Assignment 1 - Probability, Linear Algebra, & Computational Programming

Alisa Tian

Netid: WT83

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

In [ ]:
```
import numpy as np
from numpy.linalg import eig
import time
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Instructions for all assignments can be found here, and is also linked to from the course syllabus.

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 by Deisenroth, Faisal, and Ong
- Deep Learning; Part I: Applied Math and Machine Learning Basics by Goodfellow, Bengio, and Courville
- The Matrix Calculus You Need For Deep Learning by Parr and Howard
- Dive Into Deep Learning; 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. 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

Since f(x) is a probabiilty density function, f(x)>=0 for all x in its domain (which has been already satisfied). Moreover, the definite integral in its domain is equal to 1.

$\int_0^2 \alpha x^2 dx = \frac{\alpha}{3} x^3 |_0^2 = \frac{8\alpha}{3} = 1$

$\alpha = 3/8$

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

Given the probability distribution function, the culumative distribution function is: $F(x) = \int_{-\infty}^{x} f(t)dt$

1. When x $\leq$ 0, F(x)=0
2. When 0<x<3,

   $\int_{x}^{\infty} \frac{1}{3} dx = 0$

   F(x)=x/3
3. When x $\geq$ 3, F(x)=1

For all possible values of x, the CDF is: $F(x) = \begin{cases} 0 & x \leq 0, \\ \frac{x}{3} & 0 < x < 3, \\ 1 & x \geq 3 \end{cases}$

3

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

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

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

(a).

$$\mu = \mathrm{E}[X] = \int_{-\infty}^{\infty} x f(x) dx = \frac{1}{3} * \frac{9}{2} = \frac{3}{2}$$

The expected value is 3/2.

(b)

$$\mathrm{Var}[X] = \mathrm{E}[(X-\mu)^2] = \int_0^3 (x-\frac{3}{2})^2 * \frac{1}{3} dx = \frac{1}{3}(\frac{36}{4} - \frac{27}{4}) = \frac{3}{4}$$

The variance is 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 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).

$$mean = \Sigma_i \frac{x}{n} = \frac{(2+3+10-1-1)}{5} = 13/5$$

(b).

$\sigma^2 = \frac{\Sigma_i x^2}{n} - \mu^2 = \frac{1}{4}[(2-\frac{13}{5})^2 + (3-\frac{13}{5})^2 + (10-\frac{13}{5})^2 + (-1-\frac{13}{5})^2 + (-1-\frac{13}{5})^2$
= 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 dimensions of matrix A is 2 (A is a 2×4 matrix).

(b).

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

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 "$\circ$".

ANSWER

In [ ]:
```
#1
A=[[1,2,3],[2,4,5],[3,5,6]]

print(np.matmul(A,A))
```

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

In [ ]:
```
#2
AT=np.transpose(A)
print(np.matmul(A,AT))
```

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

In [ ]:
```
#3
b=[[-1],[3],[8]]
print(np.matmul(A,b))
```

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

In [ ]:
```
#4
#The operation is invalid.
bT=np.transpose(b)
```

The transpose of matrix b is a 1 by 3 matrix. However, A is a 3 by 3 matrix. They cannot be multiplied together.

5. Invalid. The matrix b is a 3 by 1 matrix. However, A is a 3 by 3 matrix. They cannot be multiplied together.

In [ ]:
```
#6

print(np.matmul(bT,A))
```

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

7. Invalid. 1 by 3 matrix cannot multiply with a 1 by 3 matrix.

In [ ]:
```
# 8
print(np.matmul(bT,b))
```

```
[[74]]
```

In [ ]:
```
# 9
print(np.matmul(b,bT))
```

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

In [ ]:
```
# 10
c=[[4],[-3],[6]]
cT=np.transpose(c)
```

10. Invalid. 3 by 1 matrix cannot add with a 1 by 3 matrix.

11. Invalid. 1 by 3 matrix cannot multiply with a 1 by 3 matrix.

In [ ]:
```
# 12
AI=np.linalg.inv(A)
```

```
print(np.matmul(AI,b))
```

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

In [ ]:
```
#13
print(np.multiply(A,A))
```

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

In [ ]:
```
#14
print(np.multiply(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. Also, the series of linear algebra videos by Grant Sanderson of 3Brown1Blue are excellent and can be viewed on youtube here. 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 $\mathbf{Av} = \lambda\mathbf{v}$. 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 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 [ ]:
```
#1
A=np.matrix(A)
values,vectors=eig(A)
```

```
print('Eigenvalues:',values)
print('Eigenvectors:',vectors)
```

```
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]]
```

For second question, I chose the first pair and use assert to show that $\mathbf{Av} = \lambda\mathbf{v}$, which also extends to higher orders: $\mathbf{AAv} = \lambda^2\mathbf{v}$.

In [ ]:
```
#2
value0=values[0]
v0=vectors[:,0]
```

In [ ]:
```
value0*v0
```

Out[ ]:
```
matrix([[-3.72093206],
        [-6.70488789],
        [-8.36085845]])
```

In [ ]:
```
assert np.all(A*v0)==np.all(value0*v0)
```

In [ ]:
```
assert np.all(A*A*v0)==np.all(value0*value0*v0)
```

For the third question, to show eigenvectors are orthogonal, the inner product should be 0. I use assert to show this.

In [ ]:
```
# 3
v00=vectors[:,0].T
v10=vectors[:,1].T
v20=vectors[:,2].T


assert np.isclose(np.inner(v10,v20),0)
assert np.isclose(np.inner(v00,v10),0)
assert np.isclose(np.inner(v00,v20),0)
```

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. The filename is `a1_egrid2016.xlsx`. This dataset is the Environmental Protection Agency's (EPA) 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. 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 [ ]:
```
df=pd.read_excel("a1_egrid2016.xlsx",skiprows=[0])
df.head()
```

Out[ ]:

	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPCA
0	1	AK	7-Mile Ridge Wind Project	63.210689	-143.247156	WND	NaN	
1	2	AK	Agrium Kenai Nitrogen Operations	60.673200	-151.378400	NG	NaN	21
2	3	AK	Alakanuk	62.683300	-164.654400	DFO	0.05326	2
3	4	AK	Allison Creek Hydro	61.084444	-146.353333	WAT	0.01547	6
4	5	AK	Ambler	67.087980	-157.856719	DFO	0.13657	

In [ ]:
```
#a
plant1=df.loc[df['PLNGENAN'].idxmax()][2]
print("{a} has generated the most energy (measured in MWh).".format(a=plant1
```

```
Palo Verde has generated the most energy (measured in MWh).
```

In [ ]:
```
#b
plant2=df.loc[df['LAT'].idxmax()][2]
print("{b} is the northern-most power plant in the United States.".format(b=
```

```
Barrow is the northern-most power plant in the United States.
```

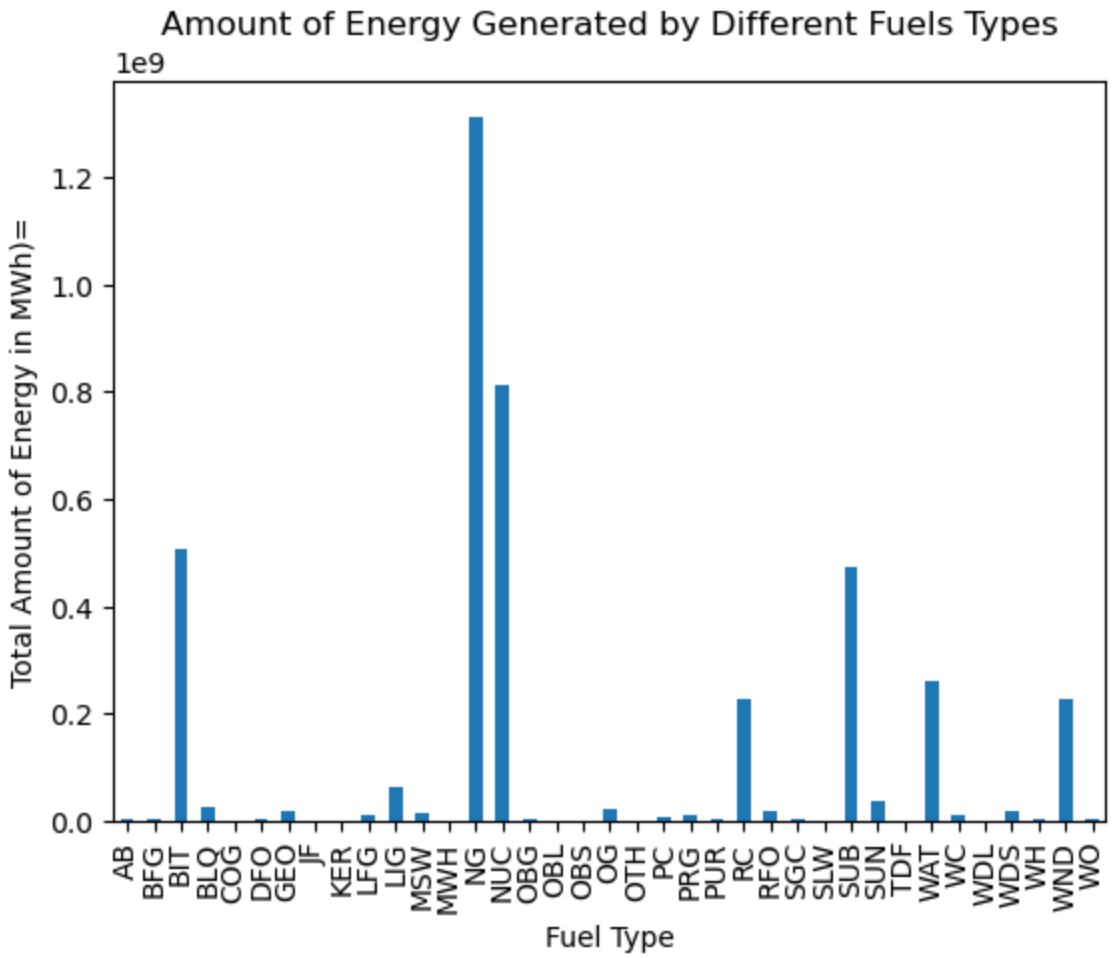
In [ ]:
```
#c
state=df.loc[df['LAT'].idxmax()][1]
print("The northern-most power plant in the United States is located in {c}.
```

```
The northern-most power plant in the United States is located in AK.
```

In [ ]:
```
#d
df2=df.groupby(['PLPRMFL'])['PLNGENAN'].sum()
df2.plot.bar()
plt.ylabel('Total Amount of Energy in MWh)=')
```

```
plt.xlabel('Fuel Type')
plt.title('Amount of Energy Generated by Different Fuels Types')
```

Out[ ]: Text(0.5, 1.0, 'Amount of Energy Generated by Different Fuels Types')



e. We can see from the barplot in (d) that the fuel: NG produces the most energy (MWh) in the United States.

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 [ ]:
```
numbers=np.random.rand(10000000)


# using for loop
start_time = time.time()
sum=0

for i in numbers:
    sum+=i*i
print(sum)

end_time = time.time() - start_time
```

```
3332605.9279979565
```

In [ ]:
```
#using inner product
start_time2 = time.time()
np.inner(numbers,numbers)
end_time2 = time.time() - start_time2
```

In [ ]:
```
print("Time [sec] (non-vectorized): {:.4f} s".format(end_time) )
print("Time [sec] (vectorized): {:.4f}s".format(end_time2))
print("The vectorized code is {time:4f} times faster than the nonvectorized
```

```
Time [sec] (non-vectorized): 1.1969 s
Time [sec] (vectorized): 0.0011s
The vectorized code is 1045.222569 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

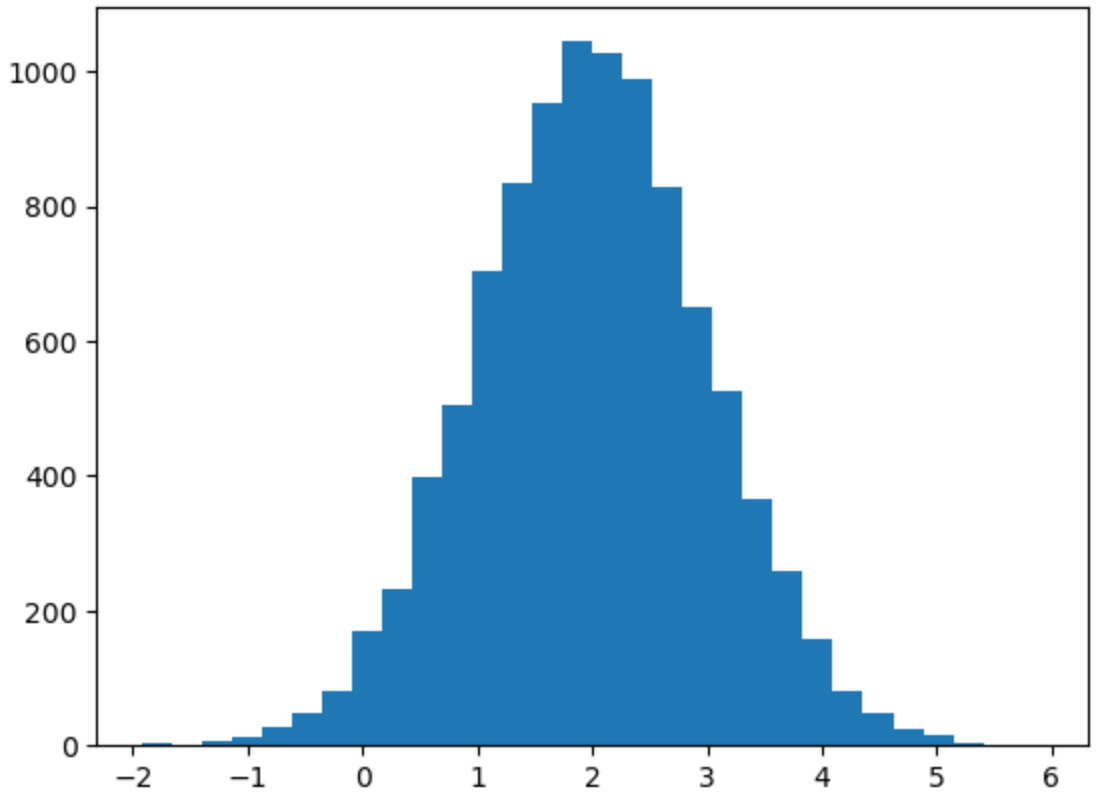
In [ ]:
```
#1
np.random.seed(42)
X=np.random.normal(2,1,10000)
X
```

Out[ ]:
```
array([2.49671415, 1.8617357 , 2.64768854, ..., 1.29468328, 2.49576557,
       2.64438845])
```

In [ ]:
```
#2
mean=np.mean(X)

std=np.std(X)

print("The mean of X is {a:2f}. The standard deviation is {b:2f}.".format(a
```

```
The mean of X is 1.997864. The standard deviation is 1.003412.
```

In [ ]:
```
# 3
plt.hist(X,bins=30)
plt.show()
```



In [ ]:
```
#4
np.percentile(X,90)
```

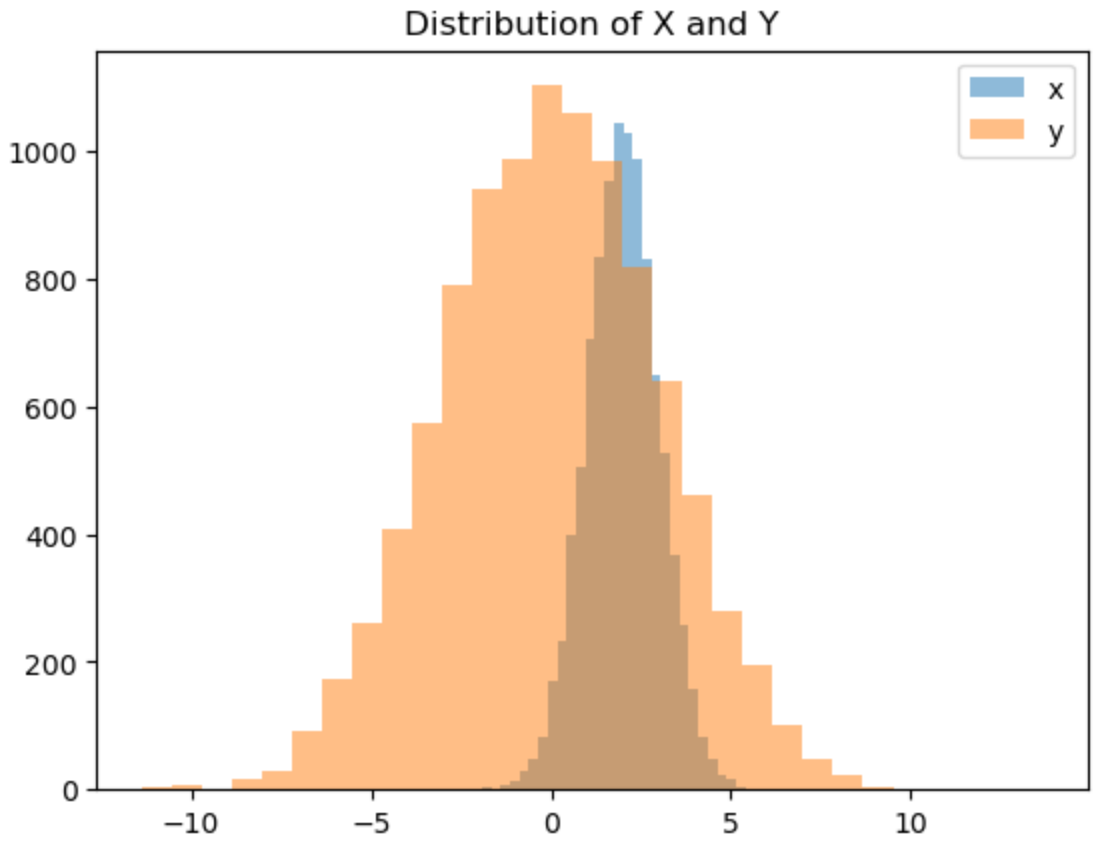
Out[ ]:
```
3.2812791676710593
```

In [ ]:
```
#5
np.percentile(X,99)
```

Out[ ]:
```
4.324830202634069
```

In [ ]:
```
#6
Y=np.random.normal(0,3,10000)
```

In [ ]:
```
#7
p1=plt.hist(X,alpha=0.5,label='x',bins=30)
p2=plt.hist(Y,alpha=0.5,label='y',bins=30)
plt.legend(loc='upper right')
plt.title("Distribution of X and Y")
plt.show()
```



E[XY]=E[X]E[Y]+Cov(X,Y)

In [ ]:
```
#8
EX=np.mean(X)
EY=np.mean(Y)
cov=np.cov(X,Y)[0][1]
E_XY=EX+EY+cov
print("The E[XY] is {a}.".format(a=E_XY ))
```

```
The E[XY] is 1.9439701169501598.
```

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. 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, 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 or Duke's Gitlab.

1. What is version control
2. What is Git
3. Install Git
4. Setting up a repository
5. Saving changes
6. Inspecting a repository
7. Undoing changes
8. Rewriting history
9. Syncing
10. Making a pull request
11. Using branches
12. Comparing workflows

I also have created two videos on the topic to help you understand some of these concepts: Git basics and a step-by-step tutorial.

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, [Alisa Tian], affirm that I have [completed the above tutorial / 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.

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?

Motivation

I love to play video games when I was a kid, and I found this video game sales dataset very interesting. This dataset is achieved from Kaggle: https://www.kaggle.com/datasets/gregorut/videogamesales, and according to its description, is scrapped from this website: https://www.vgchartz.com using BeautifulSoup. This dataset contains the videogames with sales over 100,000 copies around the globe. The variables include the ranking of sales, year of release, genre, publisher, sales in different continents, platforms, etc. There are 16719 entries in this dataset. For this analysis, I am interested in the variables that may potentially affect global sales, video games of what genres and platforms are the most popular, and want to explore the correlation between user and critic score. From this analysis, I hope to

gain more insight into factors that affect the global sales and hope to provide recommendations for video game companies.

Data Cleaning

Before cleaning, I first check the shape of the data: 16719 entries and 16 columns. Then, I checked if there are any missing values in this data set. I dropped the missing values of columns: year of release, name, genre and publisher. The percentage of missing data for critic and user score, critic and user count, developer and rating. It is not feasible to remove them or replace with mean value. So I kept those missing data. For this dataset, I did not merge any columns since I think they are very clear and there is no need to merge.

In [ ]:
```
vg=pd.read_csv("Video.csv")
vg.head()
vg.shape
```

Out[ ]:
```
(16719, 16)
```

In [ ]:
```
vg.isna().sum()
```

Out[ ]:
```
Name                 2
Platform             0
Year_of_Release    269
Genre                2
Publisher           54
NA_Sales             0
EU_Sales             0
JP_Sales             0
Other_Sales          0
Global_Sales         0
Critic_Score      8582
Critic_Count      8582
User_Score        9129
User_Count        9129
Developer         6623
Rating            6769
dtype: int64
```

In [ ]:
```
# drop the columns where the year of release, name, genre and publisher are
vg1=vg[vg['Year_of_Release'].notna()]
vg1=vg1[vg1['Publisher'].notna()]
vg1=vg1[vg1['Name'].notna()]
vg1=vg1[vg1['Genre'].notna()]
vg1.isna().sum()
```

Out[ ]:
```
Name                  0
Platform              0
Year_of_Release       0
Genre                 0
Publisher             0
NA_Sales              0
EU_Sales              0
JP_Sales              0
Other_Sales           0
Global_Sales          0
Critic_Score       8434
Critic_Count       8434
User_Score         8955
User_Count         8955
Developer          6512
Rating             6649
dtype: int64
```

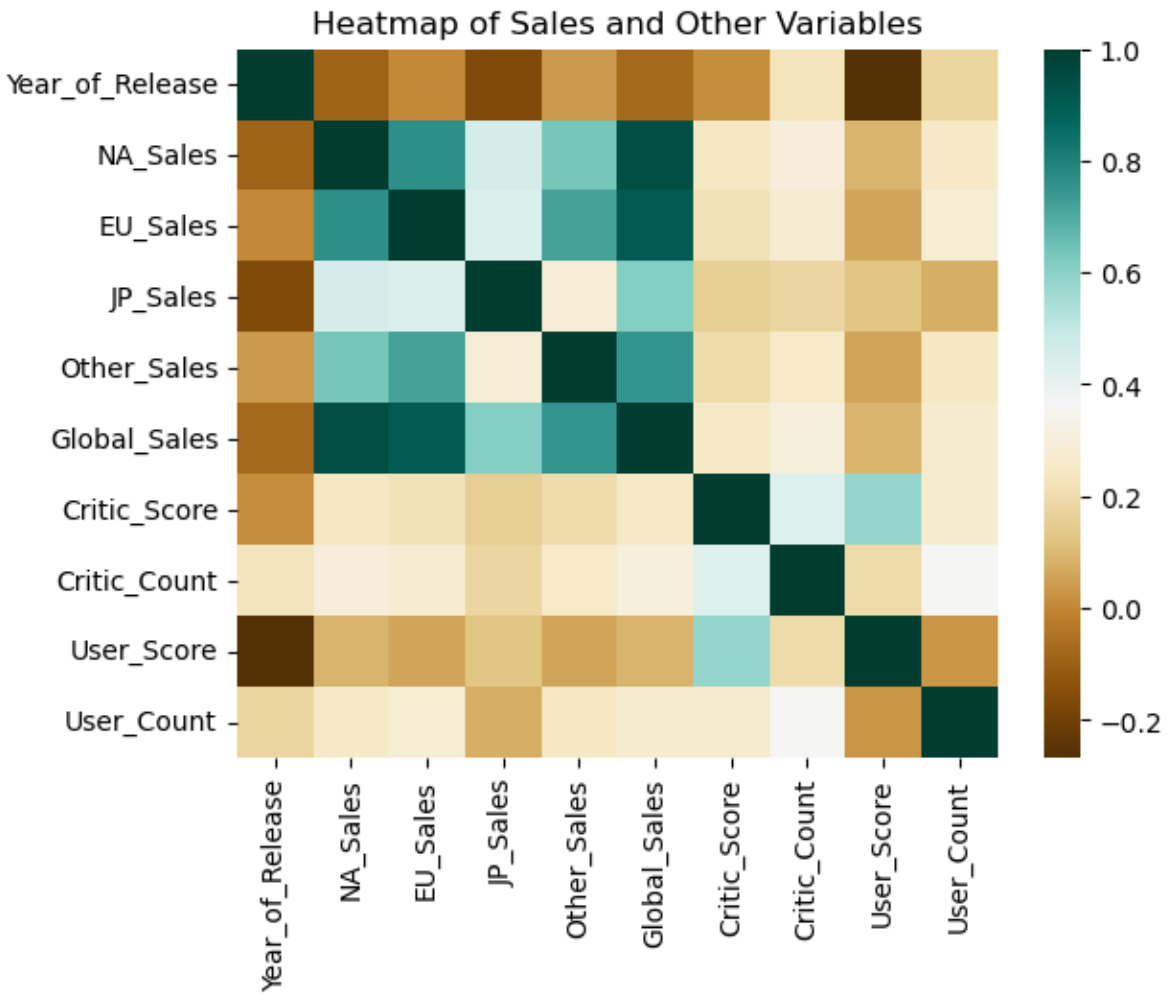
In [ ]:
```
vg1.head()
```

Out[ ]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sa
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.76	3
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.93	3.
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.

Data Exploration

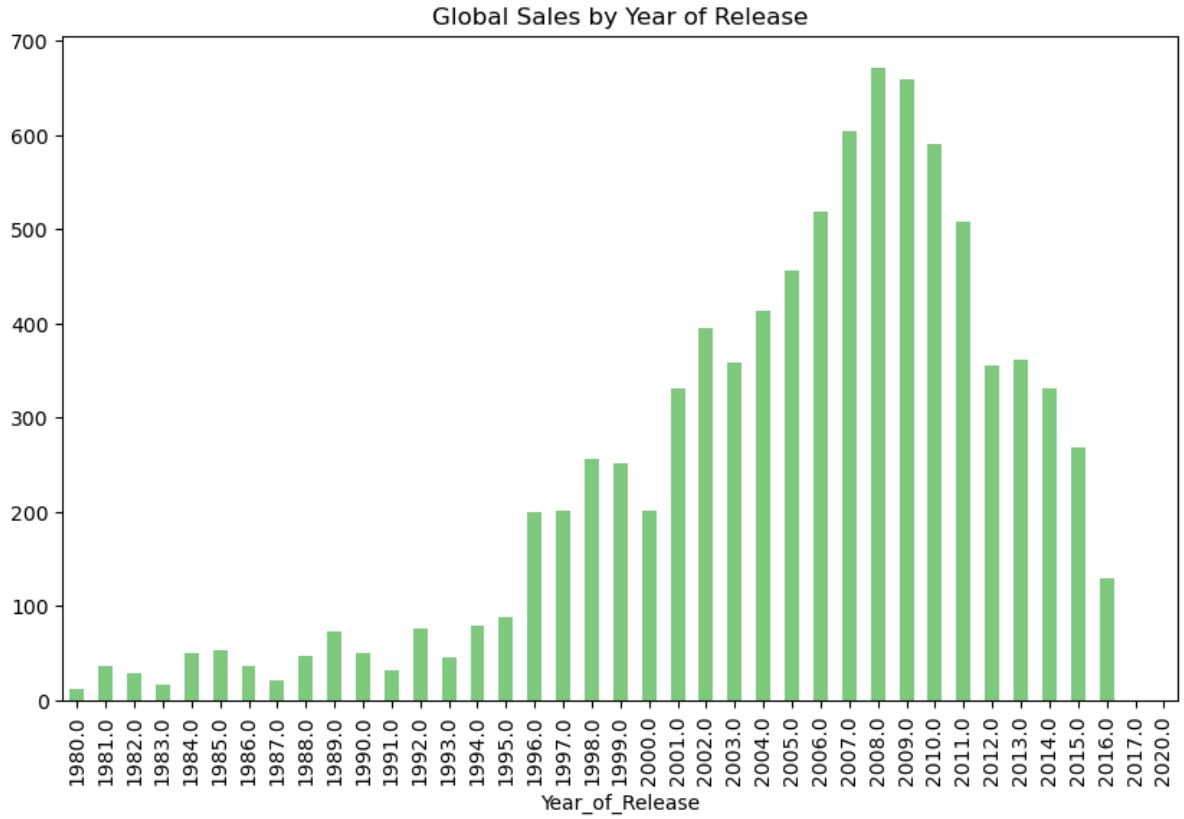
In [ ]:
```
# create correlation heatmap of sales and other variables of vg1
sns.heatmap(vg1.corr(),cmap='BrBG')
plt.title("Heatmap of Sales and Other Variables")
plt.show()
```



I am interested in the correlations among sales and different variables in this dataset, and want to find out those with relatively higher correlation in order to increase the video game sales. So, for the first step of data exploration, I plot the correlation heatmap among different variables. This plot illustrates that the correlation between global sales and critic score, user score, critic count as well as user count is positive, but low. On the other hand, the correlation between user and critic score is about 0.5, which will also be analyzed later.

Year of release

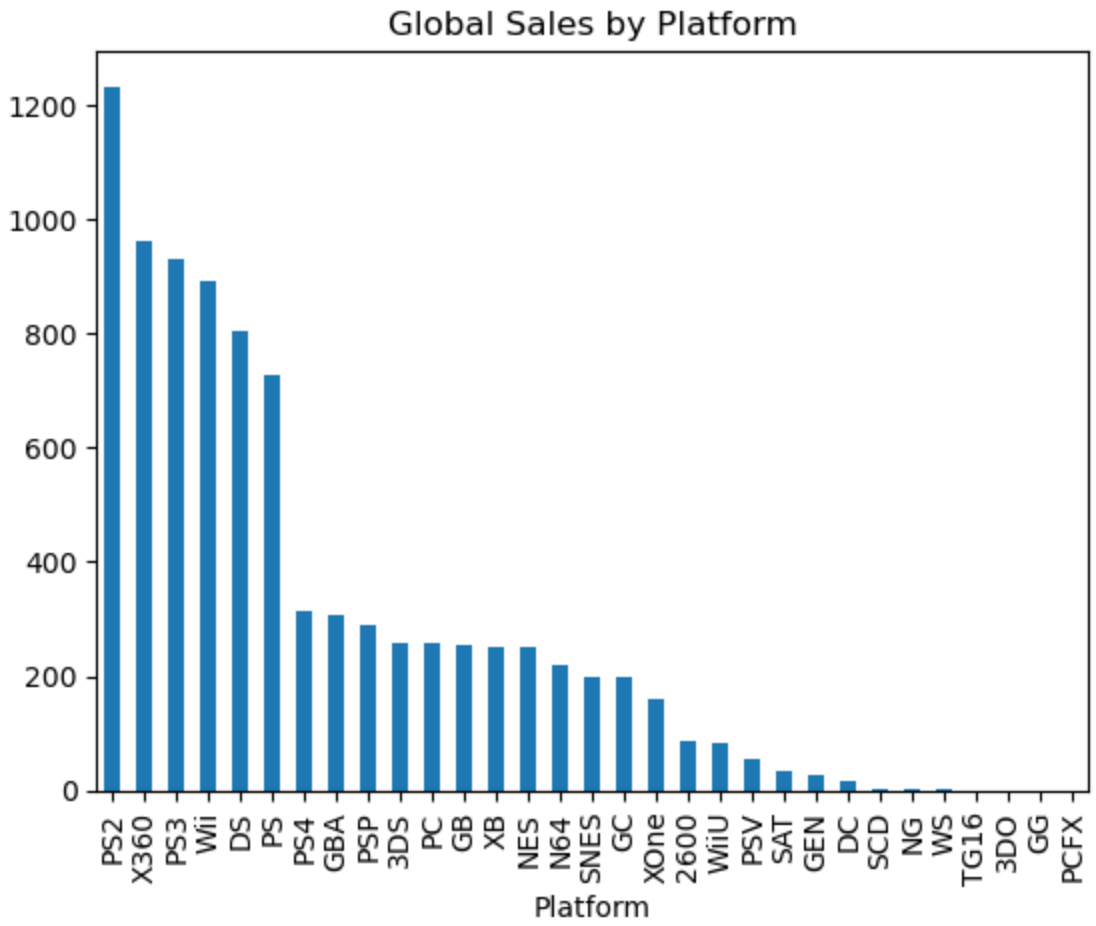
In [ ]:
```
# add title of the plot
vg1.groupby('Year_of_Release')['Global_Sales'].sum().plot(figsize=(10,6),col
plt.title("Global Sales by Year of Release")

plt.show()
```



I am also interested in the distribution of video games sales released each year. This plot show that this pattern is similar to a normal distribution whicih is left skewd. The global sales of video games released fluctuated from 1980 to 1995, but still remained at a relatively low level. This number surged in the year 1996, when it became more and more popular before the millennium. The year of 2008 has the greastest global video game sales. However, after 2008, it experienced a downward trend.

Sales by platform

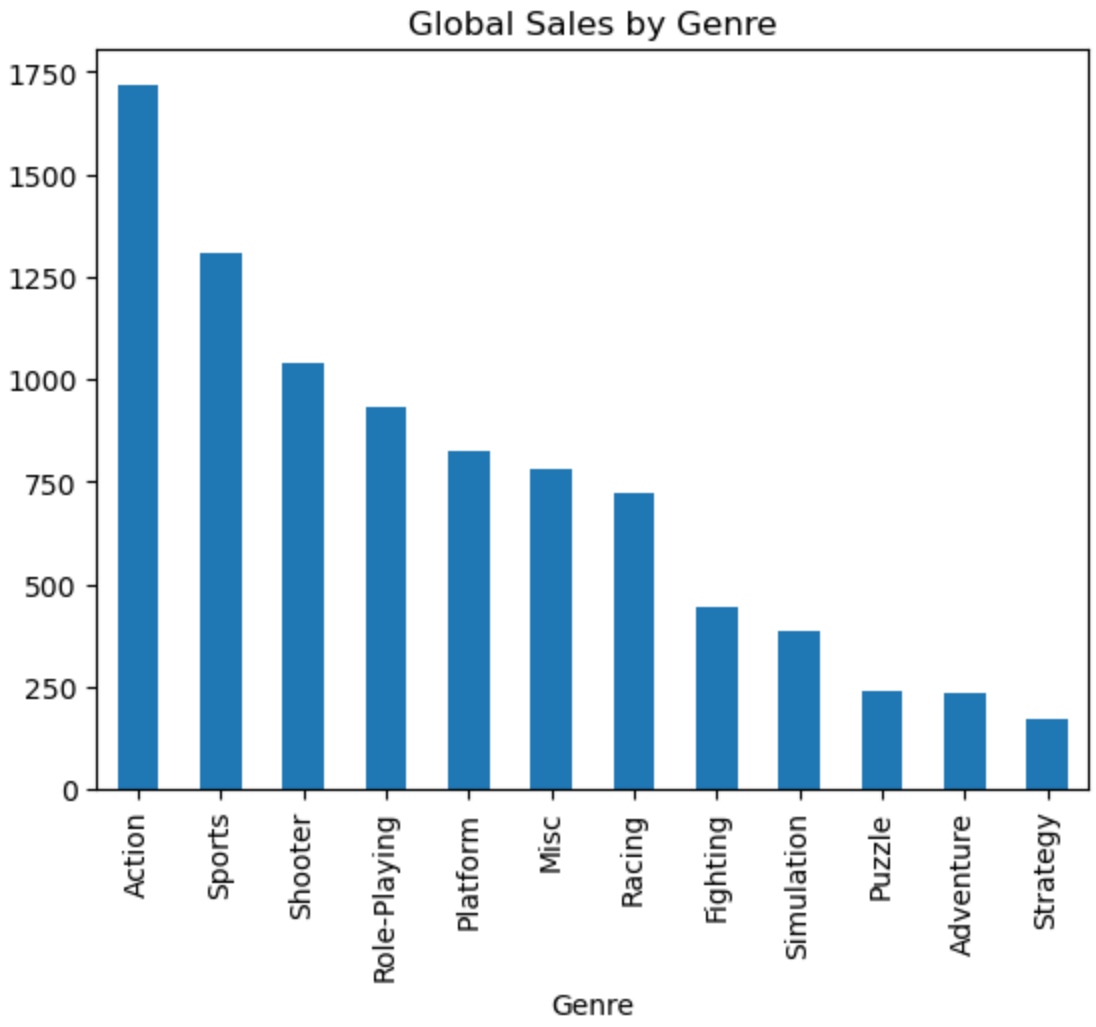
In [ ]:
```
# create bar plot of sales by platform of vg1
vg1.groupby(['Platform'])['Global_Sales'].sum().sort_values(ascending=False)
plt.title("Global Sales by Platform")
plt.show()
```



This plot shows the global sales of video games by platform. It is obvious that the PS2 platform took the three decades, with sales of approximately 1200 million USD. The global sales of X360, PS3, Wii, DS and PS are quite similar - varying from about 950 to 750 million USD. There is a huge gap between the sales of above platforms and others. Platforms like PS4, GBA and PSP are about only half of the previous group.

Sales by video games genre

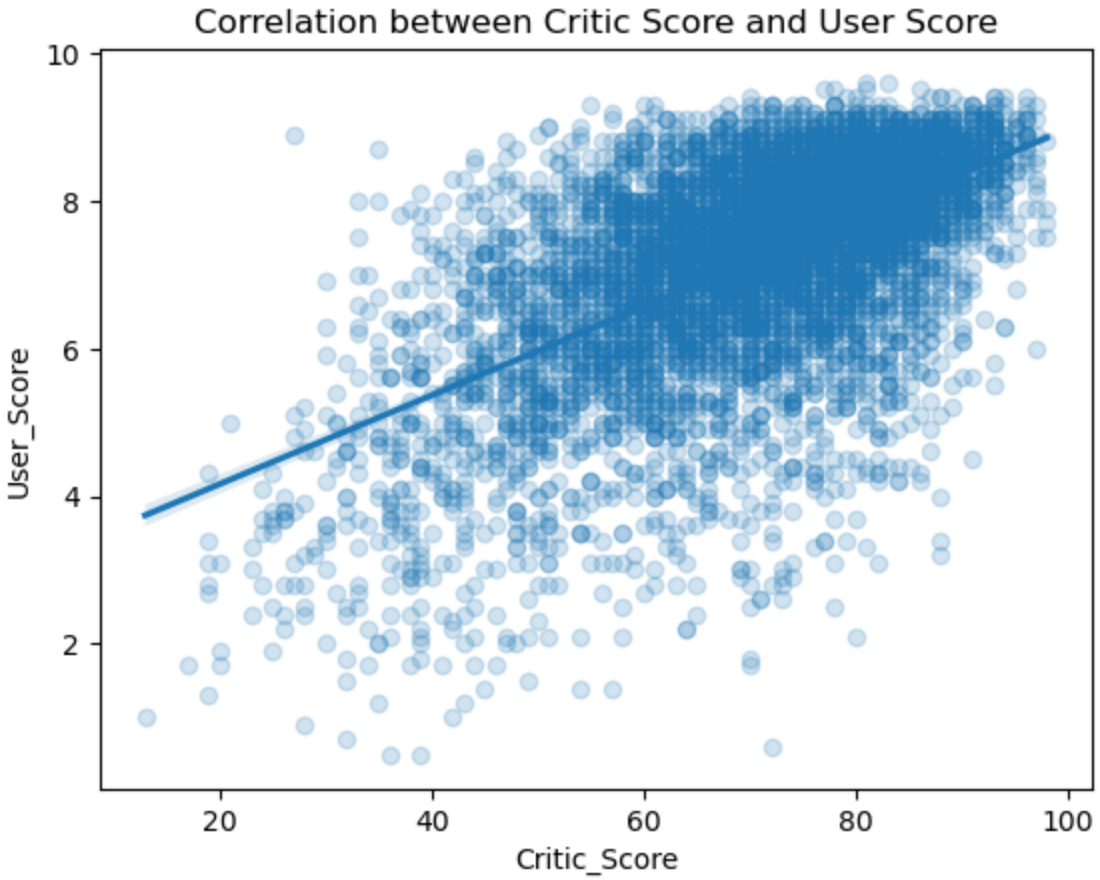
In [ ]:
```
# create bar plot of sales by Genre of vg1
vg1.groupby(['Genre'])['Global_Sales'].sum().sort_values(ascending=False).pl
plt.title("Global Sales by Genre")
plt.show()
```



This figure illustrates the global sales by different video games genre. The action video games are the most popular one. One of the reasons might be this game has great visual effects for the uses. The sports and shooter games follow the action games. The puzzle, adventure and strategy have the least global sales at around 200 million USD. These kind of games require the users to think carefully and that might be the reasons that they have the least sales.

Explore the correlation between critic score and user score

In [ ]:
```
# plot the correlation between critic score and user score in vg dataset
sns.regplot(x='Critic_Score',y='User_Score',data=vg1,scatter_kws={'alpha':0.
plt.title("Correlation between Critic Score and User Score")
plt.show()
```



This scatter plot shows the correlation between critic score and user score. From this graph, we know that they are positively correlated, which can provide into helping imrove the user score and user experience. This graph also shows that some data with the high critic score tends to have relatively user score.

Interpretation

This EDA explores different factors that may affect the global sales of video games. This dataset is interesting for me since it contains data of video game sales over the past three decades in different platforms and with a wide variety of genres.

The sales of video games released reached their peak in the year of 2008. Since then, there is a recession in the global video game market. PS2, X360, PS3, Wii, DS and PS are the most popular video game platforms during these years. The video game company can consider investing more on these platforms which are likely to achieve greater sales.

Based on the previous analysis, video games like action and shooting games tend to be popular. These kinds of game tend to have amazing visual effects that appeal to users. In comparison, games like puzzles and strategy are the least popular, which require the users to think carefully when they play the game. For companies that produce puzzle

and strategy games, one future recommendation would be to improve the visual effect that could potentially increase users' interest in these kinds of games.

This exploratory analytics also indicates that there is a positive relationship between the critic scores and users' score, which can provide implications for improving the user's experience. There is also some divergence between these two kinds of review - some games with high critic scores tend to have low users score. Before releasing the video games, the video companies could conduct more user surveys and make effort to improve the users' reviews.