Learning Objectives

Students will be able to...

- **Create an interactive plot**
- **Practice generating animated plots**
- **Save out plots**

definition

Assumptions

- Learners are comfortable reading and importing CSV data sets, extracting relevant data into data frames, and printing that data to the console.
- Learners are comfortable using ggplot2 to create visualization charts.

Interacting Plots

When working with jupyter notebook, it is pretty easy to make the plots interactive. We will use the following command above our code.

```
%matplotlib notebook
```

This command will generate a more interactive figure. Where one can hover the figure to see the points.

```
import matplotlib.pyplot as plt
import random

%matplotlib notebook

x= random.sample(range(0, 30), 10)
y= random.sample(range(0, 50), 10)

fig=plt.figure(figsize=(5,5))

plt.scatter(x,y)

plt.title('Sample plot ')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt .show()
```

We are able to turn off the interactions mode by simply clicking on the power button.

From that toolbar we we are able to:
* Zoom
* Pan
* Save

Our axes will automatically be selected in a best fit matter. However, in this mode we can change them ourselves on cursor movement after we click on the pan icon.

Saving our figures

Assuming you are not using an interactive toolbar like the one above. There is another command that can be used to save our figures.

The basic syntax :

```
plt.savefig("filename")
```

When using the `savefig()` function , we need to make sure to call it before the `show()` function which creates a new fig after the function is called.

Animating plots

Now that we know how to create an interactive plot in matplolib, we can tackle how to run simulation where we are seeing our graph changing.

The first step will be importing our libraries

```
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from IPython import display
```

The second step to do this is enabling interactive plots.

```
%matplotlib notebook
```

The third step is getting our data.

```
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import random

%matplotlib notebook
#data
x= np.linspace(0,2*np.pi,100)
fig = plt.figure()

lines = plt.plot([])
line= lines[0]
```

FuncAnimation function we are going to call need to take a couple arguments. The first argument is the figure we are trying to animate , which we conveniently named `fig`. another function which is being called for every frame. That function will help plot our data. For our third argument we are going to set the frame limit or how long the animation is going to be. The 4th argument is an interval argument in milliseconds, which is the time between frames.

The animate function, the going to reflect the changes in our axis frame by frame.

```
def animate(frame):
    # updating the frame
    y=np.sin(x+ 2*np.pi * frame/100)
    line.set_data(x,y)
    return line

anim = FuncAnimation(fig, animate,frames=200 , interval=50)
```

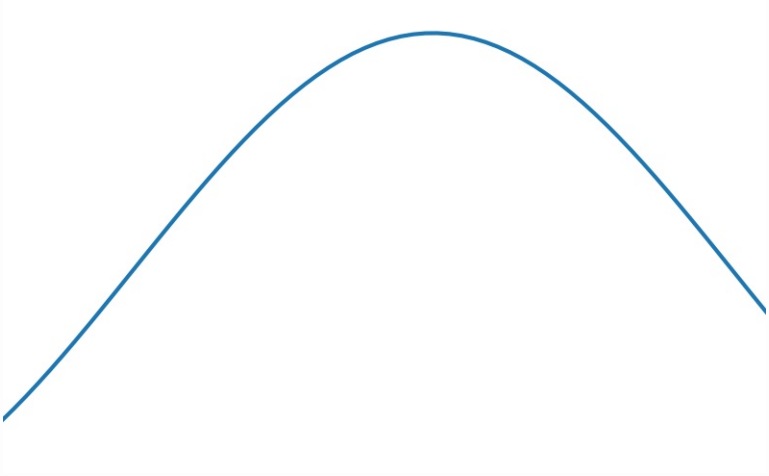
Video

The last part is going to be converting the animation . Odds are when creating a new animation the code below will always be the same.

```
video = anim.to_html5_video()
html = display.HTML(video)
display.display(html)
plt.close()
```

Initially, we can see some movements, this is where we are going to set new axis limit in order to better see our graphs. For starters, try setting x to a range of 0 to 1 and y from -1 to 1. From there try to manipulate your x and y limits till you can see more of the wave.

```
# x and y limits
plt.xlim([0, 1])
plt.ylim([-1, 1])
```



images/wave