Assignment 1 - Probability, Linear Algebra, & Computational Programming

Derrick Nemetchek

Netid: Your netid here

Names of students you worked with on this assignment: LIST HERE IF APPLICABLE (delete if not)

Note: this assignment falls under collaboration Mode 2: Individual Assignment – Collaboration Permitted. Please refer to the syllabus for additional information.

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

Total points in the assignment add up to 90; an additional 10 points are allocated to presentation quality.

Learning Objectives

The purpose of this assignment is to provide a refresher on fundamental concepts that we will use throughout this course and provide an opportunity to develop skills in any of the related skills that may be unfamiliar to you. Through the course of completing this assignment, you will...

- Refresh you knowledge of probability theory including properties of random variables, probability density functions, cumulative distribution functions, and key statistics such as mean and variance.
- Revisit common linear algebra and matrix operations and concepts such as matrix multiplication, inner and outer products, inverses, the Hadamard (element-wise) product, eigenvalues and eigenvectors, orthogonality, and symmetry.
- Practice numerical programming, core to machine learning, by loading and filtering data, plotting data, vectorizing operations, profiling code speed, and debugging and optimizing performance. You will also practice computing probabilities based on simulation.
- Develop or refresh your knowledge of Git version control, which will be a core tool used in the final project of this course
- Apply your skills altogether through an exploratory data analysis to practice data cleaning, data manipulation, interpretation, and communication

We will build on these concepts throughout the course, so use this assignment as a catalyst to deepen your knowledge and seek help with anything unfamiliar.

If some references would be helpful on these topics, I would recommend the following resources:

- Mathematics for Machine Learning (https://mml-book.github.io/book/mml-book.pdf) by Deisenroth, Faisal, and Ong
- Deep Learning (https://www.deeplearningbook.org/); Part I: Applied Math and Machine Learning Basics by Goodfellow, Bengio, and Courville
- The Matrix Calculus You Need For Deep Learning (https://arxiv.org/pdf/1802.01528.pdf) by Parr and Howard
- Dive Into Deep Learning (https://d2l.ai/chapter_appendix-mathematics-for-deep-learning/index.html); Appendix: Mathematics for Deep Learning by Weness, Hu, et al.

Note: don't worry if you don't understand everything in the references above - some of these books dive into significant minutia of each of these topics.

Probability and Statistics Theory

Note: for all assignments, write out equations and math using markdown and LaTeX (https://tobi.oetiker.ch/lshort/lshort.pdf). For this assignment show ALL math work for questions 1-4, meaning that you should include any intermediate steps necessary to understand the logic of your solution

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

In [ ]:

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

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

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 and (b) variance of the data?

Show all work. Your answer should include the definition of mean and variance in the context of discrete data. In this case, use the sample variance since the sample size is quite small

ANSWER

(a) mean = (2+3+10+(-1)+(-1))/ 5 = 2.6

(b)

sum of squares = (2-2.6)^2 + (3-2.6)^2 + (10-2.6)^2 + (-1-2.6)^2 + (-1-2.6)^2 = 81.2

variance = sum of squares / 5-1 = 20.3

Linear Algebra

5

[5 points] A common task in machine learning is a change of basis: transforming the representation of our data from one space to another. A prime example of this is through the process of dimensionality reduction as in Principle Components Analysis where we often seek to transform our data from one space (of dimension n) to a new space (of dimension m) where $m < n$. Assume we have a sample of data of dimension $n = 4$ (as shown below) and we want to transform it into a dimension of $m = 2$.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

(a) What are the dimensions of a matrix, $\mathbf{A}$, that would linearly transform our sample of data, $\mathbf{x}$, into a space of $m = 2$ through the operation $\mathbf{Ax}$?

(b) Express this transformation in terms of the components of $\mathbf{x}$: x_1, x_2, x_3, x_4 and the matrix $\mathbf{A}$ where each entry in the matrix is denoted as $a_{i,j}$ (e.g. the entry in the first row and second column would be $a_{1,2}$). Your answer will be in the form of a matrix expressing result of the product $\mathbf{Ax}$.

Note: please write your answers here in LaTeX

ANSWER

(a) The matrix is a 2 by 4 $\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \end{bmatrix}$

(b)

$$\mathbf{Ax} = \begin{bmatrix} a_{1,1} * x_1 + a_{1,2} * x_2 + a_{1,3} * x_3 + a_{1,4} * x_4 \\ a_{2,1} * x_1 + a_{2,2} * x_2 + a_{2,3} * x_3 + a_{2,4} * x_4 \end{bmatrix}$$

6

[14 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 **using Python** or indicate that it cannot be computed. Refer to NumPy's tools for handling matrices. While all answers should be computer using Python, your response to whether each item can be computed should refer to underlying linear algebra. There may be circumstances when Python will produce an output, but based on the dimensions of the matrices involved, the linear algebra operation is not possible. **For the case when an operation is invalid, explain why it is not.**

When the quantity can be computed, please provide both the Python code AND the output of that code (this need not be in LaTex)

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

In [79]:

```
import numpy as np


# reminder for later. @ seems to be same as np.dot
#Hamamard product uses np.multiply, only takes in two matricies of same shape

A = np.array([[1,2,3],
      [2,4,5],
      [3,5,6]])

b = np.array([[-1],
      [3],
      [8]])

c = np.array([[4],
      [-3],
      [6]])

I = np.array([[1,0,0],
      [0,1,0],
      [0,0,1]])

print('Reminder for matricy multiplication, # cols of first matrix and # rows of second matrix must match.')
print()
print('1. AA\n', A@A)
print()
print('2. AA^T\n', A@np.transpose(A))
print()
print('3. Ab\n', A@b)
print()
print('4. Ab^T\n', 'Invalid operation. Trying to multiply 3x3 by 1x3' )
print()
print('5. bA \n', 'Invalid operation. Trying to multiply 3x1 by 3x3')
print()
print('6. b^T*a \n', np.transpose(b)@A )
print()
print('7. bb\n', 'Invalid operation. Trying to multiply 3x1 by 3x1.' )
print()
print('8. b^T*b\n', np.transpose(b)@b )
print()
print('9. bb^T \n', b@np.transpose(b) )
print()
print('10. b + c^T \n', np.add(b,np.transpose(c)))
print()
print('11. b^T * b^T \n', 'Invalid operation. Trying to multiply 1x3 by 1x3' )
print()
print('12.A^-1 *b \n',np.linalg.inv(A)@b)
print()
print('13. A∘A \n', np.multiply(A,A))
print()
print('14. b∘c \n', np.multiply(b,c))
# print('\n', )
```

Reminder for matricy multiplication, # cols of first matrix and # rows of second matrix must match.

1. AA

```
[[14 25 31]
 [25 45 56]
 [31 56 70]]
```

2. AA^T

```
[[14 25 31]
 [25 45 56]
 [31 56 70]]
```

3. Ab

```
[[29]
 [50]
 [60]]
```

4. Ab^T

Invalid operation. Trying to multiply 3x3 by 1x3

5. bA

Invalid operation. Trying to multiply 3x1 by 3x3

6. b^T*a

```
[[29 50 60]]
```

7. bb

Invalid operation. Trying to multiply 3x1 by 3x1.

8. b^T*b

```
[[74]]
```

9. bb^T

```
[[ 1 -3 -8]
 [-3  9 24]
 [-8 24 64]]
```

10. b + c^T

```
[[ 3 -4  5]
 [ 7  0  9]
 [12  5 14]]
```

11. b^T * b^T

Invalid operation. Trying to multiply 1x3 by 1x3

12.A^-1 *b

```
[[ 6.]
 [ 4.]
 [-5.]]
```

13. A∘A

```
[[ 1  4  9]
 [ 4 16 25]
 [ 9 25 36]]
```

14. b∘c

```
[[-4]
 [-9]
 [48]]
```

7

[8 points] Eigenvectors and eigenvalues. Eigenvectors and eigenvalues are useful for some machine learning algorithms, but the concepts take time to solidly grasp. They are used extensively in machine learning and in this course we will encounter them in relation to Principal Components Analysis (PCA), clustering algorithms, 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). For these questions, numpy may once again be helpful.

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 $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$. This relationship extends to higher orders: $\mathbf{A}\mathbf{A}\mathbf{v} = \lambda^2\mathbf{v}$
3. Show that the eigenvectors are orthogonal to one another (e.g. their inner product is zero). This is true for eigenvectors from real, symmetric matrices. In three dimensions or less, this means that the eigenvectors are perpendicular to each other. Typically we use the orthogonal basis of our standard x, y, and z, Cartesian coordinates, which allows us, if we combine them linearly, to represent any point in a 3D space. But any three orthogonal vectors can do the same. We will see this property is used in PCA to identify the dimensions of greatest variation in our data when we discuss dimensionality reduction.

ANSWER

In [197]:

```
# Answer in code.

# 1. Calculate eigenvalues and eigenveectors of Matrix A
w,v = np.linalg.eig(A) # w array of eivenvalues. v array of normalized eigenvectors
print('1. ')
print('eigenvalues: \n',w)
print('eigenvectors: \n', v)

# 2
print('2. \n')
lambdaSign = w
print('A * v: \n',A@v, '\n')

print('lambda * v: \n',lambdaSign*v)
print('Both matricies above are identical')

# 3
print('3. \n Output of multipying v by v^T and rounding: \n', np.matrix.round(v@np.transpose(v), decimals=1))
print('Result is an identity matrix')
```

```
1. 
eigenvalues: 
 [11.34481428 -0.51572947  0.17091519]
eigenvectors: 
 [[-0.32798528 -0.73697623  0.59100905]
 [-0.59100905 -0.32798528 -0.73697623]
 [-0.73697623  0.59100905  0.32798528]]
2. 

A * v: 
 [[-3.72093206  0.38008036  0.10101242]
 [-6.70488789  0.16915167 -0.12596043]
 [-8.36085845 -0.30480078  0.05605767]] 

lambda * v: 
 [[-3.72093206  0.38008036  0.10101242]
 [-6.70488789  0.16915167 -0.12596043]
 [-8.36085845 -0.30480078  0.05605767]]
Both matricies above are identical
3. 
 Output of multipying v by v^T and rounding: 
 [[ 1. -0.  0.]
 [-0.  1.  0.]
 [ 0.  0.  1.]]
Result is an identity matrix
```

Numerical Programming

8

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

In data science, we often need to have a sense of the idiosyncrasies of the data, how they relate to the questions we are trying to answer, and to use that information to help us to determine what approach, such as machine learning, we may need to apply to achieve our goal. This exercise provides practice in exploring a dataset and answering question that might arise from applications related to the data.

Data. The data for this problem can be found in the `data` subfolder in the `assignments` folder on github (https://github.com/kylebradbury/ids705). The filename is `a1_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) 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/default/files/2021-02/documents/egrid2019_technical_guide.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 these sorts of missing values will be important.

Your objective. For this dataset, your goal is to 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 type across all plants.

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

ANSWER

In [70]:

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# load the data
data = pd.read_excel('a1_egrid2016.xlsx')

# (a) get the plant from a row with the highest (MWh) value. Convert to ints, remove non int values, return plant name with the highest value
a = data.loc[data['Plant annual net generation (MWh)'] == data[pd.to_numeric(data['Plant annual net generation (MWh)'], errors='coerce').notnull()]['Plant annual net generation (MWh)'].max(), 'Plant name']
print('(a) Plant with the highest MWh:', a, '\n')

# (b) name of northern-most powerplant in the US. Return plant name of row with highest latitude
b = data.loc[data['Plant latitude'] == pd.to_numeric(data['Plant latitude'], errors='coerce').max(), 'Plant name']
print('(b) Plant with the highest latitude:', b, '\n')

# (c) state of northernmost powerplant in US. Same as above but return state instead of name
c = data.loc[data['Plant latitude'] == pd.to_numeric(data['Plant latitude'], errors='coerce').max(), 'Plant state abbreviation']
print('(c)', 'State with the plant that has highest latitude', c, '\n')

# (d) Plot a bar showing the amount of energy produced by each fuel type across all plants
# Should've done this early, create a dataset where the energy is numeric
numericEnergy = pd.to_numeric(data['Plant annual net generation (MWh)'], errors='coerce')
dataWithInts = data
dataWithInts['Plant annual net generation (MWh)'] = numericEnergy
# Group data by fuel type, for each fuel type sum the energy generation. Then plug into a bar plot
dataWithInts.pivot_table(values='Plant annual net generation (MWh)', index='Plant primary fuel', aggfunc='sum').plot.bar()
print('(d) Bar plot of fuel type energy generation below.', '\n')
# (e)
print('(e) Looking at the bar plot above, the fuel type with the highest annual energy generation is NG short for Natural Gas.', '\n')
```

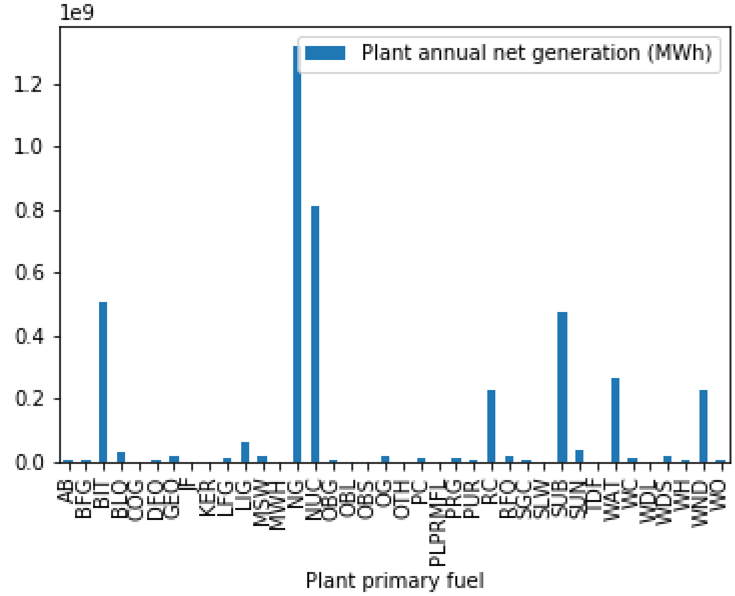
```
(a) Plant with the highest MWh: 391    Palo Verde
Name: Plant name, dtype: object
```

```
(b) Plant with the highest latitude: 12    Barrow
Name: Plant name, dtype: object
```

```
(c) State with the plant that has highest latitude 12    AK
Name: Plant state abbreviation, dtype: object
```

(d) Bar plot of fuel type energy generation below.

(e) Looking at the bar plot above, the fuel type with the highest annual energy generation is NG short for Natural Gas.



9

[6 points] *Vectorization*. When we first learn to code and think about iterating over an array, we often use loops. If implemented correctly, that does the trick. In machine learning, we iterate over so much data that those loops can lead to significant slow downs if they are not computationally efficient. In Python, vectorizing code and relying on matrix operations with efficient tools like numpy is typically the faster approach. Of course, numpy relies on loops to complete the computation, but this is at a lower level of programming (typically in C), and therefore is much more efficient. This exercise will explore the benefits of vectorization. Since many machine learning techniques rely on matrix operations, it's helpful to begin thinking about implementing algorithms using vector forms.

Begin by creating an array of 10 million random numbers using the numpy `random.randn` module. Compute the sum of the squares of those random numbers first in a for loop, then using Numpy's `dot` module to perform an inner (dot) product. 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 - your results may vary from run to run).

Your output should use the `print()` function as follows (where the # symbols represent your answers, to a reasonable precision of 4-5 significant figures):

```
Time [sec] (non-vectorized): ######

Time [sec] (vectorized):    ######

The vectorized code is ##### times faster than the nonvectorized code
```

ANSWER

In [101]:

```python
import time

bigArray = np.random.randn(10000000)

times = []
startTime = time.time()
fastSum = np.dot(bigArray,np.transpose(bigArray))
endTime = time.time()
times.append(endTime-startTime)

startTime = time.time()
sum = 0
for number in bigArray:
    sum += number*number
endTime = time.time()
times.append(endTime-startTime)

print('Time [sec] (non-vectorized):', round(times[1], 4))
print('Time [sec] (vectorized):', round(times[0], 4))
print('The vectorized code is', round(times[1] / times[0], 4), 'times faster than the nonvectorized code')
```

```
Time [sec] (non-vectorized): 3.2652
Time [sec] (vectorized): 0.0036
The vectorized code is 897.153 times faster than the nonvectorized code
```

10

[10 points] This exercise will walk through some basic numerical programming and probabilistic thinking exercises, two skills which are frequently use in machine learning for answering questions from our data.

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

In [77]:

```
import numpy as np
import matplotlib.pyplot as plt
# What is X?
# 1
m, sigma = 2, 1 # mean and standard deviation
n = int(1e4) # amount
x = np.random.normal(m,sigma,n)

# 2
mean = round(np.mean(x), 4)
deviation = round(np.std(x),4)
print('2.')
print('mean:', mean)
print('standard deviation', deviation)

# 3

print('3. Plot below')
hist, bin_edges = np.histogram(x, bins=range(30))
fig, ax = plt.subplots()
ax.title.set_text('Histogram of x. Mean:2 Deviation:1')
ax.bar(bin_edges[:-1], hist, width = 0.5)

#4
print('4. 90th percentile is: ', np.percentile(x,90))

#5
print('5. 99th percentile is: ', np.percentile(x,99))

#6
m, sigma = 0, 3 # mean and standard deviation
y = np.random.normal(m,sigma,n)
hist2, bin_edges2 = np.histogram(y, bins=range(30))

#7
print('7. Plot below')
fig2, ax2 = plt.subplots()
ax2.bar(bin_edges[:-1], hist, width = 0.5)
ax2.bar(bin_edges2[:-1], hist2, width = 0.5)
ax2.title.set_text('Histogram of x and y. Mean:0 Deviation:3')
ax2.legend('xy')
plt.show()

#8
print('TODO: what is E [X,Y]')
```

2.

mean: 1.9908

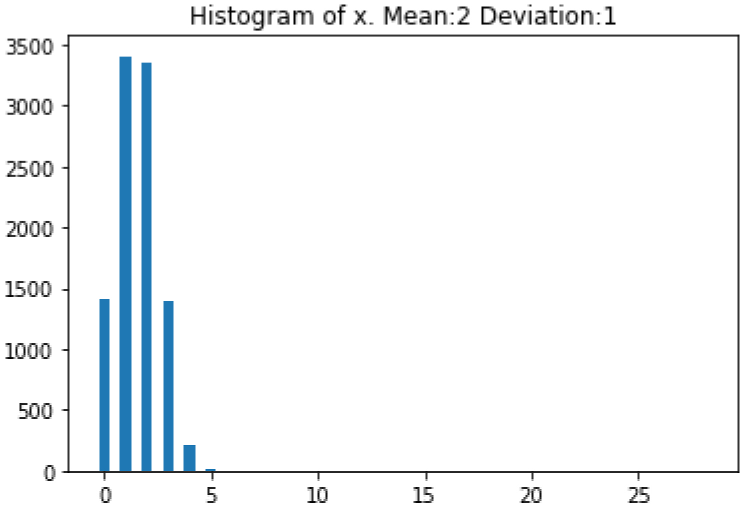
standard deviation 1.0073

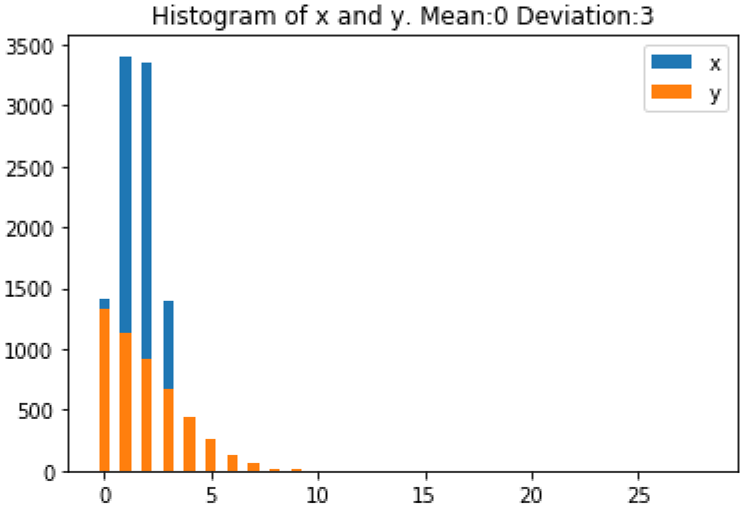
3. Plot below

4. 90th percentile is: 3.2714922977264878

5. 99th percentile is: 4.342204900502465

7. Plot below





TODO: what is E [X,Y]

Version Control via Git

11

[4 points] Git is efficient for collaboration, and expectation in industry, and one of the best ways to share results in academia. 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). As a data scientist with experience in machine learning, Git is expected. We will interact with Git repositories (a.k.a. repos) throughout this course, and your project will require the use of git repos for collaboration.

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) and a step-by-step tutorial (https://www.youtube.com/watch?v=nH7qJHx-h5s).

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

ANSWER

I, Derrick, affirm that I have previous experience that covers all the content in this tutorial

Exploratory Data Analysis

12

[15 points] Here you'll bring together some of the individual skills that you demonstrated above and create a Jupyter notebook based blog post on your exploratory data analysis. Your goal is to identify a question or problem and to work towards solving it or providing additional information or evidence (data) related to it through your data analysis. Below, we walk through a process to follow for your analysis. Additionally, you can find an example of a well-done exploratory data analysis here from past years (https://github.com/kylebradbury/ids705/blob/master/assignments/Assignment_1_Q12_Example.ipynb).

1. Find a dataset that interests you and relates to a question or problem that you find intriguing.
2. Describe the dataset, the source of the data, and the reason the dataset was of interest. Include a description of the features, data size, data creator and year of creation (if available), etc. What question are you hoping to answer through exploring the dataset?
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. If the data are clean, state how you know they are clean (what did you check?).
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. You should have at least a ~3 plots exploring the data in different ways.
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 for a general audience (imagine your publishing a blog post online) - boil down your findings in a way that is accessible, but still accurate.

Here your analysis will evaluated based on:

1. Motivation: was the purpose of the choice of data clearly articulated? Why was the dataset chosen and what was the goal of the analysis?
2. Data cleaning: were any issues with the data investigated and, if found, were they resolved?
3. Quality of data exploration: were at least 4 unique plots (minimum) included and did those plots demonstrate interesting aspects of the data? Was there a clear purpose and takeaway from EACH plot?
4. Interpretation: Were the insights revealed through the analysis and their potential implications clearly explained? Was there an overall conclusion to the analysis?

ANSWER

1 Find a dataset that interests you and relates to a question or problem that you find intriguing

I am using two datasets.

- US national covid cases and deaths by the NYT https://github.com/nytimes/covid-19-data (https://github.com/nytimes/covid-19-data)
- Google search trends for the term "covid symptoms" https://trends.google.com/ (https://trends.google.com/)

2 Describe the data set

- Covid cases is a csv file recording covid cases and deaths for each day. Their source is based on reports from state and local health agencies. The data is several hundred entries as covid has started in late in 2020 and has been continuing up to now. Intuitively the features are date, cases for that day, and deaths for that day. Since I am inspecting dates in the year 2022, I will use pandas to filter the data to that year.
- Google search trends is also a csv file recording the weekly 'popularity' of the search term "covid symptoms". The first column is the date and the second column is the 'popularity' for that week. It's worth noting that google doesn't provide specific numbers to the searches that were conducted. Rather they base it off of 'popularity' in which the week with the highest search terms has a value of 100. In the same set of data, a week with half as many searches would have a value of 50.
- **Goal** is to see how well search trends and deaths in the year 2022 line up in a line graph

3

There are no missing values, checked by viewing it in a text editor. The tables do become merged for a plot that contains both datasets.

4 Plot the data

Labelled plots below, generated from pandas

5 Insights

My goal was to compare how similar the lines would be of covid searches and covid deaths. There seems to be a weird spike in covid deaths in the month of November. When looking at the regular data there doesn't seem to be any spikes. I suspect applying a .diff() function may be causing this however this method seemed to work fine for other datasets.

Aside from the spike, covid search trends seem to be a decent feature for predicting deaths, though it's important to note line plots with independent axis can easily be tweaked to look similar.

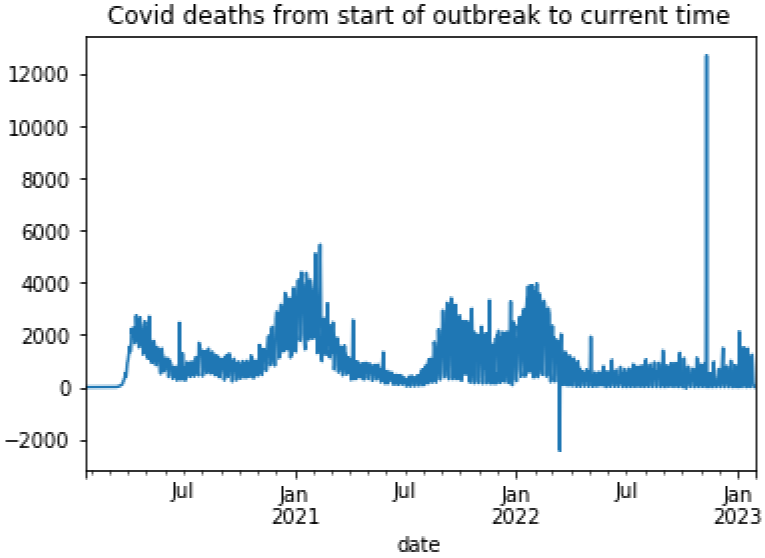
In [188]:

```
import pandas as pd

# import ny times data
path = 'https://raw.githubusercontent.com/nytimes/covid-19-data/master/us.csv'
covid = pd.read_csv(path, index_col='date', parse_dates=True)
covid.deaths.diff().plot(title = 'Covid deaths from start of outbreak to current time')
```

Out[188]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12529dc1cf8>
```

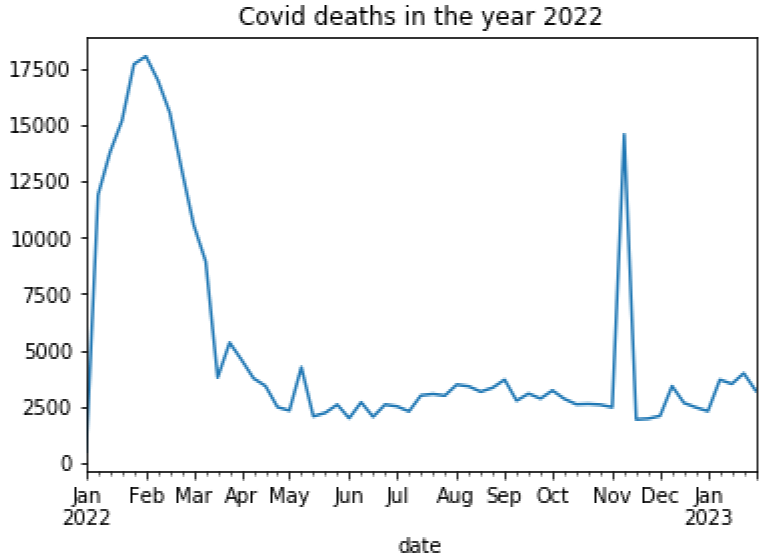


In [190]:

```
# filter covid data to year 2022 and resample to weekly
covid = covid.diff()['2022-01-01':].resample('w').sum() # get weekly number of cases/deaths
covid.deaths.plot(title='Covid deaths in the year 2022')
```

Out[190]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12529dc82e8>
```

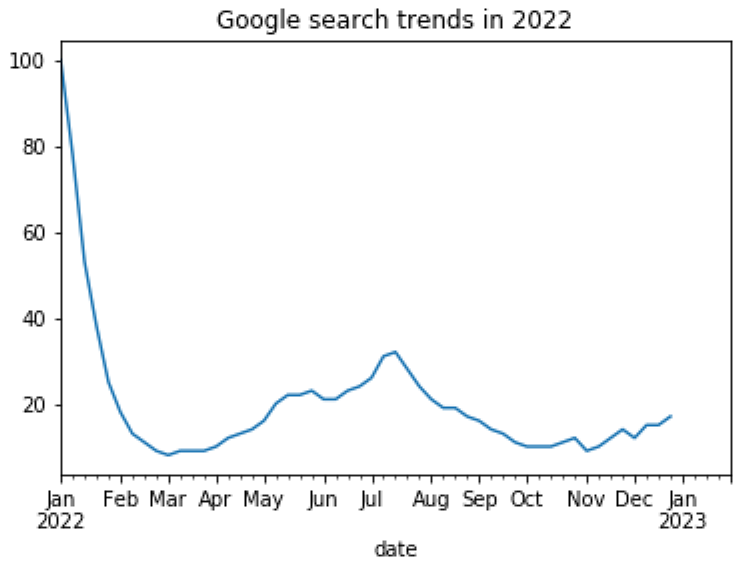


In [198]:

```
trends = pd.read_csv('search_trends.csv',index_col='Week',parse_dates=True, skiprows=2)
covid['google_searches'] = trends['covid symptoms: (United States)'] # combine data
covid.google_searches.plot(title='Google search trends in 2022')
```

Out[198]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12527d29c88>
```

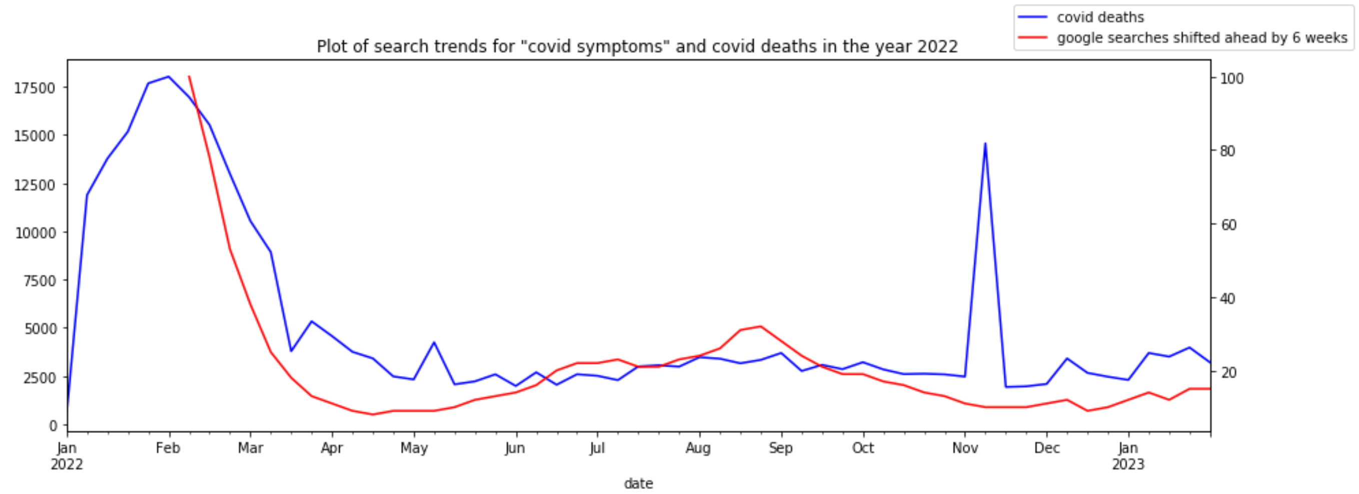


In [192]:

```
# create plot of deaths and google searches
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(15,5))
ax.set_title('Plot of search trends for "covid symptoms" and covid deaths in the year 2022')
covid.deaths.plot(ax=ax, label='covid deaths', color='blue')
covid.google_searches.shift(6).plot(ax=ax.twinx(),label='google searches shifted ahead by 6 weeks', color='red') # shift searches 6 weeks ahead to match better
fig.legend()
```

Out[192]:

```
<matplotlib.legend.Legend at 0x1252a05e0f0>
```


covid deaths
google searches shifted ahead by 6 weeks
Plot of search trends for "covid symptoms" and covid deaths in the year 2022
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
date