Homework 1

Kyle Hadley

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
import warnings
warnings.simplefilter('ignore')
```

1. Simpson's Paradox

Given,

Machine 1	Wins	Losses
You	40	60
Friend	30	70

Machine 2	Wins	Losses
You	210	830
Friend	14	70

1(a).

To calculate the winning probability on a given machine, we use the following equation,

$$Pr_A(W_x) = \frac{w_x}{p_x}$$

where $Pr_A(W_x)$ is the probability of person A winning on machine x, w_x is the number of wins on machine x, and p_x is the number of games played on machine x.

Using the above equation, we can calculate the probability of my friend and I winning on each machine and output.

The probability of myself winning on Machines 1 and 2 are,

$$Pr_{me}(W_1) = \frac{40}{40+60} = 0.4$$

$$Pr_{me}(W_2) = \frac{210}{210+830} \approx 0.202$$

The probability of my friend winning on Machines 1 and 2 are,

$$Pr_{friend}(W_1) = \frac{30}{30+70} = 0.3$$

$$Pr_{friend}(W_2) = \frac{14}{14+70} \approx 0.167$$

On both machines and given the dataset provided, **I** am more likely to win on both Machine 1 and Machine 2.

1(b).

To calculate the winning probability in the entire casino, we use the following equation,

$$Pr_A(W) = \sum_{x=1}^{n} \left(Pr_A(W_x) * \frac{p_x}{p_T} \right)$$

where $Pr_A(W)$ is the probability of person A winning in the casino, p_T is total number of games played in the casino, p_x is the number of games played on a given machine x, and n is the total number of machines played while in the casino.

Using the above equation, we can calculate the probability of my friend and I winning in the casino.

The probability of myself winning is,

$$Pr_{me}(W) = 0.400 * \frac{100}{100+1040} + 0.202 * \frac{1040}{100+1040} \approx 0.219$$

The probability of my friend winning on Machines 1 and 2 are,

$$Pr_{friend}(W) = 0.300 * \frac{100}{100+84} + 0.167 * \frac{84}{100+84} \approx 0.239$$

Given the dataset provided, **my friend** is more likely to win in the casino.

1(c).

In looking at the data, we can see that my friend is more likely to win in the casino despite having a lower probability of winning on either machine 1 or machine 2 - when assessed individually.

When reviewing the equation from part (b), we see that the total probability of winning in the casino is a weight averaged of the probability of winning on a machine machine where the average function is the number of games played on a machine.

For my games played, I played a significantly higher number of games played on the machine with the lower probability of winning. For my friend's games played, he played a higher number of games played on the machine with the higher probability of winning. Thus, his total winning percentage will be closer to the winning percentage from machine 1 (which is the higher percentage) and my winning percentage will be closer to the winning percentage from machine 2 (which is the lower percentage).

1(d).

The conclusions of (a) and (b) could be the same when $Pr_{friend}(W) \leq Pr_{me}(W) \approx 0.219$. By inspection, we can see that we need to have my friend play more games on machine 2 to decrease his winning probability; thus we will only let him keep playing on machine 2.

Solving the equality given above, and assuming that the previously winning percentages for myself on both machines and my friend on machine 1 are still valid,

$$Pr_{friend}(W) = \sum_{x=1}^{n} \left(Pr_A(W_x) * \frac{p_x}{p_T} \right) = Pr_{me}(W)$$

$$Pr_{friend}(W_1) * \frac{p_1}{p_T} + Pr_{friend}(W_2) * \frac{p_2}{p_T} = Pr_{me}(W)$$

$$\frac{Pr_{friend}(W_1)p_1 + Pr_{friend}(W_2)p_2}{p_T} = Pr_{me}(W)$$

Given that $p_T = p_1 + p_2$ and solving for p_2 (using approximations),

$$p_2 = p_1 \left(\frac{Pr_{me}(W) - Pr_{friend}(W_1)}{Pr_{friend}(W_2) - Pr_{me}(W)} \right)$$

$$p_2 = 100 \left(\frac{0.219 - .3}{.167 - .219} \right)$$

$$p_2 \approx 155.76$$

Thus, we see that in order for conclusions of (a) and (b) to be true (i.e. I have the highest winning percentage on both machines and in the casino), my friend would have to have played at least 156 games on machine 2.

2. Linear Algebra Refresher

Given $a_1 = (1, 0)$, $a_2 = (0, 1)$, $b_1 = (2, 0)$, and $b_2 = (3, 2)$.

2(a).

We can solve for the matrix W by solving the relationships in which W transforms a_1 to b_1, and a_2 to b_2,

$$a_1 W = b_1$$

$$a_2 W = b_2$$

We define W as $W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$, since we given that it is a 2×2 matrix.

Solving the first equation, we find

$$a_1 W = b_1$$

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} 2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} W_{11} & W_{21} \end{bmatrix} = \begin{bmatrix} 2 & 0 \end{bmatrix}$$

Solving the second equation, we find

$$a_2 W = b_2$$

$$\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} 3 & 2 \end{bmatrix}$$

$$\begin{bmatrix} W_{12} & W_{22} \end{bmatrix} = \begin{bmatrix} 3 & 2 \end{bmatrix}$$

Thus, we can define a matrix $W = \begin{bmatrix} 2 & 3 \\ 0 & 2 \end{bmatrix}$ such that W transforms a_1 to b_1, and a_2 to b_2.

2(b).

We can define a rotation matrix V such that it rotates clockwise by α degrees as follows,

$$V = \begin{bmatrix} \cos(-\alpha) & -\sin(-\alpha) \\ \sin(-\alpha) & \cos(-\alpha) \end{bmatrix}$$

where $\alpha = \tan^{-1}(2)$.

We can define a scaling matrix Σ such that it scales the x-axis by 4 as follows,

$$\Sigma = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

We can define a rotation matrix U such that it rotates clockwise by β degrees as follows,

$$U = \begin{bmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{bmatrix}$$

where $\beta = \tan^{-1}(\frac{1}{2})$.

Multipling the three matrices together, namely $U\Sigma V$, symbolically the resulting matrix is as follows,

$$U\Sigma V = \begin{bmatrix} \cos(-\alpha) & -\sin(-\alpha) \\ \sin(-\alpha) & \cos(-\alpha) \end{bmatrix} \cdot \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{bmatrix}$$

$$U\Sigma V = \begin{bmatrix} 4\cos(-\alpha) & -\sin(-\alpha) \\ 4\sin(-\alpha) & \cos(-\alpha) \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{bmatrix}$$

$$U\Sigma V = \begin{bmatrix} 4\cos(-\alpha)\cos(\beta) - \sin(-\alpha)\sin(\beta) & -4\cos(-\alpha)\sin(\beta) - \sin(-\alpha)\cos(\beta) \\ 4\sin(-\alpha)\cos(\beta) + \cos(-\alpha)\sin(\beta) & -4\sin(-\alpha)\sin(\beta) + \cos(-\alpha)\cos(\beta) \end{bmatrix}$$

When we solve the matrix (as done below in Python) with $\alpha = \tan^{-1}(2)$ and $\beta = \tan^{-1}(\frac{1}{2})$, we find that $U\Sigma V = \begin{bmatrix} 2 & 3 \\ 0 & 2 \end{bmatrix} = W$.

In [3]:
```
# Define angles alpha and beta
a = np.arctan(2)
b = np.arctan(1/2)
```

```
# Define matrix V, E, and U
V = np.array([[np.cos(-a), -np.sin(-a)], [np.sin(-a), np.cos(-a)]])
E = np.array([[4,0],[0,1]])
U = np.array([[np.cos(b), -np.sin(b)], [np.sin(b), np.cos(b)]])

# Calculate and print the summation of the 3 transformation matrics.
UEV = U.dot(E.dot(V))
print(UEV)
```

```
[[2.00000000e+00 3.00000000e+00]
 [2.22044605e-16 2.00000000e+00]]
```

2(c).

We start by defining the matrix W^TW as

$$W^TW = \begin{bmatrix} 2 & 3 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 6 & 13 \end{bmatrix}$$

To calculate the eigenvalues, we use the equation $|W^TW - \lambda I| = 0$ and solve for λ.

$$|W^TW - \lambda I| = 0$$

$$\left| \begin{bmatrix} 4 & 6 \\ 6 & 13 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right| = 0$$

$$\left| \begin{bmatrix} 4-\lambda & 6 \\ 6 & 13-\lambda \end{bmatrix} \right| = 0$$

$$(4-\lambda)(13-\lambda) = 0$$

$$\lambda^2 - 17\lambda + 16 = 0$$

$$(\lambda - 1)(\lambda - 16) = 0$$

Thus, we find that $\lambda_1 = 1$ and $\lambda_2 = 16$.

Next, we can find corresponding eigenvectors using the equation $W^TWv_x = \lambda_x v_x$ where v_x is the eigenvector for λ_x.

For $\lambda_1 = 1$,

$$W^TWv_1 = \lambda_1 v_1$$

$$(W^TW - \lambda_1)v_1 = 0$$

$$\begin{bmatrix} 4-\lambda_1 & 6 \\ 6 & 13-\lambda_1 \end{bmatrix} v_1 = 0$$

$$\begin{bmatrix} 3 & 6 \\ 6 & 12 \end{bmatrix} v_1 = 0$$

From this equation, we find the relationship

$$3v_{1,1} + 6v_{1,2} = 0$$

$$v_{1,1} = -2v_{1,2}$$

So, we can determine that the eigenvector for λ_1 is any non-zero multiple of $(-2, 1)$.

For $\lambda_2 = 16$,

$$W^T W v_2 = \lambda_2 v_2$$

$$(W^T W - \lambda_2) v_2 = 0$$

$$\begin{bmatrix} 4 - \lambda_2 & 6 \\ 6 & 13 - \lambda_2 \end{bmatrix} v_2 = 0$$

$$\begin{bmatrix} -12 & 6 \\ 6 & -3 \end{bmatrix} v_2 = 0$$

From this equation, we find the relationship

$$-12v_{2,1} + 6v_{2,2} = 0$$

$$2v_{2,1} = v_{2,2}$$

So, we can determine that the eigenvector for λ_2 is any non-zero multiple of $(1, 2)$.

Thus, we find that the eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = 16$ with corresponding eigenvectors $v_1 = (-2, 1)$ and $v_2 = (1, 2)$. This is verified below in the python snippet.

If we were to apply the transformation W to every point on the unit circle, we would see the circle stretched along the vector b_2 by a factor of 4.

It seems that the eigenvalues represent the stretch factor along the axeses defined by the eigenvectors. For example, a stretch factor of $\sqrt{\lambda_1}$ (4) is applied along the axis defined by v_1 $((1, 2)$).

In [4]:

```
# Define our matrix W, print out the matrix W^T*W
W = np.array([[2, 3],[0, 2]])
print(W.transpose().dot(W))

# Calculate and print the eigenvalues and eigenvecotrs for the matrix W^T*W for verific
print(np.linalg.eig(W.transpose().dot(W)))
```

```
[[ 4  6]
 [ 6 13]]
(array([ 1., 16.]), array([[-0.89442719, -0.4472136 ],
       [ 0.4472136 , -0.89442719]]))
```

2(d).

The determinant of W is calculated as $det(W) = (4)(13) - (6)(6) = 4$.

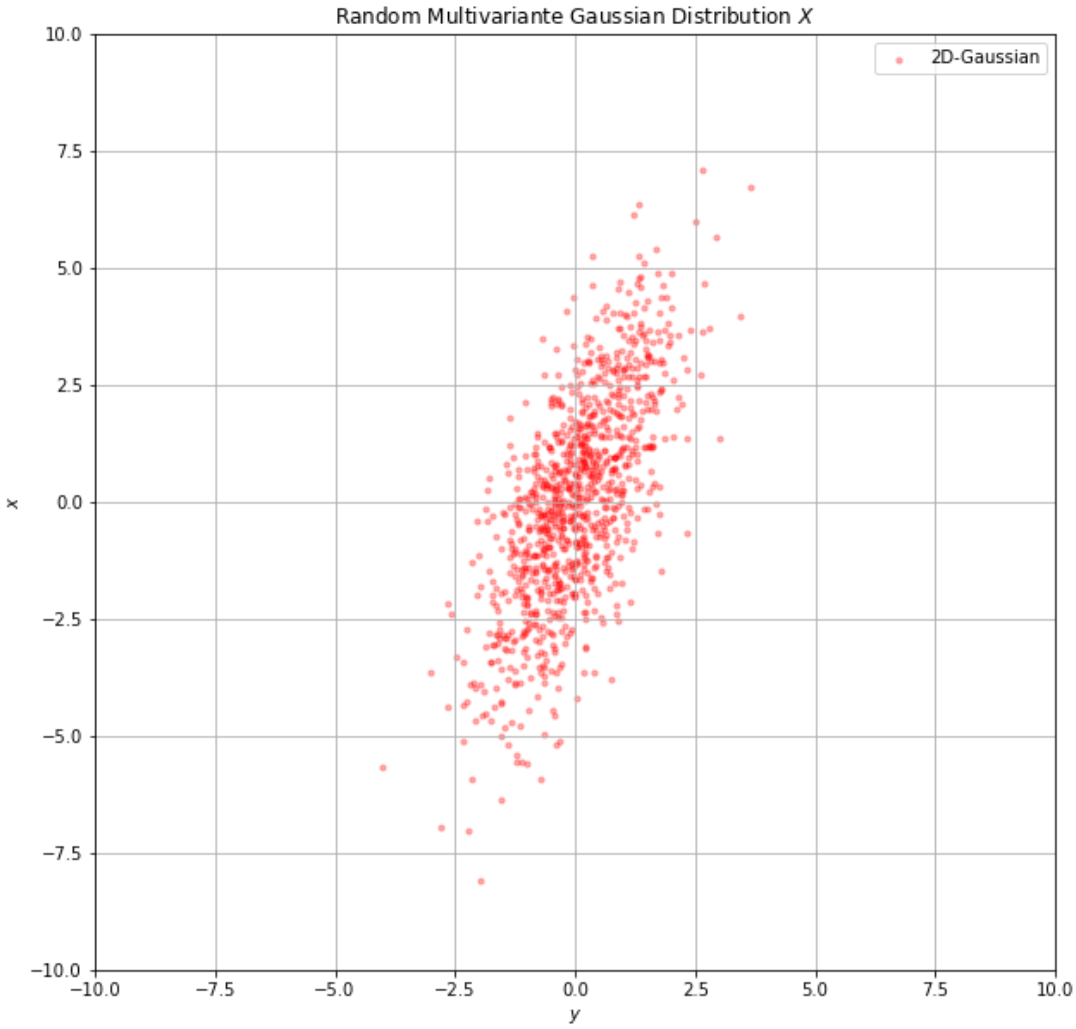
The area of the shape transformed from the unit circle is equal to 4π. It can then by hypothesized that area of a shape when transformed by a X is scaled by a factor equal to the determinant of X.

Given the hypothesis that the area of a shape when transformed by a given matrix is scaled by a factor equivalent to the determinant of the given matrix, and given that applying a transformation AB to a shape is equivalent to applying the transformation A then B, it must be that the $det(AB) = det(A)det(B)$.

3. Programming Problem: Density Estimation and Multivariate Gaussian

3(a).

Using the provided code, I generated a 1000 points as captured within X as sampled from a multivariante Gaussian distribution. A plot of these 1000 points are shown in the figure below.



We can estimate the mean and covariance of X using *numpy.mean* and *numpy.cov*. Using these functions we find the mean and covariance of X to be:

Mean of X: $\begin{bmatrix} 0.0104 & 0.0612 \end{bmatrix}$

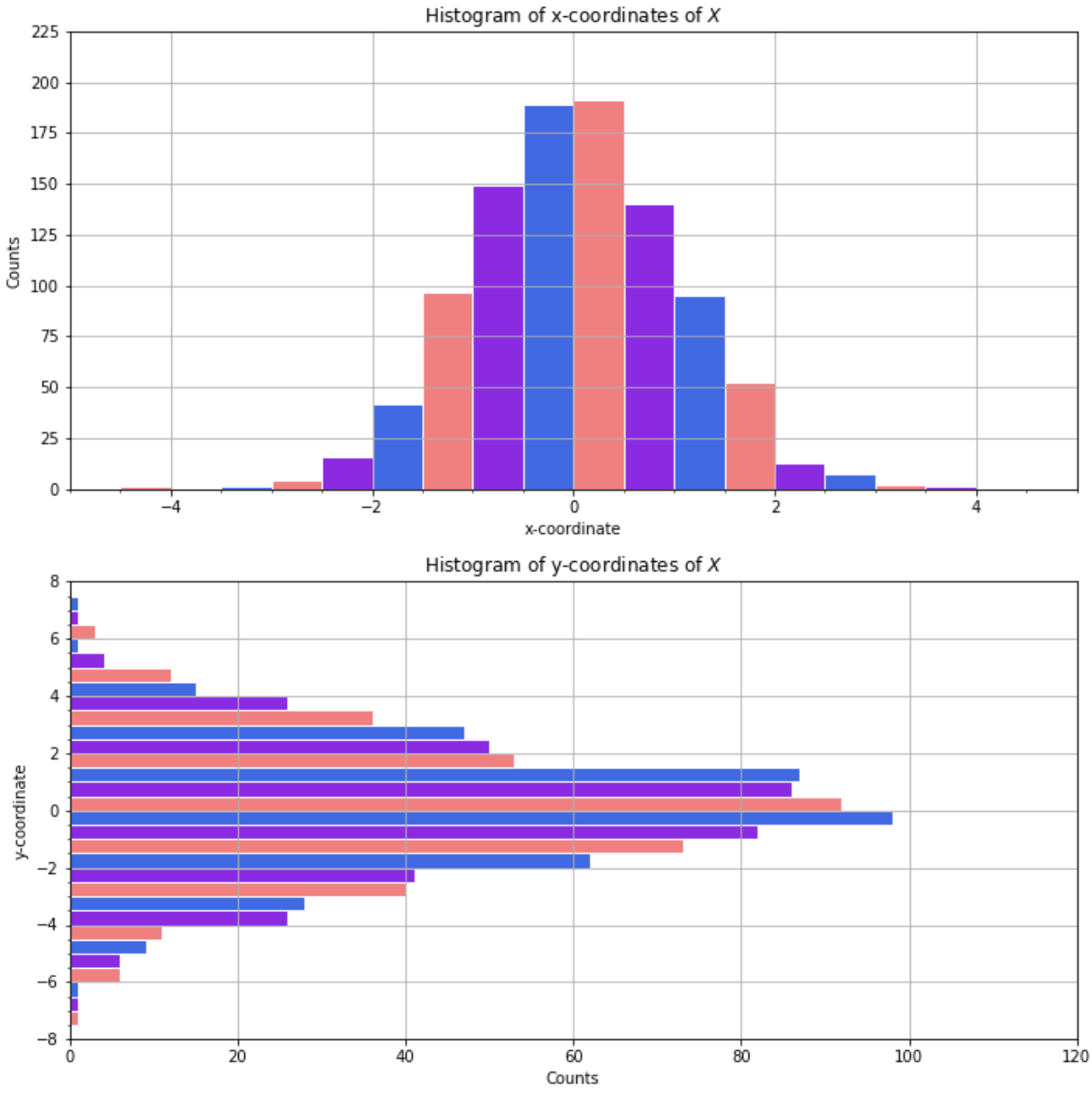
Covariance of X:

$$\begin{bmatrix} 4.2780 & 2.4099 & 2.0527 & \dots & 2.9792 & 2.196 & 3.0690 \\ \dots & & & & & & \dots \\ 3.0690 & 1.7288 & 1.4726 & \dots & 2.1373 & 1.5755 & 2.2017 \end{bmatrix}$$

Note: The covariance matrix is the size $1000x1000$, thus only a sub-sample is reported out.

3(b).

A plot of the histogram for x-coordinates and y-coordinates of X are shown in the figure below.



3(c).

From the histogram, we can see that the x-coordinates follow a seemingly normal Gaussian distribution. Thus, we can estimate a mean and variance using the following equation:

$$\bar{x} = \frac{1}{n}\sum_{i=0}^{n}(x_i * h_i)$$

where n is the total number of bins in our histogram, x_i is the midpoint of the bin with respect to x, and h_i is the height of the bin (i.e. the count within the specified bin).

We can estimate the variance ($sigma^2$) using the following equation:

$$\sigma^2 = \frac{1}{n}\sum_{i=0}^{n}(x_i - \bar{x})^2$$

where n is the total number of bins in our histogram, x_i is the midpoint of the bin with respect to x.

Given these equations, we can solve for these values in Python such that $\bar{x} = 0.013$ and $\sigma_x^2 = 1.0678$.

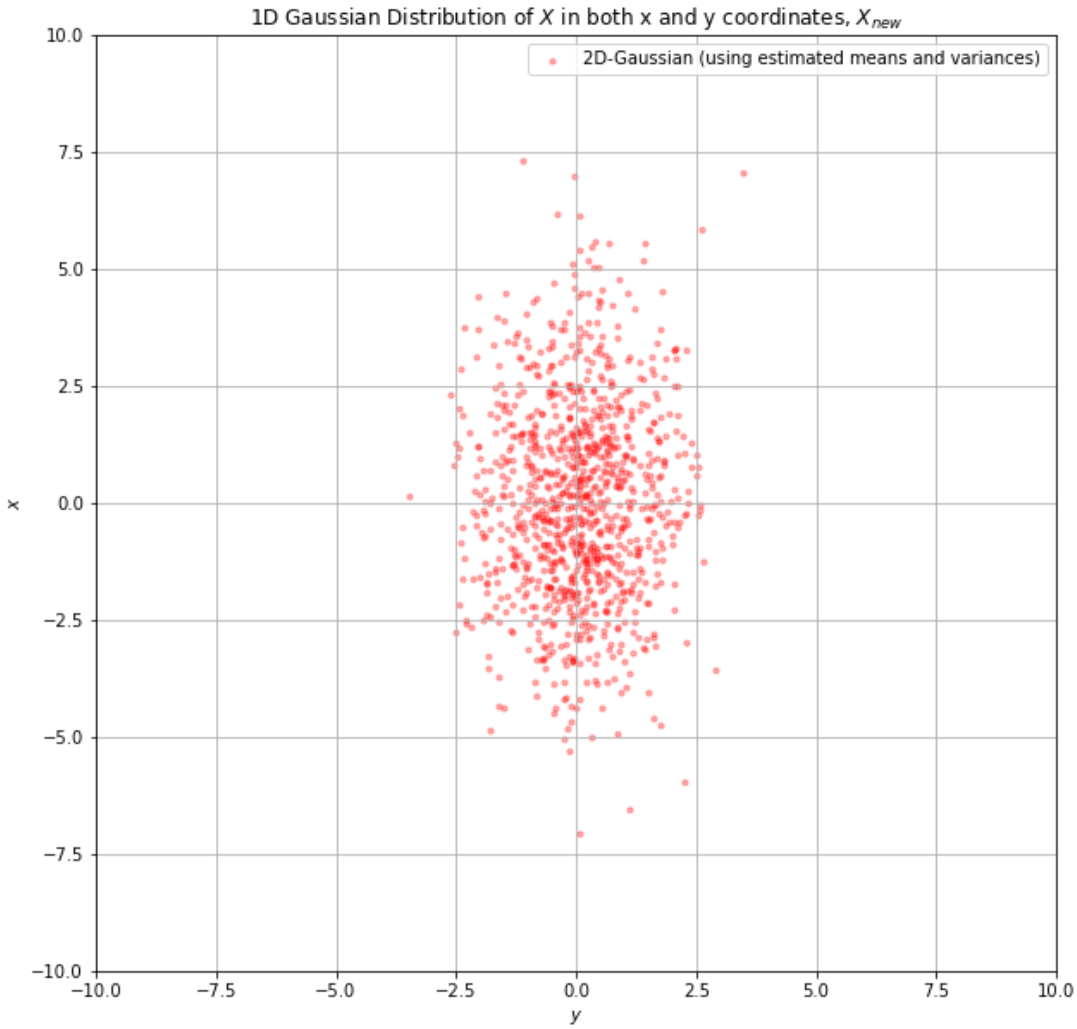
From the histogram, we can also see that the y-coordinates follow a seemingly normal Gaussian distribution. Thus we can use the same equations, but insteading solve with respect to the y-coordinate. We find that $\bar{y} = 0.07325$ and $\sigma_y^2 = 5.0321$.

3(d).

Using the mean and covariance values calculated, a new set of 1000 points using a 1D Gaussian distribution for both x and y-coordinates is generated. The plot generated is shown below.

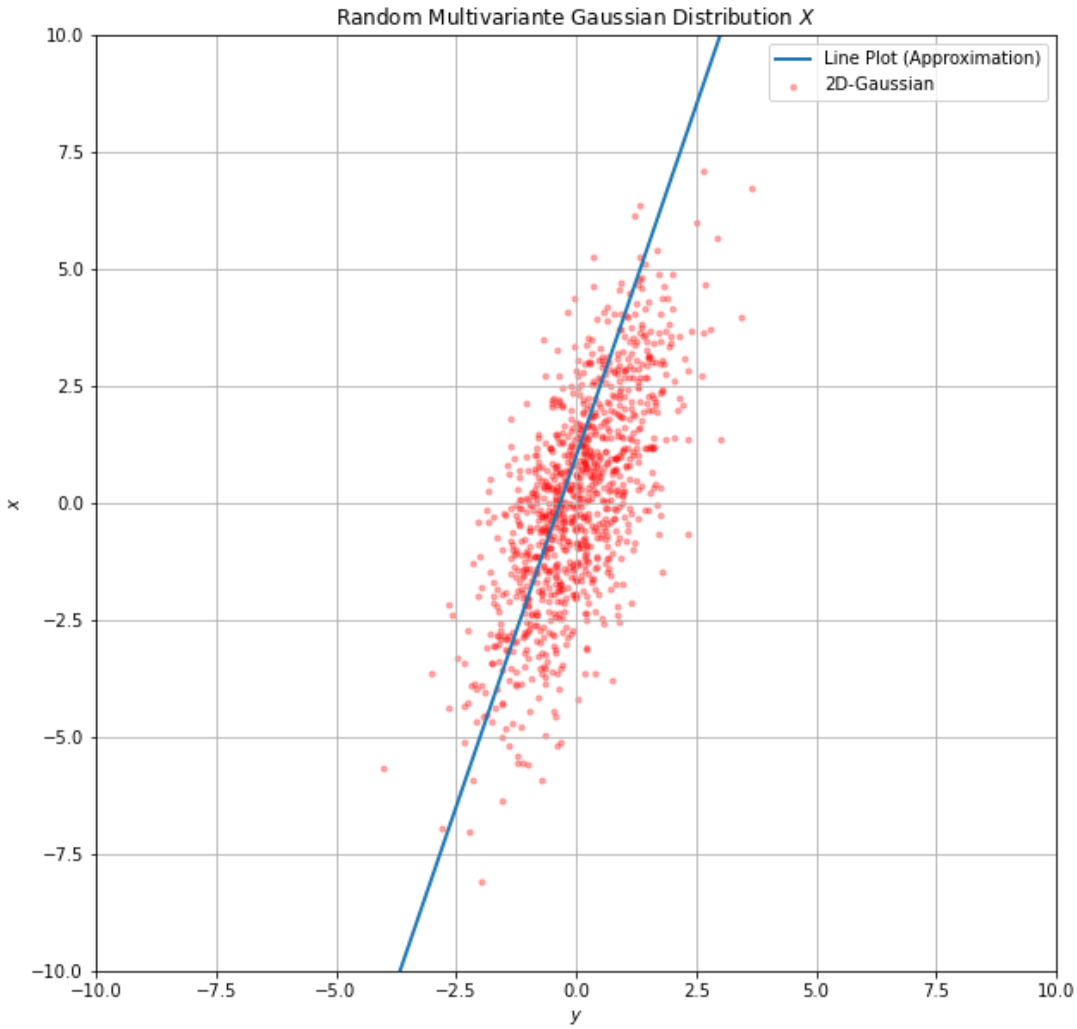
When comparing the original 1000 points and the new distribution, the original distribution skewed. We can estimate the it is skewed along the line $y = 3x + 1$ based on inferences seen in part (e) and (f).

This lack of skewing in our new dataset is driven by the way we calcualte our points within our new dataset. When we calculate our normal and variances, we decouple the relationship between our x-coordinate and y-coordinates. Rather than maintaing the relationship in which it is more likely to have a larger y-coordinate with a higher x-coordinate, we are simply capturing the mean and variance of our x and y independent of one another. Thus, when we re-couple the two values we simply see a clumping of our point about the $x = 0$ and $y = 0$ axises.



3(e).

Generating the line segment $y = 3x + 1$ and plotting this line with respect to the dataset X is shown in the plot below.

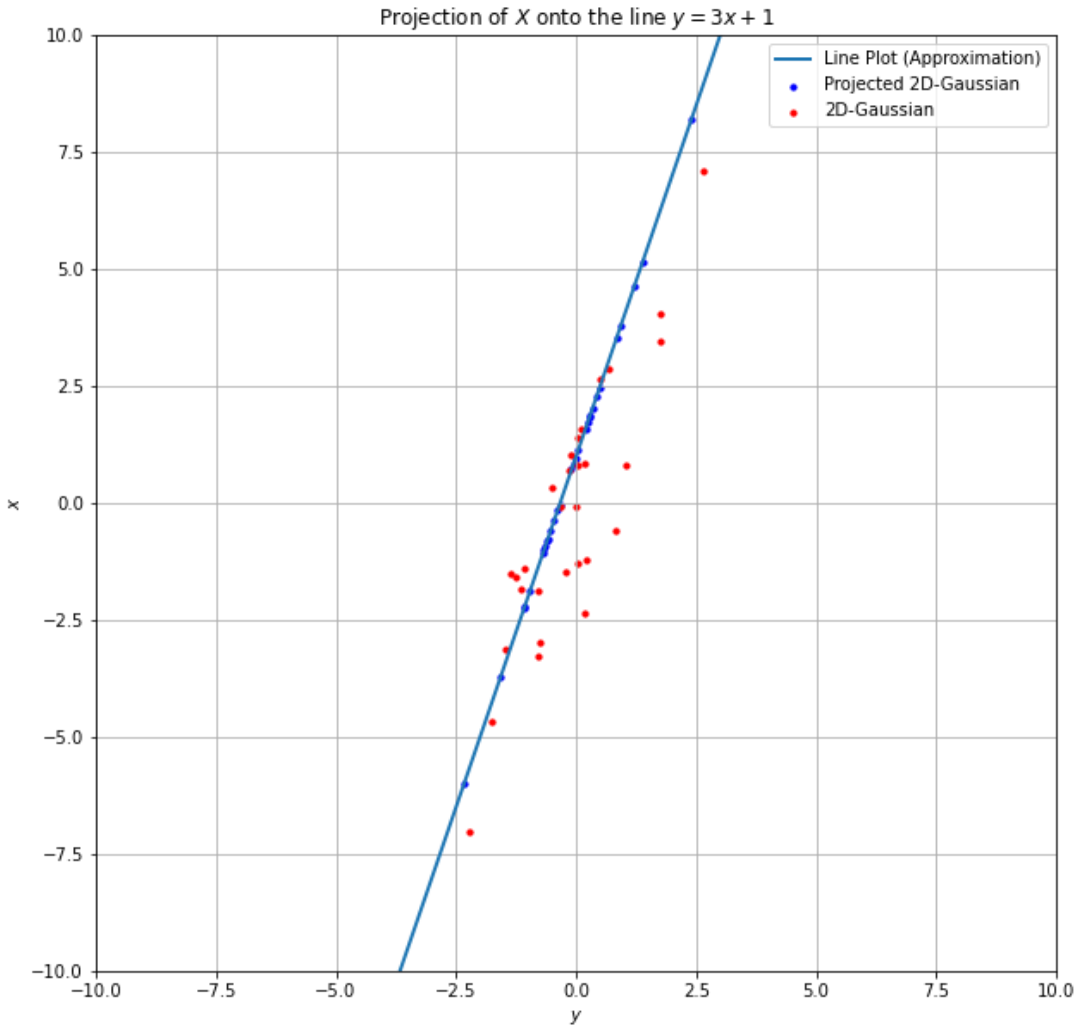


To calculate the projection of X onto the line $y = 3x + 1$, we can use the equation for projection using vector notation as

$$proj_{\vec{y}}\vec{X} = \frac{\vec{y} \cdot \vec{X}}{\vec{y} \cdot \vec{y}}\vec{y} + y_{origin}$$

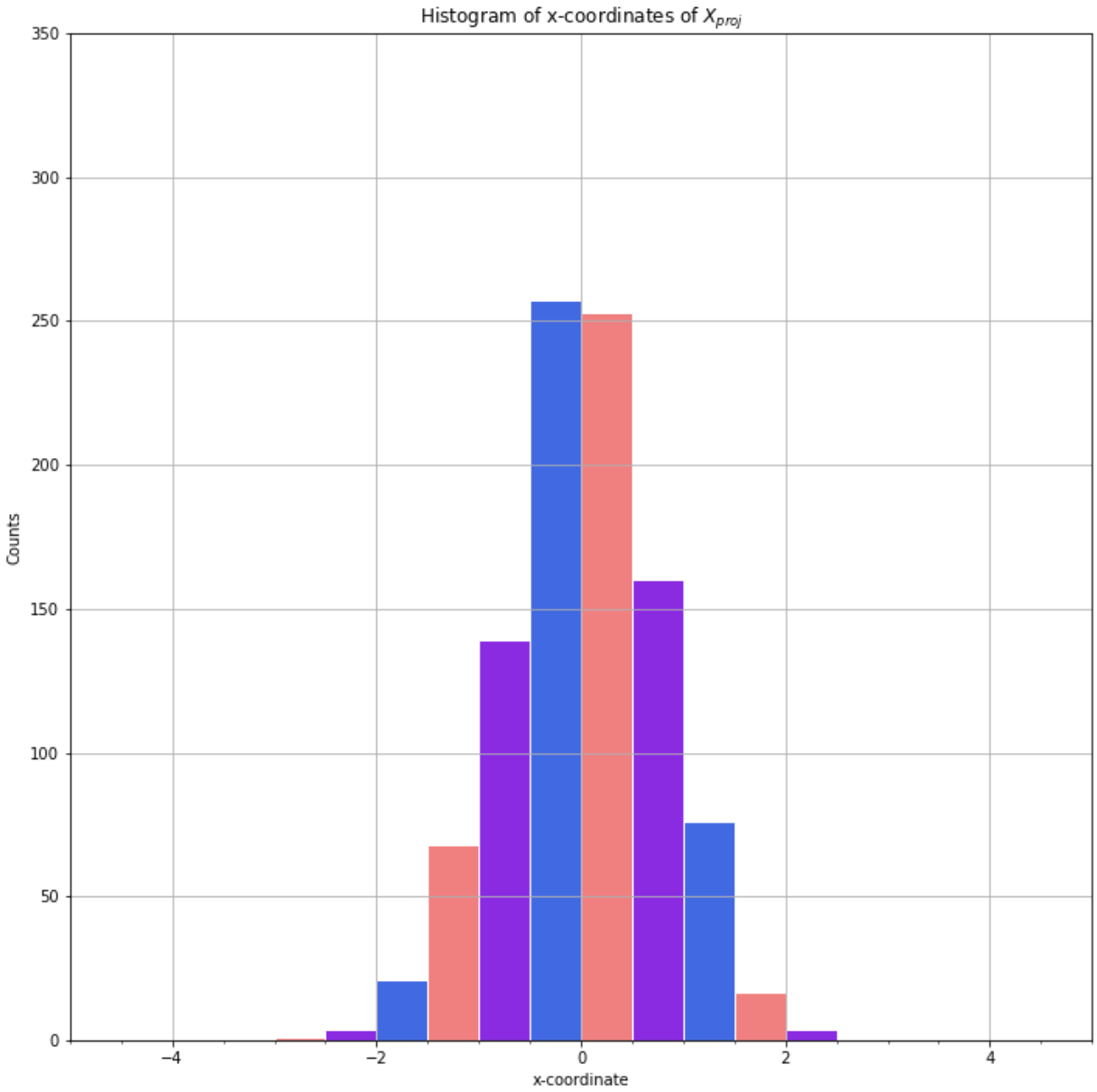
where $\vec{y} = (1, 3)$, $y_{origin} = (0, 1)$, and $\vec{X}$ is the vector notation for each point in our data.

Using this equation, we project all of X onto the line y and can demonstrate that this projection is successful by plotting a subset of the data (30 points) as shown below.



3(f).

We can now plot the histogram of the projection points, X_proj, as shown in the plot below.



Based on the histogram, we can see that the points follow a normal Gaussian distribution as seen in part (c). Given those equations, we can solve for mean and variance in Python such that $\bar{x}_{proj} = 0.015$ and $\sigma^2_{x,proj} = 0.589279$.