assignment1

April 13, 2023

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
```

1 1a

[ ]:
```python
a = np.array([1,1,1])
b = np.array([2,2,2])
c = np.array([3,3,3])
def three_array(a,b,c):
    return a+b-c

print(three_array(a,b,c))
```

```
[0 0 0]
```

2 1b

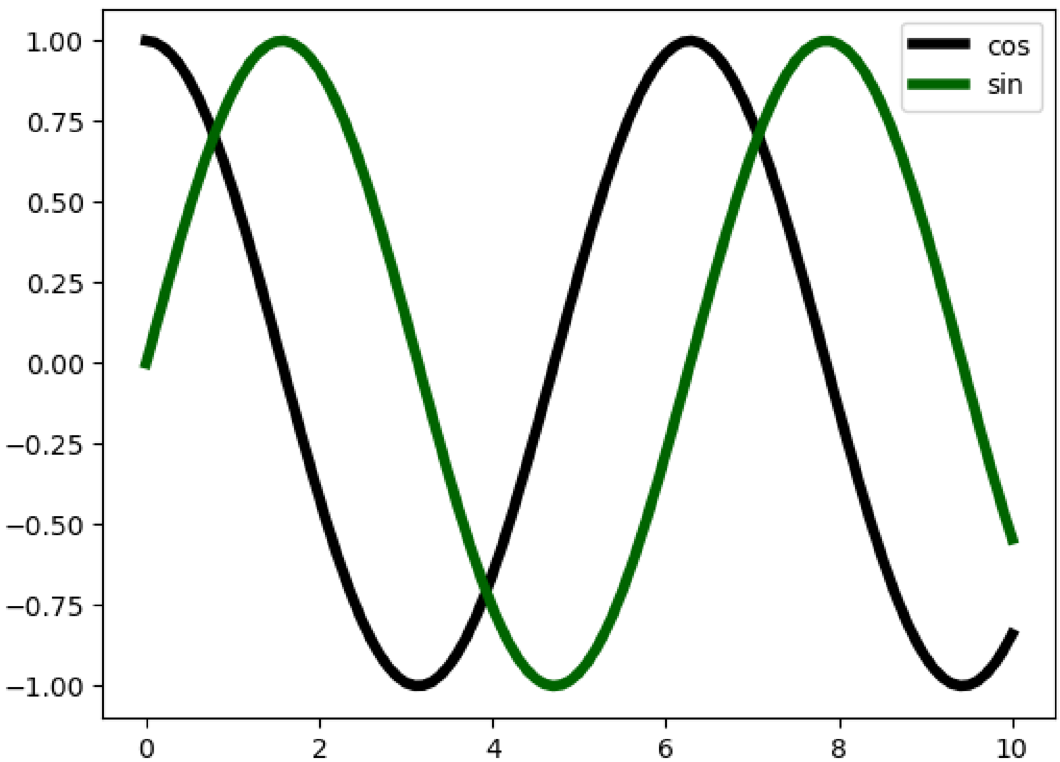
[ ]:
```python
def add_array(a,b):
    # check if dimension is correct
    if a.shape != b.shape:
        return "Dimension Error"
    else:
        return a+b

array_1 = np.array([3,5,7])
array_2 = np.array([1,1,1])
array_3 = np.array([2,2,2,2])

# testing cases
print(add_array(array_1,array_2))
print(add_array(array_1,array_3))
```

```
[4 6 8]
Dimension Error
```

3 1c

[ ]:
```
#fig, ax = plt.subplots(figsize=(10,8), nrows=1, ncols = 2)
#time = np.linspace(0,10,101)
#cos_y = np.cos(time)
#sin_y = np.sin(time)
#ax[0,0].plot(time, sin_y)
#ax[1,1].plot(time, cos_y)
```

[ ]:
```
time = np.linspace(0,10,101)
# create sin and cos data
cos_y = np.cos(time)
sin_y = np.sin(time)
# simple plotting
plt.plot(time, cos_y, color ="black", lw=4, label = "cos")
plt.plot(time,sin_y, color = "darkgreen", lw=4, label="sin")
plt.legend()
plt.show()
```



4 1d

```
[ ]: A = np.array([[1,0,0],[0,1,0],[0,0,1]])
def inver(A):
    # find the inverse
    try:
        A_inv = np.linalg.inv(A)
        return A_inv
    # catch non invertible exception
    except:
        det = np.linalg.det(A)
        print("Error: this matrix is not invertible with a determinant of " +␣
 ↪str(det))

output = inver(A)
print(np.dot(A, output))
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

5 2a

```
[ ]: def generate(a,b,N):
    sample_rand =  np.random.rand(1,N) * (b-a) + a
    return sample_rand

a = 500
b = 850
N = 100
sample = generate(a,b,N)
print("Size is: " + str(sample.size))
print(sample)
```

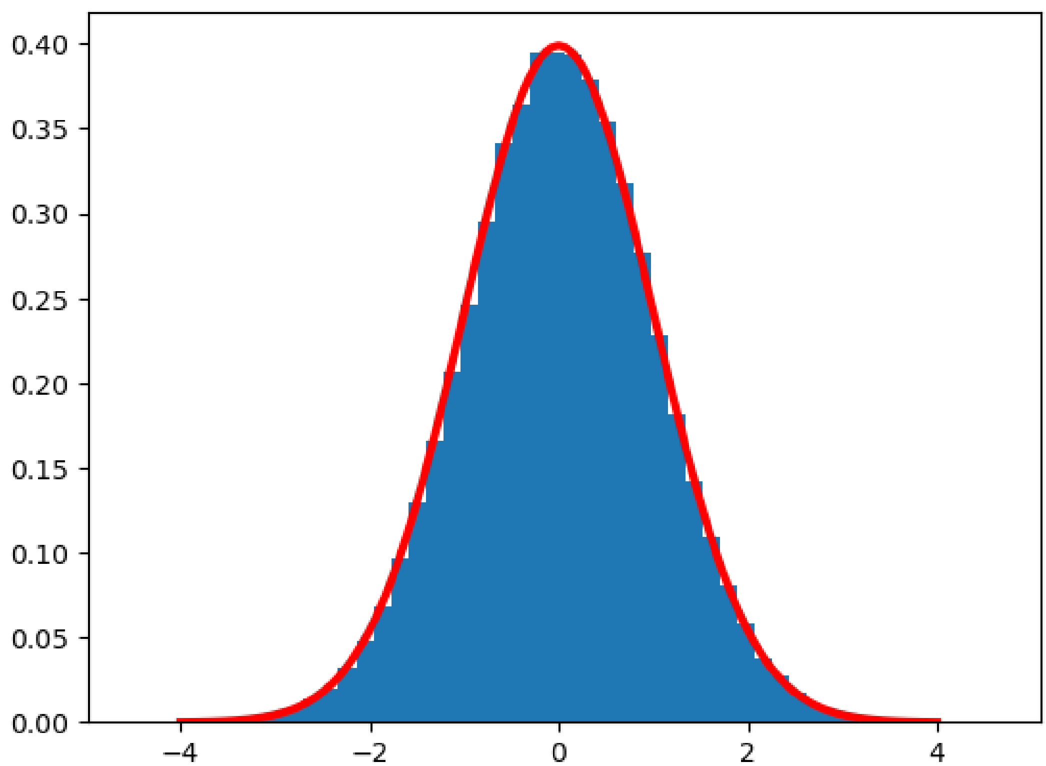
```
Size is: 100
[[773.43196361 687.47400595 555.92461221 674.02068167 627.62066354
  686.27340131 581.87542623 747.25071141 555.07366406 818.87157459
  703.43237045 561.86837427 801.89023963 604.59324485 561.05435885
  780.28146123 527.85722041 610.62869364 626.76779301 711.6528366
  736.20483478 702.86630345 559.76484144 541.07659686 573.01475157
  769.84027325 664.43723744 715.56189808 525.63072709 773.15828835
  690.32243719 634.39627923 543.02012814 703.40469666 629.25074249
  529.64570636 681.85328691 787.6184247  846.54257088 700.62226792
  752.55470016 670.23655402 717.16318478 575.50264999 643.62251562
  543.08409946 619.98266521 546.12930749 588.20789666 832.68927554
  607.06019331 627.72297966 821.16886185 700.92107119 834.70545176
  726.96713266 580.94605267 735.05232831 568.98583277 528.73246651
```

```
721.54865043 762.59727181 794.46724819 552.9712323  724.03743424
809.11524556 550.26727302 539.84974394 574.82978556 638.05620217
777.02264947 799.5326569  638.41987715 543.01601627 677.51885704
758.12303581 579.34448475 525.67760865 641.44635989 572.82226545
768.40477195 595.72767566 808.39506383 534.0287575  820.02863897
716.07789017 835.88682408 837.45426685 613.93355761 809.96715135
714.97751855 649.98782055 558.64679665 529.04764114 626.70766584
542.6862376  791.69144538 586.11324488 722.11747716 781.38929568]]
```

6 2b

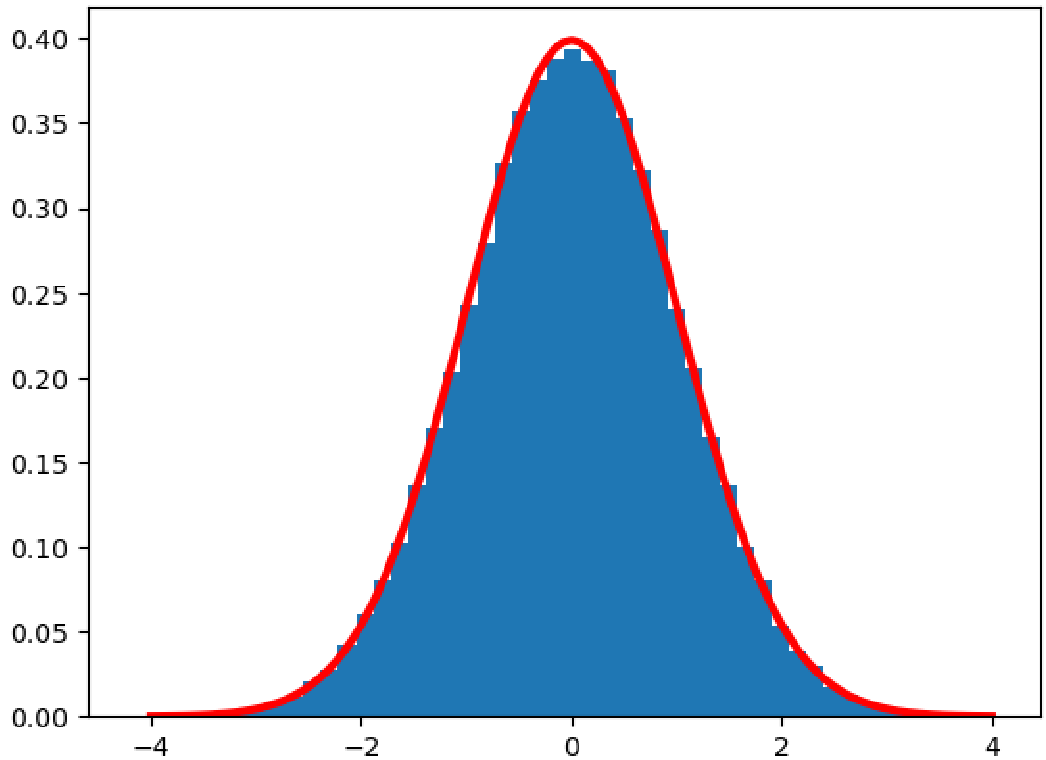
There are two ways to draw the normal distribution curve. The first way is to use it from the definition.

[ ]:
```
data = np.random.randn(100000,1)
x = np.linspace(-4,4,101)
gaus = 1/(np.sqrt(2*np.pi))*np.e**(-1/2*(x)**2) # directly from gaussian␣
↪formulation
plt.hist(data,bins=50, density=True) # histogram
plt.plot(x, gaus, lw = 3, color = "red") # plotting actual function
plt.show()
```



The second method is to use the scipy method:

```
[ ]: data = np.random.randn(100000,1)
x = np.linspace(-4,4,101)
plt.hist(data,bins=50, density=True)
plt.plot(x, norm.pdf(x, 0, 1), lw = 3, color = "red") # plot x versus the normal distribution sample from scipy
plt.show()
```



7 2c

To calculate the expectation value, we evaluate the following integral

$$\langle x \rangle = \int_0^\infty x a e^{-ax} dx = \frac{1}{a}$$

Then, we want to calculate the variance of the pdf. The variance is given by the following formula:

$$var = \langle x^2 \rangle - \langle x \rangle^2$$

This expectation of mean squared is given by the following integral:

$$\langle x^2 \rangle = \int_0^\infty x^2 a e^{-ax} dx = \frac{2}{a^2}$$

Combining the result:

$$var = \langle x^2 \rangle - \langle x \rangle^2 = \frac{2}{a^2} - (\frac{1}{a})^2 = \frac{1}{a^2}$$

8 2d

i): we can only get 12 if both of the dice are 6. Therefore, the probability is given by

$$\frac{1}{6} * \frac{1}{6} = \frac{1}{36}$$

ii): the sum of 10 can be achieved in three ways:

$$5 + 5 = 106 + 4 = 104 + 6 = 10$$

each configuration has a probability of 1 over 36, so the final probability is the sum:

$$\frac{1}{36} + \frac{1}{36} + \frac{1}{36} = \frac{1}{12}$$

iii): the sum of 6 can be achieved in five ways:

$$1 + 5 = 62 + 4 = 63 + 3 = 64 + 2 = 65 + 1 = 6$$

each configuration has a probability of 1 over 36, so the final probability is:

$$\frac{1}{36} * 5 = \frac{5}{36}$$