_list, floc=0)
pdf_fitted = distribution.pdf(x, *param) #Creates pdf
alexा_hist = plt.hist(alexа_frequency_list, len(alexа_xy), density = True, range=(alexа_xy[0][0], alexа_xy[len(alexа_xy)-1][0]), facecolor = 'red')
#Plot Alexa 488 emission spectrum histogram
plt.plot(x, pdf_fitted, color='green') #Plots pdf over histogram
plt.xlabel('Wavelength (nm)')
plt.ylabel('Relative Intensity')
plt.show()

for dist_name in dist_names:
    print("Lignin; Distribution: %s" %dist_name)
x = np.linspace(lignин_xy[0][0], lignин_xy[len(lignин_xy)-1][0], 1000)
distribution = getattr(stats, dist_name)
param = distribution.fit(lignин_frequency_list, floc=0)
pdf_fitted = distribution.pdf(x, *param) #Creates pdf
lignин_hist = plt.hist(lignин_frequency_list, len(lignин_xy), density = True, range=(lignин_xy[0][0], lignин_xy[len(lignин_xy)-1][0]), facecolor = 'blue')
#Plot Lignin emission spectrum histogram
plt.plot(x, pdf_fitted, color='green') #Plot pdf over histogram
plt.xlabel('Wavelength (nm)')
plt.ylabel('Relative Intensity')
plt.show()
raw_input() #Go to next dist_name when you press Enter

#READING IN EMISSION SPECTRA TEXT FILES; Text files came from:
#https://www.aatbio.com/spectrum/Alexa_Fluor_488
def read_spectrum(fname):
    """
    Purpose: Organizes the Alexa 488 and Lignin emission spectra data
    Parameters: fname (text file)
    Returns: List of wavelengths and their relative intensities for Alexa 488
    and Lignin Emission Spectra
    """
    with open(fname, 'r') as f: #opens text file
        data = f.readline().split() #splits text file into individual string
        #elements in list
        x_data, y_data = [], [] #empty lists for x and y values

        for pt in data[2:]: #first two points we'll throw away because they don't
            #make sense
            if pt[0].isalpha():
x_data.append(int(pt[1:])) # There are letters next to each
# wavelength data point. Each time we come across
# a letter, we want to append the numerical part of the element to
# the x_data list
else:
    y_data.append(float(pt)) # otherwise append the value as a float to
    # the y_data list; these correspond to the intensities

# we want relative intensities
max_y = max(y_data) # take max y value
y_data = [1 - y / float(max_y) for y in y_data] # and divide each value in
# list by this max y value to get relative intensity

return x_data, y_data # return list of x and y values

if __name__ == '__main__':
    main()
Bibliography


