CS 3430: S26: Scientific Computing
Assignment 6

Vladimir Kulyukin
SoC – CoE – USU

February 14, 2026

Learning Objectives

After completing this assignment, you should be able to:

1. Implement and interpret **chi-square goodness-of-fit tests** for multi-symbol data (digits, nucleotides, grayscale intensities).
2. Implement and interpret **binary randomness tests** (monobit and block-frequency tests).
3. Understand the role of **null hypotheses**, **significance levels**, and **p-values** in statistical inference.
4. Explain why expected-count conditions (e.g., $E_k \geq 5$) matter for the reliability of chi-square tests.
5. Analyze how statistical conclusions depend on:
   - the chosen probabilistic model,
   - the data encoding scheme,
   - the type of test applied.
6. Distinguish between **logical deduction** and **statistical inference**, and articulate the limitations of model-based reasoning.
7. Design computational experiments that expose instability, multiplicity effects, and structural sensitivity in randomness testing.

Introduction

In Homework 5, we explored deterministic mathematical processes that produce digits appearing statistically random. In this assignment, we reverse the direction of inquiry. Instead of asking,

> *How do we compute infinite deterministic constants?*

we will ask,

> *When does observed data behave as if it were random?*

This assignment lives at the intersection of mathematics, statistics, biology, and visual perception.

- In Problem 1, we examine the digits of π and e using chi-square goodness-of-fit tests.
- In Problem 2, we analyze the human mitochondrial genome under various encoding schemes and null hypotheses.

- In Problem 3, we test grayscale images (random, artificial, and natural) against stochastic models.

A central theme emerges:

> Randomness is not an intrinsic label. It is a statement of compatibility with a probabilistic model.

We will observe phenomena that may seem paradoxical:

- Deterministic digits behaving like random samples.
- Biological sequences rejecting simple uniform models.
- Perfectly balanced images escaping certain tests.
- Highly structured photographs decisively failing randomness tests.

These are not necessarily contradictions, but reminders that statistical randomness describes observable behavior under a model not metaphysical origin.

Please review **Lecture 10** and **Lecture 11** before beginning this assignment. These lectures develop the theoretical foundations of randomness testing, chi-square inference, and model-based reasoning that underlie every problem below.

Problem 1: χ^2 Goodness-of-Fit Test of the Mantissas of π and e

In this problem, we investigate a deceptively simple question: *Do the decimal digits of π and e behave like random samples?*

Let us fix our statistical model. Let $X_1, X_2, \ldots, X_n$ denote the first n digits of a mantissa (the digits after the decimal point). Each X_i takes values in $\{0, 1, \ldots, 9\}$. We test the null hypothesis

$$H_0 : \ P(X_i = k) = \frac{1}{10}, \quad k = 0, \ldots, 9,$$

under the assumption that the digits behave like i.i.d. samples from a uniform distribution on $\{0, \ldots, 9\}$.

Let O_k denote the observed count of digit k in the first n digits. Under H_0, the expected count is

$$E_k = \frac{n}{10}.$$

The chi-square statistic is

$$\chi^2 = \sum_{k=0}^{9} \frac{(O_k - E_k)^2}{E_k}.$$

Under H_0 and for sufficiently large n,

$$\chi^2 \sim \chi^2_9,$$

because there are $K = 10$ categories and one linear constraint $\sum_{k=0}^{9} O_k = n$, giving $df = K - 1 = 9$.

We reject H_0 at level $\alpha = 0.05$ if

$$P(\chi^2_9 \geq \text{observed}) \leq 0.05.$$

Mathematical Expectation: Expected Count Condition

The classical chi-square approximation relies on a large-sample argument. A standard rule-of-thumb requires

$$E_k \geq 5 \quad \text{for all } k.$$

Since $E_k = n/10$, this condition becomes

$$n \geq 50.$$

For $n < 50$, the chi-square reference distribution is only a rough approximation. Thus:

- The distribution of counts is highly discrete,
- The normal approximation deteriorates,
- The p-values may fluctuate erratically,
- The test becomes statistically unstable.

This instability is not a failure of mathematics. It is a consequence of using an asymptotic approximation outside its reliable boundaries.

What to Implement

Now go ahead and implement

1. `chi_square_statistic`;
2. `chi_square_p_value`;
3. `chi_square_decision`;

in `cs3430_s26_hw_6_prob_1.py`.

Computational Experiments

The experimental infrastructure for this prolem is in `cs3430_s26_hw_6_prob_1_ut.py`. Open up that file and spend a few minutes reading the comments on what the structural and numerical tests do. Basically, they allow us to:

1. Compute chi-square p-values for the first

   $$n \in \{10, 20, 30, \ldots, 100, 200, \ldots, 9999\}$$

   digits of π and e;
2. Produce formatted tables of p-values and rejection decisions.
3. Generate diagnostic plots for:
   - 10–50 digits (small-sample instability),
   - 10–3000 digits,
   - 4000–7000 digits,
   - 8000–9999 digits.

The actual numbers of digits are hard-coded for simplicity in `hw6_y_values()` in `cs3430_s26_hw_6_prob_1.py`. There are two text files `9_999DigitsOfE.txt` and `99_999DigitsOfPi.txt` in the zip. The parsing tools `pi_mantissa_parser.py` and `e_mantissa_parser.py` allow us to extract the first 10, 20, etc. digits of these numbers and then apply the chi-square tests to them. Finally, the function `format_results_table` enables us to display the experimental results. The file `prob_1_ut_output.txt` contains my output of running the UTs for Problem 1. Below are the first 10 rows of the full diagnostic table.

```
================= FULL DIAGNOSTIC TABLE OUTPUT ============================
-------------------------------------------------------------------------------
Num Digits | PI P-VAL     | E P-VAL     | Chi^2 PI Reject | Chi^2 E Reject
-------------------------------------------------------------------------------
       10 |     0.534146 |    0.0668816 | False           | False
       20 |     0.911413 |     0.834308 | False           | False
       30 |     0.602458 |     0.739918 | False           | False
       40 |     0.875539 |     0.689019 | False           | False
       50 |     0.739918 |     0.911413 | False           | False
       60 |     0.739918 |     0.602458 | False           | False
       70 |     0.821681 |     0.247472 | False           | False
       80 |     0.811993 |     0.330628 | False           | False
       90 |     0.739918 |     0.447593 | False           | False
      100 |     0.897763 |     0.289667 | False           | False
```

The last two columns record whether the H_0 hypothesis is rejected for the corresponding p values in the same row. We fail to reject the null hypothesis in all but one case and only for π when its mantissa has 3700 digits. This is the only case where the mantissa *appears* non-random. I will have more to say on appearances later.

```
    3700 |    0.0493975 |     0.908543 | True            | False
```

Diagnostic Plots and Interpretation

The unit tests also allow us to plot. Each plot includes:

- p-values for π (red),
- p-values for e (blue),
- a horizontal line at $\alpha = 0.05$ (green dashed).

Let us go through each plot that got generated on my machine and try to make sense of what took place.

10–50 Digits: Figure 1 contains the p-values for the mantissas that contain 10 to 50 digits. This testing regime violates the expected-count condition $E_k = n/10 \geq 5$. For $n < 50$, the chi-square distribution is only a rough asymptotic approximation. As a result, p-values fluctuate noticeably. Small changes in counts can cause disproportionately large changes in the chi-square statistic because the denominator E_k is small. The instability observed here is evidence of using an asymptotic approximation outside its reliable boundaries.

10–3000 Digits: Figure 2 has the p-values for the mantissas that contain 10 to 3000 digits. Once $n \geq 50$, the expected-count condition is satisfied and the chi-square approximation becomes stable. P-values still fluctuate, because each new block of digits slightly changes the empirical distribution. Occasional dips toward the rejection threshold are expected. At significance level $\alpha = 0.05$, approximately 5% of tests are expected to reject purely by chance under the null model. An isolated rejection does not prove non-randomness. It reflects the probabilistic nature of statistical inference.

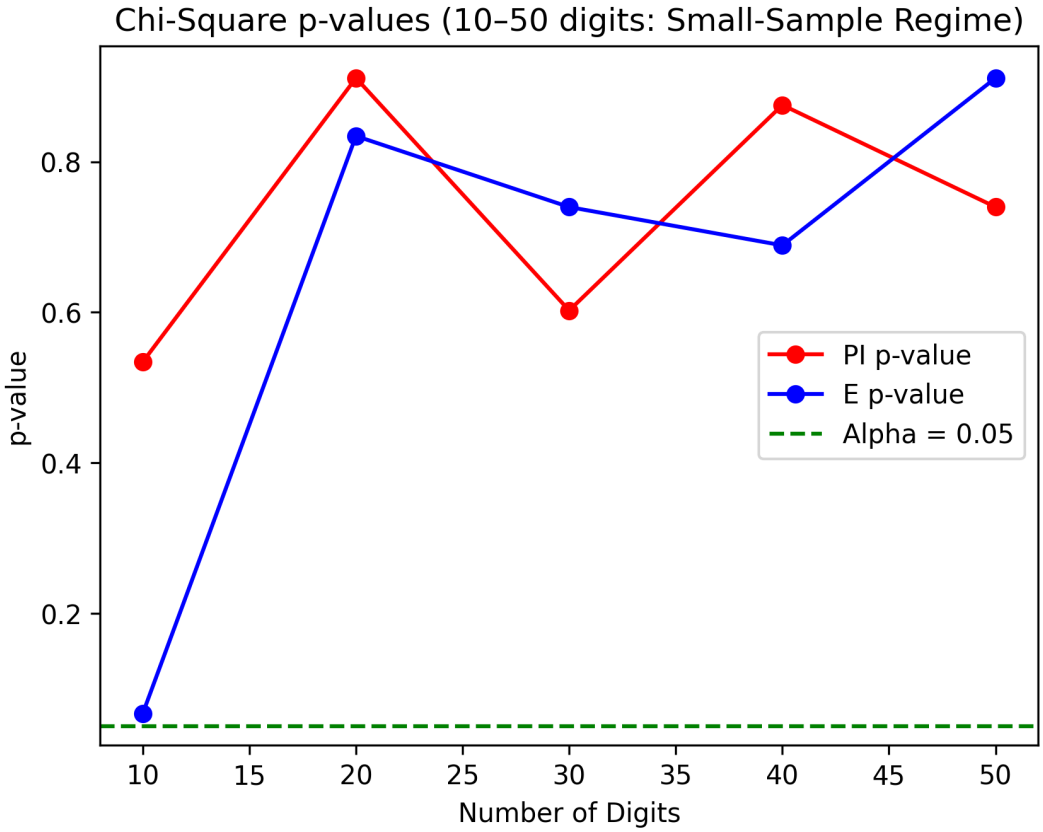

Figure 1: Chi-square p-values for π and e for $10 \leq n \leq 50$. The green dashed line represents $\alpha = 0.05$.

4000–7000 Digits: Figure 3 has the p-values for the mantissas that contain 4000 to 7000 digits. In this sample range, the p-values fluctuate within a narrower band. As n increases, the empirical digit frequencies stabilize and the chi-square statistic grows approximately on the order of $\sqrt{n}$ under the null model. The resulting p-values exhibit persistent variability but remain well above the rejection threshold. This behavior is consistent with digits that are statistically indistinguishable from uniform in finite samples. All mantissas *appear* random in this range.

8000–9999 Digits: Figure 4 has the p-values for the mantissas that contain 8000 to 9999 digits. In this range, both sequences exhibit relatively stable p-value trajectories. Neither π nor e consistently approaches the rejection threshold. The fluctuations observed are characteristic of sampling variability under a correctly specified null model. All mantissas *appear* random in this range as well.

Logical vs. Statistical Inference

Let us unpack the differences between logical and statistical inference. In pure mathematics:

- Premises are fixed.
- Deduction is exact.
- Conclusions are certain.
- There is no α.
- There is no p-value.
- There is no approximation.

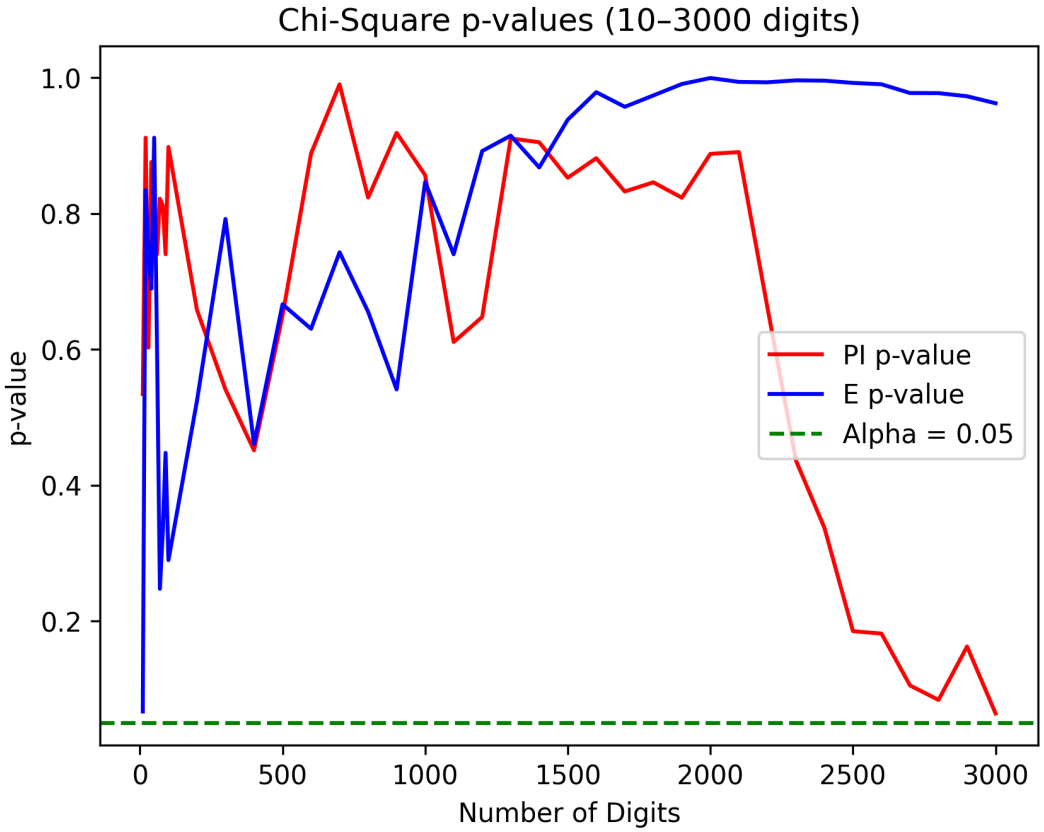

Figure 2: Chi-square p-values for π and e for $10 \leq n \leq 3000$.

If a theorem is proved, it is proved. The conclusion follows necessarily from the premises. There is no probabilistic tolerance and no risk parameter. There is no "false rejection" or "false acceptance." Logical inference is model-free in the statistical sense: truth is determined by formal derivation, not by sampling variability.

In statistical inference:

- We observe finite samples.
- We assume a probabilistic model.
- We compute a statistic.
- We evaluate a tail probability.
- We accept a risk level α.
- We draw a probabilistic conclusion.

The entire structure is: *Model-dependent and risk-tolerant (or, to put it differently, error-tolerant).* We do not prove hypotheses. We assess compatibility with a stochastic model under a specified error tolerance. Our conclusion always carries this implicit clause:

> "Under the assumed model, and at risk level α."

This qualifier never disappears. The trap (and it is a dangerous one!) of statistical inference is believing that its conclusions have the logical force of mathematical proof. They do not! Thus, the following statements are *not* logically valid:

- Rejecting H_0 proves non-randomness.

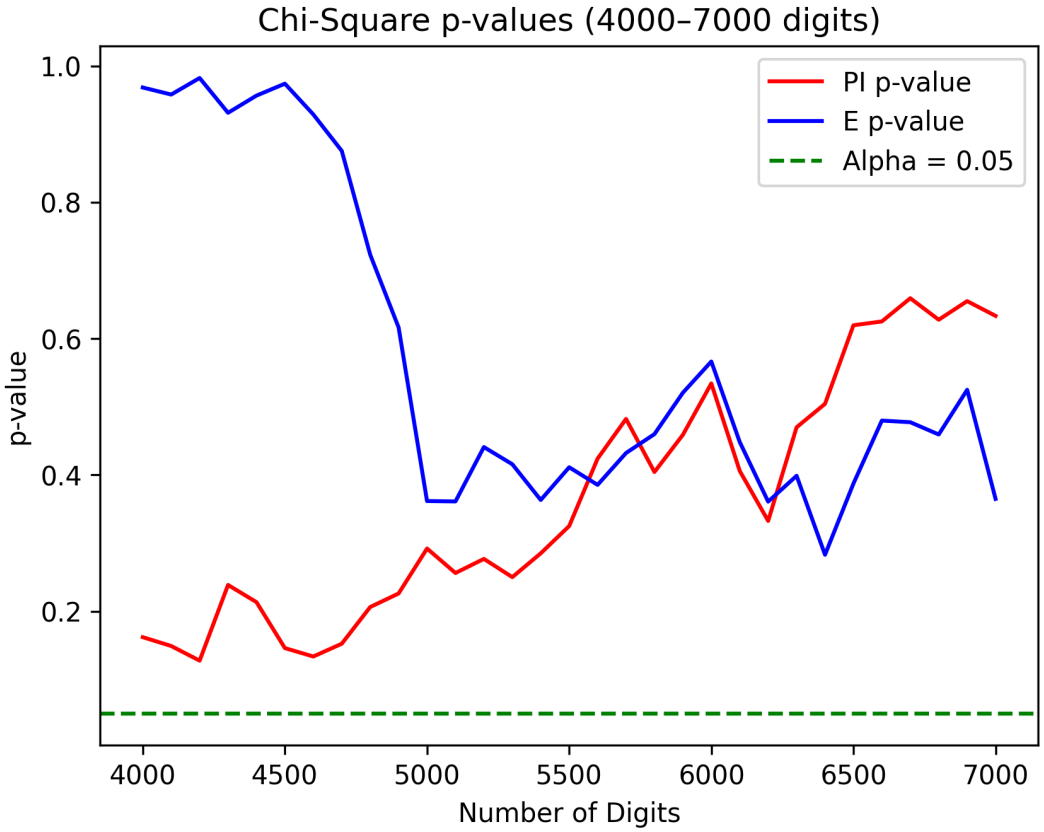

Figure 3: Chi-square p-values for π and e for $4000 \leq n \leq 7000$.

- Failing to reject H_0 proves randomness.
- A single test result reveals intrinsic truth.
- A deterministic object that passes statistical tests must be random.
- A stochastic object that fails a test must be structured.

None of these statements follow deductively. They are statistical conclusions under:

- a chosen model,
- a finite sample,
- and a specified risk parameter α.

Our inference is conditional and probabilistic, not absolute and logical.

Why Appearances Can Deceive

The digits of π and e are generated by *deterministic* algorithms. There is no randomness in their production.

Archimedes did not throw dice. Leibniz did not flip coins. Ramanujan did not sample from a distribution. The Chudnovsky brothers did not rely on stochastic simulation. These great masters constructed fully deterministic mathematical processes. Every digit follows necessarily from exact symbolic rules. Yet, when finite segments of those digits are examined through stochastic models, their behavior is statistically indistinguishable from random samples.

A paradox? Perhaps. It is certainly one of the things that injects insomnia into my brain and causes me to look at the stars deep into the night. It reminds me that statistical randomness is

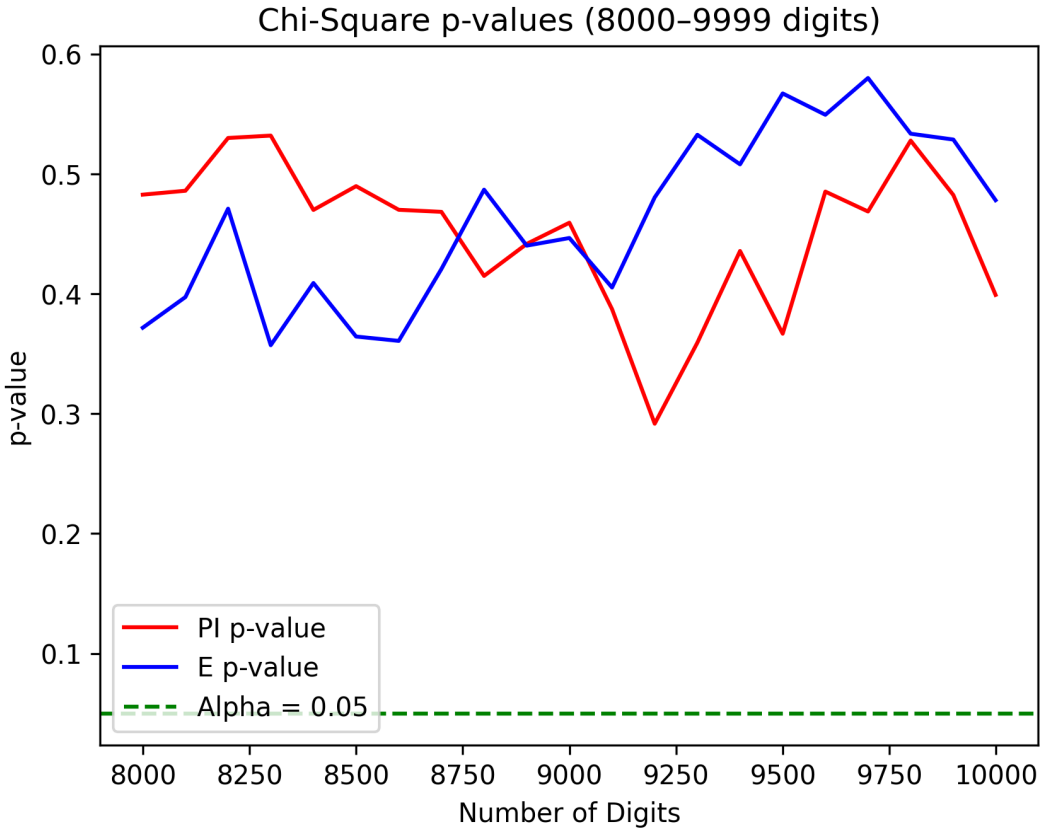

Figure 4: Chi-square p-values for π and e for $8000 \leq n \leq 9999$.

a property of observed behavior under a model, not a statement about the metaphysical origin of a sequence. Now that I wrote the previous two sentences I have to validate them logically and step out into my tiny back yard and look at the stars. Back in a minute. Well, I am back. The Big Dipper was beautiful and the cold air certainly injected a healthy doze of insomnia into my blood stream. Anyway, when examined through finite windows and evaluated under a stochastic null model, digit sequences may appear statistically random. This does not prove that the digits are random in any metaphysical sense. It demonstrates that deterministic structure can produce statistical behavior indistinguishable from randomness under finite observation. Statistical inference measures compatibility with a model. It does not reveal intrinsic ontology and, consequently, cannot, in my humble opinion, be used as a reliable tool of rational epistemology.

Reflection

Briefly (no more than a paragraph) answer the following questions in a short write-up (embedded as a multi-line comment at the top of your problem 1 Python file):

1. Why do the p-values fluctuate dramatically in the 10–50 digit regime?
2. Why does the test occasionally reject H_0 even though π and e are generated by deterministic algorithms?
3. Does failing to reject H_0 prove that the digits are random? Explain carefully.
4. If you perform many hypothesis tests at $\alpha = 0.05$, what fraction of rejections should you expect purely by chance?
5. In what sense can a deterministic sequence behave statistically like a random one?

What to Submit for Problem 1

- `cs3430_s26_hw_6_prob_1.py` with your write-up embedded as a multi-line comment at the top of this file.
- Do not submit your PNG files or your UT output. We will generate and look at these artifacts when we run the unit tests on your submission.

Problem 2 (2 points): Randomness of Biological Sequences

In Problem 1, we applied chi-square goodness-of-fit tests to the decimal mantissas of π and e. Those sequences are generated by fully deterministic mathematical methods. Yet, they passed statistical randomness tests under a simple stochastic model. In this problem, we will apply the same statistical machinery to a very different object: the human mitochondrial genome.

Our data source is `mtDNA_sequence.txt`. I generated this file from an NIH site. I included my script `fetch_mtDNA.py` for complete replicability. This txt contains the complete human mitochondrial reference genome (RefSeq NC_012920.1). The genome length is 16,569 bases. Unlike π and e, this sequence is biological. It is not generated by a closed-form deterministic mathematical formula. It is a product of evolution, mutation, selection, and biochemical constraint.

Mitochondria are cellular organelles responsible for oxidative phosphorylation, the biochemical process by which cells produce adenosine triphosphate (ATP), the primary energy currency of life. Unlike most organelles, mitochondria possess their own DNA (mtDNA), which is distinct from the nuclear genome. The human mitochondrial genome:

- is a circular DNA molecule;
- has length 16,569 base pairs;
- encodes 37 genes (13 protein-coding, 22 tRNA, 2 rRNA);
- is inherited maternally;
- exhibits relatively high mutation rates compared to nuclear DNA.

Since mtDNA is small, well-characterized, and biologically essential, it serves as an ideal model system for studying genomic structure. The mitochondrial genome plays a central role in:

- cellular energy production,
- metabolic regulation,
- apoptosis (programmed cell death),
- aging processes.

Mutations in mtDNA are associated with a wide range of diseases, including mitochondrial myopathies, neurodegenerative disorders, cardiovascular disease, and certain cancers. Since mtDNA is maternally inherited and does not undergo recombination in the same way as nuclear DNA, it is also widely used in:

- evolutionary biology,
- population genetics,
- forensic identification,
- ancestry tracing.

We will analyze the genome under two different symbolic encodings.

1. Binary GC/AT Encoding

$$G, C \mapsto 1, \qquad A, T \mapsto 0.$$

Under this encoding, we test the null hypothesis:

$$H_0 : \text{bits are i.i.d. Bernoulli}(1/2).$$

For each window, we compute:

- Monobit test
- Block frequency test

2. Four-Symbol Encoding

$$A \mapsto 1, \quad C \mapsto 2, \quad G \mapsto 3, \quad T \mapsto 4.$$

Under this encoding we test:

$$H_0 : P(A) = P(C) = P(G) = P(T) = \frac{1}{4}, \quad \text{i.i.d. across positions.}$$

We apply the chi-square goodness-of-fit test with $K = 4$ categories and $df = 3$.

Sliding Window Analysis

The file `cs3430_s26_hw_6_prob_2_uts.py` provides the experimental investigation for this problem. To study local structure, we do not test the entire genome at once. Instead, we use overlapping sliding windows:

- Window size: $W = 500$
- Step size: 50 bases
- Significance level: $\alpha = 0.05$

The above window sizes and step size are completely arbitrarily. We can fix other values for these parameters. For each window we compute p-values for:

- Monobit (binary)
- Block frequency (binary)
- Chi-square GOF (4-symbol)

This produces 322 overlapping windows across the genome. Figure 5 shows the plot of the p-values across genome position for all three tests that my implementation generated. The output of my unit tests is in `problem_problem_2_ut_output.txt`.
We can observe that:

- The 4-symbol chi-square test rejects the uniform null hypothesis in essentially every window;
- The monobit test fluctuates across the genome, sometimes above and sometimes below α;
- The block frequency test exhibits similar oscillatory behavior, with different amplitude and structure.

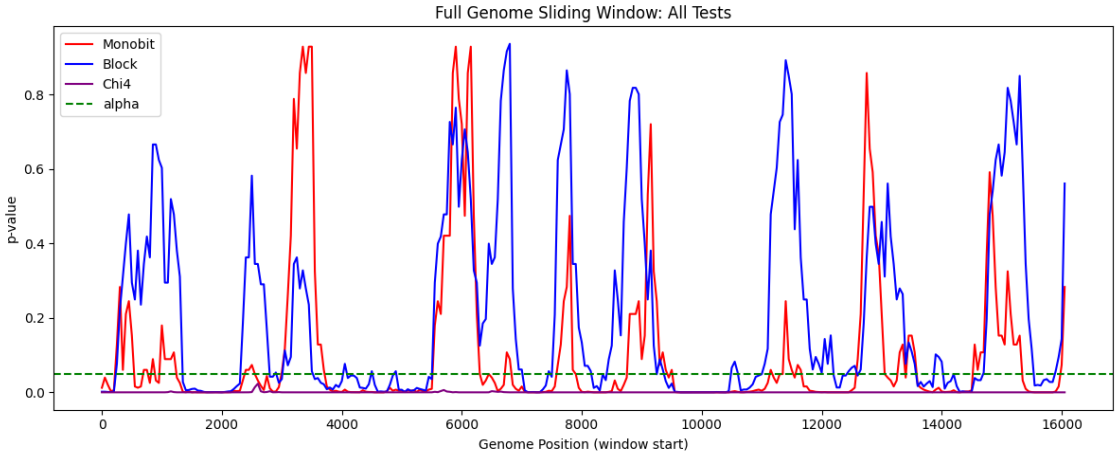

Figure 5: Sliding-window p-values across the full human mitochondrial genome. Red: Monobit (GC/AT). Blue: Block frequency. Purple: 4-symbol chi-square. Green dashed line: $\alpha = 0.05$.

How can these results be interpreted? The 4-symbol test rejects, because the human mitochondrial genome is not compositionally uniform. It is AT-rich and exhibits strand asymmetry. With window size $W = 500$, the expected count per base under uniformity is 125, so the chi-square test has high power.

The monobit test measures deviation from $P(GC) = 1/2$. Because GC content varies along the genome, the p-values oscillate spatially. The block frequency test detects local clustering within windows, leading to a different rejection profile. The story here is that the same data, under different encodings, produce different statistical conclusions. Randomness tests do not measure intrinsic randomness. They measure compatibility with a chosen null model.

What to Implement

Now go ahead and implement:

1. `monobit_test_bits`;
2. `block_frequency_test_bits`;
3. `reject_h0`.

in `cs3430_s26_hw_6_prob_2.py`.

Write a short discussion (multi-line comment at the top of your problem 2 Py file) briefly addressing:

- Why the 4-symbol null fails everywhere.
- Why monobit oscillates.
- How encoding affects statistical conclusions.
- How this contrasts with the behavior of π and e.

What to Submit for Problem 2

- `cs3430_s26_hw_6_prob_2.py` with your write-up embedded as a multi-line comment at the top of this file.
- Do not submit your PNG files or your UT output. We will generate and look at these artifacts when we run the unit tests on your submission.

Determinism, Structure, and Statistical Appearance

The digits of π and e are generated by deterministic algorithms. Archimedes, Leibniz, Ramanujan, and the Chudnovsky brothers constructed exact symbolic processes whose outputs, when viewed through stochastic models, appear indistinguishable from random sequences. And yet, the mitochondrial genome inside each of these mathematical masters (and inside every one of us) systematically violates the simplest stochastic assumptions of uniformity and independence.

The mathematical processes invented by human intellect to compute π produce sequences that pass many randomness tests. The biological sequence that enables that very intellect to function fails those same tests under naive null models.

Another paradox? Look like one! A deterministic object can appear statistically random. A biological object can exhibit structured deviation from simplistic random models. Neither conclusion speaks to intrinsic ontology. Statistical inference quantifies deviation from assumptions. It does not determine the essence of an object.

From a statistical perspective, the mitochondrial genome provides a fascinating contrast to the mantissas of π and e. The digits of π are generated by exact symbolic rules. They are mathematically deterministic and structurally rigid. The mitochondrial genome is also deterministic. When we apply randomness tests to mtDNA, we are not asking whether biology is random. We are asking:

> Is this biological sequence compatible with a simple stochastic model?

The answer depends entirely on the chosen encoding and null hypothesis. This distinction (between intrinsic origin and statistical compatibility) is key Appearances do deceive. Statistical machinery is a tool, not a revelation.

Problem 3 (1 point): Randomness in Grayscale Images

In this problem, we apply the same statistical machinery used in Problems 1 and 2 to a new and very different object: grayscale images. An image can be viewed as a matrix of integers in $\{0, 1, \ldots, 255\}$. From a statistical perspective, this is equivalent to observing outcomes of a 256-sided die. Alternatively, if we binarize the image by mapping pixel intensities $< 128 \mapsto 0$ and $\geq 128 \mapsto 1$, we obtain a binary sequence suitable for monobit and block frequency tests.

We will compare three types of images:

1. Artificially generated random images (100×100 matrices with entries drawn uniformly from $\{0, \ldots, 255\}$).
2. Real grayscale photographs (from the `imgs/` directory).
3. Carefully constructed deterministic images (constant image, half-black/half-white image, checkerboard).

Statistical Models

We consider three null hypotheses:

- **256-symbol uniform model:** Pixel intensities are i.i.d. Uniform on $\{0, \ldots, 255\}$.
- **Binary monobit model:** After binarization, pixels are i.i.d. Bernoulli$(1/2)$.
- **Binary block-frequency model:** After binarization, local blocks exhibit no significant deviation from 50% ones.

Each image is flattened row-wise into a one-dimensional sequence before applying statistical tests.

What to Implement

In `cs3430_s26_hw_6_prob_3.py`:

- `generate_random_image`.

Experiments and Observations

In `cs3430_s26_hw_6_prob_3_uts.py`, we

- Save deterministic test images to disk;
- Run all statistical tests;
- Print formatted test tables.

The file `problem_3_ut_output.txt`. Take a close look at the REAL IMAGE RANDOMNESS TABLE in this file. You should observe:

- Truly random images typically fail to reject all null hypotheses;
- Constant images strongly reject all null hypotheses;
- Half-black/half-white images may pass monobit but fail block tests;
- Checkerboard images may pass monobit and block tests despite being completely deterministic;
- Real photographs strongly reject uniform randomness assumptions.

We can draw a few lessons:

1. **Encoding dependence:** Statistical conclusions depend on how data are represented (256-level vs. binary encoding).
2. **Model dependence:** All conclusions are relative to a specified null hypothesis.
3. **Test sensitivity:** Different tests detect different types of structure.
4. **Balance vs. independence:** A sequence may be perfectly balanced (50/50) yet highly structured.

Write a short discussion (multi-line comment at the top of your problem 2 Py file) briefly (no more than a few sentences per item) addressing:

- The behavior of the **256-symbol uniform model**;
- The behavior of the **Binary monobit model**;
- The behavior of the **Binary block-frequency model**;

Reflection

In Problem 1, we saw that purely deterministic mathematical processes (invented by Archimedes, Leibniz, Ramanujan, and the Chudnovsky brothers) produce sequences of digits that behave statistically like random samples.

In Problem 2, we observed that the mitochondrial genome inside each of those great mathematical minds decisively rejects simple uniform randomness models.

In Problem 3, we see that images created by human perception and cognition once digitized are overwhelmingly structured relative to stochastic models. Natural photographs reject uniformity. Artificial checkerboards may escape certain tests despite being deterministic. Random matrices generated by code may pass tests that sophisticated visual structures fail.

Brains construct deterministic algorithms that generate sequences appearing statistically random. Those same brains are built from genomes that reject uniform randomness. And, the visual world those brains perceive is deeply structured, even when encoded into numerical matrices.

Illustrative Deterministic Images

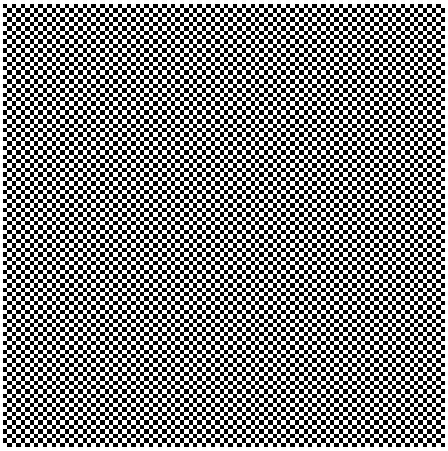

Figure 6: 100×100 checkerboard image (aligned blocks). Although perfectly deterministic and highly structured, this image passes both the monobit and block-frequency tests because it is locally balanced at every scale.

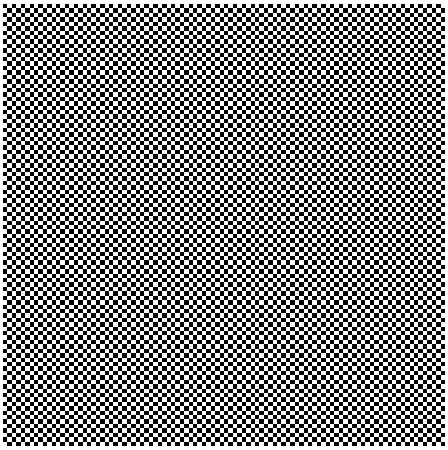

Figure 7: Checkerboard image analyzed with misaligned block size. Even when block boundaries do not align with row structure, the local 50/50 balance persists. The block test detects imbalance, not periodic alternation.

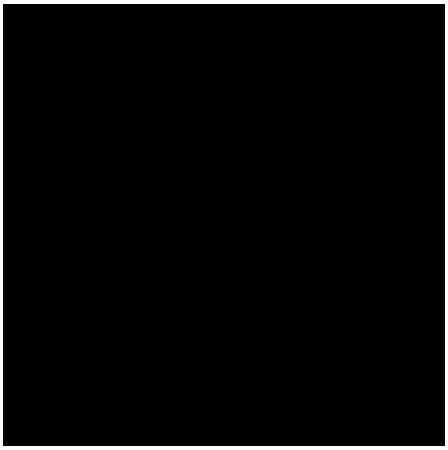

Figure 8: Constant grayscale image. All statistical tests strongly reject the null hypotheses. This image exhibits maximal structure and zero entropy.

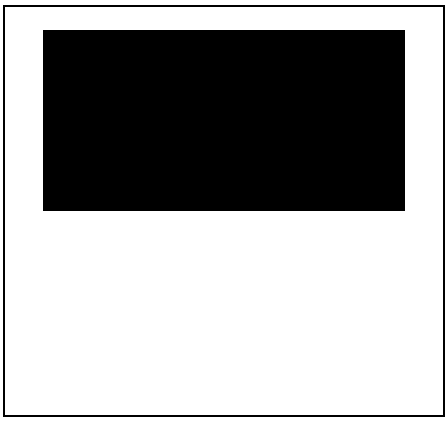

Figure 9: Half black and half white image. This image is globally balanced (50% black, 50% white), so the monobit test fails to reject the Bernoulli(1/2) null. However, the block-frequency test strongly rejects due to large contiguous regions of uniform intensity.

What to Submit for Problem 3

- `cs3430_s26_hw_6_prob_2.py` with your write-up embedded as a multi-line comment at the top of this file.
- Do not submit your PNG files or your UT output. We will generate and look at these artifacts when we run the unit tests on your submission.

Happy Hacking! Enjoy Randomess!

If this assignment leaves you unsure whether randomness exists, whether determinism is real, or whether reality itself merely fails to reject a convenient null hypothesis – that is not a bug, but a mathematical expectation! Do not worry: that uncertainty, too, may merely be statistically significant at level $\alpha = 0.05$. After all, even the universe might simply be failing to reject a very large null hypothesis.