Assignment 1 - Probability, Linear Algebra, & Computational Programming

Fides Regina Schwartz

Netid: fs113

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 that is unfamiliar.

Probability and Statistics Theory

Note: for all assignments, write out all equations and math using markdown and LaTeX. For this assignment show ALL math work

Question 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 1

I need to integrate to 1 over the function domain and solve for a:

First integrate to one:

$$\int_0^2 ax^2 dx = 1$$

Then integrate over the function domain:

$$\int_0^2 ax^2 dx = \frac{a}{3}x^3\Big|_0^2 = \frac{8}{3}a$$

Finally, solve for a:

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

Question 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.

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

To get the cumulative distribution function (CDF), $F(x)$, of a probability distribution function (PDF), $f(x)$, I need to calculate the integral of the PDF from 0 to x. This example case has three parts. (1) $x \leq 0$, $F(x) = 0$; (2) $(x \geq 3)$, $F(x) = 1$. (3) calculate the integral:

$$\int_0^x \frac{1}{3} dz = \frac{1}{3}z\Big|_0^x = \frac{1}{3}x$$

When combining them, the CDF is:

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

Question 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 3

(a) What is the xpected value of X?

$$\begin{aligned} E[x] &= \int_{-\infty}^{\infty} x f(x) dx \\ &= \int_{0}^{3} x \left(\frac{1}{3}\right) dx \\ &= \frac{1}{6} x^2 \Big|_0^3 \\ &= \frac{9}{6} \\ &= \frac{3}{2} \end{aligned} \tag{1}$$

(b) What is the expected variance of X?

To compute variance, $E[X]$, will be the starting point

$$Var(X) = E[(X - E[X])^2] = E[x^2] - E[X]^2$$

Next $E[X^2]$ can be computed:

$$\begin{aligned} E[x^2] &= \int_{-\infty}^{\infty} x^2 f(x) dx \\ &= \int_{0}^{3} x^2 \left(\frac{1}{3}\right) dx \\ &= \frac{1}{9} x^3 \Big|_0^3 \\ &= \frac{27}{9} \\ &= 3 \end{aligned} \tag{2}$$

Finally, the variance can be computed as follows:

$$Var(X) = E[x^2] - E[X]^2 = 3 - \left(\frac{3}{2}\right) = \frac{3}{4}$$

Question 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 4

(a) What is the mean of the given data?

$$E[X] = \frac{1}{N}\sum_i x_i = \frac{2+3+10-1-1}{5} = \frac{13}{5} = 2.6$$

(b) What is the variance of the given data?

$$\begin{aligned} Var(X) &= E[(X - E[X])^2] \\ &= E[(X - (2.6)^2)] \\ &= \frac{1}{N-1}\sum_i (x_i - (2.6)^2) \\ &= \frac{(-0.6)^2 + (0.4)^2 + (7.4)^2 + (-3.6)^2 + (-3.6)^2}{4} \\ &= 20.3 \end{aligned} \tag{3}$$

Linear Algebra

Question 5

[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.

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 5

In [ ]:
```
%matplotlib inline
import numpy as np

# Turn matrices into vectors:
A = np.array([[1, 2, 3],[2, 4, 5],[3, 5, 6]])
b = np.array([-1, 3, 8])[np.newaxis].T
c = np.array([4, -3, 6])[np.newaxis].T
I = np.identity(3)
```

1. Matrix AA

In [ ]:
```
print('[3 x 3][3 x 3]\n{}\n'.format(A.dot(A)))
```

```
[3 x 3][3 x 3]
[[14 25 31]
 [25 45 56]
 [31 56 70]]
```

1. Matrix AA^T

In [ ]:
```
print('[3 x 3][3 x 3]\n{}\n'.format(A.dot(A.T)))
```

```
[3 x 3][3 x 3]
[[14 25 31]
 [25 45 56]
 [31 56 70]]
```

1. Matrix Ab

In [ ]:
```
print('[3 x 3][3 x 1]\n{}\n'.format(A.dot(b)))
```

```
[3 x 3][3 x 1]
[[29]
 [50]
 [60]]
```

1. Matrix Ab^T

In [ ]:
```
print('[3 x 3][1 x 3]\n{}\n'.format('Cannot be computed'))
```

```
[3 x 3][1 x 3]
Cannot be computed
```

1. Matrix bA

In [ ]:
```
print('[3 x 1][3 x 3]\n{}\n'.format('Cannot be computed'))
```

```
[3 x 1][3 x 3]
Cannot be computed
```

1. Matrix b^TA

In [ ]:
```
print('[1 x 3][3 x 3]\n{}\n'.format( b.T.dot(A)))
```

```
[1 x 3][3 x 3]
[[29 50 60]]
```

1. Matrix bb

In [ ]:
```
print('[3 x 1][3 x 1]\n{}\n'.format( 'Cannot be computed'))
```

```
[3 x 1][3 x 1]
Cannot be computed
```

1. Matrix b^T b

In [ ]:
```
print('[1 x 3][3 x 1]\n{}\n'.format( b.T.dot(b)))
```

```
[1 x 3][3 x 1]
[[74]]
```

1. Matrix bb^T

In [ ]:
```
print('[3 x 1][1 x 3]\n{}\n'.format(b.dot(b.T)))
```

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

1. Matrix b + c^T

In [ ]:
```
print('[3 x 1] + [1 x 3]\n{}\n'.format('Cannot be computed'))
```

```
[3 x 1] + [1 x 3]
Cannot be computed
```

1. Matrix b^T b^T

In [ ]:
```
print('[3 x 1][3 x 1]\n{}\n'.format('Cannot be computed'))
```

```
[3 x 1][3 x 1]
Cannot be computed
```

1. Matrix A^-1 b

In [ ]:
```
print('[3 x 3][3 x 1]\n{}\n'.format(np.linalg.inv(A).dot(b)))
```

```
[3 x 3][3 x 1]
[[ 6.]
 [ 4.]
 [-5.]]
```

1. Matrix A o A

In [ ]:
```
print('[3 x 3] o [3 x 3]\n{}\n'.format(A*A))
```

```
[3 x 3] o [3 x 3]
[[ 1  4  9]
 [ 4 16 25]
 [ 9 25 36]]
```

1. Matrix b o c

In [ ]:
```
print('[3 x 3] o [3 x 3]\n{}\n'.format( b*c ))
```

```
[3 x 3] o [3 x 3]
[[-4]
 [-9]
 [48]]
```

Question 6

[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 6

Prelim: Replicate Matrix A from last question.

In [ ]:
```
#import numpy as np

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

1. Calculate the eigenvalues and corresponding eigenvectors of matrix A

In [ ]:
```
eigval, eigvec = np.linalg.eig(A)
print('Eigenvalues:\n{}\n'.format(eigval))
print('Eigenvectors:\n{}\n'.format(eigvec))
```

```
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]]
```

1. Choose one of the eigenvector/eigenvalue pairs, $\mathbf{v}$ and λ, and show that $\mathbf{Av} = \lambda\mathbf{v}$

In [ ]:
```
vec = eigvec[:,2]
val  = eigval[2]
Av = A.dot(vec)
lamv = val * vec
print('Av = \n{}\n'.format(Av))
print('Lambda v = \n{}\n'.format(lamv))
```

```
Av =
[ 0.10101242 -0.12596043  0.05605767]

Lambda v =
[ 0.10101242 -0.12596043  0.05605767]
```

1. Show that the eigenvectors are orthogonal to one another

In [ ]:
```
# Take the inner products between vectors and demonstrate that they equal zero
pairings = [(0,1),(0,2),(1,2)]
for (i,pair) in enumerate(pairings):
```

```
    inner_prod = np.inner(eigvec[:,pair[0]],eigvec[:,pair[1]])
    print('Inner product of eigenvector {} and {} = {:.4f}'.format(pair[0],pair[1],inne
```

```
Inner product of eigenvector 0 and 1 = -0.0000
Inner product of eigenvector 0 and 2 = -0.0000
Inner product of eigenvector 1 and 2 = -0.0000
```

Numerical Programming

Question 7

[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 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 type across all plants.

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

ANSWER 7

Setup

In [ ]:
```
import pandas as pd

egrid = pd.read_excel('./data/a1_egrid2016.xlsx', skiprows=[0])
```

Look at data

In [ ]:
```
egrid.describe()
```

Out[ ]:

	SEQPLT16	LAT	LON	CAPFAC	NAMEPCAP	PLNGENAN	PLCO2EQA
count	9709.000000	9668.000000	9668.000000	8038.000000	9696.000000	8.038000e+03	7.669000e+03
mean	4855.000000	38.873614	-95.260340	0.274646	156.250701	5.070070e+05	2.668082e+05
std	2802.891216	5.611971	18.820395	0.238365	409.463191	1.880041e+06	1.272092e+06
min	1.000000	18.974200	-188.551181	0.000000	0.000000	-7.686200e+05	0.000000e+00
25%	2428.000000	35.053572	-112.417822	0.064130	3.500000	2.948750e+03	0.000000e+00
50%	4855.000000	39.307100	-91.729028	0.226670	13.200000	1.721800e+04	0.000000e+00
75%	7282.000000	42.350222	-79.298845	0.408420	94.600000	1.568880e+05	5.500285e+03
max	9709.000000	71.292000	-67.003300	1.854950	6809.000000	3.237748e+07	2.172499e+07

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

Out[ ]:

	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPCAP	PLNGENA

	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPCAP	PLNGENA
0	1	AK	7-Mile Ridge Wind Project	63.210689	-143.247156	WND	NaN	1.8	Na
1	2	AK	Agrium Kenai Nitrogen Operations	60.673200	-151.378400	NG	NaN	21.6	Na
2	3	AK	Alakanuk	62.683300	-164.654400	DFO	0.05326	2.6	1213.0
3	4	AK	Allison Creek Hydro	61.084444	-146.353333	WAT	0.01547	6.5	881.0
4	5	AK	Ambler	67.087980	-157.856719	DFO	0.13657	1.1	1315.9

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

In [ ]:
```
most = egrid.sort_values(by='PLNGENAN', ascending=False)['PNAME'].iloc[0]
print(f"The plant that generated the most energy is: {most}.")
```

```
The plant that generated the most energy is: Palo Verde.
```

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

In [ ]:
```
north = egrid.sort_values(by='LAT', ascending=False)['PNAME'].iloc[0]
print(f"The northern-most plant in the US is: {north}.")
```

```
The northern-most plant in the US is: Barrow.
```

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

In [ ]:
```
state = egrid.sort_values(by='LAT', ascending=False)['PSTATABB'].iloc[0]
print(f"The state that contains the northern-most plant in the US is: {state}.")
```

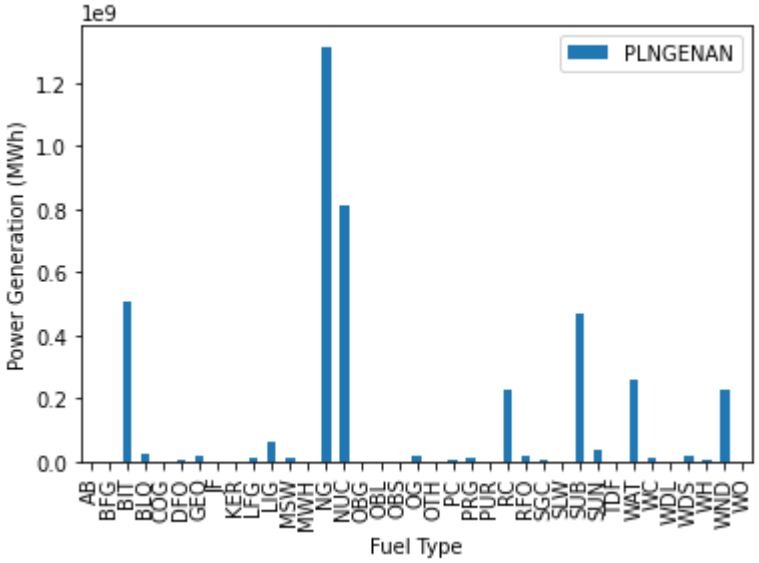
```
The state that contains the northern-most plant in the US is: AK.
```

(d) Plot a bar plot showing the amount of energy produced by each fuel type for the plants. x axis: primary fuel, y axis: power generation (in MWh).

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

fuel_type_grouping = egrid.groupby('PLPRMFL').sum()
fuel_and_power_generation    = fuel_type_grouping[['PLNGENAN']]

fuel_and_power_generation.plot.bar()
plt.ylabel('Power Generation (MWh)')
plt.xlabel('Fuel Type')
plt.show()
```



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

In [ ]:
```
fuel = fuel_and_power_generation[fuel_and_power_generation['PLNGENAN'] == fuel_and_powe
print(f"The fuel used for power generation that produces the most energy in the US is:
```

```
The fuel used for power generation that produces the most energy in the US is: NG.
```

Question 8

[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 8

Setup

In [ ]:
```
#import numpy as np
import time

# Generate random samples
x = np.random.randn(10000000)
```

For loop to compute sum of squares:

In [ ]:
```
t0 = time.time()
sum_of_squares = 0
for value in x:
    sum_of_squares += value**2
t1 = time.time()
time_nonvector = t1 - t0
print('Time [sec] (non-vectorized): {:0.4f}'.format(time_nonvector))
```

```
Time [sec] (non-vectorized): 3.9731
```

Numpy to compute sum of squares:

In [ ]:
```
t0 = time.time()
sum_of_squares = np.dot(x,x)
t1 = time.time()
time_vector = t1 - t0
print('Time [sec] (vectorized): {:0.4f}'.format(time_vector))
```

```
Time [sec] (vectorized): 0.0190
```

In [ ]:
```
print('The vectorized code is {:0.1f} times faster than the non-vectorized code'.format
```

```
The vectorized code is 209.6 times faster than the non-vectorized code
```

Question 9

[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 9

Setup

In [ ]:
```
#import numpy as np
#import matplotlib.pyplot as plt

#Set seed for reproducibility
np.random.seed(15)
```

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, $\mathbf{x}$

In [ ]:
```
x = np.random.randn(10000) + 2
print('Show some examples from x: {}'.format(x[0:3]))
```

```
Show some examples from x: [1.68767152 2.33928471 1.84409147]
```

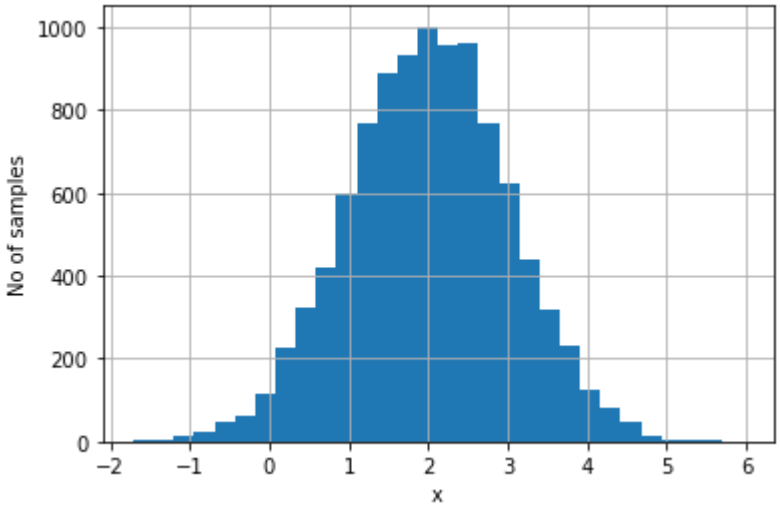
1. Calculate the mean and standard deviation of $\mathbf{x}$ to validate (1) and provide the result to a precision of four significant figures.

In [ ]:
```
print('x_mean = {:.4}'.format(x.mean()))
print('x_std  = {:.4}'.format(x.std()))
```

```
x_mean = 2.004
x_std  = 1.005
```

1. Plot a histogram of the data in $\mathbf{x}$ with 30 bins

In [ ]:
```
plt.hist(x,bins=30)
plt.xlabel('x')
plt.ylabel('No of samples')
plt.grid('on')
plt.show()
```



1. What is the 90th percentile of $\mathbf{x}$? The 90th percentile is the value below which 90% of observations can be found.

In [ ]:
```
percent90 = np.percentile(x,90)
print('The 90th percentile of x is {:.4}'.format(percent90))
```

```
The 90th percentile of x is 3.29
```

1. What is the 99th percentile of $\mathbf{x}$?

In [ ]:
```
percent99 = np.percentile(x,99)
print('The 99th percentile of x is {:.4}'.format(percent99))
```

```
The 99th percentile of x is 4.327
```

1. 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, $\mathbf{y}$.

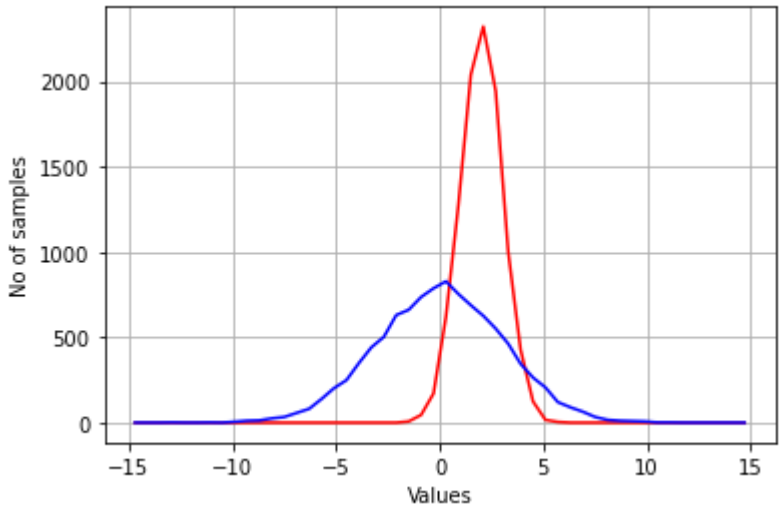
In [ ]:
```
y = 3*np.random.randn(10000)
print('Show some examples from y: {}'.format(y[0:3]))
```

```
Show some examples from y: [0.46205778 1.24814025 2.6642785 ]
```

1. 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.

In [ ]:
```
histx, binsx = np.histogram(x, bins=50, range=(-15, 15))
histy, binsy = np.histogram(y, bins=50, range=(-15, 15))
centerx = (binsx[:-1] + binsx[1:]) / 2
centery = (binsy[:-1] + binsy[1:]) / 2
plt.plot(centerx,histx,'r-',label='x')
plt.plot(centery,histy,'b-',label='y')
plt.xlabel('Values')
plt.ylabel('No of samples')
```

```
plt.grid('on')
plt.show()
```



1. Using the observations from $\mathbf{x}$ and $\mathbf{y}$, estimate $E[XY]$

In [ ]:
```
print('E[XY] = E[X]E[Y] = {:.4}'.format(x.mean() * y.mean()))
```

```
E[XY] = E[X]E[Y] = 0.05244
```

Version Control via Git

Question 10

[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 10

*I, [**Fides Schwartz**], affirm that I have [**previous experience that covers all the content in this tutorial (IDS720, Fall 2021)**]*

Exploratory Data Analysis

Question 11

[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 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. 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 11

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

Setup

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import altair as alt

chest_kernels = pd.read_excel('C:/Users/dm93/Desktop/PCCT examples/Chest/Travis_Fides_K
```

Look at data

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

Out[ ]:

	Case	Rank	Reader	Kernel
0	8	1	1	48
1	8	2	1	56
2	8	3	1	40
3	8	4	1	70
4	17	1	1	48

In [ ]:
```
#chest_kernels['Kernel'].dtype
chest_kernels.dtypes
```

Out[ ]:
```
Case      int64
Rank      int64
Reader    int64
Kernel    int64
dtype: object
```

1. Describe the dataset, the source of the data, and the reason the dataset was of interest. What question are you hoping to answer through exploring the dataset?

I am a radiologist and an associate in research with the Duke Department of Radiology. I have mostly specialized in research based on computed tomography (CT), which is an imaging method that uses x-rays to acquire 3D-imaging data. A new CT scanner system, called photon-counting CT (PCCT), was recently installed at Duke as a research system (before it got FDA-approval). Patients who received a regular clinical CT examination (e.g. to evaluate lung cancer therapy success) were

asked, if they were willing to receive a PCCT in addition to their clinical exam. If they consented, an additional CT was performed using the PCCT on the same day as the clinical CT scan.

PCCT has higher spatial and contrast resolution as well as the capability to avoid electronic noise due to photon energy thresholding. Thus, it provides the opportunity to reconstruct with sharper kernels than are currently used in clinical practice, without increasing the image noise impression to a point where it would not be clinically useful.

No study has compared the different kernel options given by the PCCT among themselves and in comparison with standard clinical CT to date. The questions of interest are:

1. Which PCCT kernel is the preferred reconstruction kernel?
2. Are PCCT kernels preferred over the standard CT kernel?
3. Do readers differ in their assessment of kernels?

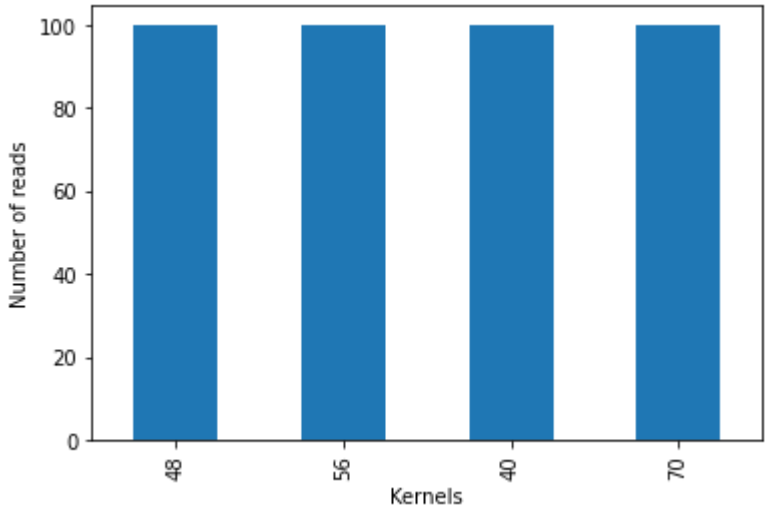
1. Check the data and see if they need to be cleaned: are there missing values? Are there clearly erroneous values? Do two tables need to be merged together? Clean the data so it can be visualized. If the data are clean, state how you know they are clean (what did you check?).

The data consists of CT image data from 20 patients, that was read by five separate readers (diagnostic radiologists from the chest and cardiovascular section). Four representative CT slices were shown on a power point slide (reconstructed with three different PCCT kernels and with one standard CT kernel at the same level in the CT scan) to each reader in a randomized order to avoid introducing bias. On each slide, the reader made a rank choice selection between the four CT slices with 1. being their most preferred reconstruction kernel and 4. being their least preferred reconstruction kernel. The data was compiled by me into one excel file from reader's notes on power point slides, so I know that there were no missing values or clearly erroneous data points. I compiled them into one file directly, so did not need to merge anything. Each reader was assigned a number (reader 1 through reader 5), each kernel was assigned a number (PCCT kernels Br40f=40, Br48f=48, Br56f=56, and standard CT kernel=70), and each case was assigned a number (1-20).

1. 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.

a) Sanity check on how often each kernel got chosen.

In [ ]:
```
%matplotlib inline
chest_kernels["Kernel"].value_counts().plot(kind="bar")
plt.ylabel('Number of reads')
plt.xlabel('Kernels')
plt.show()
```



This plot confirms that all kernels were selected once per case, i.e. 20 cases * 5 readers = 100 reads per kernel.

b) How often was which kernel chosen as 1st, 2nd, 3rd, 4th choice?

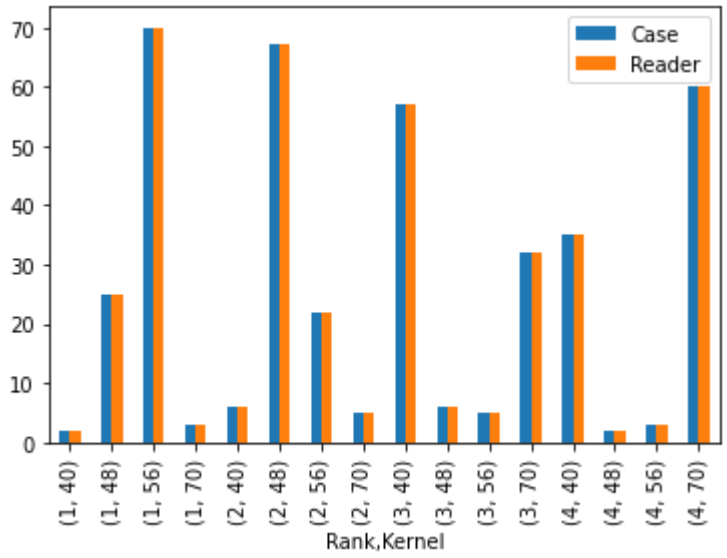
In [ ]:
```
chest_kernels_new = chest_kernels.groupby(['Rank', 'Kernel']).count()
chest_kernels_new = chest_kernels_new.drop(columns=['Case'])
chest_kernels_new
```

Out[ ]:

		Reader
Rank	**Kernel**	
1	**40**	2
	48	25
	56	70
	70	3
2	**40**	6
	48	67
	56	22
	70	5
3	**40**	57
	48	6
	56	5
	70	32
4	**40**	35
	48	2
	56	3
	70	60

In [ ]:
```
chest_kernels.groupby(['Rank', 'Kernel']).count().plot(kind="bar")
```

Out[ ]:
```
<AxesSubplot:xlabel='Rank,Kernel'>
```

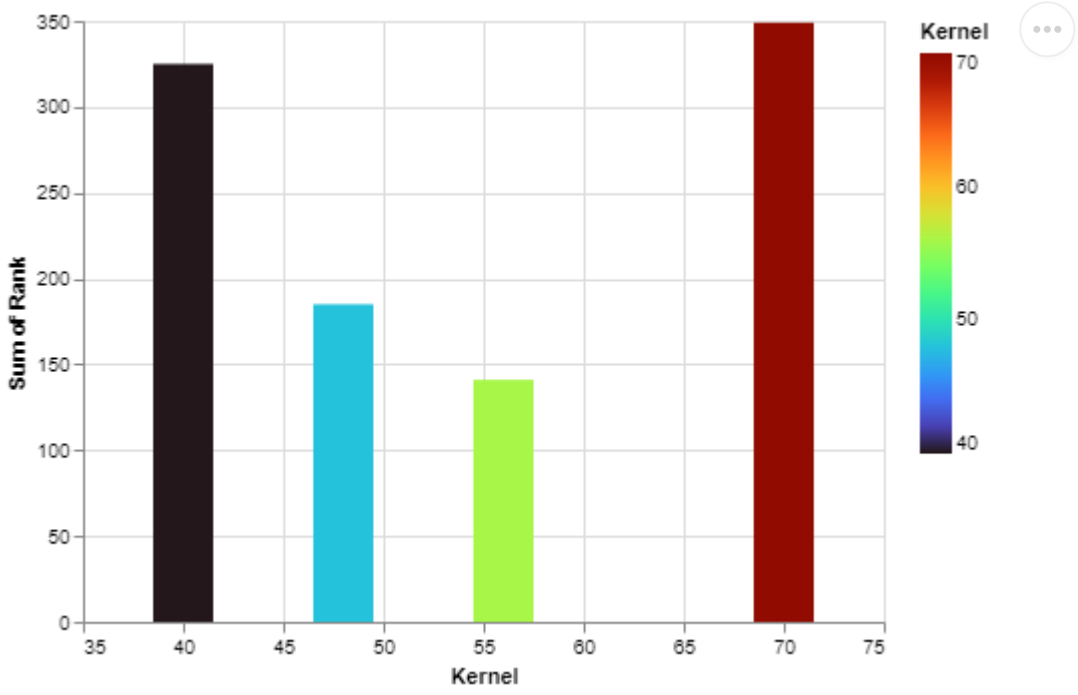


It looks like the sharpest PCCT kernel (Br56f) received the highest number of 1st choice votes. The next sharpest PCCT kernel (Br48f) received the highest number of second choice votes, PCCT kernel Br40f received the highest number of 3rd choice votes and the standard CT kernel received the highest number of 4th choice votes.

c) Plot the sum of ranks given to each kernel to see which one is lowest (most preferred).

In [ ]:
```
alt.Chart(chest_kernels).mark_bar(size=30).encode(
    alt.X('Kernel:Q',
    scale=alt.Scale(zero=False)
    ),
    y='sum(Rank):Q',
    color=alt.Color('Kernel', scale=alt.Scale(scheme='turbo'))
)
```

Out[ ]:

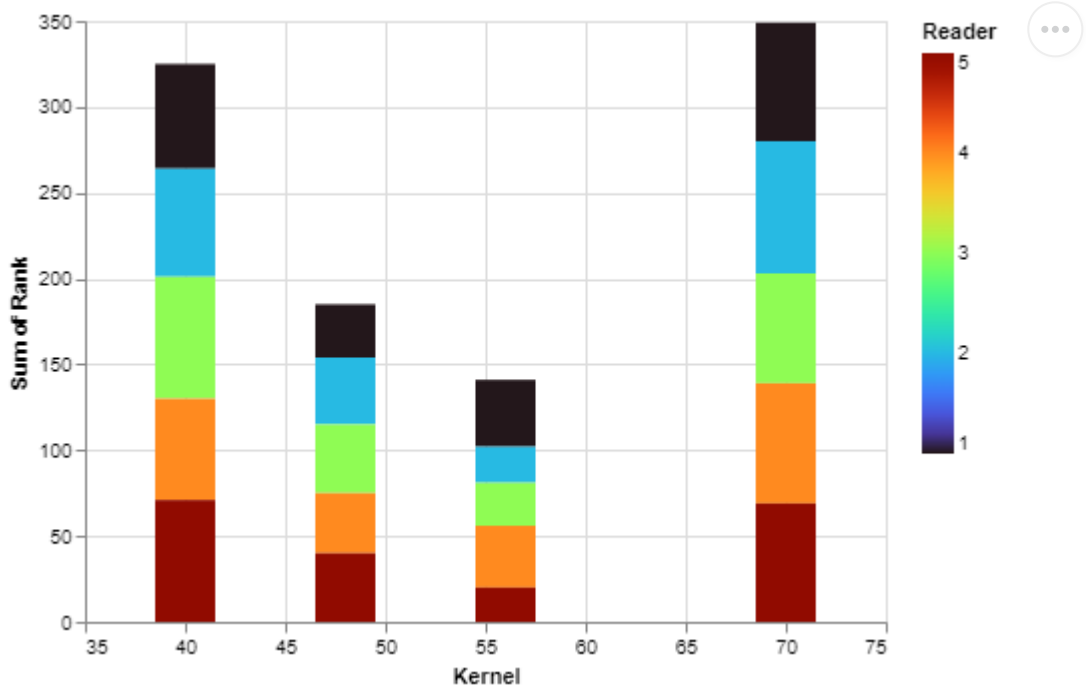


This plot shows that the overall lowest sum of ranks was achieved by the PCCT Br56f kernel (sharpest kernel), which confirms that it received the highest number of 1st and second choice votes from readers. The second most preferred kernel was the PCCT Br48f kernel, the third most preferred kernel was the PCCT Br40f kernel and the least preferred kernel, with the highest sum of ranks was the standard CT reconstruction kernel. The plots in b and c confirm that the rank order of preference is from sharpest kernel (PCCT Br56f) to smoothest kernel (standard CT kernel).

d) Sum of Ranks by reader

In [ ]:
```
alt.Chart(chest_kernels).mark_bar(size=30).encode(
    alt.X('Kernel:Q',
    scale=alt.Scale(zero=False)
    ),
    y='sum(Rank):Q',
    color=alt.Color('Reader', scale=alt.Scale(scheme='turbo'))
)
```

Out[ ]:



It looks like there are slight variations in the rank that readers assigned the different kernels with reader 1 and 4 assigning higher numbers (i.e. lower ranks) to the Br56f kernel a little bit more often than the other three readers. Otherwise the ratios seem to be very similar.

1. 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.

Exploring this data gives us answers to all three of our questions of interest.

1. Which PCCT kernel is the preferred reconstruction kernel?

The data shows a clear preference ranking of radiologists preferences for chest CT image reconstruction kernels. The majority of 1st and 2nd choice votes went to the sharpest PCCT reconstruction kernel Br56f, then the second (Br48f) and third (Br40f) sharpest kernel. "Sharp kernel" means that strutures in the lungs are depicted with a high degree of detail, which usually comes with a "noise penalty"; the images reconstructed with sharper kernels usually look more "grainy", while smoother kernels give a smoother image impression, which is often preferred by radiologists. The fact that five separate readers preferred the sharpest kernel for reconstruction of chest CT images, points to an improvement provided by PCCT in reducing noise while providing higher spatial resolution and thus better depicting details in the lung structures.

2. Are PCCT kernels preferred over the standard CT kernel?

The current standard chest CT reconstruction kernel used in clinical practice with existing CT scanners was the least preferred for all readers in almost all cases. This indicates that imaging with the new scanner system and reconstructing with a kernel that provides sharper image detail would be preferred by most radiologists.

3. Do readers differ in their assessment of kernels?

There seems to be little variation in ranking of kernels depending on readers. Though two readers gave slightly higher scores (less preference) to the Br56f kernel, the rest of the kernels received very similar scores from all readers. This slight variation might be due to the relatively low number of cases shown to the readers or possibly differences in reader residency training (institutions vary in their approach to reconstruction kernel selection for lung imaging).

Limitations and overall conclusions:

This exploratory data analysis and preliminary study, of course, cannot make any claims about improvement in diagnostic accuracy using the new scanner system or sharper reconstruction kernels. It may inform the choice of standard reconstructions performed on the CT scanner and sent to the picture archiving and communication system (PACS), once this scanner is in clinical use. Based on the choices of five readers from our institution the best choice of standard reconstruction for chest images on the new photon counting CT scanner that get sent to PACS and read clinically, would be the Br56f kernel.