Assignment 1 - Probability, Linear Algebra, Programming, and Git

Vishaal Venkatesh

Netid: vv58

Instructions for all assignments can be found here (https://github.com/kylebradbury/ids705/blob/master/assignments/_Assignment%20Instructions.ipynl which is also linked to from the course syllabus (https://kylebradbury.github.io/ids705/index.html).

Probability and Statistics Theory

Note: for all assignments, write out all equations and math using markdown and LaTeX (https://tobi.oetiker.ch/lshort/lshort.pdf). For this assignment show ALL math work

1

[3 points]

Let $f(x) = \begin{cases} 0 & x < 0 \\ \alpha x^2 & 0 \leq x \leq 2 \\ 0 & 2 < x \end{cases}$

For what value of α is $f(x)$ a valid probability density function?

ANSWER

A valid PDF sums up to one when integrated over the domain of real valued numbers:

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

$$\int_{-\infty}^{0} f(x)dx + \int_{0}^{2} f(x)dx + \int_{2}^{\infty} f(x)dx = 1$$

$$\int_{-\infty}^{0} 0dx + \int_{0}^{2} \alpha x^2 dx + \int_{2}^{\infty} 0dx = 1$$

$$\left[\frac{\alpha x^3}{3}\right]_0^2 = 1$$

$$\frac{8\alpha}{3} = 1$$

$$\alpha = \frac{3}{8}$$

2

[3 points] What is the cumulative distribution function (CDF) that corresponds to the following probability distribution function? Please state the value of the CDF for all possible values of x.

$$f(x) = \begin{cases} \frac{1}{3} & 0 < x < 3 \\ 0 & \text{otherwise} \end{cases}$$

ANSWER

The CDF for the range $-\infty < x < 0$ is the following

$$F(x) = \int_{-\infty}^{x} f(y)dy$$
$$F(x) = \int_{-\infty}^{x} 0dy$$
$$F(x) = 0$$

The CDF for the range $0 < x < 3$ is the following

$$F(x) = \int_{-\infty}^{x} f(y)dy$$
$$F(x) = \int_{-\infty}^{0} 0dy + \int_{0}^{x} \frac{1}{3}dy$$
$$F(x) = 0 + \int_{0}^{x} \frac{1}{3}dy$$
$$F(x) = \left[\frac{x}{3}\right]_0^x$$
$$F(x) = \frac{1}{3}x$$

The CDF for the range $3 < x < \infty$ is the following

$$F(x) = \int_{-\infty}^{x} f(y)dy$$
$$F(x) = \int_{-\infty}^{0} 0dy + \int_{0}^{3} \frac{1}{3}dy + \int_{3}^{\infty} 0dy$$
$$F(x) = 0 + \int_{0}^{3} \frac{1}{3}dy + \int_{3}^{\infty} 0dy$$
$$F(x) = 1$$

CDF:

$$F(x) = \begin{cases} 0 & -\infty < x < 0 \\ \frac{x}{3} & 0 < x < 3 \\ 1 & 3 < x < \infty \end{cases}$$

3

[6 points] For the probability distribution function for the random variable X,

$$f(x) = \begin{cases} \frac{1}{3} & 0 < x < 3 \\ 0 & \text{otherwise} \end{cases}$$

what is the (a) expected value and (b) variance of X. *Show all work*.

ANSWER

(a) Expected Value

$$E[X] = \int_{-\infty}^{\infty} xf(x)dx$$
$$E[X] = \int_{-\infty}^{0} xf(x)dx + \int_{0}^{3} xf(x)dx + \int_{3}^{\infty} xf(x)dx$$
$$E[X] = \int_{-\infty}^{0} x0dx + \int_{0}^{3} xf(x)dx + \int_{3}^{\infty} x0dx$$
$$E[X] = 0 + \int_{0}^{3} xf(x)dx + 0$$
$$E[X] = \int_{0}^{3} \frac{x}{3}dx$$
$$E[X] = \left[\frac{x^2}{6}\right]_0^3$$
$$E[X] = \frac{3}{2}$$

(b) Variance

$$V[X] = E[X^2] - E[X]^2$$

$$V[X] = \int_{0}^{3} x^2 f(x)dx - E[X]^2$$

$$V[X] = \int_{0}^{3} x^2 \frac{1}{3}dx - E[X]^2$$

$$V[X] = \left[\frac{x^3}{9}\right]_0^3 - E[X]^2$$

$$V[X] = \frac{27}{9} - \frac{9}{4}$$
$$V[X] = \frac{3}{4}$$

4

[6 points] Consider the following table of data that provides the values of a discrete data vector $\mathbf{x}$ of samples from the random variable X, where each entry in $\mathbf{x}$ is given as x_i.

Table 1. Dataset N=5 observations

	x_0	x_1	x_2	x_3	x_4
$\mathbf{x}$	2	3	10	-1	-1

What is the (a) mean, (b) variance, and the of the data?

Show all work. Your answer should include the definition of mean, median, and variance in the context of discrete data.

ANSWER

(A) Mean

$$Mean = \frac{1}{n}\sum_{i=0}^{4} x_i$$
$$Mean = \frac{1}{5}\sum 2 + 3 + 10 - 1 - 1$$
$$Mean = \frac{13}{5} = 2.6$$

(B) Variance

$$SampleVariance = s^2 = \sum_{i=0}^{4} \frac{(x_i - E[X])^2}{n-1}$$
$$s^2 = \sum \frac{(2-2.6)^2}{5-1} + \frac{(3-2.6)^2}{5-1} + \frac{(10-2.6)^2}{5-1} + \frac{(-1-2.6)^2}{5-1} + \frac{(-1-2.6)^2}{5-1}$$
$$s^2 = 20.3$$

(C) Median

For the median we simply arrange the data in ascending order and pick the value in the middle.

$$Median = 2$$

5

[8 points] Review of counting from probability theory.

(a) How many different 7-place license plates are possible if the first 3 places only contain letters and the last 4 only contain numbers?

(b) How many different batting orders are possible for a baseball team with 9 players?

(c) How many batting orders of 5 players are possible for a team with 9 players total?

(d) Let's assume this class has 26 students and we want to form project teams. How many unique teams of 3 are possible?

Hint: For each problem, determine if order matters, and if it should be calculated with or without replacement.

ANSWER

(a) There are 26 alphabets and 10 digits $([0, 9])$. Furthermore, there is no restriction on repetitions in license plates. Therefore the total number of ways to form the license plate is

$$26 * 26 * 26 * 10 * 10 * 10 * 10 = 175760000$$

(b) Repetitions are not possible now. We simply have to do $9!$ to get the different number of batting orders.

$$9! = 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 362880$$

(c) For the first player we have 9 options. For the second player we ahve 8. So on and so forth.... In order to select 5 players we do the following

$$9 * 8 * 7 * 6 * 5 = 15120$$

(d) In this case we can do 26 choose 3. This will allow us to restrict our selection to only unique teams.

$$\binom{26}{3} = \frac{26!}{3!(26-3)!} = 2600$$

Linear Algebra

6

[7 points] Matrix manipulations and multiplication. Machine learning involves working with many matrices, so this exercise will provide you with the opportunity to practice those skills.

Let $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 4 \\ -3 \\ 6 \end{bmatrix}$, and $\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Compute the following or indicate that it cannot be computed:

1. $\mathbf{AA}$
2. $\mathbf{AA}^T$
3. $\mathbf{Ab}$
4. $\mathbf{Ab}^T$
5. $\mathbf{bA}$
6. $\mathbf{b}^T\mathbf{A}$
7. $\mathbf{bb}$
8. $\mathbf{b}^T\mathbf{b}$
9. $\mathbf{bb}^T$

10. $\mathbf{b} + \mathbf{c}^T$
11. $\mathbf{b}^T\mathbf{b}^T$
12. $\mathbf{A}^{-1}\mathbf{b}$
13. $\mathbf{A} \circ \mathbf{A}$
14. $\mathbf{b} \circ \mathbf{c}$

Note: The element-wise (or Hadamard) product is the product of each element in one matrix with the corresponding element in another matrix, and is represented by the symbol "∘".

ANSWER

1. $\mathbf{AA}$

$$\mathbf{AA} = \begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}$$

2. $\mathbf{AA}^T$

$$\mathbf{AA}^T = \begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}$$

3. $\mathbf{Ab}$

$$\mathbf{Ab} = \begin{bmatrix} 29 \\ 50 \\ 60 \end{bmatrix}$$

4. $\mathbf{Ab}^T$ This product cannot be computed as the dimensions do not work out.
5. $\mathbf{bA}$ This product cannot be computed as the dimensions do not work out.
6. $\mathbf{b}^T\mathbf{A}$

$$\mathbf{b}^T\mathbf{A} = \begin{bmatrix} 29 & 50 & 60 \end{bmatrix}$$

7. $\mathbf{bb}$ This product cannot be computed as the dimensions do not work out.
8. $\mathbf{b}^T\mathbf{b}$

$$\mathbf{b}^T\mathbf{b} = \begin{bmatrix} 74 \end{bmatrix}$$

9. $\mathbf{bb}^T$

$$\mathbf{bb}^T = \begin{bmatrix} 1 & -3 & -8 \\ -3 & 9 & 24 \\ -8 & 24 & 64 \end{bmatrix}$$

10. $\mathbf{b} + \mathbf{c}^T$ This sum cannot be computed as the dimensions do not work out.
11. $\mathbf{b}^T\mathbf{b}^T$ This product cannot be computed as the dimensions do not work out.
12. $\mathbf{A}^{-1}\mathbf{b}$ First, we compute the inverse of the matrix A.

$$\mathbf{A}^{-1} = \begin{bmatrix} 1 & -3 & 2 \\ -3 & 3 & -1 \\ 2 & -1 & 0 \end{bmatrix}$$

$$\mathbf{A}^{-1}\mathbf{b} = \begin{bmatrix} 6 \\ 4 \\ -5 \end{bmatrix}$$

13.

$$\mathbf{A} \circ \mathbf{A} = \begin{bmatrix} 1 & 4 & 9 \\ 4 & 16 & 25 \\ 9 & 25 & 36 \end{bmatrix}$$

14.

$$\mathbf{b} \circ \mathbf{c} = \begin{bmatrix} -4 \\ -9 \\ 48 \end{bmatrix}$$

7

[8 points] Eigenvectors and eigenvalues. Eigenvectors and eigenvalues are useful for some machine learning algorithms, but the concepts take time to solidly grasp. For an intuitive review of these concepts, explore this interactive website at Setosa.io (http://setosa.io/ev/eigenvectors-and-eigenvalues/). Also, the series of linear algebra videos by Grant Sanderson of 3Brown1Blue are excellent and can be viewed on youtube here (https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab).

1. Calculate the eigenvalues and corresponding eigenvectors of matrix $\mathbf{A}$ above, from the last question.
2. Choose one of the eigenvector/eigenvalue pairs, $\mathbf{v}$ and λ, and show that $\mathbf{Av} = \lambda\mathbf{v}$. Also show that this relationship extends to higher orders: $\mathbf{AAv} = \lambda^2\mathbf{v}$
3. Show that the eigenvectors are orthogonal to one another (e.g. their inner product is zero). This is true for real, symmetric matrices.

ANSWER

1. To get the eigen values and eigen vectors we simply solve for λ in the equation $(A - \lambda I)x = 0$, where x refers to the eigen vectors and λ are the eigen values. To do this we first solve for λ in the following determinant.

$$\begin{bmatrix} 1-\lambda & 4 & 9 \\ 4 & 16-\lambda & 25 \\ 9 & 25 & 36-\lambda \end{bmatrix} = 0$$

Doing so, we get the values of $\lambda = 11.35, -0.52, 0.17$.

We then use the values of the above lambdas to solve for the three possible x's (eigen vectors).

Doing so, we get the following eigen vectors - $\begin{bmatrix} 0.45 \\ 0.80 \\ 1.00 \end{bmatrix}, \begin{bmatrix} -1.25 \\ -0.55 \\ 1.00 \end{bmatrix}, \begin{bmatrix} 1.80 \\ -2.25 \\ 1.00 \end{bmatrix}$

2. We shall choose the eigen value $\lambda = 0.17$ and the eigenvector $\begin{bmatrix} 1.80 \\ -2.25 \\ 1.00 \end{bmatrix}$ to prove the relation:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

$$\mathbf{A}\mathbf{v} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} 1.80 \\ -2.25 \\ 1.00 \end{bmatrix} = \begin{bmatrix} 0.3 \\ -0.4 \\ 0.15 \end{bmatrix}$$

Similarly, we do the product $\lambda\mathbf{v}$

$$\lambda\mathbf{v} = 0.17 * \begin{bmatrix} 1.80 \\ -2.25 \\ 1.00 \end{bmatrix} = \begin{bmatrix} 0.3 \\ -0.4 \\ 0.15 \end{bmatrix}$$

Clearly the left-hand side of the equation equals the right-hand side. We continue to prove the relation $\mathbf{A}\mathbf{A}\mathbf{v} = \lambda^2\mathbf{v}$

$$\mathbf{A}\mathbf{A}\mathbf{v} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} 1.80 \\ -2.25 \\ 1.00 \end{bmatrix} = \begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}\begin{bmatrix} 1.80 \\ -2.25 \\ 1.00 \end{bmatrix} = \begin{bmatrix} -0.05 \\ -0.25 \\ -0.2 \end{bmatrix}$$

$$\lambda^2\mathbf{v} = 0.03\begin{bmatrix} 1.80 \\ -2.25 \\ 1.00 \end{bmatrix} = \begin{bmatrix} -0.05 \\ -0.25 \\ -0.2 \end{bmatrix}$$

We can also prove the above relationship in the following manner:

$$\mathbf{A}\mathbf{A}\mathbf{v} = \mathbf{A}(\lambda\mathbf{v}) = \lambda\mathbf{A}\mathbf{v} = \lambda(\lambda\mathbf{v}) = \lambda^2\mathbf{v}$$

3. Finally, we wish to prove the three eigen vectors are orthogonal. To do so we may simply take the dot product and expect it to be zero.

We show one such case as follows:

$$\begin{bmatrix} 0.45 \\ 0.80 \\ 1.00 \end{bmatrix}\begin{bmatrix} -1.25 \\ -0.55 \\ 1.00 \end{bmatrix} = -0.5625 - 0.44 + 1 = 0$$

.

This relationship can similarly be proven for all orther combinations of eigen vectors.

Numerical Programming

8

[10 points] Loading data and gathering insights from a real dataset

Data. The data for this problem can be found in the `data` subfolder in the `assignments` folder on [github (https://github.com/kylebradbury/ids705)](https://github.com/kylebradbury/ids705). The filename is `egrid2016.xlsx` . This dataset is the Environmental Protection Agency's (EPA) [Emissions & Generation Resource Integrated Database (eGRID) (https://www.epa.gov/energy/emissions-generation-resource-integrated-database-egrid)](https://www.epa.gov/energy/emissions-generation-resource-integrated-database-egrid) containing information about all power plants in the United States, the amount of generation they produce, what fuel they use, the location of the plant, and many more quantities. We'll be using a subset of those data.

The fields we'll be using include:

field	description
SEQPLT16	eGRID2016 Plant file sequence number (the index)
PSTATABB	Plant state abbreviation
PNAME	Plant name
LAT	Plant latitude
LON	Plant longitude
PLPRMFL	Plant primary fuel
CAPFAC	Plant capacity factor
NAMEPCAP	Plant nameplate capacity (Megawatts MW)
PLNGENAN	Plant annual net generation (Megawatt-hours MWh)
PLCO2EQA	Plant annual CO2 equivalent emissions (tons)

For more details on the data, you can refer to the [eGrid technical documents (https://www.epa.gov/sites/production/files/2018-02/documents/egrid2016_technicalsupportdocument_0.pdf)](https://www.epa.gov/sites/production/files/2018-02/documents/egrid2016_technicalsupportdocument_0.pdf). For example, you may want to review page 45 and the section "Plant Primary Fuel (PLPRMFL)", which gives the full names of the fuel types including WND for wind, NG for natural gas, BIT for Bituminous coal, etc.

There also are a couple of "gotchas" to watch out for with this dataset:

- The headers are on the second row and you'll want to ignore the first row (they're more detailed descriptions of the headers).
- NaN values represent blanks in the data. These will appear regularly in real-world data, so getting experience working with it will be important.

Your objective. For this dataset, your goal is answer the following questions about electricity generation in the United States:

(a) Which plant has generated the most energy (measured in MWh)?

(b) What is the name of the northern-most power plant in the United States?

(c) What is the state where the northern-most power plant in the United States is located?

(d) Plot a bar plot showing the amount of energy produced by each fuel for the plant.

(e) From the plot in (d), which fuel for generation produces the most energy (MWh) in the United States?

ANSWER

In [176]:

```
#Read the excel file in
import pandas as pd

eg = pd.read_excel('C://Users//Vishaal//Documents//GitHub//ids705//assignment
eg.head()
```

Out[176]:

	eGRID2016 Plant file sequence number	Plant state abbreviation	Plant name	Plant latitude	Plant longitude	Plant primary fuel	Plant capacity factor	Plant nameplate capacity (MW)
0	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPCAP
1	1	AK	7-Mile Ridge Wind Project	63.2107	-143.247	WND	NaN	1.8
2	2	AK	Agrium Kenai Nitrogen Operations	60.6732	-151.378	NG	NaN	21.6
3	3	AK	Alakanuk	62.6833	-164.654	DFO	0.05326	2.6
4	4	AK	Allison Creek Hydro	61.0844	-146.353	WAT	0.01547	6.5

In [177]:

```
#Take care of the second row containing the column names. All Nan values have
new_header = eg.iloc[0]
eg = eg.iloc[1:]
eg.columns = new_header
eg  = eg.dropna()
```

8. a

In [45]:

```
max = eg['PLNGENAN'].max()
eg.loc[eg['PLNGENAN'] == max]
```

Out[45]:

	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPCAP	PLI
391	391	AZ	Palo Verde	33.3881	-112.862	NUC	0.87801	4209.6	3.23

The plant Palo Verde in AZ has generated the maximum power output in MWh.

8.b

In [46]:
```
maxl = eg['LAT'].max()
eg.loc[eg['LAT'] == maxl]
```

Out[46]:

	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPCAP	PLNG
12	12	AK	Barrow	71.292	-156.779	NG	0.28208	20.3	

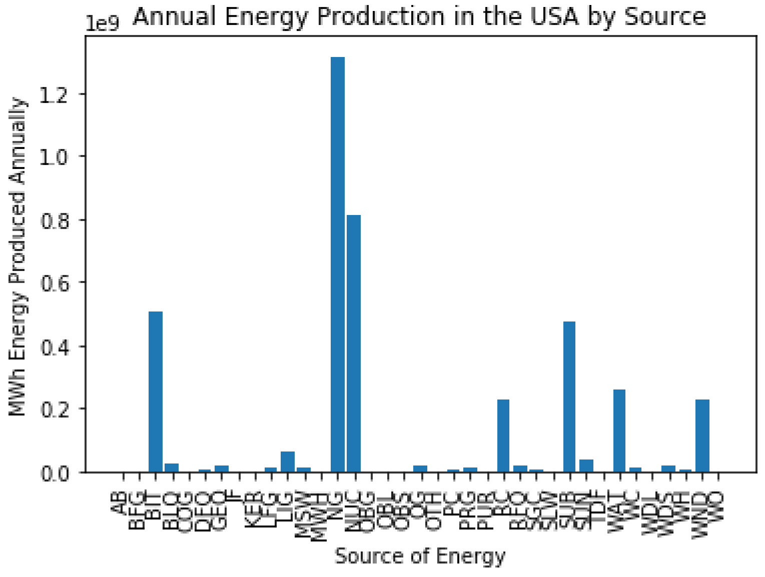
The northern most plant is in Barrow Alaska.

8. c

The northern most plant is located in Alaska.

8.d

In [179]:
```
import matplotlib.pyplot as plt

#Convert to float dtype
eg['PLNGENAN'] = eg['PLNGENAN'].astype(float)
#Groupby the energy source and take mean of annual production
eg_1 = eg.groupby('PLPRMFL', as_index=False)['PLNGENAN'].sum()
#plotting bar plot and setting axes labels and titles
plt.bar(eg_1['PLPRMFL'], eg_1['PLNGENAN'])
plt.xticks(rotation = 90)
plt.xlabel('Source of Energy')
plt.ylabel('MWh Energy Produced Annually')
plt.title('Annual Energy Production in the USA by Source')
plt.show()
```



8.e

Natural Gas is the major source of energy in the United States.

9

[8 points] Speed comparison between vectorized and non-vectorized code. Begin by creating an array of 10 million random numbers using the numpy random.randn module. Compute the sum of the squares first in a for loop, then using Numpy's `dot` module. Time how long it takes to compute each and report the results and report the output. How many times faster is the vectorized code than the for loop approach?

*Note: all code should be well commented, properly formatted, and your answers should be output using the `print()` function as follows (where the # represents your answers, to a reasonable precision):

`Time [sec] (non-vectorized): **4.807 s**

`Time [sec] (vectorized): **0.00498 s**

```
The vectorized code is 1000 times faster than the vectorized code
```

Answer

Non-Vectorized Code

In [102]:
```
import numpy as np
import time
#Generate 10 million random normally distributed numbers
num = np.random.randn(10000000)

#measure start time
t1 = time.time()

#for loop to calculate the sum of squares
for i in range (len(num)):
    num[i] = num[i]**2

s1 = np.sum(num)
#calcaulte  the end time
t2 = time.time()

print('Non-Vectorized Time = ', t2 - t1)
```

```
Non-Vectorized Time =  4.807978391647339
```

Vectorized Code

In [103]:
```python
#Generate 2 numpy arrays of 10 million random normally distributed numbers
num1 = np.random.randn(10000000)

#measure start time
t1 = time.time()

#vectorized operation to calculate the sum of squares
num3 = np.dot(num1, num1)
s2 = np.sum(num3)

#calcaulte  the end time
t2 = time.time()

print('Vectorized Time = ', t2 - t1)
```

```
Vectorized Time =  0.004986763000488281
```

10

[10 points] One popular Agile development framework is Scrum (a paradigm recommended for data science projects). It emphasizes the continual evolution of code for projects, becoming progressively better, but starting with a quickly developed minimum viable product. This often means that code written early on is not optimized, and that's a good thing - it's best to get it to work first before optimizing. Imagine that you wrote the following code during a sprint towards getting an end-to-end system working. Vectorize the following code and show the difference in speed between the current implementation and a vectorized version.

The function below computes the function $f(x, y) = x^2 - 2y^2$ and determines whether this quantity is above or below a given threshold, `thresh=0`. This is done for $x, y \in \{-4, 4\}$, over a 2,000-by-2,000 grid covering that domain.

(a) Vectorize this code and demonstrate (as in the last exercise) the speed increase through vectorization and (b) plot the resulting data - both the function $f(x, y)$ and the thresholded output - using `imshow` (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.imshow.html?highlight=matplotlib%20pyplot%20imshow#matplotlib.pyplot.imshow) from `matplotlib`.

Hint: look at the `numpy` `meshgrid` (https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.meshgrid.html) documentation

ANSWER

In [180]:

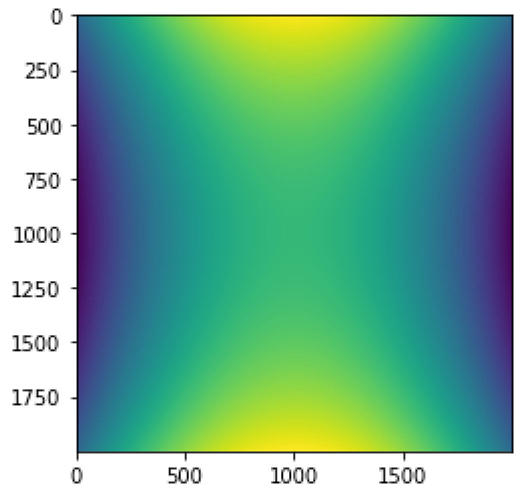
```
import numpy as np
import time
import matplotlib.pyplot as plt

nvalues = 2000
xvalues = np.linspace(-4,4,nvalues)
yvalues = np.linspace(-4,4,nvalues)
thresh  = 0

# Nonvectorized implementation
t0 = time.time()
f = np.zeros((nvalues,nvalues))
f_thresholded = np.zeros((nvalues,nvalues))
for ix, x in enumerate(xvalues):
    for iy, y in enumerate(yvalues):
        f[ix,iy]             = x**2 - 2 * y**2
        f_thresholded[ix,iy] = f[ix,iy] > thresh
t1 = time.time()
time_nonvectorized = t1 - t0

# Vectorized implementation
t0v = time.time()
fv = np.zeros((nvalues,nvalues))
f_thresholdedv = np.zeros((nvalues,nvalues))
#Create a mesh/matrix of values so that the function can be calcaulted in a v
yv, xv = np.meshgrid(yvalues, xvalues)
fv = xv**2 - 2*yv**2
f_thresholdedv = fv > thresh
t1v = time.time()
time_vectorized = t1v - t0v
```

In [181]:

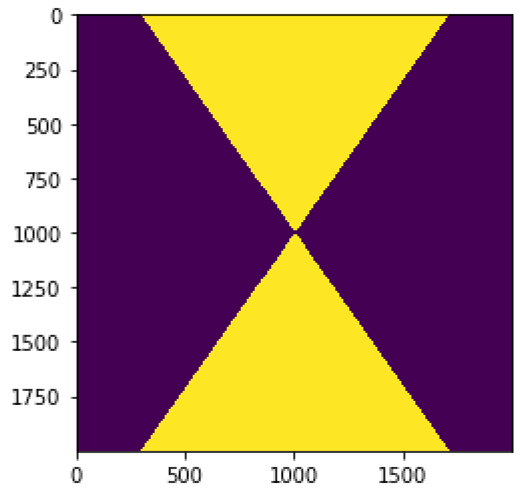
```
# Vectorization speed performance results
print('Non - Vectorized time = ', time_nonvectorized, 's')
print('Vectorized time = ', time_vectorized, 's')
```

```
Non - Vectorized time =  28.684417963027954 s
Vectorized time =  0.37689661979675293 s
```

In [138]:
```
# Plot the function f(x,y)
plt.imshow(fv)
plt.show()
```



In [139]:
```
# Plot the threshold function f(x,y)
plt.imshow(f_thresholdedv)
plt.show()
```



11

[10 points] This exercise will walk through some basic numerical programming exercises.

1. Synthesize $n = 10^4$ normally distributed data points with mean $\mu = 2$ and a standard deviation of $\sigma = 1$. Call these observations from a random variable X, and call the vector of observations that you generate, $\mathbf{x}$.
2. Calculate the mean and standard deviation of $\mathbf{x}$ to validate (1) and provide the result to a precision of four significant figures.
3. Plot a histogram of the data in $\mathbf{x}$ with 30 bins
4. What is the 90th percentile of $\mathbf{x}$? The 90th percentile is the value below which 90% of observations can be found.
5. What is the 99th percentile of $\mathbf{x}$?

6. Now synthesize $n = 10^4$ normally distributed data points with mean $\mu = 0$ and a standard deviation of $\sigma = 3$. Call these observations from a random variable Y, and call the vector of observations that you generate, $\mathbf{y}$.
7. Create a new figure and plot the histogram of the data in $\mathbf{y}$ on the same axes with the histogram of $\mathbf{x}$, so that both histograms can be seen and compared.
8. Using the observations from $\mathbf{x}$ and $\mathbf{y}$, estimate $E[XY]$

ANSWER

1

In [182]:
```
x = 1*np.random.randn(10000) + 2
x
```

Out[182]:
```
array([2.94554003, 2.91367466, 3.01688806, ..., 0.60094019, 2.63540386,
       2.1148088 ])
```

2

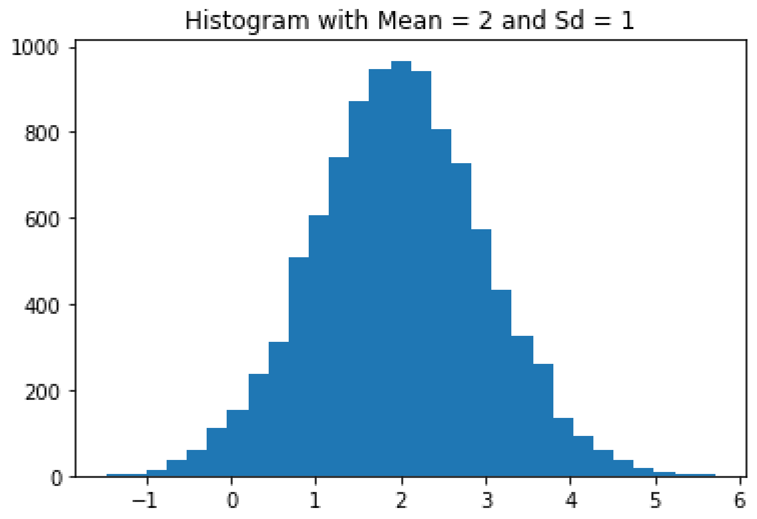
In [146]:
```
print('Mean is %.4f' % np.mean(x) )
print('Standard Deviation is %.4f' % np.std(x))
```

```
Mean is 1.9848
Standard Deviation is 1.0044
```

3

In [151]:
```
plt.hist(x, bins=30)
plt.title('Histogram with Mean = 2 and Sd = 1')
plt.show
```

Out[151]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



4

In [152]:
```
np.percentile(x,90)
```
Out[152]:
```
3.2842262309272607
```

5

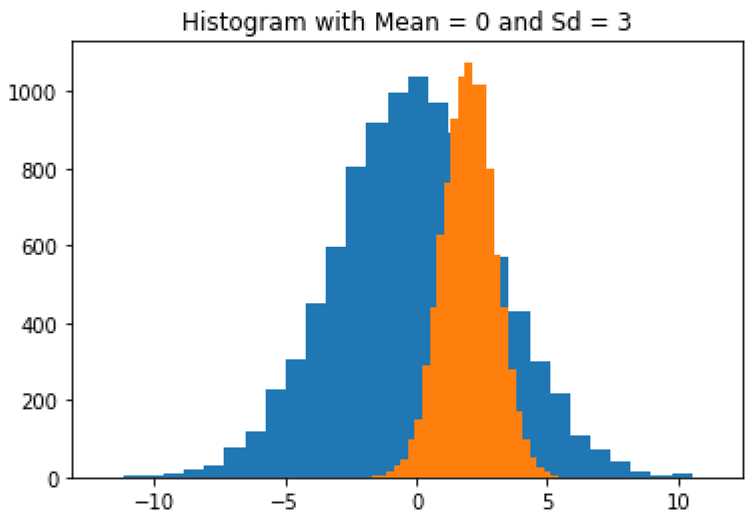
In [153]:
```
np.percentile(x,99)
```
Out[153]:
```
4.387881863668856
```

6

In [185]:
```
y = 3*np.random.randn(10000) + 0
y
```
Out[185]:
```
array([-0.76744879,  1.69776419, -0.98394937, ...,  0.12215293,
       -4.55636844, -0.404205  ])
```

7

In [187]:
```
plt.hist(y, bins=30)
plt.hist(x, bins=30)
#Axis changed to be kept the same as the distribution of X.
#plt.xlim(0,6)
plt.title('Histogram with Mean = 0 and Sd = 3')
plt.show
```
Out[187]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```

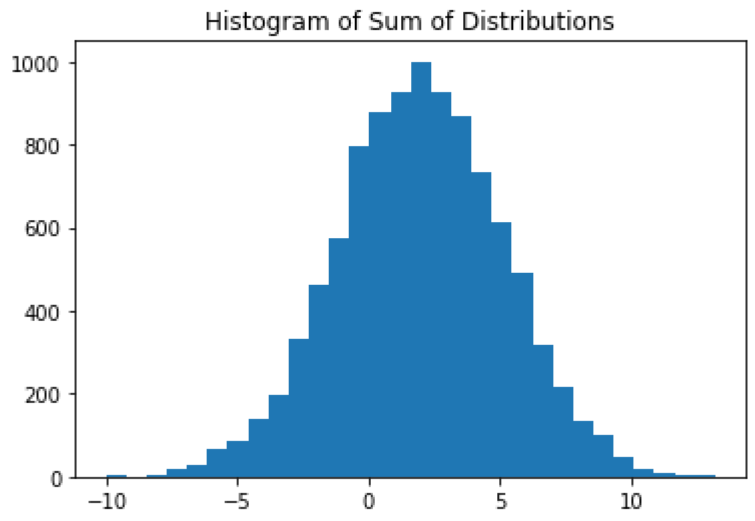


8

Ploting a histogram of the sum of the distributions X and Y, we can see that the distribution is still normal. This means the distributions are independent.

$$E[XY] = E[X]E[Y] = 0 * 2 = 0$$

In [175]:
```
plt.hist(y+x, bins = 30)
plt.title('Histogram of Sum of Distributions')
plt.show()
```



Version Control via Git

12

[1 point] You will need to use Git to submit assignments and in the course projects and is generally a version control and collaboration tool. You can even use some Git repositories (e.g. Github) as hosts for website, such as with the course website (https://kylebradbury.github.io/ids705/index.html).

Complete the Atlassian Git tutorial (https://www.atlassian.com/git/tutorials/what-is-version-control), specifically the following listed sections. Try each concept that's presented. For this tutorial, instead of using BitBucket as your remote repository host, you may use your preferred platform such as Github (https://github.com/) or Duke's Gitlab (https://gitlab.oit.duke.edu/users/sign_in).

1. What is version control (https://www.atlassian.com/git/tutorials/what-is-version-control)
2. What is Git (https://www.atlassian.com/git/tutorials/what-is-git)
3. Install Git (https://www.atlassian.com/git/tutorials/install-git)
4. Setting up a repository (https://www.atlassian.com/git/tutorials/install-git)
5. Saving changes (https://www.atlassian.com/git/tutorials/saving-changes)
6. Inspecting a repository (https://www.atlassian.com/git/tutorials/inspecting-a-repository)
7. Undoing changes (https://www.atlassian.com/git/tutorials/undoing-changes)
8. Rewriting history (https://www.atlassian.com/git/tutorials/rewriting-history)
9. Syncing (https://www.atlassian.com/git/tutorials/syncing)
10. Making a pull request (https://www.atlassian.com/git/tutorials/making-a-pull-request)
11. Using branches (https://www.atlassian.com/git/tutorials/using-branches)
12. Comparing workflows (https://www.atlassian.com/git/tutorials/comparing-workflows)

I also have created two videos on the topic to help you understand some of these concepts: [Git basics (https://www.youtube.com/watch?v=fBCwfoBr2ng)](https://www.youtube.com/watch?v=fBCwfoBr2ng) and a [step-by-step tutorial (https://www.youtube.com/watch?v=nH7qJHx-h5s)](https://www.youtube.com/watch?v=nH7qJHx-h5s).

For your answer, affirm that you *either* completed the tutorial or have previous experience with all of the concepts above. Do this by typing your name below and selecting the situation that applies from the two options in brackets.

ANSWER

*I, **Vishaal Venkatesh**, affirm that I have [**completed the above tutorial / I have previous experience that covers all the content in this tutorial**]*

Exploratory Data Analysis

13

[20 points] Here you'll bring together some of the individual skills that you demonstrated above and create a Jupyter notebook based blog post on data analysis.

1. Find a dataset that interests you and relates to a question or problem that you find intriguing
2. Using a Jupyter notebook, describe the dataset, the source of the data, and the reason the dataset was of interest.
3. Check the data and see if they need to be cleaned: are there missing values? Are there clearly erroneous values? Do two tables need to be merged together? Clean the data so it can be visualized.
4. Plot the data, demonstrating interesting features that you discover. Are there any relationships between variables that were surprising or patterns that emerged? Please exercise creativity and curiosity in your plots.
5. What insights are you able to take away from exploring the data? Is there a reason why analyzing the dataset you chose is particularly interesting or important? Summarize this as if your target audience was the readership of a major news organization - boil down your findings in a way that is accessible, but still accurate.

Here your analysis will evaluated based on:

1. Data cleaning: did you look for and work to resolve issues in the data?
2. Quality of data exploration: did you provide plots demonstrating interesting aspects of the data?
3. Interpretation: Did you clearly explain your insights? Restating the data, alone, is not interpretation.
4. Professionalism: Was this work done in a way that exhibits professionalism through clarity, organization, high quality figures and plots, and meaningful descriptions?

ANSWER

Introduction

Becoming successful on YouTube is an incredibly creative and difficult task and is far from something that could be accurately modelled. It would however, be fascinating to analyse if any relationship exists between the Likes to Dislikes ratio (L/D) of a particular video and the sentiments of the comments under a video.

In this analysis, we simply attempt an exploratory analysis to assess the faesability of a novel approach to try and model the Likes to Dislike ratio (L/D) of a video using the sentiment of the comments. This is done under the premise that the L/D ratio is a good predictor of a YouTuber's success. Measuring the sentiments of a the comments under a video can prove to be challenging and very subjective. However, this can be accomplished using the text sentiment analsyis tool VADER.

VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a lexical approach to sentiment analysis, i.e., it maps words, phrases and sentences to a dictionary of sentiments. Following this matching, the sentence is provided with four different scores – a positive, neutral, negative and compound score. The first three scores are on a scale of [0, 1] while the final compound score is the standardized sum of the first three scores on a scale of [-1, 1]. One reason for VADER's success and popularity lies in its ability to function without having to train a model. Everything we need to analyze a sentence is contained in the so-called dictionary of emotions. It is also noteworthy that VADER is very capable of dealing with emoticons, slang and a variety of contextual punctuation marks – all of which are very useful when dealing with social media text analysis.

Data

Two different datasets were used for this analysis. The first dataset contained data on the videos themselves, while the second dataset contained information on the comments under each video listed in the first dataset. The data is about videos featured in the trending category on YouTube between the dates of 13th September and 22nd October 2017 in the United States. Both datasets were linked by a unique video ID. The datasets were obtained from the popular data science website by the name of Kaggle.

The video dataset contained data on the name of the video, date when the video was uploaded, name of the channel, category (one of fifteen categories like news, entertainment etc.), views, likes and dislikes. The L/D ratio was calculated to simply be the ratio of the number of likes to the number of dislikes. It has to be noted that there were in all 73 video that had no dislikes whatsoever. These videos had to be removed from the dataset as a they would result in a L/D of infinity. There were no instances of missing data and no imputations or deletions had to be performed. Furthermore, about 2/3 of the videos in the dataset were repeated with different dates associated with them. These were simply a single video trending for multiple days and accumulating different number of views, likes and dislikes over these days. For the sake of this analysis, the likes, dislikes and views were summed up by video_id and each video were treated as a single entity despite trending on multiple days.

The comments dataset contained information of the content of the comment, the number of likes each comment received and the number of replies. It has to be noted that there were precisely 9 occurrences of incorrect textual data in the reply's column. These columns were removed from the dataset and were not used in the analysis. Furthermore, for future analysis, the most popular comment under a video was the comment that received the greatest number of likes for a given video_id. Finally, a one-to-many, inner join on the video_id was performed on the two datasets before using VADER to score the comments.

In [222]:
```
import math
import pandas as pd
import numpy as np
import seaborn as sns
import json
import itertools
import re
import random
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import requests
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

In [223]:
```
columns1 = ['video_id', 'title', 'channel_title', 'category_id',
          'tags', 'views', 'likes', 'dislikes', 'comment_total',
          'thumbnail_link', 'date']
#Read video dataset and drop the nas if any...
vid_df =  pd.read_csv("C:/Users/Vishaal/Desktop/IDS Final/USvideos.csv", usec
vid_df = vid_df.dropna()

columns2 = ['video_id', 'comment_text', 'likes', 'replies']
#Read comment dataset and drop the nas if any...
comm_df =  pd.read_csv("C:/Users/Vishaal/Desktop/IDS Final/UScomments.csv", u
comm_df = comm_df.dropna()


#vid_df.category_id = pd.Categorical(vid_df.category_id)
```

In [224]:
```
#Removing wrong entries in the reply column. Converting all to int...
comm_df.drop(41587, inplace=True)
comm_df.drop(41588, inplace=True)
comm_df.drop(51626, inplace=True)
comm_df.drop(142419, inplace=True)
comm_df.drop(142494, inplace=True)
comm_df.drop(189730, inplace=True)
comm_df.drop(245216, inplace=True)
comm_df.drop(114463, inplace=True)
comm_df.drop(388428, inplace=True)

comm_df.replies = pd.to_numeric(comm_df.replies)
comm_df.likes = pd.to_numeric(comm_df.likes)

#Creating a likes to dislike column
vid_df['L/D'] = vid_df.likes/vid_df.dislikes
```

In [225]:
```
#Some of the dates also seem to be messed up. We will rectify them and change
#Correcting the dates in vid_df
vid_df.loc[vid_df['date'] == '24.09xcaeyJTx4Co', 'date'] = '24.09'
vid_df.loc[vid_df['date'] == '26.0903jeumSTSzc', 'date'] = '26.09'
vid_df.loc[vid_df['date'] == '100', 'date'] = '09.10'
vid_df['date'] = vid_df['date'].apply(lambda x: pd.to_datetime(str(x).replace
vid_df['date'] = vid_df['date'].dt.date

video_df = vid_df.drop_duplicates(subset = 'video_id', inplace = False, keep
vid_df_1 = video_df[video_df['category_id'] == 24]
```

In [226]:
```
#An inner join makes the most sense here. Let all the video data survive. Thi
merged_left = pd.merge(left=video_df,right=comm_df, how='inner', left_on='vic
```

In [205]:
```
#Now we do the text analysis on all the comments using vader...
sid_obj = SentimentIntensityAnalyzer()

score = []
p = []
neu = []
neg = []
compound = []

for sentence in merged_left.comment_text:
        sentiment_dict = sid_obj.polarity_scores(str(sentence))
        score.append(sentiment_dict)
        p.append(sentiment_dict['pos'])
        neg.append(sentiment_dict['neg'])
        neu.append(sentiment_dict['neu'])
        compound.append(sentiment_dict['compound'])
```

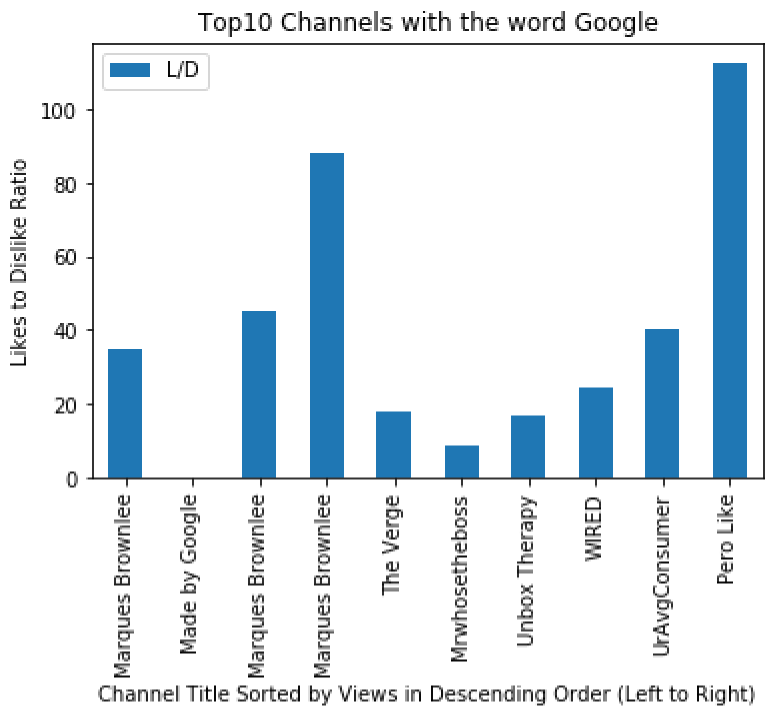
In [227]:
```
merged_left['poitive_score'] = p
merged_left['negative_score'] = neg
merged_left['neutral_score'] = neu
merged_left['compound_score'] = compound
```

Brief Insights

The following plot is a fun plot showing the number of occurences of the word 'Google' in 10 YouTube channel's titles sorted by views. The L/D ratio is given in the y-axis and the name of the channel is shown in the x-axis. Marques Brownlee's channel has the highest number of views out of all the channels with the word 'Google'and has a L/D ratio of roughly 40. This makes it a pretty popular channel and the occurence of the word Google in the channel's title comes as no surprise as Brownlee's channel is a tech channel.

In [249]:
```
Code sourced from kaggle - to see which channels have a certain trending word
ag_top = video_df[video_df['title'].str.contains('Google')].sort_values(by='v
x = tag_top.plot(kind='bar',x='channel_title',y='L/D',title='Top10 Channels w
x.set_xlabel('Channel Title Sorted by Views in Descending Order (Left to Righ
x.set_ylabel('Likes to Dislike Ratio')
```

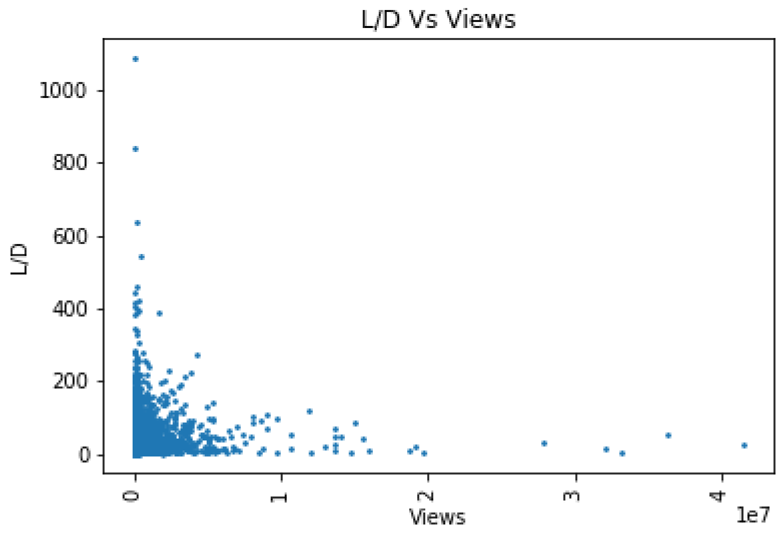
Out[249]:
```
Text(0, 0.5, 'Likes to Dislike Ratio')
```



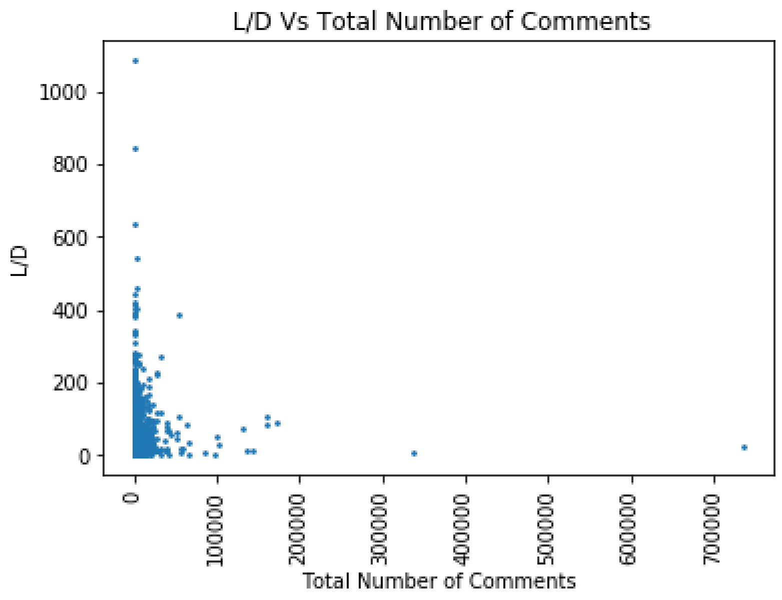
The following two plots are simple scatter plots that show how the L/D ratio vary with both the number of views and the number of comments. It can be seen that there is not much of a trend and the L/D ratio does not change much with either an increase in views or an increase on the total number of comments. This is in accordance with the point made in the beginning of the analysis - developing a model to predict L/D is difficult and strong correlations between L/D ratio and predictors are not going to be present. We shall further plot a correlation heat map below to study the relationships between the variables further.

In [258]:
```
plt.scatter(video_df['views'], video_df['L/D'], s = 3)
plt.xticks(rotation = 90)
plt.title('L/D Vs Views')
plt.xlabel('Views')
plt.ylabel('L/D')
```

Out[258]:
```
Text(0, 0.5, 'L/D')
```



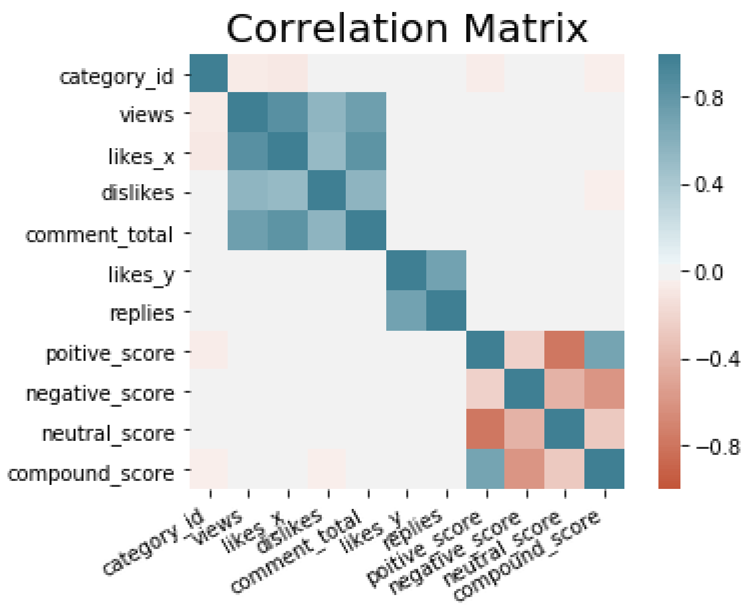
```
In [260]:  plt.scatter(video_df['comment_total'], video_df['L/D'], s = 3)
           plt.xticks(rotation = 90)
           plt.title(' L/D Vs Total Number of Comments')
           plt.xlabel('Total Number of Comments')
           plt.ylabel('L/D')

Out[260]: Text(0, 0.5, 'L/D')
```



From the correlation heat map given above, it was observed that there were some obvious positive correlations such as the number views and the likes on the video. An obvious negative correlation was between the positive score of a comment and the negative score of the same comment – a comment cannot be highly positive and negative at the same time. More interesting was the correlations between the dependent variable (L/D) and the various other possible predictors. L/D was slightly positively correlated with likes, positive score and compound score and was mildly negatively correlated with views, dislikes neutral and negative scores. It remains to be seen which of these predictors will emerge significant.

In [206]:
```
corr = merged_left.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True

)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=30,
    horizontalalignment='right'
);
ax.set_title('Correlation Matrix', fontsize=20)
```

Out[206]:
```
Text(0.5, 1, 'Correlation Matrix')
```



This analysis provides an insight into the data used, walks through the various steps in cleaning the data and performs some prelimenary analysis. A model can be developed in the future to accurately predict the L/D ratio based on the sentiments of the comments under a video.

In [ ]: