05_py_fund_visualizing_data

August 12, 2021

1 Visualizing data

Can you look at at matrix of numbers and see what's going on with our test data? I can't if it's a lot of people or a lot of days.

We humans are really good though at looking at images. This is one of the reasons why data visualization is such a fundemental part of data analysis: your brain can do all sorts of really complicated processing when it sees images.

Python has some really great ways to visualize out data. One of the standards is a library called `matplotlib`. Inside `matplotlib` is a collection of plotting tools called `pyplot` that are used all the time. Throughout the course we are going to make some great plots with `matplotlib`.

Let's `matplotlib` out and use it to investigate our data.

1.0.1 installing `matplotlib` with (Ana)conda (*shouldn't be necessary because it comes installed with Anaconda by default, but just in case...*)

to get `matplotlib` we follow the same process that we did with `numpy` (you only need to do this the first time we go throug this): 1. go to the little `+` on the left to open a `launcher` window. 1. click the 'terminal' tile to open a terminal 1. type `conda env list` and hit enter. Make sure that there is a `*` next to the line that says swbc 1. if that's right type `conda install matplotlib` 1. when it asks `proceed ([y]/n)?` type `y` and hit enter

1.1 Credit:

things here are a mix of the really excellent Software Carpentry tutorial on Python: http://swcarpentry.github.io/python-novice-inflammation/

I've made some slight adaptations here and there, but the credit goes to those organizations. I hope I am using this correctly under the licences:

https://earth-env-data-science.github.io/LICENSE.html https://swcarpentry.github.io/python-novice-inflammation/LICENSE.html

Just as we imported the `numpy` library we'll need to import the `pyplot` module from the `matplotlib` library. We can do that using the same syntax. The `%matplotlib inline` magic command makes sure we can see these plots in our notebooks.

[1]:
```
%matplotlib inline
# import numpy and matplotlib.pyplot with nicknames:
```

```
import numpy as np
import matplotlib.pyplot as plt
```

Next, we are in a new notebook now, so we need to load in our test data again. How did we do that?

[2]:
```
# load in your data again
filename = '../data/inflammation-01.csv'
data = np.loadtxt(filename, delimiter=',')
```

1.1.1 tab helps complete files and code!!!

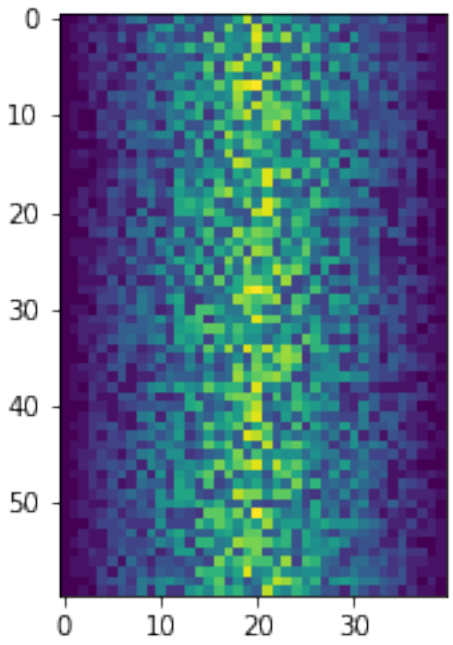
1.1.2 visulaization with matplotlib

Matplotlib can make some beautiful plots. They have great documentation where you can see and learn how to produce many plots: https://matplotlib.org/gallery/index.html

With just a single line of code we can make a nice heatmap visualization of our data using a function called `imshow`, which sounds like 'image show'

[3]:
```
# plot the data using 'imshow'

plt.imshow(data)
```

[3]:
```
<matplotlib.image.AxesImage at 0x7fecca481820>
```



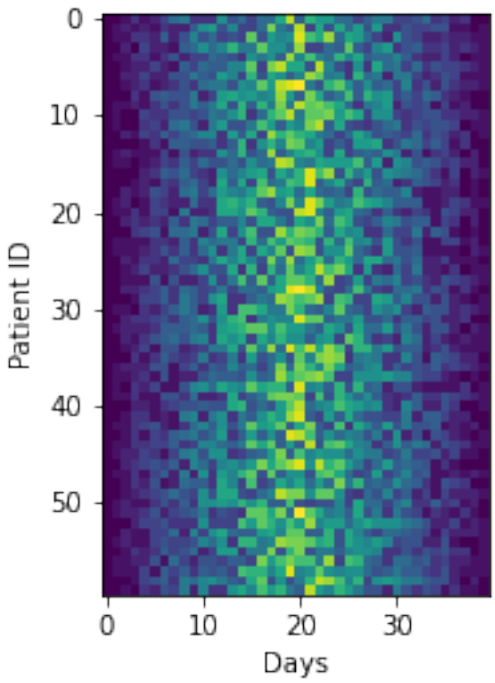
1.1.3 What are we seeing?

This is great, but we can't really use this as is, we need axis labels. What are the labels for the axis?

We can label axis with the matplotlib functions: `plt.xlabel()` and `plt.ylabel()`

[4]:
```
# plot it again, but with labels this time

plt.imshow(data)
plt.xlabel('Days')
plt.ylabel('Patient ID')
```

[4]:
```
Text(0, 0.5, 'Patient ID')
```



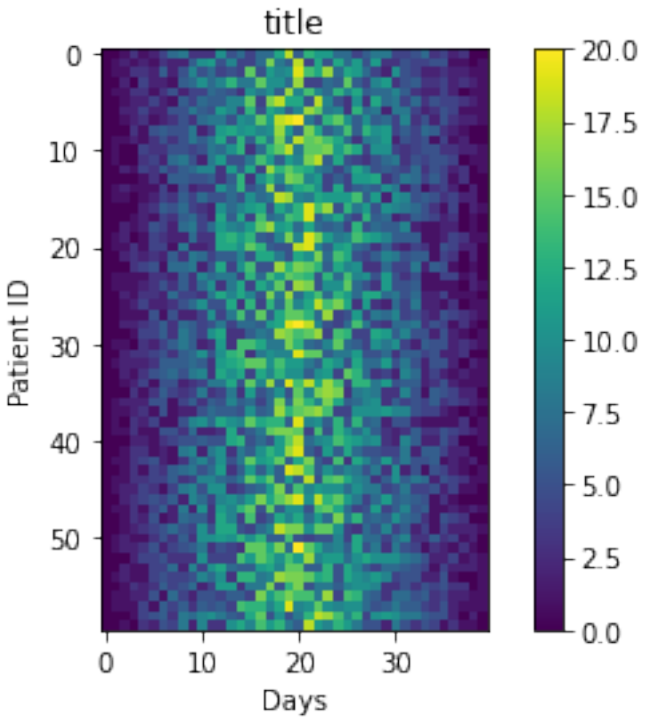
1.1.4 Much better!

Except we need something else. What do those colors mean? We need a label for the colorscale.

We can get this using the matplotlib function `plt.colorbar()`

[5]:
```
# make the plot with labels and a colorbar

plt.imshow(data)
plt.xlabel('Days')
plt.ylabel('Patient ID')
plt.colorbar()
plt.title('title')
```

[5]: `Text(0.5, 1.0, 'title')`



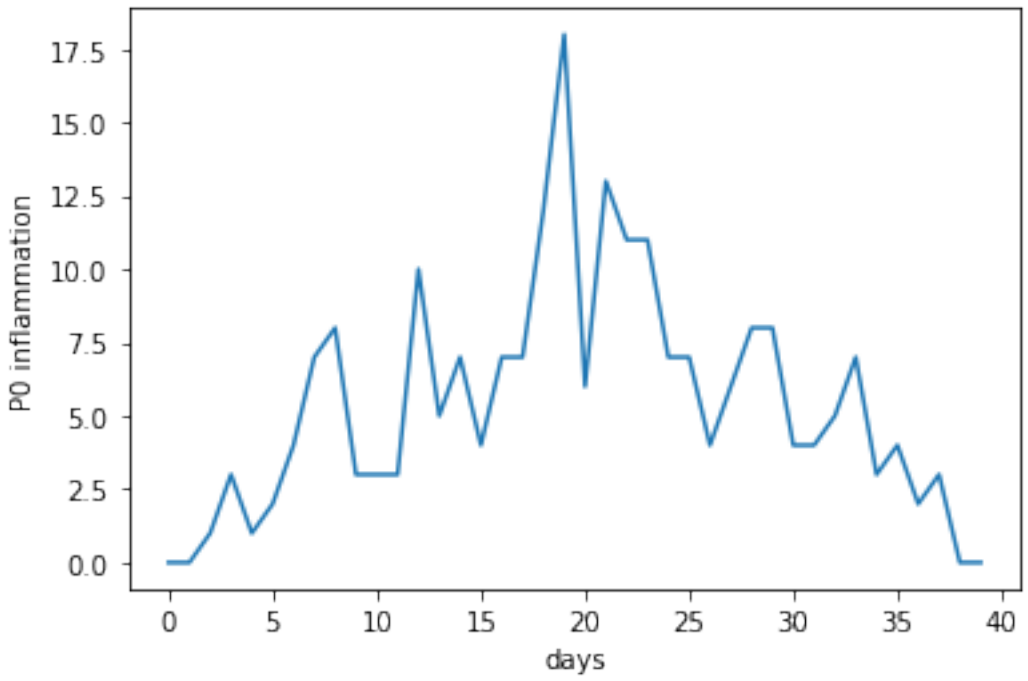
There are many, many types of plots. Above we are visualizing all our data.

What if we just want to plot the inflammation through time of patient 0?

We need to select just patient 0, then we can use another plotting function from matplotlib called `plot()`. You can read about all the options for using `plot()` by checking out its documentation: https://matplotlib.org/3.3.0/api/_as_gen/matplotlib.pyplot.plot.html

[6]:
```
# select just patient 0 data and plot it with labeled axes

patient_0 = data[0, :]

plt.plot(patient_0)
plt.ylabel('P0 inflammation')
plt.xlabel('days')
```

[6]:
```
Text(0.5, 0, 'days')
```



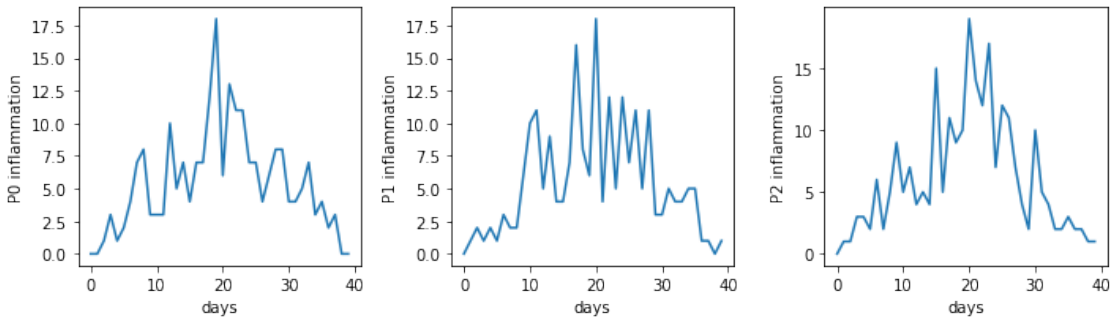
1.2 subplots: grouping plots

You can group similar plots in a single figure using subplots. Maybe we want to look in detail at the first three patient's data.

This script below uses a number of new commands. The function `plt.figure()` creates a space into which we will place all of our plots. The parameter `figsize` tells Python how big to make this space. Each subplot is placed into the figure using its `add_subplot` method. The `add_subplot` method takes 3 parameters. The first denotes how many total rows of subplots there are, the second parameter refers to the total number of subplot columns, and the final parameter denotes which subplot your variable is referencing (left-to-right, top-to-bottom). Each subplot is stored in a different variable `(axes1, axes2, axes3)`. Once a subplot is created, the axes can be labeled using the `set_xlabel()` command (or `set_ylabel()`). Here are our three plots side by side:

[7]:
```
fig = plt.figure(figsize=(10.0, 3.0))

axes1 = fig.add_subplot(1, 3, 1)
axes2 = fig.add_subplot(1, 3, 2)
axes3 = fig.add_subplot(1, 3, 3)


axes1.plot(data[0,:])
axes1.set_ylabel('P0 inflammation')
axes1.set_xlabel('days')
```

```
axes2.plot(data[1,:])
axes2.set_ylabel('P1 inflammation')
axes2.set_xlabel('days')

axes3.plot(data[2,:])
axes3.set_ylabel('P2 inflammation')
axes3.set_xlabel('days')

fig.tight_layout()
```



The end…

2 Exercise 05 / Breakout