1. Linear Algebra

```
[1] # 1. Scalar (dot) product:
    import numpy as np
    x = np.array([1, 5])
    y = np.array([2, 3])
    a = np.dot(x, y) # 1x2 + 5x3 = 17
    print(a)

    17
```

This calculates the dot product of the given x and y.

```
# 2. Linear combination of two vectors:
d = 2*x + 2*y
print(d)

[ 6 16]
```

2*x + 2*y = 2[1, 5] + 2[2, 3] = [6. 16]

```
# 3. Norm of a vector:
x = np.array([1, 2, 3])
b = np.sqrt(x[0]**2 + x[1]**2 + x[2]**2)
c = np.linalg.norm(x)
print(b, c)

3.7416573867739413 3.7416573867739413
```

Both b and c have the same value

```
[3] # 4. Angle between two non-zero vectors:
    x = np.array([1, 5])
    y = np.array([2, 3])
    theta_radians = np.arccos(np.dot(x, y)/(np.linalg.norm(x)*np.linalg.norm(y)))
    theta_degrees = np.degrees(theta_radians)
    print(theta_degrees)

    22.380135051959567
```

First, calculate the angle in radians and convert it into degrees.

```
[4] # 5. Matrix vector multiplication:
    A = np.array([[1, -1, 2], [0, -3, 4]], dtype=float)
    u = np.array([2, 1, 3])
    v = np.dot(A, u)
    print(v)

    [7. 9.]
```

Here we multiply the matrix with a vector.

```
# 6. Solve a linear matrix equation, or system of linear scalar equations:
d = np.array([[2, 5], [3, -5]])
e = np.array([3, 2])
results = np.linalg.solve(d, e)
print(results)

[1.  0.2]
```

We have two linear matrix equations and linalg.slove can directly solve them.

```
[ ] # 7. Inverse of a matrix:
    I = np.linalg.inv(d)
    print(I)

    [[ 0.2   0.2 ]
     [ 0.12 -0.08]]
```

linalg.inv() finds the inverse matrix.

```
# 8. Trace of a matrix
B = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
trace = np.trace(B)
print(trace)
```

```
15
```

Trace is the sum of its diagonal elements.

```
# 9. Determinant of a matrix: det(B) or |B|
B_det = np.linalg.det(B)
print(B_det)
```

```
0.0
```

Linalg.det() finds the determinant of a matrix

```
# 10. Eigenvalues and Eigenvectors:
a, M = np.linalg.eig(B)
print(a)
print(M)
print(np.dot(M[:,0], M[:,1]))
print(M @ M.T)
```

```
[ 1.61168440e+01 -1.11684397e+00 -1.30367773e-15]
[[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
-0.27343437080986494
[[ 0.83800623 -0.14330218 -0.12461059]
 [-0.14330218  0.95015576  0.04361371]
 [-0.12461059  0.04361371  1.21183801]]
```

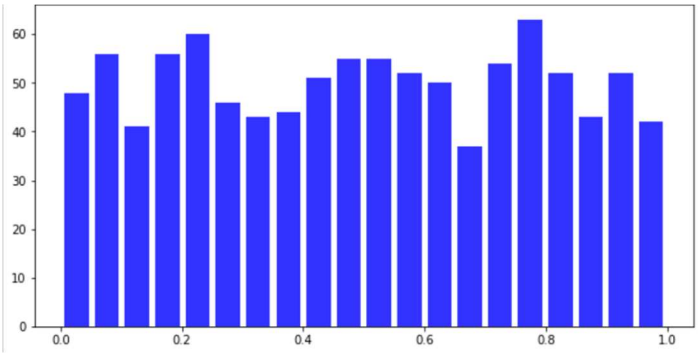
11. @ is a binary operator used for matrix multiplication. It operates on two matrices, and in general, N-dimensional NumPy arrays, and returns the product matrix. The product of a square matrix with its transpose is always symmetric and always real symmetric matrices have eigenvectors and values. $(A^tA)^t = (A^t)^tA^t = A^tA$

And their nonnegative eigenvalues and vectors are the same as the singular values and vectors since they are orthogonal.

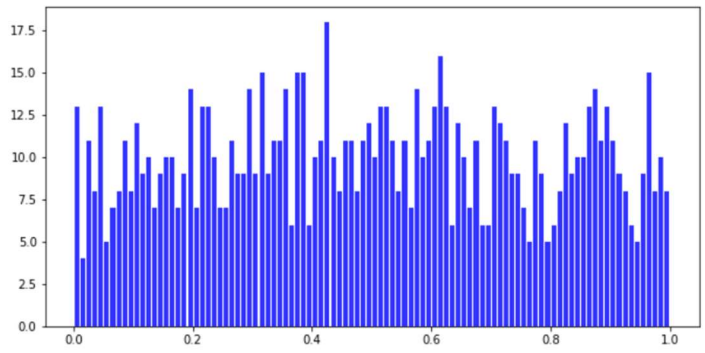
2. Random Numbers and Univariate Distributions

Histograms of 1000 uniform random numbers

20 bins



100 bins

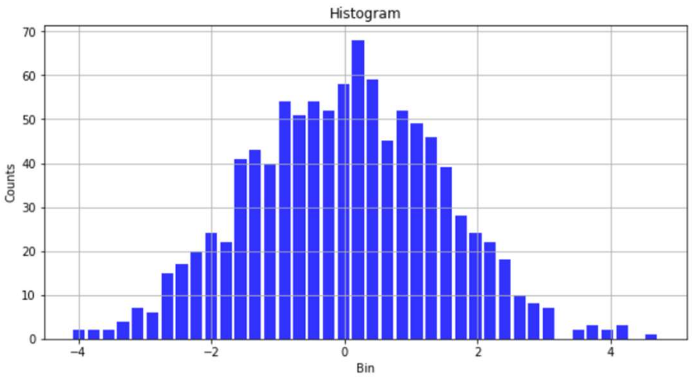



```
Variation within bin counts:  0.0003150000000000007
Variation within bin counts:  0.00019
Variation within bin counts:  0.00013
Variation within bin counts:  0.000185
Variation within bin counts:  0.0001425000000000002
Variation within bin counts:  0.00020999999999999998
Variation within bin counts:  0.0001325
Variation within bin counts:  0.000265
Variation within bin counts:  0.0002250000000000002
Variation within bin counts:  0.0003775
```

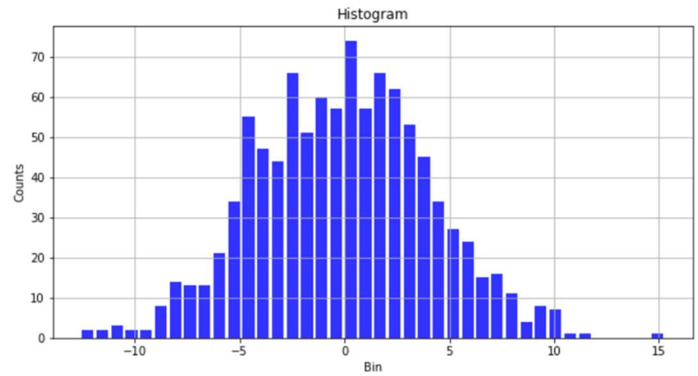
```
Variation within bin counts:  4.4999999999999996e-05
Variation within bin counts:  5e-05
Variation within bin counts:  4.9500000000000004e-05
Variation within bin counts:  5.800000000000001e-05
Variation within bin counts:  5.2000000000000004e-05
Variation within bin counts:  4.4000000000000006e-05
Variation within bin counts:  4.7499999999999996e-05
Variation within bin counts:  5.3499999999999986e-05
Variation within bin counts:  5.1000000000000006e-05
Variation within bin counts:  3.999999999999999e-05
```

When the number of bin counts goes up, the variance of the histogram goes down.

12 numbers

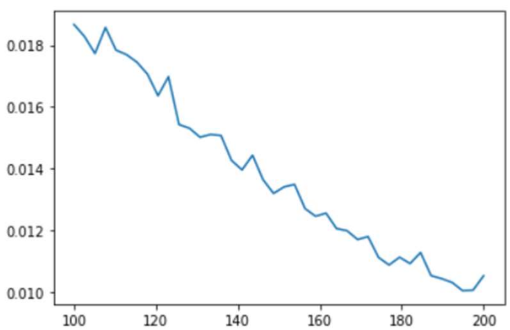


100 numbers



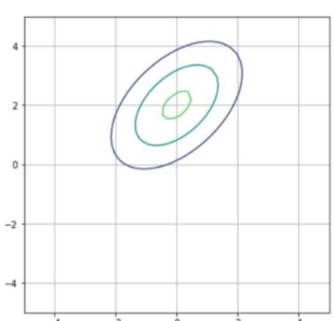
When adding two Gaussian distributions resultant distribution mean being the sum of the two means, and its variance being the sum of the two variances. When we subtract these distributions, the resultant mean is the difference between two means while variance is still the sum of both variances. Above two figures we can obtain the resultant distribution again the Gaussian and when we increase the number of data the variance is increased and the mean keep the same as 0 because 0 + 0 = 0.

3. Uncertainty in Estimation

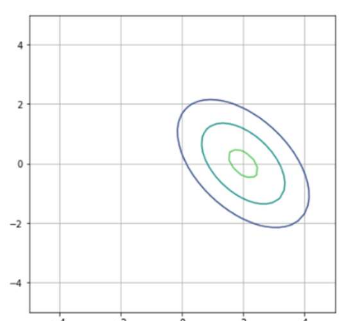


If the sample size is large, estimating variance is quite easy because it becomes fixed (not changing much). In other words, if the sample size is small variance is varying and does not get similar values for each case. Therefore, estimating the variance of the distribution is easy (successful) for large samples. Otherwise, it will not be successful since the variance is not fixed over different sample sets.

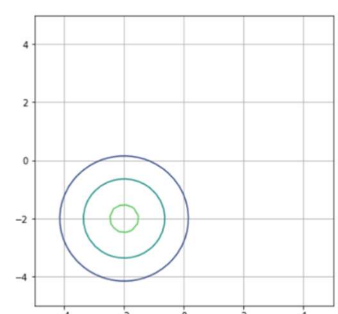
4. Bivariate Gaussian Distribution



$\boldsymbol{m1} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ $\boldsymbol{C1} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$



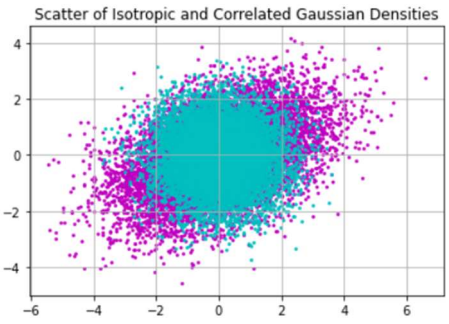
$\boldsymbol{m1} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ $\boldsymbol{C1} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$



$\boldsymbol{m1} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ $\boldsymbol{C1} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$

Here, m defines the mean vector (around which the distribution is) and c is called the covariance matrix. It defines the variances of the distribution over two axes and the rotation through axes.

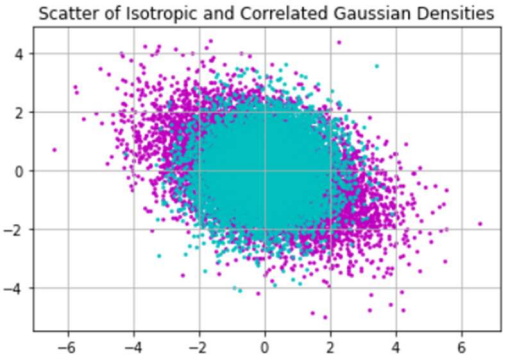
5.Sampling from a Multivariate Gaussian Distribution



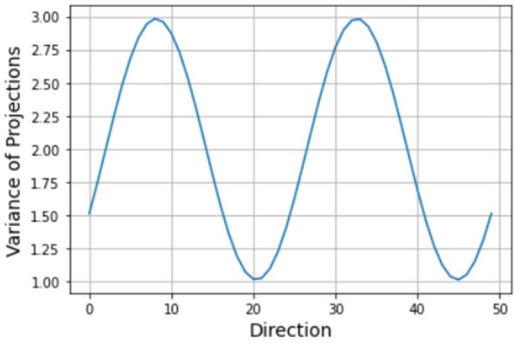

```
[[2, 1], [1, 2]]
[[1.41421356 0.        ]
 [0.70710678 1.22474487]]
[[2. 1.]
 [1. 2.]]
(10000, 2)
(10000, 2)
```

Two samples were taken from Gaussian distribution and plot it using the color cyan through two axes. Then change variance and rotation using covariance matrix C. Finally, plot modified distribution in magenta color.

6.Distribution of Projections



When $\boldsymbol{C1} = \begin{bmatrix} \mathbf{2} & \mathbf{-1} \\ \mathbf{-1} & \mathbf{2} \end{bmatrix}$




```
The vector:  [0.8660254037844386, 0.5000000000000001]
Sum of squares:  1.0
Degrees:  59.99999999999999
(10000,)
Projected Variance:  2.9838282842348405
```

The angle of the distribution is determined by the covariance matrix's Eigenvectors. The theta value in the previous example was 1/2. Here, it is -1/2 this time. The rotation is therefore the same in size but in the opposite direction. However, because the variances (elements of the main diagonal) are the same as the earlier one, the size of the distribution is also the same as in the earlier one.