Linear Algebra

1.

```
import numpy as np
A = np.random.randint(-10, 11, size=(5, 2))
x = np.random.randint(-10, 11, size=(2,1))
diagonal_elements = [1, 2, 3, 4, 5]
B = np.diag(diagonal_elements)
np.dot(A, x)
np.dot(B, A)
```

```
>>> import numpy as np
>>> A = np.random.randint(-10, 11, size=(5, 2))
>>> x = np.random.randint(-10, 11, size=(2,1))
>>> diagonal_elements = [1, 2, 3, 4, 5]
>>> B = np.diag(diagonal_elements)
>>> np.dot(A, x)
array([[ -7],
       [-43],
       [ -1],
       [-26],
       [-27]])
[>>> np.dot(B, A)
array([[ -4,  -9],
       [ 16,  -2],
       [ 15,  24],
       [ 16,  -8],
       [ 15, -20]])
[>>> A
array([[-4, -9],
       [ 8, -1],
       [ 5,  8],
       [ 4, -2],
       [ 3, -4]])
[>>> x
array([[-5],
       [ 3]])
```

2.

```
from numpy.linalg import matrix_rank
matrix_rank(A)
matrix_rank(B)
matrix_rank(np.dot(B, A))
```

```
[>>> from numpy.linalg import matrix_rank
[>>> matrix_rank(A)
2
[>>> matrix_rank(B)
5
[>>> matrix_rank(np.dot(B, A))
2
```

$$For\ A \in R^{m\times n}, B \in R^{n\times p},\ rank(AB) <= min(rank(A), rank(B))$$

Obviously, Here we have $rank(B) > rank(A) = rank(AB)$. Since A and B are full rank matrix, so the matrix AB is also full rank.

3.

(a)

$$AB = \begin{bmatrix} -2 & 1 & 8 \\ -1 & -1 & 7 \\ 3 & 0 & 4 \end{bmatrix} \begin{bmatrix} 5 & 0 & -7 \\ 6 & 3 & -8 \\ -2 & -2 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} (-2)\times 5 + 1\times 6 + 8\times(-2) & (-2)\times 0 + 1\times 3 + 8\times(-2) & (-2)\times(-7) + 1\times(-8) + 8\times 0 \\ (-1)\times 5 + (-1)\times 6 + 7\times(-2) & (-1)\times 0 + (-1)\times 3 + 7\times(-2) & (-1)\times(-7) + (-1)\times(-8) + 7\times 0 \\ 3\times 5 + 0\times 6 + 4\times(-2) & 3\times 0 + 0\times 3 + 4\times(-2) & 3\times(-7) + 0\times(-8) + 4\times 0 \end{bmatrix}$$

$$= \begin{bmatrix} -20 & -13 & 5 \\ -25 & -17 & 16 \\ 7 & 0 & -21 \end{bmatrix}$$

(b)

$$BA = \begin{bmatrix} 5 & 0 & -7 \\ 6 & 3 & -8 \\ -2 & -2 & 0 \end{bmatrix} \begin{bmatrix} -2 & 1 & 8 \\ -1 & -1 & 7 \\ 3 & 0 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 5\times(-2) + 0\times(-1) + (-7)\times 3 & 5\times 1 + 0\times(-1) + (-7)\times 0 & 5\times 8 + 0\times 7 + (-7)\times 4 \\ 6\times(-2) + 3\times(-1) + (-8)\times 3 & 6\times 1 + 3\times(-1) + (-8)\times 0 & 6\times 8 + 3\times 7 + (-8)\times 4 \\ (-2)\times(-2) + (-2)\times(-1) + 0\times 3 & (-2)\times 1 + (-2)\times(-1) + 0\times 0 & (-2)\times 8 + (-2)\times 7 + 0\times 4 \end{bmatrix}$$

$$= \begin{bmatrix} -31 & -23 & 12 \\ -42 & 3 & 33 \\ 6 & 0 & -30 \end{bmatrix}$$

(c)

$$AB-BA = \begin{bmatrix} -20 & -13 & 5 \\ -25 & -17 & 16 \\ 7 & 0 & -21 \end{bmatrix} - \begin{bmatrix} -31 & -23 & 12 \\ -42 & 3 & 33 \\ 6 & 0 & -30 \end{bmatrix}$$

$$= \begin{bmatrix} (-20)-(-31) & (-13)-(-23) & 5-12 \\ (-25)-(-42) & (-17)-3 & 16-33 \\ 7-6 & 0-0 & (-21)-(-30) \end{bmatrix}$$

$$= \begin{bmatrix} 11 & 10 & -7 \\ 17 & -20 & -17 \\ 1 & 0 & 9 \end{bmatrix}$$

(d)

$$ABC = AB * C = \begin{bmatrix} -20 & -13 & 5 \\ -25 & -17 & 16 \\ 7 & 0 & -21 \end{bmatrix} \begin{bmatrix} 6 & 3 & -1 \\ 2 & 4 & 5 \\ -1 & -1 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} (-20)\times 6 + (-13)\times 2 + 5\times(-1) & (-20)\times 3 + (-13)\times 4 + 5\times(-1) & (-20)\times(-1) + (-13)\times 5 + 5\times 8 \\ (-25)\times 6 + (-17)\times 2 + 16\times(-1) & (-25)\times 3 + (-17)\times 4 + 16\times(-1) & (-25)\times(-1) + (-17)\times 5 + 16\times 8 \\ 7\times 6 + 0\times 2 + (-21)\times(-1) & 7\times 3 + 0\times 4 + (-21)\times(-1) & 7\times(-1) + 0\times 5 + (-21)\times 8 \end{bmatrix}$$

$$= \begin{bmatrix} -151 & -117 & 35 \\ -200 & -159 & 68 \\ 63 & 42 & -175 \end{bmatrix}$$

4.

(a)

$$A - \lambda I = \begin{bmatrix} -2-\lambda & 1 & 8 \\ -1 & -1-\lambda & 7 \\ 3 & 0 & 4-\lambda \end{bmatrix}$$

Determinant ($|A - \lambda I|$)

$$= (-2-\lambda)((-1-\lambda)\times(4-\lambda) - 7\times 0) - (-1)(1\times(4-\lambda) - 0\times 8) + 3\times(1\times 7 - (-1-\lambda)\times 8)$$

$$= (-2-\lambda)(\lambda^2 - 3\lambda - 4) + (4-\lambda) + 24\lambda + 45$$

$$= -\lambda^3 + \lambda^2 + 10\lambda + 8 - \lambda + 4 + 24\lambda + 45$$

$$= -\lambda^3 + \lambda^2 + 33\lambda + 57$$

Set the determinant equal to zero and solve

$$-\lambda^3 + \lambda^2 + 33\lambda + 57 = 0$$

Calculate with python code:

```
import numpy as np
p = np.poly1d([-1, 1, 33, 57])
roots = p.roots
print(roots)
```

```
>>> import numpy as np
>>> p = np.poly1d([-1, 1, 33, 57])
>>> roots = p.roots
>>> print(roots)
[ 6.93926448 -3.7471852  -2.19207928]
```

For each eigenvalue λ, we'll solve the system of linear equations (A - λI)v = 0 to find the eigenvectors.

For any λ, $A - \lambda I = \begin{bmatrix} -2-\lambda & 1 & 8 \\ -1 & -1-\lambda & 7 \\ 3 & 0 & 4-\lambda \end{bmatrix}$

$$\lambda_1 \approx 6.939,\ A-\lambda I = \begin{bmatrix} -8.939 & 1 & 8 \\ -1 & -7.939 & 7 \\ 3 & 0 & -2.939 \end{bmatrix}$$

$$\lambda_2 \approx -3.747,\ A-\lambda I = \begin{bmatrix} 1.747 & 1 & 8 \\ -1 & 2.747 & 7 \\ 3 & 0 & 7.747 \end{bmatrix}$$

$$\lambda_3 \approx -2.192,\ A-\lambda I = \begin{bmatrix} 0.192 & 1 & 8 \\ -1 & 1.192 & 7 \\ 3 & 0 & 6.192 \end{bmatrix}$$

let$(A-\lambda I)V = 0$, and the result is listed below:

$[0.615, -0.580, 0.260]$

$[0.476, -0.783, 0.957]$

$[0.628, 0.225, -0.126]$

(b)

The trace of A equal to the sum of it's eigenvalues, $trA = \sum_{i=1}^{n} \lambda_i = -2 + (-1) + 4 = 1$

5.

```
import numpy as np

# Define matrices A, B, and C
A = np.array([[-2, 1, 8], [-1, -1, 7], [3, 0, 4]])
B = np.array([[5, 0, -7], [6, 3, -9], [-2, -2, 0]])
C = np.array([[6, 3, -1], [2, 4, 5], [-1, -1, 8]])

# Concatenate A, B, and C into matrix D
D = np.concatenate((A, B, C), axis=0)

# Define vector b
b = np.array([3, -10, 2])

# Solve for the least squares solution x using numpy.linalg.lstsq
x, residuals, _, _ = np.linalg.lstsq(D.T, b, rcond=None)

# Print the least squares solution x
print("Least Squares Solution x:")
print(x)

# Calculate the squared norm of the residual (‖DTx - b‖²)
squared_residual_norm = np.linalg.norm(np.dot(D.T, x) - b)**2
```

```
# Print the squared norm of the residual
print("Squared Norm of Residual (‖DTx − b‖²):")
print(squared_residual_norm)
```

```
>>> # Print the least squares solution x
>>> print("Least Squares Solution x:")
Least Squares Solution x:
>>> print(x)
[-0.6588419   0.79125205  1.2848211   1.12195654 -0.49220611  0.53822844
  0.10452284 -1.36113905  0.86584317]
>>>
>>> # Calculate the squared norm of the residual (‖DTx − b‖²)
>>> squared_residual_norm = np.linalg.norm(np.dot(D.T, x) - b)**2
>>>
>>> # Print the squared norm of the residual
>>> print("Squared Norm of Residual (‖DTx − b‖²):")
Squared Norm of Residual (‖DTx − b‖²):
>>> print(squared_residual_norm)
4.4965071597597673e-29
```

Statistics

1.

Obviously, the probability of getting 10 heads in a trail is $(\frac{1}{2})^{10} = \frac{1}{1024}$;

Consider use the binomial probability formula:

$$P_{(X=k)} = (n\ choose\ k) * p^k * (1-p)^{(n-k)}$$

Where:

- n is the number of trials.
- k is the number of successes.
- p is the probability of success in a single trial $\frac{1}{1024}$.

$$P_{(X=1)} = (1000\ choose\ 1) * (\frac{1}{1024})^1 * (1 - \frac{1}{1024})^{(1000-1)} = 0.368$$

Obviously, the opposite of at least one success is fail 1000 times.

$$P_{(at\ least\ one\ success)} = 1 - (1 - \frac{1}{1024})^{1000} = 0.624$$

python code:

```
# Import math module
from math import comb

# Define the number of trials
n = 1000

# Define the number of successful outcomes (specifically, 1 success)
k = 1
```

```python
# Define the probability of success for each individual trial
p = 1/1024

# Calculate the probability of exactly one success using the binomial
probability formula
one_success = comb(n, k) * (p ** k) * ((1 - p) ** (n - k))

# Calculate the probability of at least one success using complementary
probability
at_least_one_success = 1 - ((1 - p) ** n)

# Print the calculated probabilities
print("Probability of exactly one success:", one_success)
print("Probability of at least one success:", at_least_one_success)
```

```
>>> from math import comb
>>> n = 1000
>>> k = 1
>>> p = 1/1024
>>> one_success = comb(n, k) * (p ** k) * ((1 - p) ** (n - k))
>>> at_least_one_success = 1 - ((1 - p) ** n)
>>> one_success
0.36796070191273117
>>> at_least_one_success
0.623576201943276
```

2.

Consider use z-distribution because we know the average and standard deviation of the variable.

First, convert the values 32 and 38 to z-scores using the formula:

$$Z = \frac{X - \mu}{\sigma}$$

Where:

- Z is the z-score.
- X is the value we want to convert.
- μ is the average of the distribution.
- σ is the standard deviation of the distribution.

Obviously: μ=36 and σ=5, so we have

$$Z_{32} = \frac{32 - 36}{5} = -0.8$$

$$Z_{38} = \frac{38 - 36}{5} = 0.4$$

Using a standard normal distribution table or calculator, we can find these probabilities and then subtract them:

$$P_{(-0.8<Z<0.4)} = P_{(Z<0.4)} - P_{(Z<-0.8)} = 0.444$$

python code:

```
import scipy.stats as stats

# Define the given values
x = 36  # Mean (average) number of times a cat returns to its food bowl
u = 5   # Standard deviation

# Calculate the z-score for 32 and 38 using the z-score formula
z_32 = (32 - x) / u  # Z-score for 32
z_38 = (38 - x) / u  # Z-score for 38

# Calculate the cumulative probability (CDF) for the z-scores using a standard
normal distribution
p_32 = stats.norm.cdf(z_32)  # Cumulative probability for 32
p_38 = stats.norm.cdf(z_38)  # Cumulative probability for 38

# Calculate the answer by finding the difference in probabilities
answer = p_38 - p_32  # Probability of returning between 32 and 38 times

# Print the calculated answer
print("Probability of returning between 32 and 38 times:", answer)
```

```
>>> import scipy.stats as stats
>>> # x represents Mean and u represents standard deviation
>>> x = 36
>>> u = 5
>>> # Calculate the z-scores for 32 and 38
>>> z_32 = (32 - x) / u
>>> z_38 = (38 - x) / u
>>> # Use the cumulative distribution function (CDF) to find probabilities
>>> p_32 = stats.norm.cdf(z_32)
>>> p_38 = stats.norm.cdf(z_38)
>>> # Calculate the probability that a cat returns between 32 and 38 times
>>> answer = p_38 - p_32
>>> answer
0.4435663430269275
```

3.

Consider the formula for a confidence interval for the population mean (μ) is:

$$Confidence\ Interval = \overline{x} \pm Z\left(\frac{s}{\sqrt{n}}\right)$$

Where:

- $\overline{x}$ is the sample mean.
- Z is the critical value from the standard normal distribution corresponding to the desired confidence level. For a 95% confidence interval, Z is approximately 1.96.
- s is the sample standard deviation.

- n is the sample size.

Steps to get confidence interval:

1. Calculate the sample mean of the prices $\overline{x} = 842.6$
2. Calculate the sample standard deviation of the prices $s = 534.297$
3. Use the formula to calculate the confidence interval, result is $(511.445, 1173.755)$

python code:

```
import numpy as np
import scipy.stats as stats

# Define the list of prices for a particular model of digital camera
prices = [999, 1499, 1997, 398, 591, 498, 798, 849, 449, 348]

# Calculate the number of samples, sample mean, and sample standard deviation
n = len(prices)
sample_mean = np.mean(prices)
sample_std_dev = np.std(prices, ddof=1)

# Set the confidence level to 95%
confidence_level = 0.95

# Calculate the critical value (Z-score) for the confidence level
z_critical = stats.norm.ppf((1 + confidence_level) / 2)

# Calculate the margin of error
margin_of_error = z_critical * (sample_std_dev / np.sqrt(n))

# Calculate the confidence interval
confidence_interval = (sample_mean - margin_of_error, sample_mean +
margin_of_error)

# Print the results
print("Sample Mean Price:", sample_mean)
print("Sample Standard Deviation:", sample_std_dev)
print(f"95% Confidence Interval for Mean Price: {confidence_interval}")
```

4.

Consider perform a hypothesis test. Here are the steps to conduct the hypothesis test:

State the Hypotheses:

- Null Hypothesis (H_0): The average number of movies seen per person at the large university is equal to the national average ($\mu = 8.5$).

- Alternative Hypothesis (H_1): The average number of movies seen per person at the large university is not equal to the national average ($\mu \neq 8.5$).

Set the Significance Level (α):

- The significance level (α) is given as 0.05.

Collect Data and Calculate Test Statistic:

- Sample size (n) = 40
- Sample mean ($\overline{x}$) = 9.6
- Population standard deviation (σ) = 3.2
- Calculate the test statistic (t) using the formula: $t = \frac{(\overline{x} - u)}{\left(\frac{\sigma}{\sqrt{n}}\right)} = 2.174$

Determine the Critical Value(s):

- Since it's a two-tailed test (we are testing if the mean is not equal to 8.5), we need to find the critical values for a two-tailed test at a 0.05 level of significance. We can use a t-distribution table or calculator to find the critical values **(-2.023, 2.023)**.

Calculate the P-value:

- Calculate the p-value associated with the test statistic. We can use a t-distribution table or calculator for this. P-value = 0.036.

Make a Decision:

- Since p-value is less than α, reject the null hypothesis (H_0).

Draw a Conclusion:

- Since reject the null hypothesis, it suggests that there is a **significant difference** between the average number of movies seen at the large university and the national average.

python code:

```
import scipy.stats as stats

# Given data
sample_mean = 9.6
population_mean = 8.5
population_std_dev = 3.2
sample_size = 40
alpha = 0.05

# Calculate the test statistic (t)
t_statistic = (sample_mean - population_mean) / (population_std_dev /
(sample_size**0.5))

# Calculate the degrees of freedom
```

```
degrees_of_freedom = sample_size - 1

# Calculate the critical values for a two-tailed test
critical_value_left = stats.t.ppf(alpha / 2, degrees_of_freedom)
critical_value_right = stats.t.ppf(1 - alpha / 2, degrees_of_freedom)

# Calculate the p-value
p_value = 2 * (1 - stats.t.cdf(abs(t_statistic), degrees_of_freedom))

# Make a decision
if p_value < alpha:
    decision = "Reject the null hypothesis"
else:
    decision = "Fail to reject the null hypothesis"

# Draw a conclusion
if p_value < alpha:
    conclusion = "There is a significant difference from the national
average."
else:
    conclusion = "There is not enough evidence to conclude a significant
difference."

# Print results
print("Test Statistic (t):", t_statistic)
print("Critical Value (left):", critical_value_left)
print("Critical Value (right):", critical_value_right)
print("P-value:", p_value)
print("Decision:", decision)
print("Conclusion:", conclusion)
```

```
>>> # Print results
>>> print("Test Statistic (t):", t_statistic)
Test Statistic (t): 2.17406589136576
>>> print("Critical Value (left):", critical_value_left)
Critical Value (left): -2.0226909117347285
>>> print("Critical Value (right):", critical_value_right)
Critical Value (right): 2.022690911734728
>>> print("P-value:", p_value)
P-value: 0.035831846689315716
>>> print("Decision:", decision)
Decision: Reject the null hypothesis
>>> print("Conclusion:", conclusion)
Conclusion: There is a significant difference from the national average.
```

5.

To test whether the variance in grades exceeds 100, you can perform a hypothesis test for the population variance. Here are the steps:

State the Hypotheses:

- Null Hypothesis (H_0): The population variance (σ^2) is less than or equal to 100.

- Alternative Hypothesis (H_1): The population variance (σ^2) exceeds 100.

Set the Significance Level (α):

- The significance level (α) is given as 0.05.

Collect Data:

- The sample data consists of midterm grades.

Calculate the Test Statistic:

- Calculate the test statistic for testing the variance using the chi-squared distribution. The formula is:

$$\chi^2 = \frac{(n-1)S^2}{\sigma^2} = 27.729$$

  Where:

  n is the sample size.

  S^2 is the sample variance.

  σ^2 is the hypothesized population variance (100 in this case).

Determine the Critical Value(s):

- Determine the critical value from the chi-squared distribution table for a right-tailed test at the 0.05 level of significance with degrees of freedom $df = n - 1$. **Critical value = 23.685**.

Make a Decision:

- Since the calculated test statistic is greater than the critical value, we reject the null hypothesis.

Draw a Conclusion:

- Since we reject the null hypothesis, it suggests that **the variance in grades exceeds 100**.

python code:

```
import numpy as np
import scipy.stats as stats

# Given data (sample grades)
grades = np.array([92.3, 89.4, 76.9, 65.2, 49.1, 96.7, 69.5, 72.8, 67.5, 52.8,
88.5, 79.2, 72.9, 68.7, 75.8])

# Sample size
n = len(grades)
```

```python
# Sample variance
sample_variance = np.var(grades, ddof=1)  # Use ddof=1 for sample variance

# Hypothesized population variance
hypothesized_variance = 100

# Calculate the test statistic (chi-squared)
test_statistic = (n - 1) * sample_variance / hypothesized_variance

# Degrees of freedom
degrees_of_freedom = n - 1

# Calculate the critical value from the chi-squared distribution
alpha = 0.05
critical_value = stats.chi2.ppf(1 - alpha, df=degrees_of_freedom)

# Make a decision
if test_statistic > critical_value:
    decision = "Reject the null hypothesis"
else:
    decision = "Fail to reject the null hypothesis"

# Draw a conclusion
if test_statistic > critical_value:
    conclusion = "The variance in grades exceeds 100."
else:
    conclusion = "There is not enough evidence to conclude that the variance
exceeds 100."

# Print results
print("Sample Variance:", sample_variance)
print("Test Statistic (chi-squared):", test_statistic)
print("Critical Value (chi-squared):", critical_value)
print("Decision:", decision)
print("Conclusion:", conclusion)
```

```
>>> print("Sample Variance:", sample_variance)
Sample Variance: 183.7755238095238
>>> print("Test Statistic (chi-squared):", test_statistic)
Test Statistic (chi-squared): 25.728573333333333
>>> print("Critical Value (chi-squared):", critical_value)
Critical Value (chi-squared): 23.684791304840576
>>> print("Decision:", decision)
Decision: Reject the null hypothesis
>>> print("Conclusion:", conclusion)
Conclusion: The variance in grades exceeds 100.
```