Article citation info:
Chmielowiec A, Klich L. Application of python libraries for variance, normal distribution and Weibull distribution analysis in diagnosing and operating production systems. Diagnostyka. 2021;22(4):89-105. https://doi.org/10.29354/diag/144479.

DIAGNOSTYKA, 2021, Vol. 22, No. 4

ISSN 1641-6414
e-ISSN 2449-5220
DOI: 10.29354/diag/144479

APPLICATION OF PYTHON LIBRARIES FOR VARIANCE, NORMAL DISTRIBUTION AND WEIBULL DISTRIBUTION ANALYSIS IN DIAGNOSING AND OPERATING PRODUCTION SYSTEMS

Andrzej CHMIELOWIEC [1], Leszek KLICH [2]

[1] Rzeszow University of Technology, achmie@prz.edu.pl
[2] Rzeszow University of Technology, l.klich@prz.edu.pl

Abstract

The use of statistical methods in the diagnosis of production processes dates back to the beginning of the 20th century. Widespread computerization of processes made enterprises face the challenge of processing large sets of measurement data. The growing number of sensors on production lines requires the use of faster and more effective methods of both process diagnostics and finding connections between individual systems. This article is devoted to the use of Python libraries to effectively solve some problems related to the analysis of large data sets. The article is based on the experience related to data analysis in a large company in the automotive industry, whose annual production reaches 10 million units. The methods described in this publication were the basis for the initial analysis of production data in the plant, and the obtained results fed the production database and the created automatic anomaly detection system based on artificial intelligence algorithms.

Keywords: variance analysis, normal distribution, Weibull distribution, statistical analysis, python

ZASTOSOWANIE BIBLIOTEK JĘZYKA PYTHON DO ANALIZY WARIANCJI, ROZKŁADU NORMALNEGO I ROZKŁADU WEIBULLA W DIAGNOSTYCE I EKSPLOATACJI SYSTEMÓW PRODUKCYJNYCH

Streszczenie

Wykorzystywanie metod statystycznych w diagnostyce procesów produkcyjnych sięga swoimi korzeniami początków XX wieku. Powszechna informatyzacja procesów postawiła przedsiębiorstwa przed wyzwaniem przetwarzania dużych zbiorów danych pomiarowych. Rosnąca liczba czujników na liniach produkcyjnych wymaga stosowania szybszych i skuteczniejszych metod zarówno diagnostyki procesu, jak i znajdowania powiązań pomiędzy poszczególnymi systemami. Niniejszy artykuł został poświęcony wykorzystaniu bibliotek języka Python do efektywnego rozwiązywania niektórych problemów związanych z analizą dużych zbiorów danych pomiarowych. Artykuł powstał na bazie doświadczeń związanych z analizą danych w dużym przedsiębiorstwie branży motoryzacyjnej, którego roczna produkcja sięga 10 milionów sztuk. Opisane w niniejszej publikacji metody stanowiły podstawę wstępnej analizy danych produkcyjnych we wspomnianym zakładzie, a uzyskane wyniki zasiliły bazę danych produkcyjnych oraz tworzony system automatycznego wykrywania anomalii oparty na algorytmach sztucznej inteligencji.

Słowa kluczowe: analiza wariancji, rozkład normalny, rozkład Weibulla, analiza statystyczna, python

1. INTRODUCTION

Serial production reaching tens or even hundreds of millions of pieces a year is not unusual these days. Modern factories full of robots and automation continuously produce huge amounts of goods. However, the level of automation requires proper control of the production processes. It is usually connected with the necessity of installation of a large number of sensors that record the condition of machines and the quality of manufactured products. The number of sensors is as high as 10^3 in the case of a medium-sized company, and even 10^5 in the case of very large factories. Therefore, the production state at time t can be described as series of measurement data $S_t = (s_{1,t}, s_{2,t}, \dots, s_{m,t})$, where $s_{i,t}$ are the measurement results from individual sensors. By writing down the state of the company in successive moments of time $t_1 < t_2 < \cdots < t_r$ we obtain a matrix that reflects the changes in the production process:

$$(s_{i,t_j}) = \begin{pmatrix} s_{1,t_1} & s_{2,t_1} & \cdots & s_{m,t_1} \\ s_{1,t_2} & s_{2,t_2} & \cdots & s_{m,t_2} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1,t_r} & s_{2,t_r} & \cdots & s_{m,t_r} \end{pmatrix}. \quad (1)$$

Data from the measurement matrix are analyzed in order to detect abnormalities or specific relationships between events. Even an apparently small matrix becomes computationally very demanding if it is

necessary to independently analyze the sub-matrices contained in it. Statistical tools are very often used to determine interrelationships between events and detect irregularities. They allow to automatically process the data contained in such a matrix and indicate interesting relationships between the variables that make up the state of the production process. This article presents the basic tools of statistical analysis that can be used in detecting dependencies and interrelationships between the values of the presented matrix.

The study of the statistical properties of time series from the production process is a very common method of assessing the quality of this process today. Advanced methods of data analysis allow you to control the quality of the process, assess the reliability or test the resistance of the design. The theory of reliability draws a lot from statistical methods, as evidenced, inter alia, by the monographs of Barlow and Proschan [10], Ansell and Phillips [5], Johnson and others [41], Birolini [11], Woo [94], Grynchenko and Alfyorov [30]. Her research area is the variability of the quality function over time, which is very well expressed in terms of the probability calculus. Schedules modeling the life cycle of machines and devices allow for effective management of production lines - their operational reliability and quality of manufactured elements. A particularly important concept for this field is the Weibull distribution [91, 92], whose precursors were Frechet [29] and Fisher and Tippett [27]. This distribution plays a special role in the theory of reliability, as evidenced by, for example, the publications of Johnson [40] and Lai [49]. A lot of detailed information about it can be found, inter alia, in the monographs of Murthy [65], Lai [50] and McPherson [61]. The statistical process control [63, 95, 62] initiated by Shewart [81] is today a highly developed method of managing the production process. It covers both the analysis of single variable functions [63, 95] and the analysis of multivariable functions [56, 59]. The robust design methods introduced by Taguchi [83] also make significant use of a variety of statistical tools. Their use in production companies resulted in a two-fold [43], and in some cases even a four-fold reduction in the variability of the production process [70]. The breakthrough achievements in this field include the results published by: Kacker [42], Leon et al. [52], Box [13], Nair [66] and Tsui [87]. Taguchi's methods have also been extended to design resistance based on multiple characteristics. These problems are raised, inter alia, by: Logothetis and Haigh [55], Pignatiello [71], Elsayed and Chen [23] and Tsui [88]. The common feature of the issues described above is the intensive use of statistical tools on data sets from the production process. It should be emphasized that as the number of sensors and the size of databases increases, more and more emphasis is placed on processing efficiency. Therefore, the main task of this publication is to show how modern IT tools and numerical methods help to deal with some problems of data analysis.

Statistical process control is part of a much wider problem of statistical analysis, which is searching for various types of anomalies in a time series. This problem has been intensively researched over the last 20 years. Many algorithms and techniques for finding anomalies have been proposed, provided that the range of an anomaly is known more or less. Examples include the results obtained by Keogh et al. [45, 46] and Senin et al. [77]. Nevertheless, it is still a huge challenge to search the entire set of available data. As an example illustrating the level of complexity of the problem, we can use a one-dimensional sequence 10^5 of observations of a single quantity - it may be, for example, one of the dimensions of the manufactured element. Let us emphasize that a production run of this size is nothing extraordinary and is easily achieved in batch production conditions. There are over $1.6 \cdot 10^{14}$ subseries for such a series, which may contain various types of anomalies. This example shows how complicated the situation is when there is no information about the potential location of an anomaly. The publication [17] gives some intuitions about trying to solve this problem from the algorithmic point of view, but it can be said that this is only the discovery of the tip of the iceberg. It should be emphasized that the level of complexity of such problems increases significantly when data matrices appear instead of simple sequences/vectors. A good example of the complexity of this issue is the review article by Ebner and Henze [22], which describes the methods and problems of testing normality in multidimensional spaces.

Contemporary methods of finding anomalies in time series can be divided into three main groups depending on the results they generate. We distinguish algorithms for finding anomalies at a point, structural anomalies and series anomalies (in the case of multiple series). By an anomaly at a point, we understand the deviation of the values of a single measurement from the values in the series [32, 25]. In turn, structural anomalies are such subsets of a given series, whose statistical properties differ from those determined for the entire series [44, 97, 77]. In some way, series anomalies are related to this issue, which consist in finding deviations between entire sequences of measurements [38, 51]. Algorithms using artificial intelligence methods are gaining popularity recently. This is undoubtedly a future direction of research, as evidenced by, for example, Intel's commitment to this area. The experience gained by the company in this area was published by Wang et al. [90]. Also, the publication by Chalapathy and Chawla [15] provides an overview of deep machine learning methods for detecting anomalies in time series.

The article [21] introduces the division of anomaly detection algorithms in time series according to the type of the method used. Ding and others

distinguished between the following methods: classification, nearest neighborhood, clustering, and statistical methods. It should be emphasized that all the last three methods are more or less based on statistical inference and the calculus of probability. Therefore, the rapid determination of statistics is a very important issue in the context of the performance of anomaly detection systems.

This publication is divided into two parts. The first part presents selected methods of statistical analysis from the theoretical point of view, and the second part presents a practical approach - the implementation of the previously described methods in Python. Of course, the approach to big data analysis described in the following sections does not exhaust the catalog of methods that can be used in such a context. Nevertheless, the three issues described constitute a kind of basis that can be used for more advanced calculations and machine learning.

2. THEORETICAL FOUNDATIONS OF SELECTED METHODS OF STATISTICAL DIAGNOSTICS

In this section, the methods of variance analysis and two probability distributions will be considered. They play a key role from the point of view of quality management and reliability theory. These distributions are the normal (Gaussian) distribution and the Weibull distribution. The description of individual will be presented with particular emphasis on the implementation aspect. The automation of the results evaluation and the calculation speed are extremely important in the case of processing and searching large measurement data sets. At the end of the part, the algorithm for processing a large set of measurement data is presented. Its task is to supplement the database with additional statistics that can be used for graphical analysis and machine learning.

2.1. Variance analysis

Variance is a measure of the concentration of data around the mean value. The lower the variance, the more concentrated the results. On the other hand, a high value of the variance indicates the statistical dispersion and significant distances between the points of the analyzed data set. Under production conditions, high process variability may indicate quality problems. Therefore, it is worth starting the data analysis with the analysis of variance. At this point, it should be noted that the process analysis cannot assume the study of variance only within measurement sequences with a certain length. Determining the time window size can give an incomplete picture of the process. Therefore, in further considerations, a chronologically ordered measurements sequence $x_1, \dots, x_n$ representing the values of one of the random variables defined by the production process will be taken into account. Suppose that for each element x_i a sequence of variances will be determined that will be computed for the surroundings of this element. It means that a certain sequence of radii $\delta_1, \dots, \delta_m$, has been established, for which we calculate the variances

$$v_{i,j}(x_{i-\delta_j}, \dots, x_{i+\delta_j}) = \frac{1}{D_j}\sum_{k=i-\delta_j}^{i+\delta_j} x_k^2 - \left(\frac{1}{D_j}\sum_{k=i-\delta_j}^{i+\delta_j} x_k\right)^2, \quad (2)$$

where $D_j = 2\delta_j + 1$. Note that the biased variance estimator was intentionally used in the formula above. This is due to the fact that it is more efficient to implement, and the transition to the unbiased estimator requires only multiplying the obtained quantity by $1 + (2\delta_j)^{-1}$. Additionally, in this article we will assume that successive rays δ_j increase by a certain predetermined amount w. It means that $\delta_j = j \cdot w$. For such assumptions, it can be shown that the independent determination of all variances $v_{i,j}$ for $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$ requires about $mn^2/6$ operations. If we assume that $m = n/w$ for some fixed w, then the computational complexity of the algorithm determining the set of variances $v_{i,j}$ has order $O(n^3/w)$. In practice, it means that for a small number of 10^4 measurements and value $w = 100$ it is necessary to perform 10^{10} operations. Therefore, to determine the entire set of variances, we will use the approach presented in Lemma 1 [17]. It will allow to reduce the computational complexity of the problem under consideration to the level of $O(n^2/w)$, which will significantly improve the efficiency of calculations. For example, for the aforementioned 10^4 measurements, the computation time will be reduced 10 000 times.

The Welford formula [93, 14] in the form given by Knuth [48] will be used for the effective implementation of the algorithm for determining the set of variances $v_{i,j}$. It assumes that if $\mu(s), v(s)$ are the arithmetic mean and variance of the series of measurements s, respectively, then for $s = (x_1, \dots, x_k)$ and $s^+ = (x_1, \dots, x_k, x_{k+1})$ the following dependencies occur

$$\mu(s^+) = \mu(s) + \frac{1}{k+1}(x_{k+1} - \mu(s)),$$

$$v(s^+) = v(s) + \big(x_{k+1} - \mu(s)\big)\big(x_{k+1} - \mu(s^+)\big) \quad (3)$$

This formula will be used in the initial stage of calculations - until the sequence s reaches the size $2\delta + 1$. In the next stage, the window method presented in [17] will be used. It uses the fact that for $s = (x_1, \dots, x_k)$ and $s' = (x_2, \dots, x_{k+1})$ the following equations are satisfied

$$\mu(s') = \mu(s) + \frac{x_{k+1} - x_1}{k},$$

$$v(s') = v(s) + \frac{x_{k+1} - x_1}{k}\left(x_1 + x_{k+1} - \mu(s) - \mu(s')\right) \quad (4)$$

The above-mentioned relationships allow for the definition of a computationally effective algorithm, which was presented in [17].

2.2. Normal distribution

The normal distribution as a limit distribution for the law of large numbers appears very often in the practice of quality management. Many models assume that the features of a process or a product can be modeled by means of a random variable $X = \mu + Y$, where μ is a fixed value and the random variable Y has a normal distribution. However, this condition is not always met. There are moments when a random variable ceases to have a normal distribution. Detecting such a situation is crucial from the quality control point of view, as it may indicate a disturbance in the production process. Therefore, methods for verifying the normality of decomposition are a very important part of the control of the production process.

Testing the normality of a distribution is already a classic problem in probability theory and statistics. The first analyzes of this issue were already carried out by Fisher [26] and Pearson [69] in the interwar period. There are many examples of testing a one-dimensional normal distribution. One of the most frequently used tests are the Anderson Darling [4], Shapiro-Wilk [79], Shapiro-Francia [78] and Kolmogorov-Smirnov [60, 54] tests. On the other hand, the analysis of skewness and kurtosis (third and fourth moments) was used to construct the first normality tests for multivariate distributions [9, 57, 58]. At the end of the 20th century, Bowman [12] proposed a normality test of multidimensional distributions based on the smoothness of density. Vasicek, in turn, developed a test using the entropy of the normal distribution [89]. The literature also offers several proposals for tests based on characteristic functions, which include [24, 18, 37], the BHEP test [8, 36] and energy tests [82]. Testing the normality of distribution, especially in the multivariate case, is currently a very intensively researched issue. This is evidenced by the publications of recent years by authors such as Mori et al. [64], Henze and Visagie [35], Tenreiro [84], Thas and Ottoy [85] and Zhu et al. [98]. Extensive reviews of methods for testing normality of distribution can be found in the works of Henze [34] and Das and Imon [19].

Recall that the normal distribution is a two-parameter distribution denoted as $\mathcal{N}(\mu, \sigma^2)$, where μ is the expected value and σ is the standard deviation. Figure 1 shows the probability density plot for the parameters $\mu = 0$ and $\sigma = 1$ along with the marked intervals $[-1,1]$, $[-2,2]$ i $[-3,3]$, the integral of which is respectively 0.683, 0.954 and 0.997. This integral determines the probability with which the random variable assumes values from the specified interval. The density function for the normal distribution is given by the formula

$$f_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (5)$$

It is not the purpose of this article to present a systematic description of all known tests to verify the normality of a distribution. The extensive literature provided at the beginning of this section allows the reader to get acquainted with specific methods. Examples of the use of selected methods will be presented later in the article as part of the use of specific libraries for statistical analysis. However, in order to outline the complexity of statistical testing, the Shapiro-Wilk test description will be presented. This test is today considered to be one of the strongest tests of normality. Its main disadvantage, however, is numerical limitations that make it impossible to test large samples. Currently used libraries for statistical analysis allows to carry out the test on vectors consisting of about 5000 samples. Nevertheless, it is recommended to verify the maximum number of input data that can be properly examined by a given implementation in the numeric packet documentation.

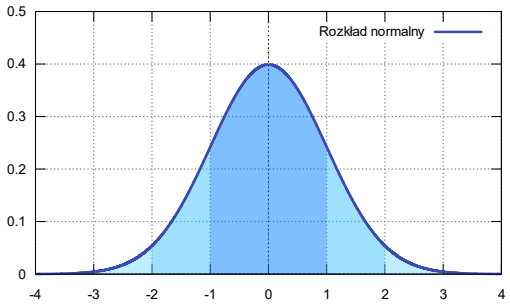

Fig. 1. Normal distribution density function for parameters $\mu = 0$ i $\sigma = 1$

The essence of the Shapiro-Wilk test is to compare the variance of the sample with the variance that should have a normal distribution if the data actually came from such a distribution. Thus, this test answers the question to what extent the sample has a chance to represent a normal distribution. The Shapiro-Wilk test statistic is

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_i\right)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}, \quad (6)$$

where the sequence $x = (x_1, \ldots, x_n)$ is a sequence of positional statistics (the sequence terms are sorted in ascending order) and $\overline{x}$ is the average value of the given statistics sequence. Generally, the values of the W statistics are numbers in the range [0,1]. When a sample derived from a normal distribution is tested, the test statistic W tends to unity as the sample size increases. The biggest problem with this test is finding the vector of coefficients a, which the equation should satisfy

$$a = \frac{V^{-1}m}{(m^T V^{-1} V^{-1} m)^{1/2}}, \tag{7}$$

where m is the vector of expected values for the sample of positional statistics of size n, and the matrix V is the covariance matrix of the position statistics from the sample and the vector m. Computational problems related to the determination of vector a coefficients are a fairly current topic. They were already mentioned by Shapiro and Wilk [79], Royston [75, 74, 76] had a large contribution to this issue, and recently Gunner and others [31] presented a method using the Shapiro-Wilk test in fast signal processing. It should be noted that operating on vector a approximations is also a fairly common approach. One of the most commonly used relationships is

$$a \approx \frac{\hat{m}}{(\hat{m}^T \hat{m})^{1/2}}, \tag{8}$$

where $\hat{m}_i = \Phi^{-1}((i - 0.375)/(n + 0.25))$.

The abbreviated description of the Shapiro-Wilk test presented above is only a sketch of the problems related to this test and is only intended to draw the reader's attention to the fact that the problem of testing the normality of distribution is a very extensive topic. Therefore, before using specific functions that test normality, it is worth taking the time to become familiar with the properties of the test you want to use. Particular attention should be paid to the limitations of a given method.

2.3. Weibull distribution

As mentioned in the introduction, the Weibull distribution is an example of a distribution that models the lifetime of a product. Recent years have shown its intensive use in issues related to modeling: glass strength [47], progressive pitting corrosion [80], adhesive wear of metals [72], failure of coatings [3], failure of brittle materials [28], failure of composite materials [67], wear of concrete elements [53], fatigue life of aluminum alloys [33], fatigue life of Al-Si castings [1], strength of polyethylene terephthalate fibers [96] or failure rate of joints under the influence of shear [7]. This very wide range of applications, however, does not require automatic processing of large data sets. However, it shows how universal the Weibull distribution is when it comes to testing the failure rate and aging of products.

The Weibull distribution was treated in a completely different way in the publication [16], which shows how this distribution can be used to optimize operating costs. The generalization of the approach presented there, for example for all replacement parts used in a large enterprise, is already associated with the statistical and optimization analysis of a large set of data (in general, the number of replacement parts used reaches thousands). Another potential application may be the optimization of warranty servicing costs of cars sold by a given manufacturer. In this context, we are dealing with an even larger database, as generally a car model is sold in hundreds of thousands, and sometimes even millions of copies.

The probability density of the three-parameter Weibull distribution is given by the equation

$$f(t) = \frac{\beta}{\alpha}\left(\frac{t-\tau}{\alpha}\right)^{\beta-1} e^{-((t-\tau)/\alpha)^{\beta}}, \tag{9}$$

where $\beta > 0, \tau \geq 0, t \geq \tau$. For the adopted notations, α is called the scale parameter, β is the shape parameter, and τ is the position parameter. Figure 2 shows example plots of the probability density function f for the Weibull distribution.

An overview of possible methods can be found in the publications of Ross [73] and Jacquelin [39]. It is worth emphasizing here that the form of the Weibull distribution gives the possibility of using a substitution that transforms it into a linear relationship. Thanks to this, it is very easy to connect the methods based on probability plots [86, 6, 20, 68, 2] with the method of determining parameters using linear regression. It can therefore be concluded that this distribution fits exceptionally well in the automatic analysis of large data sets.

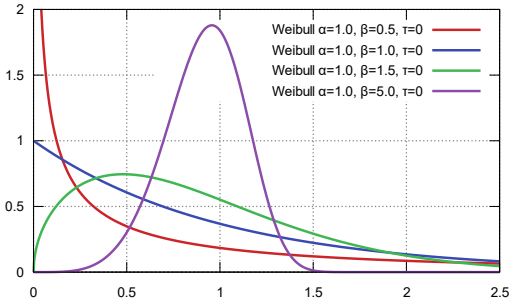

Fig. 2. Sample probability density plots for the Weibull distribution for various parameters α, β and fixed $\tau = 0$

2.4. Anomaly detection algorithm

Let us assume that we have the time series $F(t) = (x_1, x_2, \ldots, x_m)$ defined on the discrete set of arguments $t_1, t_2, \ldots, t_n$. We treat t_i arguments as the moments of time in which the enterprise's operating parameters were measured, represented by a tuple $(x_1, x_2, \ldots, x_m)$. Let us also assume that the search for anomalies in the production process is carried out based on the time window of the size w. That is, the parameters corresponding to the arguments $t_{i+1}, t_{i+2}, \ldots, t_{i+w}$ are subjected to statistical analysis. Of course, the Algorithm 1 presented in this section may be applied to multiple time window widths as needed. Nevertheless, we present it in the version for fixed in.

Algorithm 1: Anomaly detection data base

INPUT: w – window size, h – half window size, $t_1, t_2, \ldots, t_n$ – discrete time series, $F(t) = (x_1, x_2, \ldots, x_m)$ – measured parameters function represented as matrix $(x_{i,j})$ where $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$.

OUTPUT: $R_j(t) = (k_1, k_2, V, N, W)$ – function family defined for each $j \in \{1, 2, \dots, m\}$, where k_1 is number of tests for which x_j variance was greater than given bound value *B*, k_2 is number of tests for witch x_j does not pass normality test, *V* is variance of window for which x_j was the center, *N* is normal distribution parameters vector of window for which x_j was the center and *W* is Weibull distribution parameters vector of window for which x_j was the center.

PROCEDURE:
For $j \in \{1, \dots, m\}$ **and** $i \in \{1, 2, \dots, n\}$ **find** $R_j(t_i) = (k_{1,i}, k_{2,i}, V_i, N_i, W_i)$:
$V_i \leftarrow Var(x_{i-h}, \dots, x_{i+h})$
$N_i \leftarrow NormParam(x_{i-h}, \dots, x_{i+h})$
$W_i \leftarrow WeibullParam(x_{i-h}, \dots, x_{i+h})$
If $V_i > B$ **then**
Increase $k_{1,i}$ for all $i \in \{i-h, \dots, i+h\}$
If N_i is not Normal **then**
Increase $k_{2,i}$ for all $i \in \{i-h, \dots, i+h\}$

It should be noted that it makes sense to determine either the vector of the parameters of the normal distribution *N* or the vector of parameters of the Weibull distribution *W*. It depends on the type of measured value x_i and its nature. Nevertheless, the most important quantities in the presented procedure are k_1 and k_2, because they show how often a given sample behaved abnormally during statistical tests.

3. PYTHON IMPLEMENTATION OF SELECTED METHODS OF STATISTICAL DIAGNOSTICS

Data analysis methods were implemented in free Python language and available statistical tools (**numpy** - basic library for creating and analyzing multidimensional arrays; **pandas** - library for reading, creating and manipulating data of various types). Using Python with these libraries allows to create simple scripts for data analyze and knowledge discovery. These tools, along with a wide range of libraries for data visualization, create a set of ready-made tools for preparing analyzes and visualizing production processes. The practical implementation of statistical research is based on a real log from the production machines saved in a CSV file. The structure of the log file: Name; Date; Number; Time; Type; Time; Machine; Source; User; Par1; Par2; Par3; Par4; Par5; Par6; Par7; Par8; Par9; Par10; Par11; Par12; Par13; Par14; Par15; Par16; Par17; Par18; Par19; Par20, where:
- name – proces name;
- date – proces date;
- number – machine numer;
- time – proces start time;
- type – proces type;
- source – proces source;
- user – production worker;
- par1:par20 – numerical parameters of the current process.

Not all parameters from file are analyzed. For this reason, in the part devoted to the implementation of selected studies, methods of selectively retrieving only the columns that will be analyzed will be presented. The data in the file was also filtered to analyze a certain range. To read data from the log file, the reading is the same and is done using the read_csv method. The program presented in Listing 1 (Annex A SOURCE CODES) imports the pandas library and reads from the log.csv file to the selected columns of the DataFrame object. The columns, separated by semicolons, contain the observations. The "shape" method in the df object created from the DataFrame class was used to display the number of cases and the parameters of the set. The last line displays the first 10 observations of the set, which facilitates the initial evaluation of the analyzed data set. This example will be used, with some slight modifications, in subsequent descriptions of the conducted statistical research.

3.1. Python implementation of variance analysis

The variance's implementation, i.e. examining the distribution of values in a data set around its mean value, can be valuable information while analyzing data from production logs. The methods available in Python statistics packages provide ready-made tools to calculate variance. You can calculate the variance of a set by calling the var() method on the DataFrame object. The method returns the unencumbered by variance against the requested axis, with 0 representing the rows and 1 representing the columns. The algorithm takes an optional "ddof" argument, which default's value is 1. This parameter indicates the degrees of freedom that will be used in the calculation. A value of 0 for the "ddof" parameter calculates the variance for the population, and a value of 1 estimates the population variance from the selected sample. The Listing 2 shows the variance calculation for all columns in a population. Variance calculation applies only to the numeric columns. For this reason, only columns containing numeric values were loaded into memory. Because the variance results are close to zero, they will be displayed exponentially. This record may make the analysis difficult, therefore the conversion to real values was performed with the use of the lambda function. Additionally, the number precision parameter was modified, narrowing down the result to 8 decimal places. The result of the program is a list of variances:

```
 Observations: 38600, parameters: 19
Par1    0.58692063
Par2    0.00006758
Par3    0.00000350
Par4    0.00000080
Par5    0.00019867
Par6    0.00000148
Par7    0.00000029
Par8    0.00000011
Par9    0.00000005
Par10   0.00000018
```

```
Par11   0.00003386
Par12   0.00000003
Par13   0.00000000
Par14   0.00000005
Par15   0.00000035
Par16   0.00000008
Par17   0.00000002
Par18   0.00000351
Par20   0.58054608
dtype: float64
```

Special attention should be paid to the display of real numbers. Their display has been truncated to the specified precision by the {: .8f} parameter. In some cases, this can lead to potential errors because the precision is too low. Therefore, the precision should always be selected on the basis of the complete results.

Apart from the standard method, the pandas package also offers calculation of variance with the use of the window method, which is especially useful when analyzing large production data sets. The calculation of the window variance is available in the "rolling" method. The argument in this case is the size of the window. In the example of calculating the window variance, it was assumed that the variance studies will take place in the windows of size 4 and 15, and the tested parameters will be columns P2, P5 and P7. In the case of window variance, it is best to assign the calculation results to a new DataFrame object. As a result of the program in Listing 3, the variance for the selected series will be displayed.

```
          vP2         vP5         vP7
0         NaN         NaN         NaN
1         NaN         NaN         NaN
2         NaN         NaN         NaN
3 0.00001125 0.00004869 0.00000001
4 0.00005850 0.00004525 0.00000002
5 0.00005069 0.00004550 0.00000001
6 0.00005550 0.00046869 0.00000019
7 0.00006019 0.00047825 0.00000033
8 0.00001225 0.00045819 0.00000045
9 0.00000819 0.00041050 0.00000046
```

In the case of a window variance, where window size = 4, the rows indexed [0: 3] will not be filled with data. The number of blank lines containing NaN (not a number) values will increase as the window grows. For example, increasing the window size to 10 will cause the display of the first 10 results to look as follows:

```
   vP2  vP5  vP7
0  NaN  NaN  NaN
1  NaN  NaN  NaN
2  NaN  NaN  NaN
3  NaN  NaN  NaN
4  NaN  NaN  NaN
5  NaN  NaN  NaN
6  NaN  NaN  NaN
7  NaN  NaN  NaN
8  NaN  NaN  NaN
9  NaN  NaN  NaN
```

A simple solution in these types of cases is to remove redundant lines containing NaN values with an additional line (before the print function):

```
dfVar.dropna(axis=0, how='any', inplace=True)
```

```
          vP2         vP5         vP7
3  0.00001125 0.00004869 0.00000001
4  0.00005850 0.00004525 0.00000002
5  0.00005069 0.00004550 0.00000001
6  0.00005550 0.00046869 0.00000019
7  0.00006019 0.00047825 0.00000033
8  0.00001225 0.00045819 0.00000045
9  0.00000819 0.00041050 0.00000046
10 0.00000225 0.00001225 0.00000010
11 0.00001850 0.00000919 0.00000019
12 0.00002319 0.00000875 0.00000018
```

This time, the listing only contains rows that contain numeric values, and the index of the case in the object has not changed.

The Seaborn library was used to visualize the results of calculating the variance, which is an extension of the Matplotlib library, enriching the standard package with additional types of graphs. Generating plots for the purpose of visualizing variance poses the problem of legibility in the plot of a relatively large amount of data. The Figure 3 shows a graph of variance for several columns, where time covers the entire population. As one can see, the chart is not legible, although it allows for a preliminary evaluation of the set.

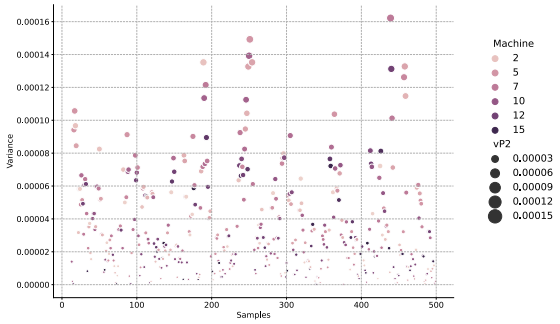

Fig. 3. Plot of variance for entire population generated with matplotlib package

A graph will look clearer and provide better perception when it is generated with the use of the filtering methods available in the pandas package. In the example in Listing 4, there are filters that date range from 5/1/2020 to 5/2/2020. Filtering before performing a statistical analysis is a frequent operation, especially in the case of long time series analysis, because a sample which is too large is not always desirable during numerical calculations, and additionally placing too large time period on the graph may distort the perception of reception and, consequently, interfere with the possibility of noticing certain regularities or anomalies. By filtering a certain range (sample) of the population, the plot must contain a subset that contains only the range of the series of interest to be analyzed. The example uses date range filtering, but the filter can also include the set values specified with extreme parameters. The Figure 4 shows a graph presenting the variance of the P2 parameter in the separated date range. Thanks to the use of filtering, the chart is

much more readable and allows the evaluation the parameters value from a segment of the production process. Additionally, the chart shows the division into machines on which the production process took place. The size of the points on the chart and the colors allow a better perception of reception, and thus, an observation of potential disturbances in the configuration of the parameters of production machines. As one can see in the graph, there are no major differences in the variance of a single parameter depending on the machine, however, if there is an abnormality on one of the machines, it will be clearly indicated with a point along with the machine number specification.

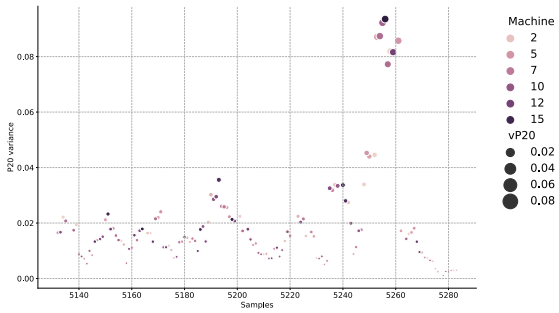

Fig. 4. Parameter P2 variance plot - date range

The filtering capabilities of the DataFrame object are much broader. They also allow the use of logical operators. Thanks to this, greater filtering precision is possible, which is necessary when it is required to get rid of anomalies resulting from errors which occur while saving or transferring log files. In such cases, the erroneous observations should be removed during the set cleaning operation. An additional protection against the analysis should be the removal by filtering of the values exceeding the limit values. In this way, you can eliminate invalid lines that clearly deviate from the range of minimum and maximum production values in the production process. Filtering will eliminate potentially false anomalies during the analysis.

```
df = df[(df['P2'] > 9.26) & (df['P2'] < 3.62) &
(df["Data"].isin(pd.date_range( '2020-05-01', '2020-05-02')))]
```

The above code is an example of the use of both date range and extreme manufacturing process filtering.

The analysis of variance can be freely extended, e.g. to search for correlations between the available parameters and other non-numeric parameters. As a result, you can get a broader picture of the studied phenomenon and notice some correlations. For example, comparing the variance of the series parameters on the graph with the division into production machines or the operators who support them, it is possible to analyze the differences in the configuration of individual machines, or errors made by the operators of these machines. The collected and processed results can be of great help in predicting production processes and allow for confirming or rejecting the proposed hypotheses concerning disturbances in production processes.

3.2. Python implementation of Normal distribution

When examining data sets, one of the basic principles is to pre-examine its distribution of features. There are many methods in Python statistical libraries to test the normality of a distribution. These methods return the P-value probability, which is the lowest significance level at which the null hypothesis for the observed value of the test statistic can be rejected (the null hypothesis can be rejected when the computed test probability in the p variable turns out to be not higher than the adopted significance level, which is usually 0.05). In the case when the value of $p \leq 0.05$, the hypothesis can be rejected, and when the value of $p > 0.05$, probably the variable has values derived from the normal distribution. The examination of the series for the assessment of the normality of the distribution can be done in two ways. The first is the graphical test, i.e. visualization of the variable distribution using one of the available charts. The second method is to perform one of a number of statistical tests to obtain the P-value.

The following program serves as an example of a graphical test, which generates random values of the normal distribution. For this purpose, the random.normal method from the numpy library was used, which takes the amount of generated data as one of the parameters. Generating the series allows the visualization of the plot for a normal distribution. The result of the program in Listing 5 is a histogram presenting the generated values of the data series and is shown in the Figure 5. The analysis of the graph shows that the series can come from a normal distribution.

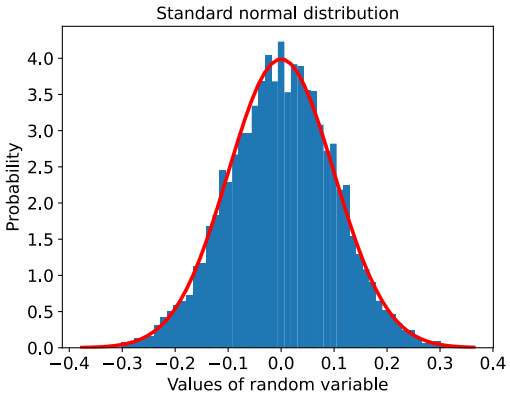

Fig. 5. Visualization of the normal distribution on a graph

Another way to generate a series from the normal distribution is to use the norms method from the scipy library as is shown in Listing 6. In this case, the numpy library was also used to generate the series from the normal distribution, but this time the linspace method was used to generate the series. The

linspace method generates a series of 1000 random values from -100 to 100 from the normal distribution which is shown in the Figure 6.

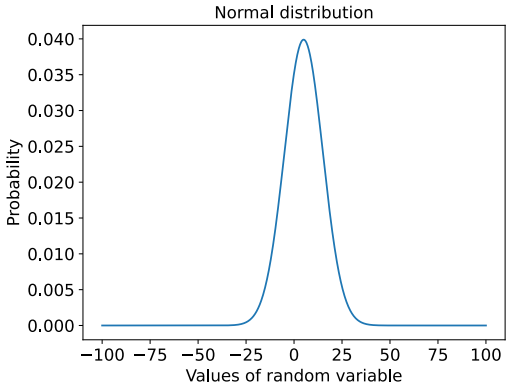

Fig. 6. Series generated with the linspace method (1000 random values from -100 to 100 from the normal distribution)

In order to perform a statistical test of the normality of the distribution, an actual sample from the population was selected (based on a date range), and additionally, anomalies that could be clearly classified as errors were eliminated from the set. As a result, a time series containing only the correct values of one of the parameters of the production process were obtained – Listing 7.

Graphical methods for assessing the normality of a distribution often use a histogram for a numerical parameter. In this type of chart, data is represented by a certain number of rectangles with sorted data containing the number of observations. Histograms divide the values of continuous variables into discrete sections and show the number of values in each of these ranges as is shown in the Figure 7. Additionally, the chart uses the kde = True option to generate the chart kernel density estimation to calculate the probability density function of a random variable.

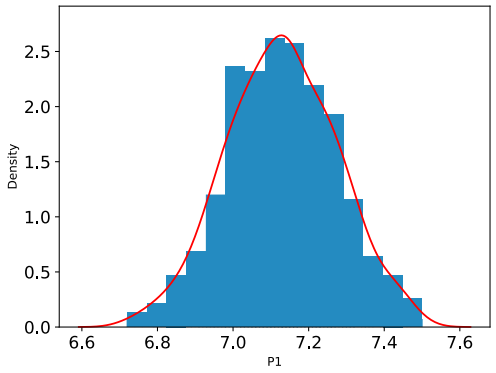

Fig. 7. Histogram with the generated series of data values

A popular type of plot for visualizing and testing a distribution is the Q-Q Plot (quantile quantile graph). This type of chart makes it easier to evaluate the studied distribution of the variable. One can use the stats.probplot function to create a chart. The program presented in Listing 8 will generate a chart of quantiles for the variable on the *y* axis, compared to the theoretical quantiles of the normal distribution. The result of the program operation is shown in the Figure 8.

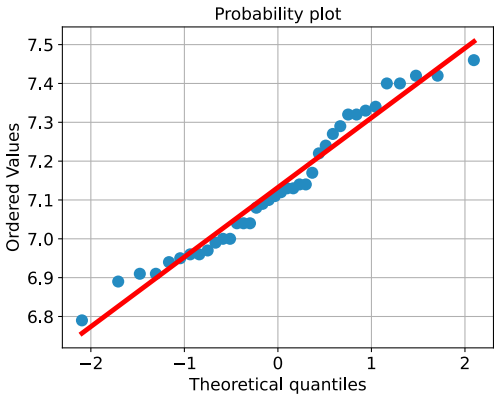

Fig. 8. Normal probability plot

The interpretation of the quantile chart is based on the observation of the concentration of points around the straight line. If a variable is normally distributed, its values agree with theoretical quantiles. An important information when interpreting a Q-Q Plot is the even alternating distribution of points close to the straight line, which may mean that the data comes from a normal distribution. Even if there are slight deviations of a few points above and below the line, if the deviations are small, the data series may be derived from the normal distribution.

The visual assessment of the graph in terms of the normality of the distribution is always associated with misinterpretation, therefore statistical tests are needed to verify the null hypothesis when examining the distribution. To this end, we will introduce several series-testing algorithms for distribution that are available in the Python statistical package.

The first test is the Shapiro-Wilk test, which is designed to test the normality of distributions in collections of up to 5000 samples. It is available in the shapiro package, and the test implementation looks as is shown in Listing 9. The result of the test is the display of the number of observations and the results of the statistical test, similar to subsequent tests.

```
Observations: 949, parameters: 2
0.9902371764183044 6.105211468820926e-06
```

The Shapiro-Wilk test may yield erroneous results when testing larger samples and is the preferred way to test the normality of a probability distribution due to its strength.

Another test that will be used is the D'Agostino-Pearson test, which assumes that the sample size should not be smaller than 20 observations. Its implementation on an identical row is shown in Listing 10.

The next test performed is the Kolmogorov-Smirnov test, which belongs to the non-parametric tests for assessing the compatibility of the distribution of variables with the normal distribution. It is a test used in situations where the mean or standard deviation is unknown. The method should be used for samples with $n > 100$. Like other tests, it tests the null hypothesis of a distribution close to the normal distribution – Listing 11.

Another test that will be performed is based on the Anderson-Darling method and is based on statistical tests of the consistency of the distribution with a given standard distribution. It tests the null hypothesis of a sample from a population with a specific distribution. Critical values depend on the tested distribution. The method works for normal, exponential, logistic, or Gumbel distribution. By default, the tested distribution is the normal distribution, but it can also take other distributions: expon, logistic, gumbel, gumbel_l, gumbel_r, extreme1) – Listing 12. The result of the executed series testing program using the Anderson-Darling test is:

```
Statistics: 2.104
Hypothesis rejected: critical value 1.087
signification level 1.0
```

The next statistical test available is the chi-square test. This method is often used to verify the hypothesis whether the observed trait in a community has a specific type of distribution – Listing 13.

The second to last of the presented statistical tests is the Lilliefors test. It is a test based on the Kolmogornov-Smirnov test, which can be used when the mean value and standard deviation are unknown. The test implementation is presented in Listing 14.

The last of the tests is the Jarque-Ber distribution normality test, which is one of the tests frequently used in econometrics due to the uncomplicated form of the asymptotic distribution. The construction of the test statistic is based on the values of the moments of distribution of a random variable calculated on the basis of the empirical sample and comparing them with the theoretical moments of the normal distribution. This test verifies the hypothesis of univariate normality of a random variable against any other distribution and is presented in Listing 15.

3.3. Python implementation of Weibull distribution

Reliability engineering and survival analysis are an important stage of research into the prediction of production processes. There are several reliability analysis libraries for Python. In this case, the Reliability library was used for the research, which contains a set of functions useful in this type of analysis. The library is an extension of scipy.stats and contains additional tools useful in testing reliability.

With the help of the library, you can create distribution matches for both complete and incomplete data, and the available fit modules are named with the number of their parameters. For example, Fit Weibull 2P uses α, β and Fit Weibull 3P uses α, β, γ. The distributions are fitted using the requested function along with the errors passed to it. A minimum of 4 samples is recommended as the accuracy of the fit depends on this. For the purpose of the experiment, the Fit Weibull 2P method was used. The program in Listing 16 imports the log from the production machine and determines parameter errors in the form of an additional logical column. In this case, the errors result from exceeding the production values. In the next step, the Weibull fitting function is called, which displays the results of the calculations.

```
Observations: 289, variables: 2
False    263
True      26
Name: error, dtype: int64
Results from Fit_Weibull_2P (95% CI):
Analysis method: Maximum Likelihood
Estimation (MLE)
Optimizer: TNC
Failures/Right censored: 26/0 (0% right
censored)

Parameter Point Estimate Standard Error
Lower CI Upper CI
    Alpha  13096.77421476    148.50853144
12808.91349390 13391.10416461
     Beta     18.29210998      2.82802505
13.51030110    24.76638272

Goodness of fit         Value
 Log-likelihood -210.75609058
           AICc  426.03392029
            BIC  428.02837424
             AD    1.04087407
```

By analyzing the result of the program, you can read the number of error samples. The output also includes confidence intervals and standard error for parameter estimates. The probability plot is generated automatically using the plt.show() method and is presented in the Figure 9.

On the generated graph, you can observe how the data is modeled, and the interpretation of the visualization consists in analyzing the location of points that should be placed on a straight line. In this case, however, it is not so. A misalignment occurs when a line or curve formed by points deviates significantly from a straight line. In the plot interpretation, slight deviations from the straight line are tolerated at the ends of the distribution, but most points should follow the straight line. To display the failure points next to PDF, CDF, SF, HF, or CHF points without using axis scaling, one can use the plot_points function to generate a plot – Listing 17. The plot_points function plots the failure points based on the position and then generates the points similar to the previous case. However, it does not scale the axis or fitted distribution. The result of the

program is the creation of the Weibull distribution graph, shown in the Figure 10.

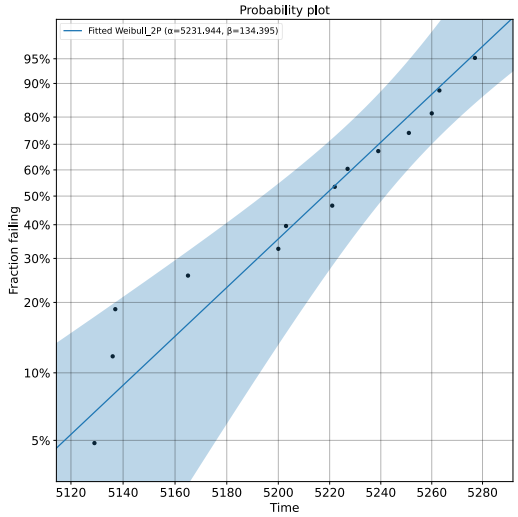

Fig. 9. Weibull probability plot created with the reliability package

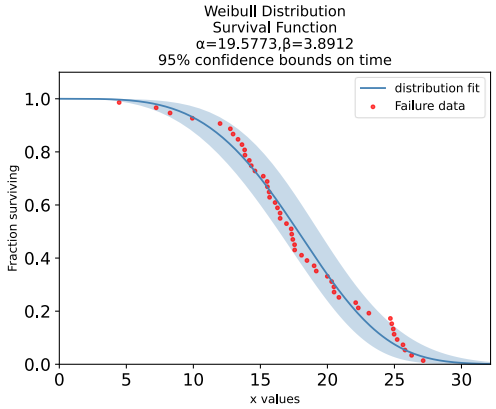

Fig. 10. Weibull distribution with control points without axis scaling or fitted distribution (plot_points method)

4. CONCLUSIONS

Systems for the analysis of big data are playing an increasingly important role in industry, and this trend applies to all sectors. A strong impulse for the development of this field of knowledge is the constant increase in the importance of information and the need to compete in global markets. Exploring the collections in terms of obtaining specific knowledge is a way to automate the delivery of answers to previously asked questions. The information from production systems, analyzed and prepared in a human-readable way, can be used for faster prediction of failures and component wear or process monitoring.

It is worth using commonly available tools such as Python and a number of available libraries for this purpose. While only the Python environment was used in the research, there are a number of convenient dedicated environments, such as Jupyter and Spyder. These tools are equipped with convenient interfaces and a built-in help system, which greatly facilitates and speeds up work on the harvest.

This article presents selected issues of automatic analysis of data from production processes. In this way, the authors wanted to show how extensive the subject is. The second goal was to demonstrate how modern programming tools can be used to support data analysis in enterprises. The desire to show specific possibilities results from the fact that many modern IT systems make very little use of these modern technologies.

A. SOURCE CODES

Listing 1: Read csv file and get info.

```
import pandas as pd
cols = ["Data", "Machine", "Time", "Machine", "P1", "P2", "P3",
"P4", "P5"]

df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

print('Observations: {}, parameters: {}'.format(df.shape[0],
df.shape[1]))
```

Listing 2: Set display data format.

```
import pandas as pd

cols = ["P1", "P2", "P3", "P4", "P5", "P6"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

print('Observations: {}, parameters: {}'.format(df.shape[0],
df.shape[1]))

pd.set_option('display.float_format', lambda x:
f'%.{len(str(x%1))-2}f' % x)

print(df.var())
```

Listing 3: Variance analysis of selected numerical columns.

```
import pandas as pd
df = pd.read_csv("log.csv", decimal=".", delimiter=";")
dfVar = pd.DataFrame()

pd.set_option('display.float_format', lambda x:
f'%.{len(str(x%1))-2}f' % x)

win = 4 # Rolling window size
dfVar['vP2'] = df.P2.rolling(win).var(ddof=0)
dfVar['vP5'] = df.P5.rolling(win).var(ddof=0)
dfVar['vP7'] = df.P7.rolling(win).var(ddof=0)
dfVar['vP10'] = df.P10.rolling(win).var(ddof=0)
dfVar['vP17'] = df.P17.rolling(win).var(ddof=0)

print(dfVar.head(10))
```

Listing 4: Rolling windows variance of selected columns analysis.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the machine log file to DataFrame object
df = pd.read_csv("log.csv", decimal=".", delimiter=";")

mask = (df.index >= 10) & (df.index <= 500)
```

```
df = df.loc[mask]

# Create Dataframe
dfVar = pd.DataFrame()
win = 4
# Variance for P2 parameter
dfVar['vP2']=df.P2.rolling(win).var(ddof=0)

# Plot variance
chart = sns.relplot(x=dfVar.index,
                    y=dfVar.vP2,
                    size=dfVar.vP2,
                    sizes=(15, 200),
                    hue=df.Machine,
                    data=dfVar)

# Change labels and size settings than are provided by default
chart.fig.set_figwidth(15)
chart.fig.set_figheight(8)
chart.set_ylabels("Variance", size=14)
chart.set_xlabels("Samples", size=14)
plt.grid(linestyle='--', linewidth=1)
plt.xticks(size = 14)
plt.yticks(size = 14)
# Show chart
plt.show()
```

Listing 5: Example of generating a data series and histogram displaying.

```
import numpy as np
import matplotlib.pyplot as plt

# Define mean and standard deviation
me, sigma = 0, 0.1
# Generate series
series = np.random.normal(me, sigma, 6500)

count, bins, ignored = plt.hist(series, 60, density=True)

plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) * np.exp( - (bins -
me)**2 / (2 * sigma **2 )), linewidth=3, color='red')
plt.title("Standard normal distribution")
plt.xlabel('Values of random variable')
plt.ylabel('Probability')
plt.show()
```

Listing 6: An example of generating a normal distribution and displaying a graph.

```
from scipy.stats import norm
import matplotlib.pyplot as plt
import numpy as np

mu = 5 #mean
std = 10 #standard deviation
snd = norm(mu, std)
x = np.linspace(-100, 100, 1000)

plt.plot(x, snd.pdf(x))
plt.title('Normal distribution')
plt.xlabel('Values of random variable')
plt.ylabel('Probability')
plt.show()
```

Listing 7: Histogram from a real log csv file.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

print('Observations: {}, parameters: {}'.format(df.shape[0],
df.shape[1]))

# Convert data column to datetime
df['Data'] = pd.to_datetime(df['Data'])

from_date = '2020-05-01'
to_date = '2020-05-10'
df = df[(df['P1'] > 6.0) & (df['P1'] < 8.0) &
(df['Data'].isin(pd.date_range(from_date, to_date)))]

sns.distplot(df.P1, rug=True, rug_kws = {"color": "b"}, kde_kws
= {"color": "k"},
hist_kws = {"linewidth": 4, "alpha": 1, "color": "b"})
plt.show()
```

Listing 8: Sample Q-Q Plot chart from log csv file.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

print('Observations: {}, parameters: {}'.format(df.shape[0],
df.shape[1]))

# Convert data column to datetime
df['Data'] = pd.to_datetime(df['Data'])

from_date = '2020-05-01'
to_date = '2020-05-10'
df = df[(df['P1'] > 6.0) & (df['P1'] < 8.0) &
(df['Data'].isin(pd.date_range(from_date, to_date)))]

stats.probplot(df["P1"], dist="norm", plot=plt)
plt.show()
```

Listing 9: Statistical test of the normality of a distribution using the Shapiro-Wilk method.

```
import pandas as pd
from scipy.stats import shapiro

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

stats, p = shapiro(df.P1)
print(stats ,p)
if p>0.05:
   print ('Normal distribution')
```

Listing 10: Standard statistical test of the normality.

```
import pandas as pd
from scipy import stats

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

stats, p = stats.normaltest(df.P1)
print(stats ,p)
if p>0.05:
   print ('Normal distribution')
```

Listing 11: Statistical test of the normality of a distribution using the Kolmogorov-Smirnov.

```
import pandas as pd
from scipy.stats import kstest

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

stats, p = kstest(df.P1, 'norm')
print(stats ,p)
if p>0.05:
   print ('Normal distribution')
```

Listing 12: Statistical test of the normality of a distribution using the Anderson method.

```
import pandas as pd
from scipy.stats import anderson

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

result = anderson(df.P1)
print('Statistics: %.3f' % result.statistic)
for i in range(len(result.critical_values)):
   si_lev, cr_val = result.significance_level[i],
result.critical_values[i]
if result.statistic < cr_val:
   print(f'Normal distribution: critical value {cr_val}, signification
level {si_lev}')
else:
   print(f'Hypothesis rejected: critical value {cr_val} signification
level {si_lev}')
```

Listing 13: Statistical test of the normality of a distribution using the Chi-squared method.

```
import pandas as pd
from scipy.stats import chisquare

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

stat, p = chisquare(df.P1)
print('Statistics: %.3f, p = %.3f' % (stat, p))
if p>0.05:
   print('Normal distribution')
```

Listing 14: Statistical test of the normality of a distribution using the Lilliefors method.

```
import pandas as pd
from statsmodels.stats.diagnostic import lilliefors

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

stat, p=lilliefors(df.P1)
print('Statistics: %.3f, p = %.3f' % (stat, p))

if p>0.05:
   print('Normal distribution')
```

Listing 15: Statistical test of the normality of a distribution using the Jarque-Bera method.

```
import pandas as pd
from scipy.stats import jarque_bera

cols = ["Data","P1"]
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

stats, p = jarque_bera (df.P1)
print('Statistics: %.3f, p = %.3f' % (stats, p))
if p>0.05:
   print('Normal distribution')
```

Listing 16: Weibull probability plot example.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from reliability.Fitters import Fit_Weibull_2P
from reliability.Probability_plotting import
Weibull_probability_plot, Exponential_probability_plot

# Set specific column from CSV file
cols = ["Data","P1"]
# and load the machine log file to DataFrame object
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

mask = (df.index >= 5120) & (df.index <= 5280)
df = df.loc[mask]

# Round to eight decimal places in pandas
pd.options.display.float_format = '{:.8f}'.format

# Define errors and anomaly based on critical values
# The correct values are within the limits of min max
df['error'] = np.where((df['P1'] < 3.2) | (df['P1'] > 9.63), True,
False)
df = df[(df['error'] == True)]

# Print error count
print(df.error.value_counts())

# Create fail data
fail_data = df.index.tolist()

# Create the probability plot
Weibull_probability_plot(failures=fail_data)

# Set title, change labels and size settings than are provided
by default
plt.title('Probability plot', size=14)
plt.xlabel("Time", size=14)
plt.ylabel("Fraction failing", size=14)
plt.xticks(size = 14)
plt.yticks(size = 14)
# Show chart
plt.show()
```

Listing 17: Weibull dstribution fit 2P and plots the survival function.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from reliability.Fitters import Fit_Weibull_2P
from reliability.Distributions import Weibull_Distribution
from reliability.Probability_plotting import plot_points

# Set specific column from CSV file
cols = ["Data","P1"]
# and load the machine log file to DataFrame object
df = pd.read_csv("log.csv", decimal=".", delimiter=";", usecols =
cols)

# Optional select index range data from DataFrame
mask = (df.index >= 120) & (df.index <= 1500)
```

```
df = df.loc[mask]

data = df

# Round to eight decimal places in pandas
pd.options.display.float_format = '{:.8f}'.format

# Create additional column with logical value True/False
# The correct values are within the limits of min max
df['error'] = np.where((df['P1'] < 3.2) | (df['P1'] > 9.63), True,
False)
# Filter error values
df = df[(df['error'] == True)]

# Create fail data
fail_data = df.index.tolist()

# Fits a Weibull distribution to the data and generates the
probability plot
weibull_fit = Fit_Weibull_2P(dataframe=data, failures=fail_data,
show_probability_plot=False, print_results=True)
# Uses the distribution object from Fit_Weibull_2P and plots
the survival function
weibull_fit.distribution.SF(label='distribution
fit',color='steelblue')
# Plots the survival function of the original distribution
plot_points(failures=fail_data,func='SF',label='Failure
data',color='red',alpha=0.7)

# Change labels and size settings than are provided by default
plt.xlabel("x values", size=14)
plt.ylabel("Fraction surviving", size=14)
plt.xticks(size = 14)
plt.yticks(size = 14)
plt.legend() # Enable legend
# Show chart
plt.show()
```

REFERENCES

1. Aigner R, Leitner M, Stoschka M. Fatigue strength characterization of Al-Si cast material incorporating statistical size effect. In G. Henaff, editor, 12th International Fatigue Congress (FATIGUE 2018), volume 165 of MATEC Web of Conferences, 2018. https://doi.org/10.1051/matecconf/201816514002
2. Almeida A, Loy A, Hofmann H. ggplot2 Compatible Quantile-Quantile Plots in R. The R Journal. 2018; 10(2):248-261. https://doi.org/10.32614/RJ-2018-051
3. Almeida JB. Application of weilbull statistics to the failure of coatings. Journal of Materials Processing Technology. 1999; 92-93:257-263. https://doi.org/10.1016/S0924-0136(99)00177-6
4. Anderson TW, Darling DA. Asymptotic Theory of Certain 'Goodness of Fit' Criteria Based on Stochastic Processes. Annals of Mathematical Statistics. 1952; 23(2):193-212. https://doi.org/10.1214/aoms/1177729437
5. Ansell JI, Phillips MJ. Practical Methods for Reliability Data Analysis. Oxford University Press, Oxford. 1994.
6. Augustin NH, Sauleau EA, Wood SN. On quantile quantile plots for generalized linear models. Computational Statistics & Data Analysis. 2012; 56(8):2404-2409. https://doi.org/10.1016/j.csda.2012.01.026
7. Bai YL, Yan ZW, Ozbakkaloglu T, Han Q, Dai JG, Zhu DJ. Quasistatic and dynamic tensile properties of large-rupture-strain (LRS) polyethylene terephthalate fiber bundle. Construction and Building Materials, 2020; 232. https://doi.org/10.1016/j.conbuildmat.2019.117241
8. Baringhaus N, Henze N. A consistent test for multivariate normality based on the empirical characteristic function. Metrika. 1988; 35:339-348. https://doi.org/10.1007/BF02613322
9. Baringhaus N, Henze N. Limit distributions for mardia's measure of multivariate skewness. Annals of Statistics, 1992; 20(4):1889-1902. https://doi.org/10.1214/aos/1176348894
10. Barlow RE, Proschan F. Statisitcal Theory of Reliability and Life Testing. Holt, Rinehart, Austin, 1975.
11. Birolini A. Reliability Engineering. Springer, Berlin, Heidelberg, 2017. https://doi.org/10.1007/978-3-662-54209-5
12. Bowman AW, Foster PJ. Adaptive Smoothing and Density-Based Tests of Multivariate Normality. Journal of the American Statistical Association, 1993; 88(422):529-537. https://doi.org/10.1080/01621459.1993.10476304
13. Box G. Signal-to-Noise Ratios, Performance Criteria, and Transformations. Technometrics, 1988; 30(1):1-17. https://doi.org/10.2307/1270311
14. Box GEP, Hunter JS. Condensed calculations for evolutionary operation programs. Technometrics, 1959; 1(1):77-95. https://doi.org/10.1080/00401706.1959.10489850
15. Chalapathy R, Chawla S. Deep learning for anomaly detection: A survey. CoRR, 2019; abs/1901.03407. http://arxiv.org/abs/1901.03407
16. Chmielowiec A. Weibull distribution and its application in the process of optimizing the operating costs of non-repairable elements. Prediction in mechanical and automatic systems 2020 - mathematical and statistical modelling volume 1. pages 45-73, Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszow, 2020. Rozkład Weibulla i jego zastosowanie w procesie optymalizacji kosztów eksploatacji elementów nienaprawialnych https://depot.ceon.pl/handle/123456789/19465
17. Chmielowiec A. Algorithm for error-free determination of the variance of all contiguous subsequences and fixed-length contiguous subsequences for a sequence of industrial measurement data. Computational Statistics, 2021. https://doi.org/10.1007/s00180-021-01096-1
18. Csorgo S. Testing for Normality in Arbitrary Dimension. Annals of Statistics, 1986; 14(2):708-723. https://doi.org/10.1214/aos/1176349948
19. Das KR, Imon AHMR. A brief review of tests for normality. American Journal of Theoretical and Applied Statistics, 2016; 5(1):5-12. https://doi.org/10.11648/j.ajtas.20160501.12
20. Dhar SS, Chakraborty B, Chaudhuri P. Comparison of multivariate distributions using quantile–quantile plots and related tests. Bernoulli, 2014; 20(3):1484-1506. https://doi.org/10.3150/13-BEJ530
21. Ding J, Liu Y, Zhang L, Wang J, Liu Y. An anomaly detection approach for multiple monitoring data series based on latent correlation probabilistic model. Applied Intelligence, 2016; 44:340-361. https://doi.org/10.1007/s10489-015-0713-7
22. Ebner B, Henze N.. Tests for multivariate normality - a critical review with emphasis on weighted L^2-statistics. TEST, 2020; 29:845-892. https://doi.org/10.1007/s11749-020-00740-0

23. Elsayed EA, Chen A. Optimal levels of process parameters for products with multiple characteristics. International Journal of Production Research, 1993; 31(5):1117-1132. https://doi.org/10.1080/00207549308956778
24. Epps TW, Pulley LB. A test for normality based on the empirical characteristic function. Biometrika, 1983; 70(2):723-726. https://doi.org/10.2307/2335564
25. Evans K, Love T, Thurston SW. Outlier identification in model-based cluster analysis. Journal of Classification, 2015; 32:63-84. https://doi.org/10.1007/s00357-015-9171-5
26. Fisher RA. The moments of the distribution for normal samples of measures of departure from normality. Proceedings of the Royal Society, 1930; 130(812):16-28. https://doi.org/10.1098/rspa.1930.0185
27. Fisher RA, Tippett LMC. Limiting forms of frequency distribution of the largest or smallest member of a sample. Mathematical Proceedings of the Cambridge Philosophical Society, 1928; 24:180-190. https://doi.org/10.1017/S0305004100015681
28. Fok SL, Mitchell BC, Smart J, Marsden BJ. A numerical study on the application of the weibull theory to brittle materials. Engineering Fracture Mechanics, 2001; 68(10):1171-1179. https://doi.org/10.1016/S0013-7944(01)00022-4
29. Frechet M. Sur la loi de probabilite de l'ecart maximum. Annales de la Societe Polonaise de Mathematique, 1927; 6:93-116.
30. Grynchenko O, Alfyorov O. Mechanical Reliability. Springer, Cham, 2020. https://doi.org/10.1007/978-3-030-41564-8
31. Guner B, Frankford MT, Johnson JT. A study of the Shapiro-Wilk test for the detection of pulsed sinusoidal radio frequency interference. IEEE transactions on Geoscience and Remote sensing, 2009; 47(6):1745-1751. https://doi.org/10.1109/TGRS.2008.2006906
32. Hawkins DM. Identification of outliers. Springer, Dordrecht, 1980.
33. Hemphill MA, Yuan T, Wang GY, Yeh JW, Tsai CW, Chuang A, Liaw PK. Fatigue behavior of Al0.5CoCrCuFeNi high entropy alloys. Acta Materialia, 2012; 60(16):5723-5734. https://doi.org/10.1016/j.actamat.2012.06.046
34. Henze N. Invariant tests for multivariate normality: a critical review. Statistical Papers, 2002; 43(4):467-506. https://doi.org/10.1007/s00362-002-0119-6
35. Henze N, Visagie J. Testing for normality in any dimension based on a partial differential equation involving the moment generating function. Annals of the Institute of Statistical Mathematics, 2019; 5:1-28. https://doi.org/10.1007/s10463-019-00720-8
36. Henze N, Wagner T. A New Approach to the BHEP Tests for Multivariate Normality. Journal of Multivariate Analysis, 1997; 62(1):1-23. https://doi.org/10.1006/jmva.1997.1684
37. Henze N, Zirkler B. A class of invariant and consistent tests for multivariate normality. Communications in Statistics - Theory and Methods, 1990; 19(10):3595-3617. https://doi.org/10.1080/03610929008830400
38. Hyndman RJ, Wang E, Laptev N. Large-scale unusual time series detection. In 2015 IEEE international conference on data mining workshop (ICDMW). 2015: 1616-1619. https://doi.org/10.1109/ICDMW.2015.104
39. Jacquelin J. Inference of sampling on Weibull parameter estimation. IEEE transactions on dielectrics and electrical insulation, 1996; 3(6):809-816. https://doi.org/10.1109/94.556564
40. Johnson NL, Kotz S, Balakrishnan N. Continuous Univariate Distributions, volume 1. Wiley, New York, 1994.
41. Johnson VE, Hamada M, Martz H, Reese S, Wilson A. Modern Reliability Analysis: A Bayesian Perspective. Springer, Berlin, Heidelberg, New York, 2005.
42. Kackar RN. Off-Line Quality Control, Parameter Design, and the Taguchi Method. Journal of Quality Technology, 1985; 17(4):176-209. https://doi.org/10.1080/00224065.1985.11978964
43. Kackar RN, Shoemaker AC. Robust design: A cost effective method for improving manufacturing process, pages 159-174. Springer, Boston, 1989. https://doi.org/10.1007/978-1-4684-1472-1_8
44. Keogh E, Lin J. Clustering of time-series subsequences is meaningless: implications for previousand future research. Knowledge and Information Systems, 2005; 8(2):154-177. https://doi.org/10.1007/s10115-004-0172-7
45. Keogh E, Lin J, Fu A. Hot sax: efficiently finding the most unusual time series subsequence. In Fifth IEEE international conference on data mining (ICDM'05), 2005; 8. https://doi.org/10.1109/ICDM.2005.79
46. Keogh E, Lin J, Lee SH, Herle HV. Finding the most unusual time series subsequence: algorithms and applications. Knowledge and Information Systems, 2006; 11(1):1-27. https://doi.org/10.1007/s10115-006-0034-6
47. Keshevan K, Sargent G, Conrad H. Statistical analysis of the hertzian fracture of pyrex glass using the weibull distribution function. Journal of Materials Science, 1980; 15:839-844. https://doi.org/10.1007/BF00552092
48. Knuth DE. The Art of Computer Programming, volume II: Seminumerical Algorithms. Addison-Wesley, 2 edition, 1981.
49. Lai C, Murthy DN, Xie M. Weibull Distributions and Their Applications, pages 63–78. Springer London, London, 2006. https://doi.org/10.1007/978-1-84628-288-1_3
50. Lai CD. Generalized Weibull Distributions. Springer, Heidelberg, 2014. https://doi.org/10.1007/978-3-642-39106-4
51. Laptev N, Amizadeh S, Flint I. Generic and scalable framework for automated time-series anomaly detection. In KDD'15: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2015: 1939-1947. https://doi.org/10.1145/2783258.2788611
52. Leon RV, Shoemaker AC, Kackar RN. Performance Measure Independent of Adjustment: An Explanation and Extension of Taguchi's Signal-to-Noise Ratio. Technometrics, 1987; 29(3):253-265. https://doi.org/10.2307/1269331
53. Li QS, Fang JQ, Liu DK, Tang J. Failure probability prediction of concrete components. Cement and Concrete Research, 2003; 33(10):1631-1636. https://doi.org/10.1016/S0008-8846(03)00111-X
54. Lilliefors HW. On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown. Journal of the American Statistical Association, 1967; 62(318):399-402. https://doi.org/10.1080/01621459.1967.10482916

55. Logothetis N, Haigh A. Characterizing and optimizing multi-response processes by the Taguchi method. Quality and Reliability Engineering International, 1988; 4(2):159-169. https://doi.org/10.1002/qre.4680040211
56. Lowry CA, Montgomery DC. A review of multivariate control charts. IIE Transactions, 1995; 27(6):800-810. https://doi.org/10.1080/07408179508936797
57. Malkovich JF, Afifi AA. On Tests for Multivariate Normality. Journal of the American Statistical Association, 1973; 68(341):176-179. https://doi.org/10.2307/2284163
58. Mardia KV. Measures of multivariate skewness and kurtosis with applications. Biometrika, 1970; 57(3):519-530. https://doi.org/10.2307/2334770
59. Mason RL, Champ CW, Tracy ND, Wierda SJ, Young JC. Assessment of multivariate process control techniques. Journal of Quality Technology, 1997;29(2):140-143. https://doi.org/10.1080/00224065.1997.11979743
60. Massey JF. The Kolmogorov-Smirnov Test for Goodness of Fit. Journal of the American Statistical Association, 1951; 46(253):68-78. https://doi.org/10.1080/01621459.1951.10500769
61. McPherson JW. Reliability Physics and Engineering: Time-To-Failure Modeling. Springer, Cham, 2019. https://doi.org/10.1007/978-3-319-93683-3
62. Montgomery DC. Introduction to statistical quality control. Wiley, New York, 2019.
63. Montgomery DC, Woodall WH. A Discussion on Statistically-Based Process Monitoring and Control. Journal of Quality Technology, 1997; 29(2):121-121. https://doi.org/10.1080/00224065.1997.11979738
64. Mori TF, Szekely GJ, Rizzo ML. On energy tests of normality. Journal of Statistical Planning and Inference, 2021; 213:1-15. https://doi.org/10.1016/j.jspi.2020.11.001
65. Murthy DNP, Xie M, Jiang R. Weibull Models. Wiley, New York, 2003. https://doi.org/10.1002/047147326X
66. Nair VN. Taguchi's Parameter Design: A Panel Discussion. Technometrics, 1992; 34(2):127-161. https://doi.org/10.2307/1269231
67. Newell JA, Kurzeja T, Spence M, Lynch M. Analysis of recoil compressive failure in high performance polymers using two-, four-parameter weibull models. High Performance Polymers, 2002; 14:425-434. https://doi.org/10.1177/095400830201400408
68. Oldford WR. Self-calibrating quantile–quantile plots. The American Statistician, 2016; 70(1):74-90. https://doi.org/10.1080/00031305.2015.1090338
69. Pearson ES. A further development of tests for normality. Biometrika, 1930; 22(1-2):239-249. https://doi.org/10.2307/2332073
70. Phadke MS, Kackar RN, Speeney DV, Grieco MJ. Off-line quality control integrated circuit fabrication using experimental design. Bell System Technical Journal, 1983; 62(5):1273-1309.
71. Pignatiello JJ. Strategies for robust multiresponse quality engineering. IIE Transsactions, 1993; 25(3):5-15. https://doi.org/10.1080/07408179308964286
72. Queeshi FS, Sheikh AK. A probabilistic characterization of adhesive wear in metals. IEEE Transactions on Reliability, 1997; 46(1):38-44. https://doi.org/10.1109/24.589924
73. Ross R. Bias and standard deviation due to Weibull parameter estimation for small data sets. IEEE Transactions on Dielectrics and Electrical insulation, 1996; 3(1):28-42. https://doi.org/10.1109/94.485512
74. Royston P. Algorithm AS 181: the W test for normality. Applied Statistics, 1982; 31(2):176-180. https://doi.org/10.2307/2347986
75. Royston P. An extension of Shapiro and Wilk's W test for normality to large samples. Journal of the Royal Statistical Society: Series C (Applied Statistics), 1982; 31(2):115-124. https://doi.org/10.2307/2347973
76. Royston P. Approximating the Shapiro-Wilk W-test for non-normality. Statistics and computing, 1992; 2(3):117-119. https://doi.org/10.1007/BF01891203
77. Senin P, Lin J, Wang X, Oates T, Gandhi S, Boedihardjo AP, Chen C, Frankenstein S. Time seriesanomaly discovery with grammar-based compression. In Proceedings of the 18th international conference on extending database technology, EDBT 2015: 481-492, 2015. https://doi.org/10.5441/002/edbt.2015.42
78. Shapiro SS, Francia RS. An Approximate Analysis of Variance Test for Normality. Journal of the American Statistical Association, 1972; 67(337):215-216. https://doi.org/10.1080/01621459.1972.10481232
79. Shapiro SS, Wilk MB. An analysis of variance test for normality (complete samples). Biometrika, 1965; (3-4):591-611. https://doi.org/10.2307/2333709
80. Sheikh AK, Boah JK, Hansen DA. Statistical modelling of pitting corrosion and pipeline reliability. Corrosion, 1990; 46(3):190-197. https://doi.org/10.5006/1.3585090
81. Shewart WA. Economic Control of Quality Manufactured Product. D. Van Nostrand, New York, 1931.
82. Szekely GJ, Rizzo ML. A new test for multivariate normality. Journal of Multivariate Analysis, 2005; 93(1):58-80. https://doi.org/10.1016/j.jmva.2003.12.002
83. Taguchi G. Introduction to Quality Engineering: Designing Quality into Products and Processes. Asian Productivity Organization, Tokyo, 1986.
84. Tenreiro C. A new test for multivariate normality by combining extreme and nonextreme BHEP tests. Communications in Statistics - Simulation and Computation, 2017; 46(3):1746-1759. https://doi.org/10.1080/03610918.2015.1011334
85. Thas O, Ottoy JP. Some generalizations of the Anderson–Darling statistic. Statistics & Probability Letters, 2003; 64(3):255-261. https://doi.org/10.1016/S0167-7152(03)00169-X
86. Tiryakioglu M, Campbell J. Weibull analysis of mechanical data for castings: A guide to the interpretation of probability plots. Metallurgical and Materials Transactions A, 2010; 41(12):3121-3129. https://doi.org/10.1007/s11661-010-0364-6
87. Tsui KL. A critical look at Taguchi's modelling approach. Journal of Applied Statistics, 1996; 23(1): 81-95. https://doi.org/10.1080/02664769624378
88. Tsui KL. Robust design optimization for multiple characteristic problems. International Journal of Production Research, 1999; 37(2):433-445. https://doi.org/10.1080/002075499191850
89. Vasicek O. A Test for Normality Based on Sample Entropy. Journal of the Royal Statistical Society. Series B, 1976; 38(1):54-59. https://www.jstor.org/stable/2984828
90. Wang X, Lin J, Patel N, Braun M. Exact variable-length anomaly detection algorithm for univariate and

multivariate time series. Data Minind and Knowledge Discovery, 2018; 32:1806-1844. https://doi.org/10.1007/s10618-018-0569-7
91. Weibull W. A statistical theory of the strength of material. Ingeniors Vetenskaps Akademiens Handligar, 1939; 151:5-45.
92. Weibull W. A statistical distribution function of wide applicability. Journal of Applied Mechanics, 1951; 18:293-296.
93. Welford BP. Note on a method for calculating corrected sums of squares and products. Technometrics, 1962; 4(3):419-420. https://doi.org/10.2307/1266577
94. Woo S. Reliability Design of Mechanical Systems. Springer, Singapore, 2020. https://doi.org/10.1007/978-3-319-50829-0
95. Woodall WH, Tsui KL, Tucker GR. A Review of Statistical and Fuzzy Quality Control Charts Based on Categorical Data. In: Lenz H.J, Wilrich P.T. Frontiers in Statistical Quality Control, pages 83-89, Heidelberg, 1997. Physica-Verlag HD. https://doi.org/10.1007/978-3-642-59239-3_7
96. Xie S, Lin H, Wang Y, Chen Y, Xiong W, Zhao Y, Du S. A statistical damage constitutive model considering whole joint shear deformation. International Journal of Damage Mechanics, 2020; 29(6):988-1008. https://doi.org/10.1177/1056789519900778
97. Zhang Y, Meratnia N, Havinga P. Outlier detection techniques for wireless sensor networks: a survey. IEEE Communications Surveys and Tutorials, 2010; 12(2):159-170. https://doi.org/10.1109/SURV.2010.021510.00088
98. Zhu LX, Wong HL, Fan KT. A test for multivariate normality based on sample entropy and projection pursuit. Journal of Statistical Planning and Inference, 1995; 45(3):373-385. https://doi.org/10.1016/0378-3758(94)00058-4

Received 2021-10-02
Accepted 2021-12-03
Available online 2021-12-06

Andrzej CHMIELOWIEC received the M.Sc. degree in mathematics from the Faculty of Mathematics, Informatics and Mechanics at Warsaw University, Ph.D. degree in informatics from Institute of Fundamental Technological Research Polish Academy of Sciences. He works at the the Department of Computerization and Robotization of Industrial Processes, Faculty of Mechanics and Technology, Rzeszow University of Technology.

Leszek KLICH assistant professor at the Department of Computerization and Robotization of Industrial Processes, Faculty of Mechanics and Technology, Rzeszow University of Technology. Research interests include issues related to data analysis and the use of artificial intelligence in predicting industrial processes.