High-dimensional Pearson's chi-squared test

Hoge dimensionale Pearson chi-squared test.

by

Christiaan van Wingerde

Bachelor of Applied Mathematics, Technical University of Delft, 2025

Supervisor:	F. Mies, Dr. rer. nat., Assistant Professor of Statistics at TU Delft
Examining Board:	F.Mies, Dr. rer. nat., Assistant Professor of Statistics at TU Delft
	D.Kurowicka, Dr., TU Delft

Abstract

This paper revisits Pearson's chi-square test and studies its properties, highlighting the behavior of the test when applied to large supports, i.e., the number of cells versus the sample size. First, we explore the general behavior through a controlled simulation, wherein we find that the test exhibits an increased number of type I errors. These errors occur when the sample size is small relative to the number of cells. This behavior will be explained using a generalized central limit theorem, showing that the support needs to be $o\left(\sqrt{\frac{n}{\log n}}\right)$.

Contents

1 Introduction

A Personal Story. During my free time as a student, my friends and I often played a tabletop game called Dungeons and Dragons. This game tells fantastical stories and relies on the uniform fairness of dice, such as d6 (six-sided dice), d12, d20, and even d100. However, one of our friends seemed to view the game as a competition to be won rather than a collaborative storytelling experience. This raised our suspicion that his dice might not be entirely fair.

As students of TU Delft, we decided to investigate this issue statistically before confronting our friend. We used Pearson's chi-square test and were able to demonstrate, with high confidence ($p \leq 0.02$), that his d6 was indeed biased. However, we noticed that as the number of faces on a die increased, so did the number of samples required to achieve reliable results. At the time, we unknowingly began contemplating the relationship between the number of die faces (i.e., the underlying support) and the required sample size.

This thesis aims to analyze the chi-squared approximation of Pearson's test statistic to distributions with very large support S.

To achieve this, the report begins by formally introducing Pearson's chi-squared test statistic and discussing the challenges posed by discretizing a continuous distribution. The methodology is then outlined, including simulations, followed by the introduction of a Gaussian approximation. This approximation will be validated using the same statistical methods as previously applied.

Notation definitions

Table 1: Variables definitions

Notation	**Definition**
z_j	expected cell counts under the null
n_j	realized cell counts
π_j	probability that X is observed within interval $(a_j, a_{j+1}]$
$T(Z_1, \ldots, Z_n)$	Pearson test statistic
$\hat{T}(Y_1, \ldots, Y_n))$	Gaussian approximation for test statistic

2 Pearson's chi-squared test statistic

Firstly, we will revise Karl Pearson's Chi-squared test. Proposed by Pearson in 1900 (Agresti, 2007), the test takes categorical data $(x_1, \ldots, x_n)$ i.i.d., and compares them to the null distribution using the chi-squared distribution to calculate the probability of rejection. That is to say, the observations are classified into cells, which are then compared to the theoretical cell counts derived from the null distribution.

We introduce the following notation:

$$z_j : \textit{expected cell counts under the null hypothesis}$$
$$n_j : \textit{realized cell counts}$$

See Equation (2), and Equation (3) respectively.

Intuitively, as the differences between z_j, n_j, the more probable the rejection of the null hypothesis will be. To illustrate this intuition, Example 1 will continue on with the story of the Introduction.

Example 1 (DnD dice game). To continue on the introduction's scenario, the derived data would be the recorded amount of throws of the dice. The table below portrays our recorded values ($n = 200$):

Value	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Count	7	9	6	10	4	10	10	5	11	8	8	10	4	8	12	10	14	12	7	35

Table 2: Recorded data of d20 throws

Here n_j are the counts within the second row, and z_j is expected to be $\frac{200}{20} = 10$. The image belows displays a histogram of the data:

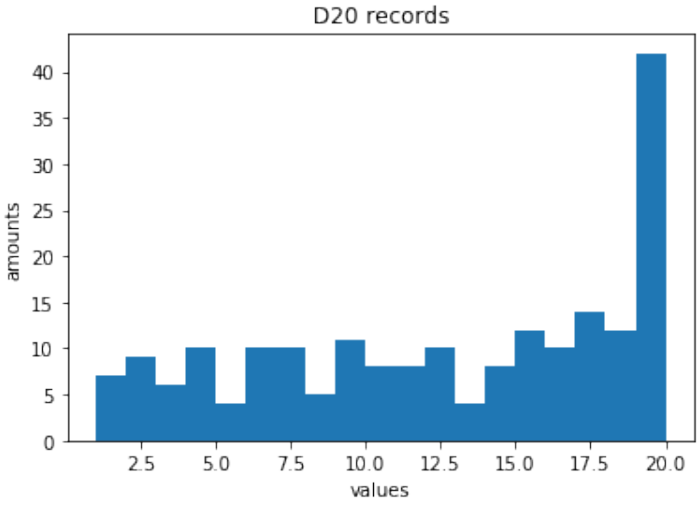

Figure 1: Histogram of record values of d20

The histogram clearly shows an overwhelming strange peak at value 20! Through application of Pearson's test, we statically concluded that these results could almost never occur with a fair dice. A question one could pose is now: How would the test change the more faces the die has?

This report studies how many cells there can be before errors occur (See section 3). We must first introduce notation to help. Firstly, initialize a partition of the domain into a total of m parts, with exterior bins such that most of the data is confined within the interior bins. Thus introduce the partition of the axis:

$$(-\infty, \infty) = (-\infty, a_1] \cup (a_1, a_m] \cup (a_m, \infty) = \bigcup_{j=1}^{m} (a_j, a_{j+1}] \cup [a_m, \infty) \cup (\infty, a_1]$$

The theoretical cell counts are determined by the expected frequencies, thus we determine the probability that an observation is observed in cell j as:

$$\pi_j = \mathbb{P}_0(x_t \in (a_j, a_{j+1}]) \tag{1}$$

For $t \in 1, ..., n$ and $j \in 1, ..m$.
From this definition it follows that the expected frequencies are simply,

$$z_j := n * \pi_j \tag{2}$$

For cell j. Continuing, the realized cell counts are defined;

$$n_j = \sum_{t=1}^{n} \mathbb{1}\{x_t \in (a_j, a_{j+1}]\} \tag{3}$$

Notably, n_j follows a $Binomial(n, \pi_j)$ distribution, as it is a realization of z_j. Therefore, one applies the Central Limit Theorem to approximate n_j as Gaussian random variable. Continued by normalizing, we find that for large n:

$$\frac{n_j - z_j}{\sqrt{z_j}} \sim N(0,1)$$

The Pearson's χ^2 statistic, denoted henceforth by $T(x_1, \ldots, x_n)$, calculates the differences of the theoretical and realized cell counts, finding:

$$T(X_1, \ldots, X_n) = \sum_{j=1}^{m} \frac{(n_j - z_j)^2}{z_j} \tag{4}$$

Notably, $T(x_1, \ldots, x_n)$ converges asymptotically in distribution to a χ^2_{m-1}.

Theorem 2.1. $T(x_1, \ldots, x_n) \xrightarrow[n\to\infty]{d} \chi^2_{m-1}$

See Section 4 for the associated proof.
Concluding, an introduction to new notation of X_t, which will be helpful in Section 4.

$$T(x_1, \ldots, x_n) = \sum_{j=1}^{m} \frac{(n_j - z_j)^2}{z_j} \tag{5}$$

$$= \left\| \begin{pmatrix} \frac{n_1 - z_1}{\sqrt{z_1}} \\ \vdots \\ \frac{n_m - z_m}{\sqrt{z_m}} \end{pmatrix} \right\|^2 \tag{6}$$

$$= \left\| \begin{pmatrix} \frac{\sum_{t=1}^{n} \mathbb{1}\{x_t \in (a_1, a_2]\} - n\pi_1}{\sqrt{n\pi_1}} \\ \vdots \\ \frac{\sum_{t=1}^{n} \mathbb{1}\{x_t \in (a_m, a_{m+1}]\} - n\pi_m}{\sqrt{n\pi_m}} \end{pmatrix} \right\|^2 \tag{7}$$

$$= \left\| \frac{1}{\sqrt{n}} \sum_{t=1}^{n} \begin{pmatrix} \frac{\mathbb{1}\{x_t \in (a_1, a_2]\} - \pi_1}{\sqrt{\pi_1}} \\ \vdots \\ \frac{\mathbb{1}\{x_t \in (a_m, a_{m+1}]\} - \pi_m}{\sqrt{\pi_m}} \end{pmatrix} \right\|^2 \tag{8}$$

$$:= \left\| \frac{1}{\sqrt{n}} \sum_{t=1}^{n} X_t \right\|^2 \tag{9}$$

In particular, for any $x \in \mathbb{R}$, we define the vector $X \in \mathbb{R}^d$ as

$$X = \begin{pmatrix} \frac{\mathbb{1}\{x \in (a_1, a_2]\} - \pi_1}{\sqrt{\pi_1}} \\ \vdots \\ \frac{\mathbb{1}\{x \in (a_m, a_{m+1}]\} - \pi_m}{\sqrt{\pi_m}} \end{pmatrix}.$$

Regarding the attributes of the above noted X_t's, firstly one notices the distribution of the indicator functions within the entries. Should they confirm with the null hypothesis, these are $Bernoulli(\pi_i)$ distributed. We will describe the correlation structure of X_t within Section 4.

3 Simulation of Pearson's statistic for large discretizations

Within this section, we shall display how the test statistic will incur type I errors when support is disproportionate with sample size. We will build a simple simulation of sample from a standard Gaussian pdf, and test the generated sample against being part of a standard Gaussian. Evidently as the null hypothesis is correct we expect the test not reject.

3.1 Discretization of a standard Gaussian

We generate a standard Gaussian, discretized between (-3,3], such that 99.7% of the sample are contained, adding both a left and right bin. Hence, taking the null as:

H_0*: the underlying distribution is standard Gaussian*

H_1*: the underlying distribution is not standard Gaussian*

The samples are generated from a standard Gaussian, we expect the test to generate p-values that are standard uniformly distributed.

Indeed, if the sample size is great enough compared to m, this does hold, which we will see in subsection 3.2. Figure 2 depicts the discretized null distribution partitioned with $m = 30$ bins.

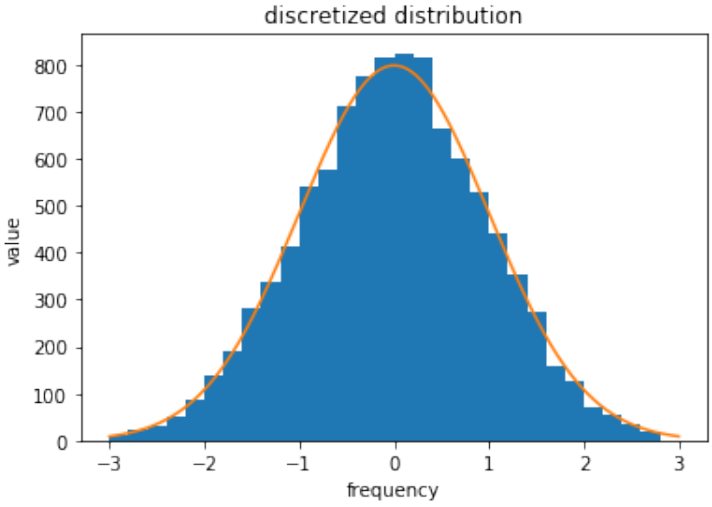

Figure 2: This picture shows visually how a standard Gaussian is partitioned

We partition the domain (-3,3) into m parts:

$$\mathcal{P} = \bigcup_{j=0}^{m} \mathcal{P}_j = \bigcup_{j=0}^{m} (-3 + j\Delta x, -3 + (j+1)\Delta x]$$

with $j \in [0, m]$ and $\Delta x = \frac{6}{m}$. Calculating the expected frequency of each bin: First calculating the probability of each bin.

$$\pi_j = \int_{a_j}^{a_{j+1}} f(x)dx = \Phi(-3 + (j+1)\Delta x) - \Phi(-3 + j\Delta x)$$

Where Φ is the cumulative distribution function of the Gaussian distribution. Then the expected frequency is $z_j = n\pi_j$. Performing Pearson's test on Figure 2 attains $T(x_1, \ldots, x_n) = 32.83$, with a p-value of $p = 0.2845$. Which aligns with the null hypothesis, as we expect the test to output p-values uniformly in $[0, 1]$. However, in Section 3.2 it shall be shown that this is not always the case. As it will be demonstrated that if the sample size is lacking, then the p-values will diverge from a uniform distribution and will tend to 0 erroneously. For the code of this subsection, please see Appendix 7.1.

3.2 Errors in method

Why would the p-values tend to 0? Because $T(x_1, \ldots, x_n)$ relies on the central limit theorem for each bin to converge in distribution to a standard Gaussian. Suppose we fix sample size n, while we increase cell amounts m, then each π_j will

decrease. Thus the expected value z_j will decrease. Leading to the difference between z_j, n_j becoming more sensitive. Which in turn results in greater overall differences, or a greater χ^2 value. Thus increasing the probability to reject. In order to illustrate this behavior. we repeated the simulation of Figure 2 with increasing cell counts m. Figure 3 shows the distributions of the p-values, which becomes more and more centered on 0 as we increase the cell counts.

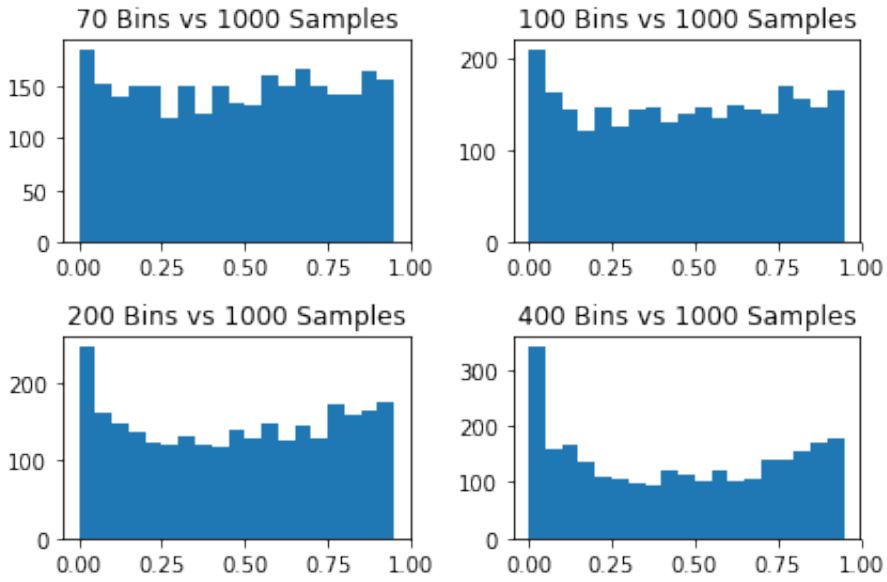

Figure 3: Histograms depicting 500 p-values of $T(X_1, \ldots, X_n)$ for n, m combinations

In order to simplify the comparison of the distribution of the p-values, we will make use of the Kolmogorov-Smirnov test (Kolmogorov, 1933)[**Ks test**]. Which agrees with our visual interpretation of Figure 3, which reject the uniformity of sub-figure 3 and 4 ($p_3 = 0.028, p_4 = 0.049$). This numerical value will aid us for comparing numerous combinations of sample amounts and cell amounts in the form of Figure 4.

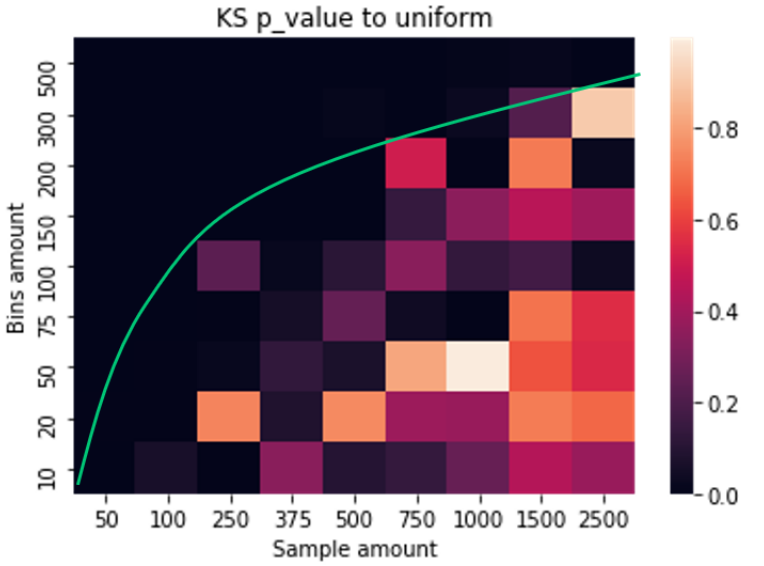

Figure 4: Heat-map of of KS test p-values of n, m combinations

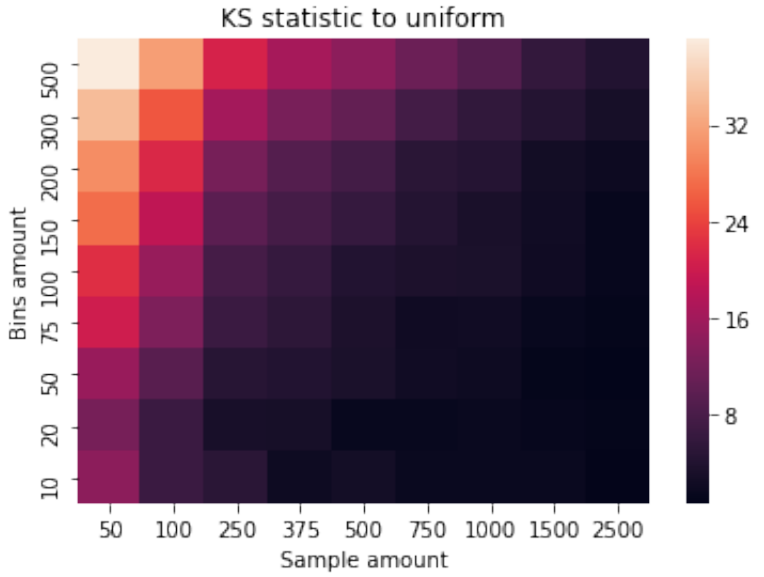

Figure 5: Heat-map of KS test values of n, m combinations, showing that for lower sample size compared to bin amount, the KS value drastically increases

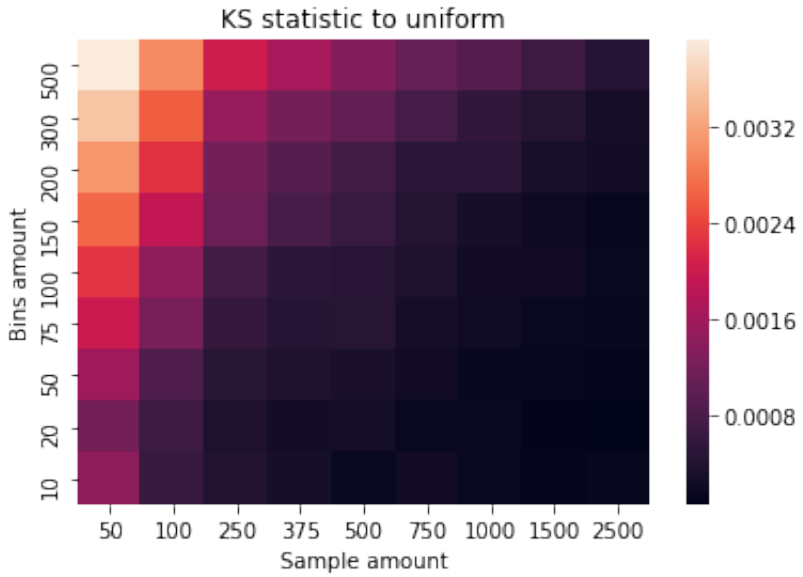

Figure 6: Heat-map of KS test values of n, m combinations, showing that for lower sample size compared to bin amount, the KS value drastically increases

The Figures above show the relationship of how the p-values deviate from the standard uniform distribution. In particular, Figure 4 shows us that the p-values tend heavily to 0 when the bin counts are too low. The figure depicts the relationship line on the green line. This effect is also shown in figure 5, where as the lighter the gradient becomes, the higher the value of the KS test. The details of the test are the code of Appendix 7.1 was used to run a within the next sections, we will discuss a Gaussian approximation which will perform better within the same parameters as described here.

The figure above highlights the main issue of $T(x_1, \ldots, x_n)$, which is shown behind the colored line in a clear dark curve. The colors within the heat map represent the KS test's p-value for uniformity. As our null hypothesis is indeed correct, we should get random p-values for the KS test rather than the KS test that constantly rejects higher values of m. The dark curve will be the main argument for the usefulness of the next two sections which will prove and employ a new Gaussian approximation which does not run into the dark curve under the same situation. In details, 10000 iterations of Pearson's test was run and over the p-values was the KS test performed to attain the above results, the code of which can be found in Appendix 7.1.

4 Gaussian approximation for Pearson's static

The previous section shows that $T(X_1, \ldots, X_n)$ leads to falsely reject the null if the discrepancy between the sample size and bins amount is too great comparatively. To study this relation, we shall take a look at a high-dimensional Gaussian approximation derived within [**Rpaper**].

Theorem 4.1. *Let $Z_1, \cdots, Z_n$ be independent, centered, d-variate random vectors that admit the bound $(\mathbb{E}||Z_t||^q)^{\frac{1}{q}} \leq b_t, t = 1, \cdots, n$ for some $q > 2$. On a different probability space, there exist independent random vectors $\tilde{Z}_t = Z_t$, in distribution, and independent Gaussian random vectors $Y_t \sim \mathcal{N}(0, Cov(Z_t))$, such that for some universal $C > 0$,*

$$\left(\mathbb{E} \left\| \frac{1}{\sqrt{n}} \sum_{t=1}^{n} (\tilde{Z}_t - Y_t) \right\|^2 \right)^{\frac{1}{2}} \leq \frac{C}{\min(\sqrt{q-2}, 1)} \left(\frac{d}{n} \right)^{\frac{1}{2} - \frac{1}{q}} \sqrt{\log n} \sqrt{\frac{1}{n} \sum_{t=1}^{n} b_t^2}.$$

Theorem 4.1 is in essence a version of the multivariate central limit theorem, as we are approximating a sum of independent random vectors with a Gaussian approximation. It even admits the expected bound between the two. We shall employ this to find the previous mention relationship. To begin, we shall make the approximation first explicit, followed, by proofs of needed tools. To finish we shall show that the approximation follows a chi-squared distribution and is approximately equal to the Pearson test statistic.

For clarity in the context of Theorem 4.1, we shall use 9 as the independent, centered random vectors.

Explicit form of the approximation

By Theorem 4.1 states that the approximation Y_t multivariate normal with mean 0 and covariance $\Sigma_t := \mathbb{C}ov(X_t)$. Thus we only need to determine the correlation structure of X_t. We begin to determine the entries on the main diagonal, which by the following calculations is $1 - \pi_j$. We introduce the notation $X_{t,j}$, as the vector X_t, indexed as j.

$$\begin{aligned}
\mathbb{C}ov(X_{t,j}, X_{t,j}) &= \mathbb{V}ar(X_{t,j}) \\
&= \mathbb{V}ar\left(\frac{\mathbb{1}\{x_t \in (a_j, a_{j+1}]\} - \pi_j}{\sqrt{\pi_j}} \right) \\
&= \frac{1}{\pi_j} \mathbb{V}ar(\mathbb{1}\{x_t \in (a_j, a_{j+1}]\}) \\
&= \frac{1}{\pi_j} \pi_j (1 - \pi_j) = (1 - \pi_j)
\end{aligned}$$

For non-main diagonal elements, we fix $j, k \in (1, \ldots, m)$, with $j \neq k$. To maintain a clear view, we denote the following $\mathbb{1}_j := \mathbb{1}\{x_t \in (a_j, a_{j+1}]\}$. Then:

$$
\begin{aligned}
\mathbb{C}ov(X_{t,j}, X_{t,k}) &= \mathbb{E}(X_{t,j}X_{t,k}) - \mathbb{E}(X_{t,j})\mathbb{E}(X_{t,k}) \\
&= \mathbb{E}\left(\frac{\mathbb{1}_j - \pi_j}{\sqrt{\pi_j}}\frac{\mathbb{1}_k - \pi_k}{\sqrt{\pi_k}}\right) - \mathbb{E}(\frac{\mathbb{1}_j - \pi_j}{\sqrt{\pi_j}})\mathbb{E}(\frac{\mathbb{1}_k - \pi_k}{\sqrt{\pi_k}}) \\
&= \mathbb{E}\left(\frac{1}{\sqrt{\pi_j\pi_k}}(\mathbb{1}_j\mathbb{1}_k - \pi_j\mathbb{1}_k - \pi_j\mathbb{1}_j + \pi_j\pi_k)\right) \\
&\quad - \frac{1}{\sqrt{\pi_j\pi_k}}(\mathbb{E}(\mathbb{1}_j) - \pi_j)(\mathbb{E}(\mathbb{1}_k) - \pi_k)) \\
&= \frac{1}{\sqrt{\pi_j\pi_k}}\left\{\mathbb{E}(\mathbb{1}_j\mathbb{1}_k) - \pi_j\mathbb{E}(\mathbb{1}_k) - \pi_k\mathbb{E}(\mathbb{1}_j) + \pi_j\pi_k\right\} \\
&\quad - \frac{1}{\sqrt{\pi_j\pi_k}}\left\{\mathbb{E}(\mathbb{1}_j)\mathbb{E}(\mathbb{1}_k) - \pi_j\mathbb{E}(\mathbb{1}_k) - \pi_k\mathbb{E}(\mathbb{1}_j) + \pi_j\pi_k\right\} \\
&= -\sqrt{\pi_j\pi_k}
\end{aligned}
$$

We combine these calculation to reach:

$$
\Sigma_t := \mathbb{C}ov(X_t) = \begin{pmatrix}
1-\pi_1 & -\sqrt{\pi_1\pi_2} & \dots & & \\
-\sqrt{\pi_2\pi_1} & 1-\pi_2 & \dots & & \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
 & & \dots & 1-\pi_{m-1} & -\sqrt{\pi_{m-1}\pi_m} \\
 & & \dots & -\sqrt{\pi_m\pi_{m-1}} & 1-\pi_m
\end{pmatrix} \tag{10}
$$

Note that Σ_t is independent of t, this followed from that X_t are i.i.d. Thus $\Sigma_t = \Sigma$ for all t. Thus:

$$
Y_t \sim N_m(0, \Sigma)
$$

Thus the Gaussian approximation of the Pearson's statistic will be:

$$
T(Y_1, \dots, Y_n) = \|Y\|^2 = \left\|\frac{1}{n}\sum_{t=1}^{n} Y_t\right\|^2 \tag{11}
$$

Lemmas

Now, we present two lemmas that will play a crucial role in the proof of Theorem 4.4.

Lemma 4.2. *Let $Y_t \sim N_m(0, \Sigma)$ for $t \in (1, \dots, n)$, with Σ as described in Equation (10). Then $T(Y_1, \dots, Y_n) = \|Y\|^2 = \left\|\frac{1}{n}\sum_{t=1}^{n} Y_t\right\|^2 \sim \chi^2_{m-1}$.*

Proof of Lemma 4.2. As Σ is real and symmetrical, by Spectral Theorem, it is able to be diagonalized.
Diagonalisation of (10): We are able to diagonalize Σ, thus $\exists O$ such that $O\Sigma O^T = diag(\lambda_i)$. Begin by defining the vector $\Pi = (\sqrt{\pi_1}, \ldots, \sqrt{\pi_m})^T$, then we observe that:

$$\Sigma = I - \Pi\ \Pi^T$$

The eigenvalues of Σ are the same as $1 - \lambda_m$ where λ_m are the eigenvalues of $\Pi\ \Pi^T$. Thus the eigenvalues of the outer product of the vector Π are given by $(\|\Pi\|, 0, ..., 0)$.

If we ensure the constraint that $\sum_{j=1}^{m} \pi_j = 1$, which is fulfilled as the discretization fully covers and does not overlap. Then:

$$||\Pi|| = \sqrt{\sum_{j=1}^{m} \sqrt{\pi_j}^2} = \sqrt{\sum_{j=1}^{m} \pi_j} = \sqrt{1} = 1$$

Thus plugging this into the previous observation leaves us with the eigenvalues of Σ, being $(0, 1, ..., 1)$.

We consider the eigenvectors corresponding to the eigenvectors. For $\lambda = 0$, we seen that eigenvector Π fits, as:

$$\begin{aligned} \Sigma\Pi &= \lambda\Pi \\ (\Sigma - \lambda I)\Pi &= 0 \\ \Sigma\Pi &= 0 \\ (I - \Pi\Pi^T)\Pi = \Pi - \Pi\Pi^T\Pi = \Pi - \Pi * 1 &= 0 \end{aligned}$$

To find the eigenvectors associated with $\lambda = 1$, one can employ the the Gramm-schmid process. These will not be explicitly calculated in this report. We shall index them as $(v^{(1)}, \ldots, v^{(d-1)})$. Thus we construct matrix O:

$$O = \begin{pmatrix} \vdots & \vdots & & \vdots & \vdots \\ v^{(1)} & v^{(2)} & \ldots & v^{(d-1)} & \Pi \\ \vdots & \vdots & & \vdots & \vdots \end{pmatrix}$$

By Spectral theorem, we find that $Y = O\Sigma O^T = diag(1, \ldots, 1, 0)$. We shall now determine the approximation to the statistic. Firstly, as O is an orthogonal matrix, we find:

$$||Y||^2 = ||OY||^2$$

Contingence of proof: If $Y = O^T Z$, then Z is normally multivariate $N(0, I_{m-1})$. Then Z is a vector $(Z_1, Z_2, \ldots, Z_{k-1}, 0)$, where the first $k-1$ coordinates are i.i.d. Gaussian random variables following $\mathcal{N}(0,1)$, and the last coordinate is zero

$$\begin{aligned}
\|Y\|^2 &= \left\|O^T Z\right\|^2 \\
&= \sum_{j=1}^{m} Z_i^2 \\
&= \sum_{j=1}^{m-1} Z_i^2 + 0 \\
&= \sum_{j=1}^{m-1} Z_i^2
\end{aligned}$$

Thus as we are summing up $m-1$ standard Gaussian random variables, and by definition:

$$T(Y_1, \ldots, Y_n) = \|Y\|^2 = \left\| \frac{1}{n} \sum_{t=1}^{n} Y_t \right\|^2 \sim \chi^2_{m-1}.$$

□

Lemma 4.3. *Let $z_1, \ldots, z_n \in \mathbb{R}^m$ and $y_1, \ldots, y_n \in \mathbb{R}^m$. Then*

$$\left| \sqrt{T(z_1, \ldots, z_n)} - \sqrt{T(y_1, \ldots, y_n)} \right| \leq \left\| \frac{1}{n} \sum_{t=1}^{n} (z_t - y_t) \right\|.$$

Proof of Lemma 4.3.

$$\begin{aligned}
\left| \sqrt{T(z_1, \ldots, z_n)} - \sqrt{T(y_1, \ldots, y_n)} \right| &= \left| \sqrt{\left\| \frac{1}{\sqrt{n}} \sum_{t=1}^{n} z_t \right\|^2} - \sqrt{\left\| \frac{1}{\sqrt{n}} \sum_{t=1}^{n} y_t \right\|^2} \right| \\
&= \left| \left\| \frac{1}{\sqrt{n}} \sum_{t=1}^{n} z_t \right\| - \left\| \frac{1}{\sqrt{n}} \sum_{t=1}^{n} y_t \right\| \right| \\
\textit{(Reverse Triangle Inequality)} &\leq \left\| \frac{1}{n} \sum_{t=1}^{n} z_t - \frac{1}{n} \sum_{t=1}^{n} y_t \right\| \\
&= \left\| \frac{1}{n} \sum_{t=1}^{n} (z_t - y_t) \right\|
\end{aligned}$$

□

The final tool that we need is the bound on $(\mathbb{E} \|X_t\|^q)^{\frac{1}{q}}$. Firstly, we note that for fixed t, there is only one singular indicator function that is nonzero. Denote the interval for which this is true, as $(a_k, a_{k+1}]$, then:

$$
\begin{aligned}
(\mathbb{E}||X_t||_2^q)^{\frac{1}{q}} &= \left(\mathbb{E} \left\{ \sum_{j=1}^{m} X_{t,j} \right\}^{\frac{q}{2}} \right)^{\frac{1}{q}} \\
&\leq \left(\mathbb{E} \left\{ \frac{(1-\pi_k)^2}{\pi_k} + \sum_{j=1}^{m} \pi_j \right\}^{\frac{q}{2}} \right)^{\frac{1}{q}} \\
&\leq \sqrt{\frac{1}{\min_{j\in(1,\ldots,m)} \pi_j} + 1} \\
&\leq \sqrt{\frac{2}{\min_{j\in(1,\ldots,m)} \pi_j}}
\end{aligned}
$$

Thus we define:

$$
b_t = \sqrt{\frac{2}{\min_{j\in(1,\ldots,m)} \pi_j}} \tag{12}
$$

Now the idea is to use X_t, as defined in (9), in Theorem 4.1, which forms an upper bound by Lemma 4.3, to show that $T(X_1, \ldots, X_n)$ is close to $T(Y_1, \ldots, Y_n)$. We shall now present this as follows:

Theorem 4.4. *Let Y_t, as in Theorem 4.1 and let f, the null probability density function, be bounded from below. Then*

$$
\left| \sqrt{T(x_1, \ldots, x_n)} - \sqrt{T(Y_1, \ldots, Y_n)} \right| \xrightarrow{p} 0. \tag{13}
$$

If one of the following holds:
a) d fixed and $n \to \infty$
b) $d, n \to \infty$ and $d = o(\sqrt{\frac{n}{\log(n)}})$

Proof of Theorem 4.4. We use Theorem 4.1, with b_t as described in Equation (12). We observe that b_t is independent of both q and t, meaning we can take q arbitrary. However, we choose that $q \to \infty$ such that we are able to find:

$$
\begin{aligned}
\left(\mathbb{E} \left\| \frac{1}{\sqrt{n}} \sum_{t=1}^{n} (X_t - Y_t) \right\|^2 \right)^{\frac{1}{2}} &\leq C b_t \sqrt{d} \sqrt{\frac{\log n}{n}} \\
(By\ Eq\ (12)) &\leq C \sqrt{\frac{2d}{\min_j \pi_j} \frac{\log n}{n}}.
\end{aligned}
$$

Proof of a): Let d fixed in $\mathbb{N}$, thus b_t is also fixed. Taking the limit of n, we find that using L'Hôpital's Rule :

$$\lim_{n\to\infty} Cb_t\sqrt{d}\sqrt{\frac{\log n}{n}} = \lim_{n\to\infty} Cb_t\sqrt{d}\sqrt{\frac{1}{n}} \to 0$$

Proof of b): Suppose we look at the limiting behavior as $n \to \infty$ and $d \to \infty$ simultaneously. In order to correctly evaluate the limit we need the order of $\min_j \pi_j$. So we consider;

$$\begin{aligned}
\min_j \pi_j &= \min_j \mathbb{P}_0(X \in (a_j, a_{j+1}])) \\
&= \min_j \int_{a_j}^{a_{j+1}} f_0(x)dx \\
&\geq \min_j f_0(x_j)\frac{a_{j+1} - a_j}{d} \\
&= \mathcal{O}(\frac{1}{d})
\end{aligned}$$

Thus as long as $\mathcal{O}(d^2\log(n)) < \mathcal{O}(n)$, we have that the right hand side goes to 0. Thus if $d = o(\sqrt{\frac{n}{\log(n)}})$, the RHS goes to 0.

We note the following of importance:

$$\begin{aligned}
&\left(\mathbb{E}\left\|\frac{1}{\sqrt{n}}\sum_{t=1}^{n}(\hat{X}_t - Y_t)\right\|^2\right)^{\frac{1}{2}} \to 0. \\
\implies &\mathbb{E}\left\|\frac{1}{\sqrt{n}}\sum_{t=1}^{n}(\hat{X}_t - Y_t)\right\| \to 0 \\
\implies &\left\|\frac{1}{\sqrt{n}}\sum_{t=1}^{n}(\hat{X}_t - Y_t)\right\| \xrightarrow{p} 0
\end{aligned}$$

In both cases of a) and b), we are able to combine these result with Lemma 4.3 to find that:

$$\left|\sqrt{T(x_1,\ldots,x_n)} - \sqrt{T(Y_1,\ldots,Y_n)}\right| \leq \left\|\frac{1}{n}\sum_{t=1}^{n}(X_t - Y_t)\right\| \xrightarrow{p} 0$$

□

In conclusion, we have found that for the above conditions:

$$T(x_1, \ldots, x_n) \approx T(Y_1, \ldots, Y_n) \tag{14}$$

And,

$$T(Y_1, \ldots, Y_n) \sim \chi^2_{m-1}.$$

5 Application to Power net Hertz Time series

This section will cover the application of the previously discussed theory. Using publicly available data on Power Grid Frequency from the public power grid frequency database (see [**powergridfrequency**]), we analyze data from January 2019 for Germany ($n \approx 10^6$). The dataset is a time series of frequency deviations from a baseline of 50 Hertz. The hypothesis is that these deviations are Gaussian-distributed. We examine the data for January to test for normality, observing the effects of different numbers of bins on the test results. In Figure 7, we plot the time series of the deviations for the month.

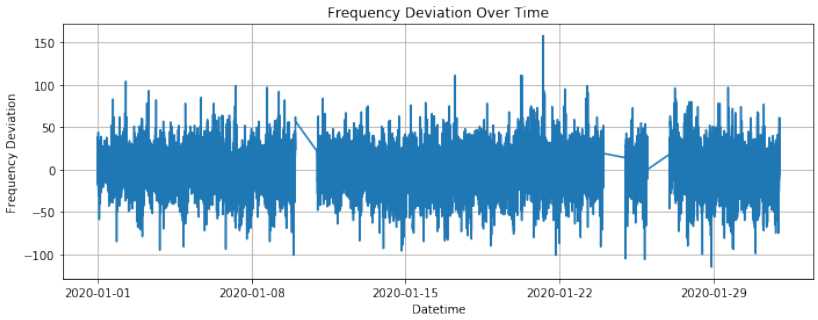

Figure 7: Time series of frequency deviations

Figure 8, plots the deviations as a histogram.

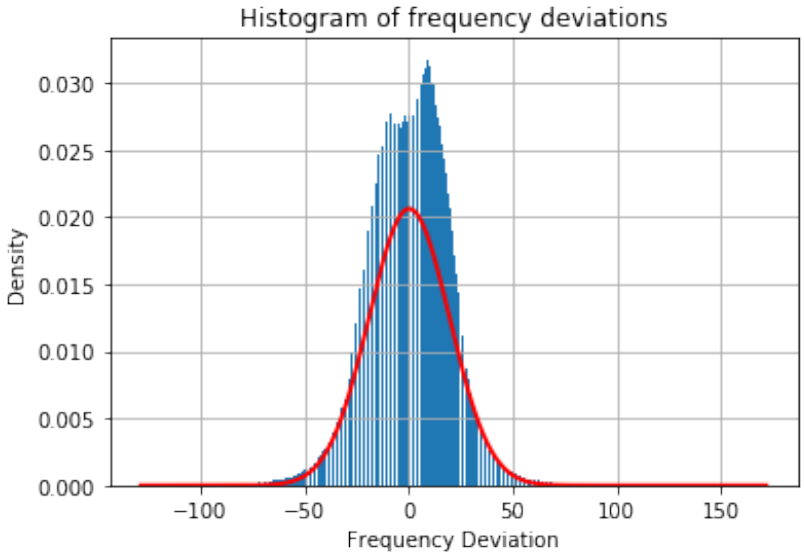

Figure 8: Histogram of frequency deviations

Figure 8 is quite apparent that the probability mass is overly concentrated at the center, exceeding the peak of the Gaussian distribution by nearly 0.010.

Likewise testing for normality for the deviations find the following results for different amount of bins.

d	Chi value	p value
202	5.148024e+06	0.0
405	1.260701e+07	0.0
164538	6.063210e+09	0.0

Table 3: Chi-squared values and corresponding p-values

Table 5 show that a Gaussian distribution is overwhelmingly rejected.

6 Conclusion

This paper has analyzed Pearson's chi-squared test and statistic, showing that the test statistic tends to return type I errors when the support increases too much relative to the sample size. This was visualized by a heatmap using the Kolmogorov-Smirnov test, which was employed to show the boundary function for how large the sample size must be given a cell count. This relationship was described by comparing the test statistic to its Gaussian approximation, which was used to derive a bound of $d = o\left(\sqrt{\frac{n}{\log(n)}}\right)$.

References

[1] Agresti, A. (2007) An Introduction to Categorical Data Analysis. 2nd Edition, John Wiley and Sons, Hoboken, New Jersey. https://mregresion.files.wordpress.com/2012/08/agresti-introduction-to-categorical-data.pdf

[2] Mies, Fabian, and Ansgar Steland. Sequential Gaussian Approximation for Nonstationary Time Series in High Dimensions. Bernoulli 29, no. 4 (2023): 3114-40. https://doi.org/10.3150/22-BEJ1577.

[3] Kolmogorov, A. N., & Smirnov, V. (1939). Tests of the goodness of fit. *Doklady Akademii Nauk SSSR*, 66, 3-4.

[4] Power Grid Frequency Project. Power Grid Frequency Database. 2024. Available at: https://power-grid-frequency.org/database/. Accessed: April 29, 2025.

7 Appendix

7.1 Validation 1 code

```
 %Replace it
#imports
import scipy
from scipy.stats import kstest
from scipy.stats import norm
from scipy.stats import chi2
import math as m
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#critical values
bins_amount  = 30
df=bins_amount-1
sample_amount= 10000

#Creating the bins
bound        = 3
step_size    = (2*bound/bins_amount)
bins=np.arange(-bound,bound,step_size)

#creating a list of expected frequencies
expected_frequencies=[]
for i in range(len(bins)-1):
    expected_frequencies.append(sample_amount*(norm.cdf(bins[i+1])-norm.cdf(bins[i])))
degrees_of_freedom=len(expected_frequencies)-1
```

```
#Generate n samples from std gaussian
samples=np.random.normal(0,1,sample_amount)
counts, bin_edges = np.histogram(samples, bins=bins)
counts

#compare the counts with the expected frequencies
chi=[]
for j in range(len(counts)):
    item=(counts[j]-expected_frequencies[j])**2/expected_frequencies[j]
    chi.append(item)
#print the value of chi-squared statistic
Chi=sum(chi)
print(Chi,degrees_of_freedom)

#discretized distribution
plt.hist(samples,bins=bins)
x_axis = np.arange(-3, 3, 0.001)
# Mean = 0, SD = 2.
plt.plot(x_axis, sample_amount/5*norm.pdf(x_axis,0,1))
plt.xlabel("frequency")
plt.ylabel("value")
plt.title("discretized distribution")
plt.show()

#p-value
chi2.sf(Chi,df=df)

#Do the previous steps numerous times to confirm uniformity of p-values
#creating repeated version
def ChiTest(bins_amount,sample_amount): #chi2 test for standard gaussian
    df=bins_amount-1
    #Creating the bins
    bound        = 4
    step_size    = (2*bound/bins_amount)
    bins=np.arange(-bound,bound,step_size)
    #creating a list of expected frequencies
    expected_frequencies=[]
    for i in range(len(bins)-1):
        expected_frequencies.append(sample_amount*(norm.cdf(bins[i+1])-norm.cdf(bins[i])))
    degrees_of_freedom=len(expected_frequencies)-1
    #Generate n samples from std gaussian
    samples=np.random.normal(0,1,sample_amount)
    counts, bin_edges = np.histogram(samples, bins=bins)
    #compare the counts with the expected frequencies
    chi=[]
    for j in range(len(counts)):
```

```
        item=(counts[j]-expected_frequencies[j])**2/expected_frequencies[j]
        chi.append(item)
    return(chi2.sf(sum(chi),df=df)) #returns the p-value of this iteration

#Repeating the chi2 test to bould sample space of p-values
#We maintain the amount of samples and bins amount for now
#we expect the p-values to ascertain a uniform(0,1) dstribution
#parameters
iterations=500
local_bins_amount   = 50
local_samples_amount= 3000*local_bins_amount

p_values=[]
for k in range(iterations): #do the test iterations amount of times
    p_values.append(ChiTest(local_bins_amount,local_samples_amount))

#Testing for uniformity for p values
plt.hist(p_values)
plt.ylabel("Frequency")
plt.title(f"p-values of chi2, d= {local_bins_amount}")
#Heavy parameters
iterations=50

#Iterative testing
def IteratedTesting(local_bins_amount,iterations):
    local_samples_amount=30*local_bins_amount
    p_values=[]
    for k in range(iterations): #do the test iterations amount of times
        p_values.append(ChiTest(local_bins_amount,local_samples_amount))
    return kstest(p_values,cdf="uniform")

#creating a list of bins amount with p-values of KS test
Bins=[50,100,200,400,800,2000]
data=[]
for bin in Bins:
    data.append((bin,IteratedTesting(bin,iterations)[1]))
#creating a dataframe
colums =["Bins amount", "p-value"]
Df=pd.DataFrame(data=data,columns=colums)
Df
```

7.2 Critical values

The following code was done in MatLab:

```
%%
```

```
function a_quantile = FindCriticalValue(N)
    alpha = 0.05;
    pi = [0; 0];
    Sigma = [1 - pi(1), -sqrt(pi(1)*pi(2)); -sqrt(pi(1)*pi(2)), 1 - pi(2)];
    mu = [0; 0];M=200;

    % Generate a range of values from 0 to 1 (with a reasonable number of points)
    A = linspace(0, 10, 3000);  % Adjust this range as needed for your problem
    lst = NaN(1, length(A));  % Initialize lst as NaN to track critical values

    for a = 1:length(A)
        % p(T(Y) > a) <= 0.05
        running_total = 0;
        for n = 1:N
            % Draw samples from multivariate normal distribution
            Y = mvnrnd(mu, Sigma, M);  % Generate one sample of size 1
            Y2 = (1/M)*sum(Y.^2);  % Sum of squares of the components

            if Y2 > A(a)  % Compare the sum of squares with the current threshold
                running_total = running_total + 1;
            end
        end

        probability = running_total / N;  % Estimate probability

        % If probability is <= alpha, we store this value of A as a candidate for the critic
        if probability <= alpha
            lst(a) = A(a);  % Store the threshold corresponding to this probability
        end
    end

    % Find the minimum value of A where the condition holds
    lst = lst(~isnan(lst));  % Remove NaN values
    a_quantile = min(lst);  % The critical value corresponding to alpha
end

%%
% Example usage
a_quantile = FindCriticalValue(1000);
disp(['The critical value is: ', num2str(a_quantile)]);
```

7.3 Application code

```
    #BEP APPLICATION TO ENERGY GRID
#Libs
```

```
import math as m
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import chi2
from scipy.stats import norm
from scipy.stats import laplace
#dataset
# Load the Excel file (replace 'your_file.xlsx' with your actual file name)
# Read the CSV file (assuming no header)
df = pd.read_csv('germany_2020_01.csv', header=None, names=['datetime', 'frequency deviation

#remove jumps using '#######'
df = df[~df.astype(str).apply(lambda x: x.str.contains('#')).any(axis=1)]

# Convert the columns to appropriate data types
df['datetime'] = pd.to_datetime(df['datetime'], errors='coerce')
df['frequency deviation'] = pd.to_numeric(df['frequency deviation'], errors='coerce')

# Drop rows with failed conversions (optional but recommended)
df = df.dropna(subset=['datetime', 'frequency deviation'])

#remove the first row, as it contains only NaN values
df = df.iloc[1:].reset_index(drop=True)

# Remove jumps and convert data types
df = df[~df.astype(str).apply(lambda x: x.str.contains('#')).any(axis=1)]
df['datetime'] = pd.to_datetime(df['datetime'], errors='coerce')
df['frequency deviation'] = pd.to_numeric(df['frequency deviation'], errors='coerce')
df = df.dropna(subset=['datetime', 'frequency deviation'])
df = df.iloc[1:].reset_index(drop=True)
#Initial vieuwing of the data set
sample_mean = np.mean(df['frequency deviation'])    #sample mean
sample_variance = np.var(df['frequency deviation']) #sample variance
n = len(df['frequency deviation'])                  #sample amount
df[["datetime", "frequency deviation"]].describe()  #quick look at the core properties of th
# Plot histogram normalized to form a probability density
plt.hist(df['frequency deviation'], bins=200, density=True)

# Generate x values within the data range
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 10000)

# Compute the Gaussian (normal) distribution
p = norm.pdf(x, sample_mean,np.sqrt(sample_variance))
```

```
# Plot the Gaussian curve
plt.plot(x, p, 'r', linewidth=2)
plt.title('Histogram of frequency deviations')
plt.xlabel('Frequency Deviation')
plt.ylabel('Density')
plt.grid(True)
plt.show()
#line plot of the frequency deviations over time
plt.figure(figsize=(10, 4))
plt.plot(df['datetime'], df['frequency deviation'])
plt.xlabel('Datetime')
plt.ylabel('Frequency Deviation')
plt.title('Frequency Deviation Over Time')
plt.grid(True)
plt.tight_layout()
plt.show()
#The Chi-squared test: Gaussian test
def Chi_Test(df, bins_amount, mean, variance):
    # define parameters
    n = len(df['frequency deviation'])  # sample amount
    d = bins_amount                     # bins amount

    # Create probability-equal bins
    bins = norm.ppf(np.linspace(0, 1, d + 1), loc=mean, scale=np.sqrt(variance))

    # Creating a list of expected frequencies under the null hypothesis
    cdf_values = norm.cdf(bins, loc=mean, scale=np.sqrt(variance))
    expected_frequencies = n * np.diff(cdf_values)  # Corrected here

    # Counting hits to bins
    counts, bin_edges = np.histogram(df['frequency deviation'], bins=bins)

    # Chi-square statistic
    counts = np.asarray(counts)
    expected_frequencies = np.asarray(expected_frequencies)

    Chi = np.sum((counts - expected_frequencies) ** 2 / expected_frequencies)

    # Degrees of freedom = d - 1 (usually) - for goodness of fit
    p_value = chi2.sf(Chi, d - 1)

    return Chi, p_value
    #testing the dataset with different bins amount for normally
# Calculate the d values
d_optimal = m.floor(m.sqrt(n/m.log(n)))  # proposed optimal bin amount
d_small   = m.floor(d_optimal/2)
```

```
d_great   = m.floor(n/m.log(n))

# Create a list of d values to test
d_values = [d_small, d_optimal, d_great]

# Initialize lists to store results
chi_values = []
p_values = []

# Perform Chi-square test for each d value
for d in d_values:
    chi, p = Chi_Test(df, d, sample_mean, sample_variance)
    chi_values.append(chi)
    p_values.append(p)

# Create a results table
results_table = pd.DataFrame({
    'd': d_values,
    'Chi_value': chi_values,
    'p_value': p_values
})

# Display the table
print(results_table)
```