COMP 364: Computer Tools for Life Sciences

Using libraries: NumPy & Data visualization with MatPlotLib (part two)

Christopher J.F. Cameron and Carlos G. Oliver

Key course information

Midterm

- grades will be appear in MyCourses over the weekend
- to view your marked exam
  - please see Chris or Carlos in their office hours

Office hours

- for the week of October 30th, 2017
  - Chris - Wednesday: 1:-2:30 pm
  - Carlos - Monday: 10-11:30 am

Quiz #5

- will be available October 30th, 2017
- access closes at 11:59:59 pm on the same day
- MC questions will cover topics from the last two weeks

Matplotlib recap

Matplotlib module

- covered the basics of plotting
  - **plt.plot()**
  - line vs. scatter plots
  - colouring data points
  - labeling: trends, axes, legend, etc.
  - saving/closing figures
- make sure to review slides that were not covered in class
- Matplotlib API:
  - `https://matplotlib.org/2.0.2/api/index.html`

Today, we are going to cover

- basic statistics with **NumPy**
- other plot types in Matplotlib

NumPy

Python's **NumPy** module

- a fundamental package for scientific computing, which contains:
  - a powerful N-dimensional array object
  - useful linear algebra and random number capabilities

NumPy API:
`https://docs.scipy.org/doc/numpy-1.13.0/reference/`

Since we'll be performing basic statistics

- NumPy $\rightarrow$ Routines $\rightarrow$ Statistics

Calculating means and stdev

Using the NumPy module, for each gene in our dataset

- calculate mean expression
- calculate variance of expression
- calculate **standard devitation (stdev)** of expression

Variance measures the spread of expression values
Stdev measures the dispersion of expression values

- a low stdev means values are close to the mean

Let's begin by reading in data from our dataset file

- then use the NumPy API to perform our tasks

```
import numpy as np

data_dict = {}
with open("./Spellman.csv","r") as f:
    header = f.readline()
    for line in f:
        gene_name,*exps = line.rstrip().split(",")
        exps = [float(val) for val in exps]
        # calculate statistics using NumPy
        data_dict[gene_name] = [np.mean(exps),
            np.var(exps),np.std(exps)]

gene_name = next(iter(data_dict))
print(gene_name,data_dict[gene_name])
# prints:
# YAL001C [0.006521739130434778,
# 0.07153572778827976, 0.26746163797501832]
```

Barplots

Now we have some useful gene statistics to work with

- mean gene expression
- variance of gene expression
- std of gene expression

Using the Matplotlib module

- sort genes by their mean expression
  - for simplicity, ignore gene's with a mean$\leq$0
- create a bar plot
- for each gene
  - displays the gene's mean expression as bar height
  - set the xticks to be gene names
- don't forget to label your plot

Sorting genes

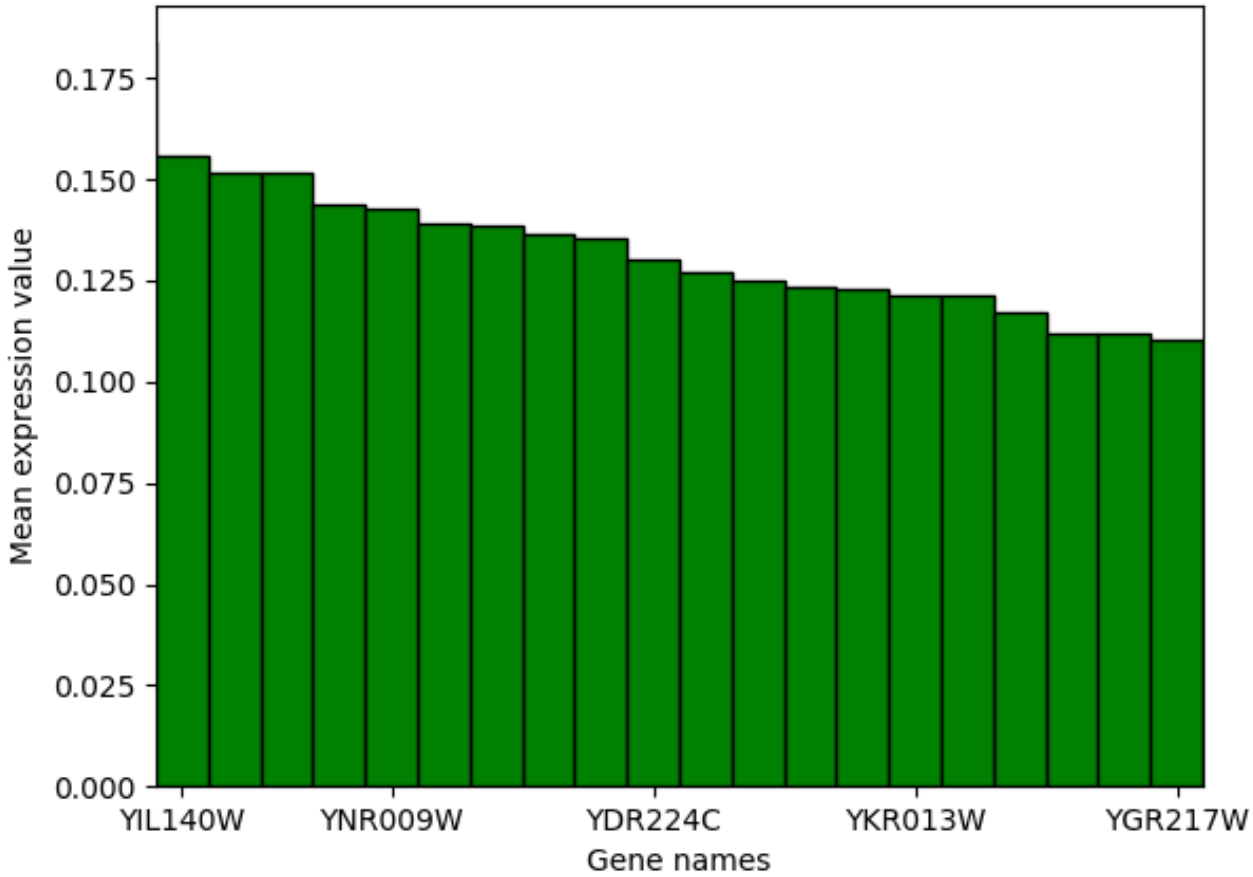
```
labels,bar_heights = [],[]

# filter for genes with expression>0.0
for key in data_dict.keys():
    mean,var,stdev = [val[0]
      for val in zip(data_dict[key])]
    if mean > 0.0:
      labels.append(key)
      bar_heights.append(mean)

# sort lists by gene means
labels,bar_heights = zip(*sorted(zip(labels,
     bar_heights),key=lambda x:x[1],
     reverse=True))
```

Matplotlib barplot

```
1  import matplotlib.pyplot as plt
2
3  N = np.asarray(range(len(labels)))+0.5
4  plt.bar(N,bar_heights,1.0,color="g",edgecolor="k")
5  plt.xlim([1.0,21.0]) #   limit plot to first 20 genes
6  plt.xlabel("Gene names")
7  # adjust xticks and labels
8  xticks,xticklabels = [],[]
9  for val in [1,5,10,15,20]:
10     xticks.append(val+0.5)
11     xticklabels.append(labels[val-1])
12 plt.xticks(xticks,xticklabels)
13 plt.ylabel("Mean expression value")
14 plt.savefig("./barplot.png")
15 plt.close()
```



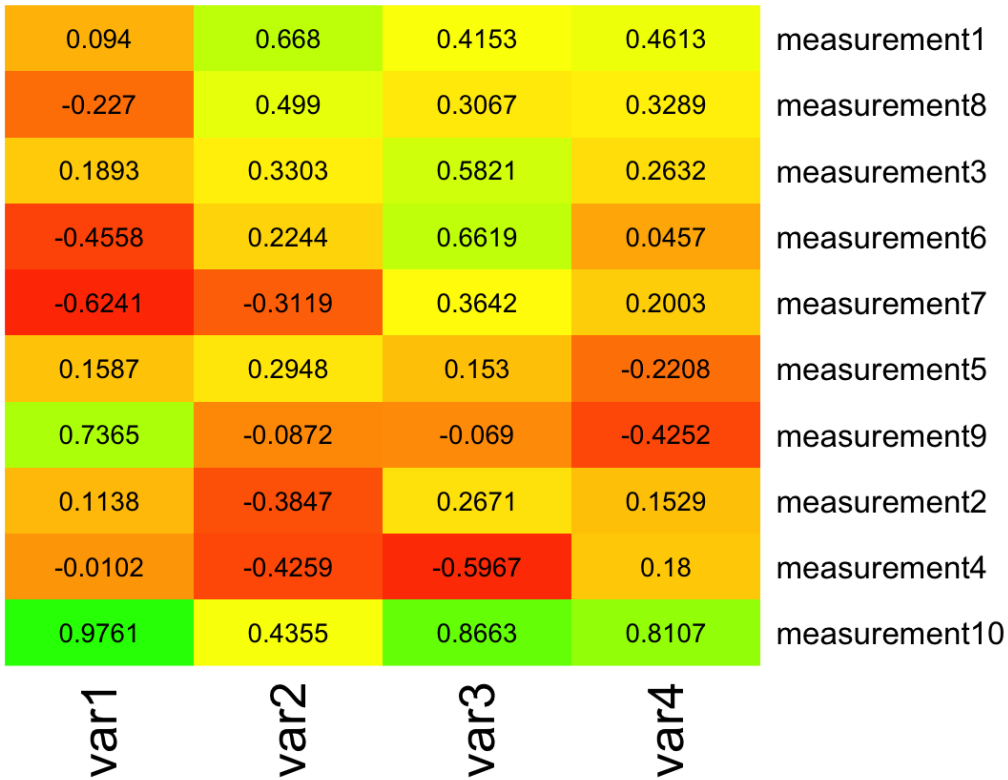
For homework

- add error bars to the plot that represent the stdev
- **hint:** review the Matploltib API for **plt.bar()**

Heatmaps

Heatmaps (or heat maps) is a graphical representation of data

- where individual values contained in a matrix are represented as colours



Heatmaps in Matplotlib

In Matplotlib, we use **plt.imshow()**

- where we an *array_like* object to be plotted

imshow API:
`https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.imshow.html`

Let's try creating a heatmap from our gene expression data

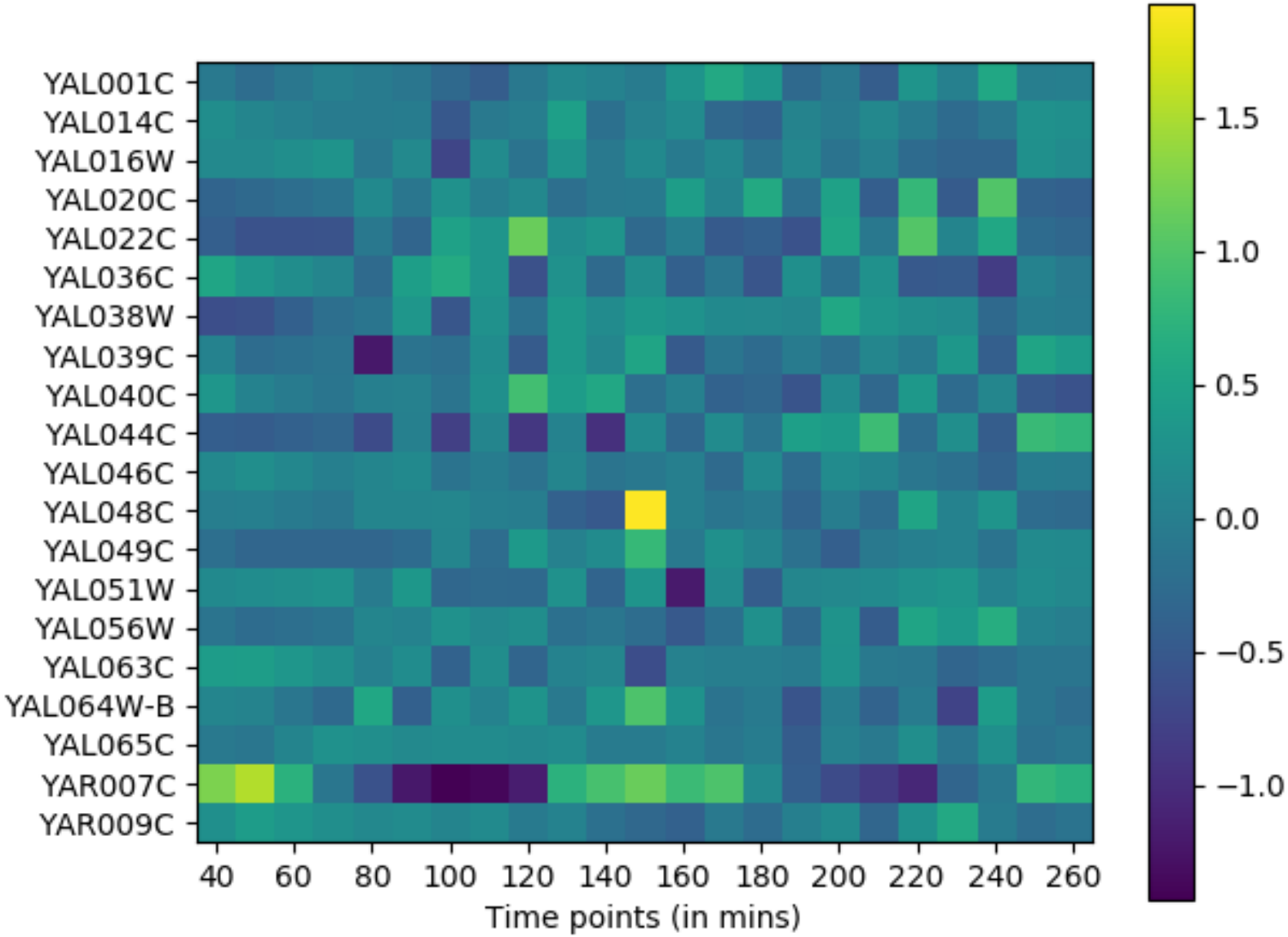
- start by reading the expression data into a matrix
- rows will be genes
- columns will be time points
- then use **plt.imshow()** to produce the heatmap figure

```python
import numpy as np

gene_names,n = [], 20
with open("./Spellman.csv","r") as f:
    header = f.readline().rstrip().split(",")
    # determine number of columns
    time_points = header[1:]
    m = len(time_points)
    matrix = np.zeros((n,m))
    for i,line in enumerate(f):
        gene_name,*exps = line.rstrip().split(",")
        gene_names.append(gene_name)
        # fill in matrix with corresponding values
        try:
            for j,val in enumerate(exps):
                matrix[i][j] = float(val)
        except: break
```

Plotting heatmap from matrix

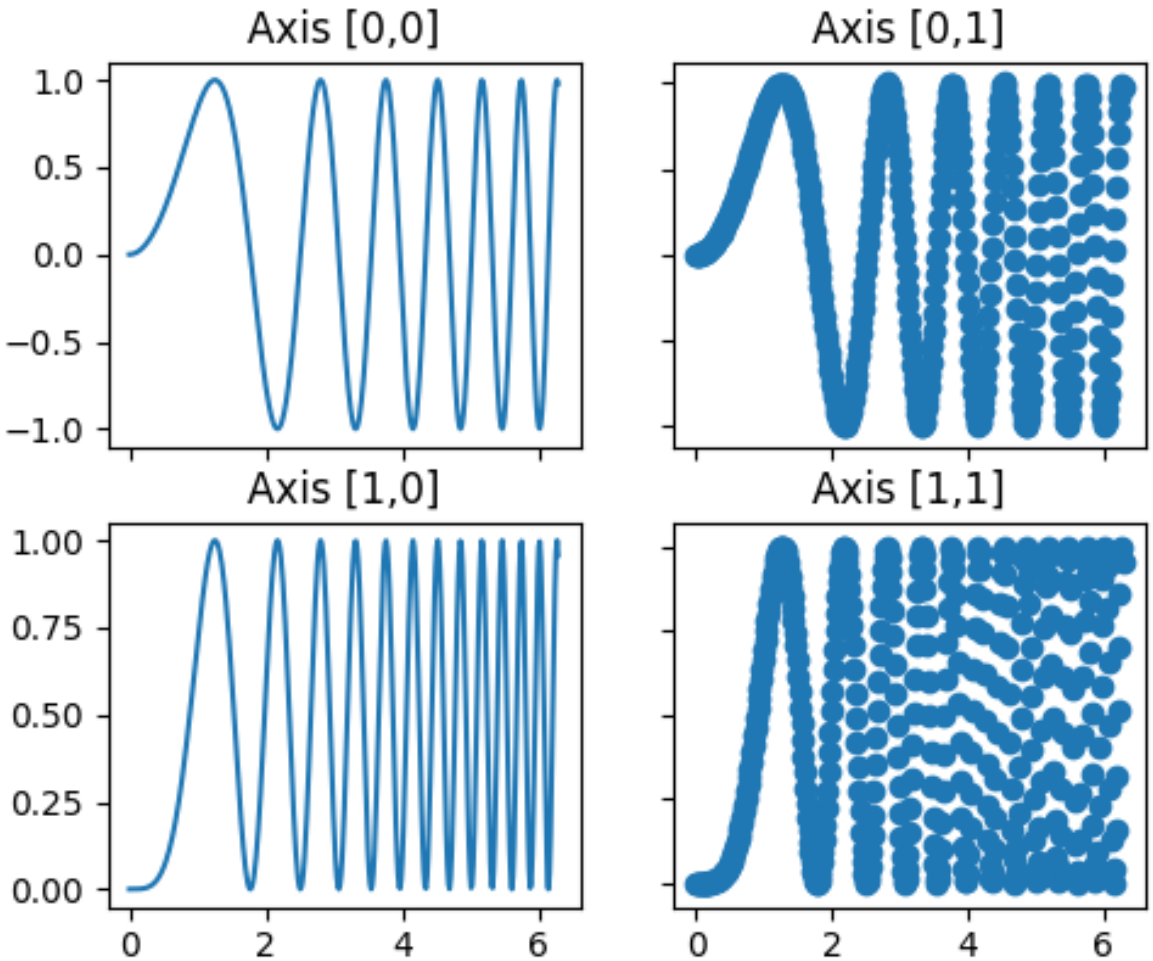
```
1  import matplotlib.pyplot as plt
2
3  plt.imshow(matrix)
4  plt.colorbar() # add colorbar
5  # adjust labels
6  plt.yticks(list(range(n)),gene_names)
7  plt.xticks(list(range(m))[::2],time_points[::2])
8  plt.xlabel("Time points (in mins)")
9  #   clean up plot canvas
10 plt.tight_layout()
11 plt.savefig("./heatmap.png")
12 plt.close()
```


YAL001C
YAL014C
YAL016W
YAL020C
YAL022C
YAL036C
YAL038W
YAL039C
YAL040C
YAL044C
YAL046C
YAL048C
YAL049C
YAL051W
YAL056W
YAL063C
YAL064W-B
YAL065C
YAR007C
YAR009C
40
60
80
100
120
140
160
180
200
220
240
260
Time points (in mins)
1.5
1.0
0.5
0.0
−0.5
−1.0

Subplots

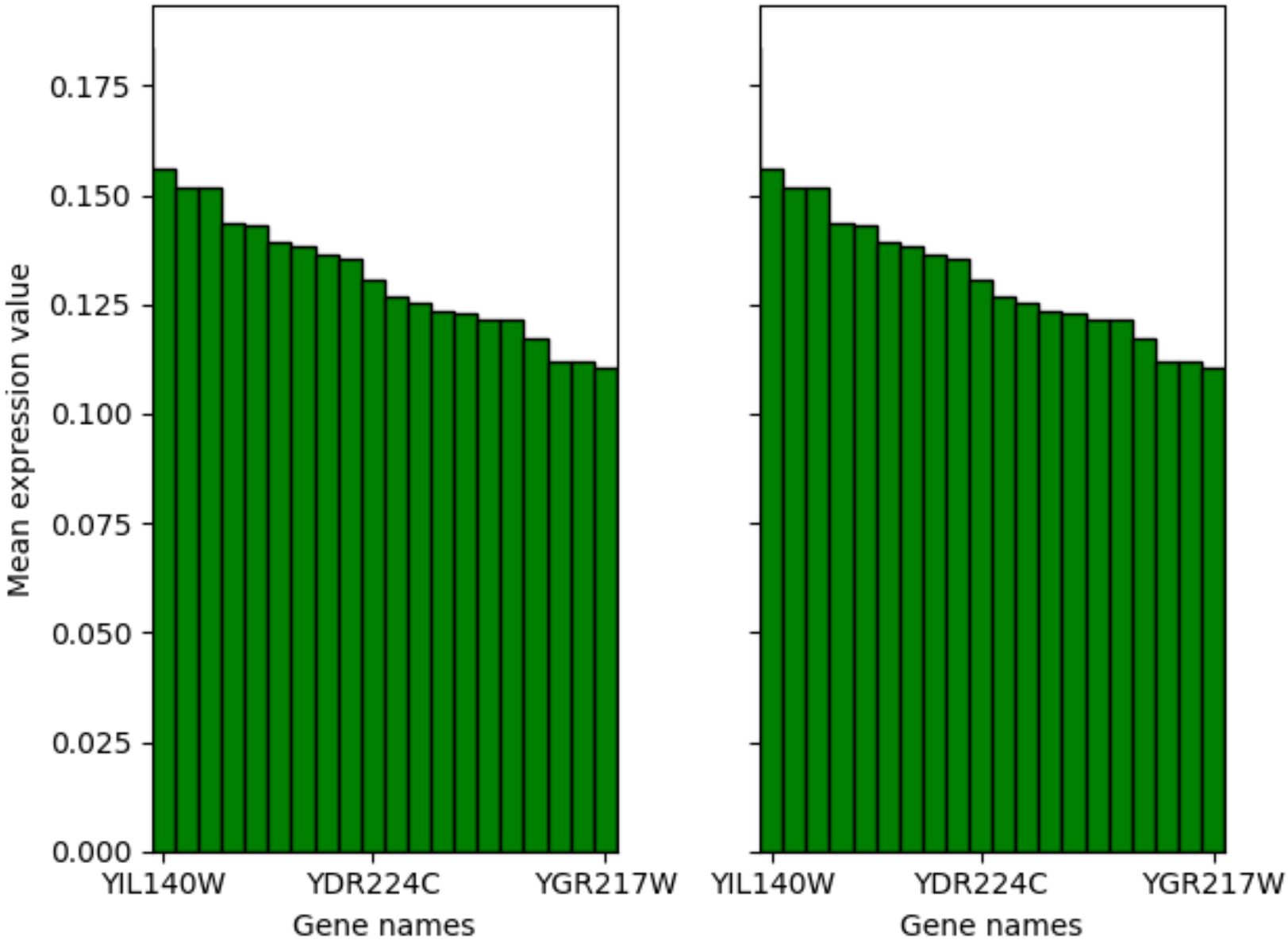
It may be useful to show multiple plots in one figure

- in Matplotlib, we can accomplish using **plt.subplots()**



Let's try plotting our barplot from before as two paired subplots

```
xticks,xticklabels = [],[]
for val in [1,10,20]:
    xticks.append(val+0.5)
    xticklabels.append(labels[val-1])
# define subplots
fig,axes = plt.subplots(1,2,sharey=True)
for i,ax in enumerate(axes):
    ax.bar(N,bar_heights,1.0,color="g",edgecolor="k")
    # notice the 'set_' prefix
    ax.set_xlim([1.0,21.0])
    ax.set_xlabel("Gene names")
    ax.set_xticks(xticks)
    ax.set_xticklabels(xticklabels)
ax[0].set_ylabel("Mean expression value")
plt.tight_layout()
plt.savefig("./barplot_2.png")
plt.close()
```


Mean expression value
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
YIL140W
YDR224C
YGR217W
Gene names
YIL140W
YDR224C
YGR217W
Gene names