**Fitting Data to Probability Distributions Using Maximum Likelihood Estimate (MLE)**

Mahshid Jafar Pour, Mehdi Sadeghi

Georgia Institute of Technology, USA

December 2023

**Abstract**

DistFit is a Python package designed for distribution fitting and goodness-of-fit testing. It estimates distribution parameters using Maximum Likelihood Estimation (MLE) for different probability distributions, as well as Chi-Square goodness of fit for assessment of how well a given distribution fits a set of observed data. The package is adept at handling both discrete and continuous distributions, offering users a flexible and versatile tool for conducting in-depth statistical analyses. The implemented distributions are: Bernoulli, Binomial, Geometric, Poisson, Uniform, Exponential, Normal, Weibull, and Gamma. This paper serves as a comprehensive guide to DistFit, including some details about each distribution type. The MLE approach is employed to estimate the parameters of these distributions.

Furthermore, the DistFit package includes an Examples file showcasing the integration of these modules. Users can visualize the fitted distributions, explore possible alternatives, and conduct hypothesis tests to assess the fit of the chosen distribution.

Key features of DistFit include easy use, flexibility in handling diverse distribution types, and efficient algorithms for parameter estimation. The package serves as a valuable asset for distribution analysis and hypothesis testing in Python.

1. **Introduction**

At the core of DistFit is the MLE methodology, a robust statistical technique to estimate the parameters of probability distributions by maximizing the likelihood of the observed data.

The Goodness of Fit (GoF) module within the DistFit package plays a crucial role in evaluating how well a probability distribution model fits observed data. This module utilizes statistical tests to compare the observed data against the theoretical distribution, helping users make informed decisions about the appropriateness of the chosen model for their dataset. Through GoF analysis, users are able to assess the validity of the fitted distribution.

The package is applicable to a broad range of statistical needs. It has the capability to model discrete phenomena, such as binary outcomes, counts, and event occurrences, as well as continuous variables, such as time intervals and measurements. DistFit also provides users with insightful visualizations, enabling the exploration of how well the chosen distribution aligns with the observed data.

This paper serves as a guide to users from the Bernoulli distribution, modeling binary events, to the Weibull and Gamma distributions used in reliability engineering , DistFit is applicable to a diverse range of statistical scenarios. This paper assists users to select the most suitable distribution for their datasets.

1. **Methodology**

DistFit has three distinct modules: datagen, distfit, and gof. These modules perform data generation, distribution fitting, data visualization, and goodness-of-fit analysis. Additionally, an examples file demonstrates real-world applications of the datagen, distfit, and gof modules. It serves as a practical guide, describing how to use DistFit to fit distributions to datasets and analyze them. The examples show the versatility of DistFit in fitting distributions to data and assessing the goodness of fit. The subsequent sections describe the specific details of each module, their functionalities and how to use them.

* 1. **datagen**

The datagen module within DistFit is used for random data generation based on specified probability distributions. This module provides a convenient way for users to create artificial datasets of various probability distributions. The primary purpose of the datagen module is to facilitate testing and experimentation with DistFit's distribution fitting capabilities.

* + 1. **Datagen**

The Datagen takes distribution type, the desired number of data, and distribution-specific parameters. It supports a variety of distribution types, including Normal, Geometric, Binomial, Poisson, Exponential, Gamma, Weibull, Uniform, and Bernoulli.

Initialization Parameters are:

* dist\_type: Type of probability distribution (e.g., 'Normal', 'Geometric', 'Binomial').
* row\_count: Number of rows (data points) to be generated.
* par: Distribution-specific parameters. The format of parameters varies based on the distribution type.

The data\_generation method in this module generates synthetic data based on the specified distribution type and parameters. It utilizes NumPy's random number generation functions for each distribution type.

* 1. **distfit**

The methodology implemented in the Fitting class of the DistFit Python package revolves around fitting data to various probability distributions using MLE method. The class supports both discrete and continuous distributions. The key components of the methodology involve Initialization, Guessing Distributions, and parameter estimation of various distribution types. Additionally, the plot method can be used to visually compare the fitted distributions versus the data histogram.

* + 1. **Initialization**

The class is initialized with a given distribution and dataset (data), which can be provided as a pandas DataFrame or a pandas/numpy Series. If the input is not a DataFrame, it is converted to the correct format.

Essential statistics about the dataset are computed during initialization, such as mean (mu), standard deviation (sigma), data size (size), minimum value (data\_min), maximum value (data\_max), and the original data itself.

* + 1. **Guessing Distributions**

The ‘guess\_distributions’ method determines the possible distributions based on the data type (discrete or continuous) and characteristics of the dataset.

For discrete data, it checks for specific conditions to identify Bernoulli distribution or selects from a predefined list of discrete distributions.

For continuous distributions, it considers all the specified continuous distributions, adjusting the list based on the properties of the dataset. For example, if the data histogram is skewed, it cannot be normally distributed.

This function should be used as a high-level guess of the distribution type. It is always the subject matter expert who knows which distribution is best suited for the data,

* + 1. **Fit (Parameter Estimation)**

MLE provides a systematic and versatile approach to fitting data to a range of distributions. It is also empowering users with tools for exploratory data analysis. The visualizations provide an understanding of how well the fitted distributions align with the observed data.

* 1. **Gof (Goodness of Fit Test)**

The gof (Goodness of Fit) module in DistFit evaluates how well a chosen probability distribution fits the observed data. It employs chi-squared goodness of fit test to assess the statistical significance of the difference between observed and expected frequencies.

The following are the key components of this module:

* **Initialization**: Initializes the Gof class with the distribution type (dist\_type) and its parameters (par). The class automatically determines the number of estimated parameters (s) based on the distribution type.
* **Frequency Calculation**: Calculates the frequency of the expected data within the specified bin edges. This step is required to calculate the test statistics for goodness-of-fit test. This calculation is a numerical approximation of the expected data frequency as it is performed numerically.
* **Binning Optimization**: Optimizes bins by combining adjacent bins if the expected frequency is less than 5. However, it ensures a minimum of 3 bins even when frequencies are low.
* **Goodness of Fit Test**: Conducts a goodness-of-fit test using the chi-squared statistic. It generates expected data based on the specified distribution, calculates observed and expected frequencies, optimizes bins, and performs the chi-squared test. As a result of approximate frequency calculation for the expected data, the goodness of fit test is approximate, and the results should be treated cautiously when the test statistics and critical values are close in values.

The goodness-of-fit analysis should be used in conjunction with the "distfit" class to ensure the selected distribution and its estimated parameters are a good fit. The “distfit” class, as previously discussed, efficiently fits data to a variety of probability distributions using the MLE method.

* 1. **Supported Distribution Types and Tests**

As discussed, DistFit is a useful Python package designed for efficiently fitting data to a variety of probability distributions using MLE method. The package supports both discrete and continuous distributions, providing users with a versatile tool for statistical analysis. The characteristics of the datasets influence the choice of an appropriate distribution.

In this section, we introduce the main distribution types supported by DistFit, some examples and results.

* + 1. **Continuous Distributions**
       1. **Uniform Distribution**

The Uniform distribution models outcomes with equal likelihood over a specified range.

DistFit estimates the lower *a* and upper *b* bounds through the MLE method. This type of distribution is useful when each outcome within a range is equally likely, as seen in scenarios like random number generation.

As an example, a uniform dataset is generated based on a uniform distribution. The following code snippet outlines the process of fitting this dataset, the MLE estimation of the parameters, visualization of the fitted distribution, and a subsequent Goodness of Fit (GoF) test.

import DistFit.distfit as dft

import DistFit.datagen as dgn

import DistFit.gof as gof

# Generate data from a uniform distribution

data\_dist = dgn.Datagen(dist\_type='uniform', row\_count=2000, par=(2.3, 4.1), seed=1)

data = data\_dist.data\_generation()

# Estimate MLE parameters using DistFit

model = dft.Fitting(data, dist\_type='uniform')

a, b = model.fit()

print("The fitted MLE parameters are %.4f %.4f." % (a, b))

# Explore possible distributions using DistFit

possible\_distributions = model.guess\_distributions()

# Visualize the fitted uniform distribution

model.uniform\_plot((a, b))

# Perform Goodness of Fit test using DistFit

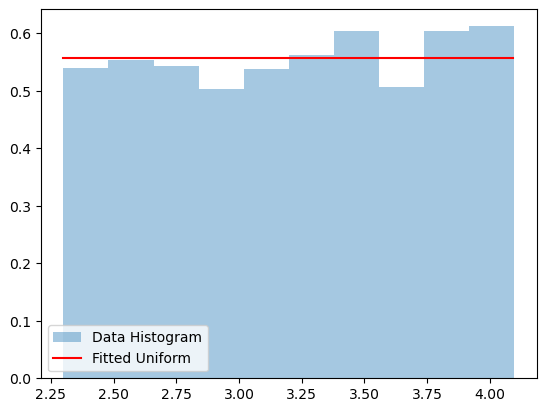
gof\_test = gof.Gof(dist\_type='uniform', par=(a,b))

gof\_test.gof(data)

The fitted MLE parameters are 2.3002 4.0973.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: ['Uniform', 'Exponential', 'Normal', 'Weibull', 'Gamma']



*Figure 1: Fitted Uniform distribution versus generated data histogram*

Test Statistics: 0.6157; Critical Value: 5.9915

Accept H0 that the distribution is a good fit at the given significance level.

True

In this example, we generated a dataset with 2000 samples from a uniform distribution with parameters a=2.3 and b=4.1. The DistFit package efficiently estimates the uniform parameters (*a* and *b*), identifies the possible distributions, and visualizes the fitted uniform distribution versus the data histogram. It successfully identifies the possible distributions for the given dataset as expected. The ability to accurately propose potential distribution types is a useful aspect of the package, providing users with valuable insights into the underlying nature of their data.

Furthermore, the GoF test confirms that the fitted distribution is a good fit at the specified significance level (default=0.05), as indicated by the acceptance of the null hypothesis.

* + 1. **Exponential Distribution**

The Exponential distribution models the interarrival time in a Poisson process.

DistFit estimates the rate parameter, (1/rate), through the MLE method. It is commonly used in scenarios where the focus is on the time between events, such as the time between arrivals in a queue.

The code below, demonstrates how to fit data to an exponential distribution using DistFit package. It includes data generation, MLE parameter fitting, exploration of possible distributions, visualization of the fitted distribution, and a goodness-of-fit test. The printed MLE estimation of the parameter, the possible distributions, and the result of the goodness-of-fit test are.

import DistFit.distfit as dft

import DistFit.datagen as dgn

import DistFit.gof as gof

# Generate data from an exponential distribution

data\_dist = dgn.Datagen(dist\_type='exponential', row\_count=2000, par=(0.35), seed=1)

data = data\_dist.data\_generation()

# Estimate MLE parameters using DistFit

model = dft.Fitting(data, dist\_type='exponential')

mle\_param = model.fit()

print("The fitted MLE parameter is %.4f." %mle\_param)

# Explore possible distributions using DistFit

possible\_distributions = model.guess\_distributions()

# Visualize the fitted uniform distribution

model.exponential\_plot(mle\_param)

# Perform Goodness of Fit test using DistFit

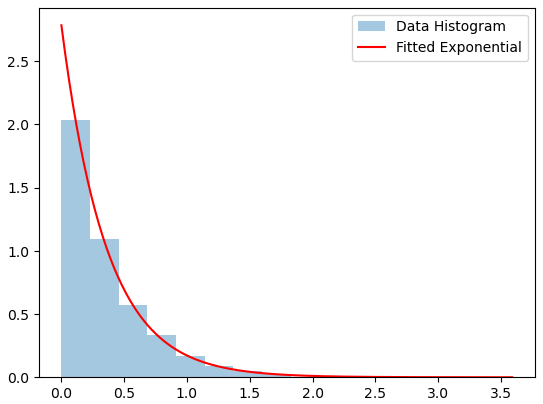
gof\_test = gof.Gof(dist\_type='exponential', par=mle\_param)

gof\_test.gof(data)

The fitted MLE parameter is 0.3593.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: ['Uniform', 'Exponential', 'Weibull', 'Gamma']



*Figure 2: Fitted Exponential distribution versus generated data histogram*

Test Statistics: 4.0574; Critical Value: 7.8147

Accept H0 that the distribution is a good fit at the given significance level.

True

Data with 2000 samples are generated from an exponential distribution with 1/rate parameter of 0.35. The MLE method estimates the parameter of the exponential distribution, and for the generated data, the fitted MLE estimation of the parameter is found to be 0.3593 which is a reasonable estimation of the true parameter. DistFit explores various distribution types based on the nature of the data. In this case, normal distribution is eliminated due to non-symmetrical data histogram; therefore, the possible distributions include 'Uniform', 'Exponential', 'Weibull', and 'Gamma'.

The Goodness of Fit (GOF) test is included for the evaluation of how well the fitted distribution aligns with the observed data. In the provided example, the test statistic is calculated as 4.0574, and the critical value at a 5% significance level is 7.8147. The outcome of the test is conclusive: accept the null hypothesis (H0) that the distribution is a good fit at the given significance level. Therefore, the fitted exponential distribution adequately represents the underlying nature of the generated data.

* + 1. **Normal Distribution**

The Normal distribution is a versatile distribution modeling a wide range of phenomena.

DistFit estimates the mean ('') and standard deviation ('') through the MLE method. It is widely used in scenarios where data distribution is symmetric and follows a bell-shaped curve, such as human height.

To showcase DistFit package fitting a dataset generated from a normal distribution, we conducted a fitting and goodness-of-fit (GoF) analysis. The following Python code demonstrates each step of the process.

import DistFit.distfit as dft

import DistFit.datagen as dgn

import DistFit.gof as gof

# Generate data from a normal distribution

data\_dist = dgn.Datagen(dist\_type='normal', row\_count=2000, par=(3,2), seed=1)

data = data\_dist.data\_generation()

# Estimate MLE parameters using DistFit

model = dft.Fitting(data, dist\_type='normal')

mu, sigma = model.fit()

print("The fitted MLE parameter is %(m).4f %(s).4f." %{'m':mu, 's':sigma})

# Explore possible distributions using DistFit

possible\_distributions = model.guess\_distributions()

# Visualize the fitted uniform distribution

model.normal\_plot(params=(mu, sigma))

# Perform Goodness of Fit test using DistFit

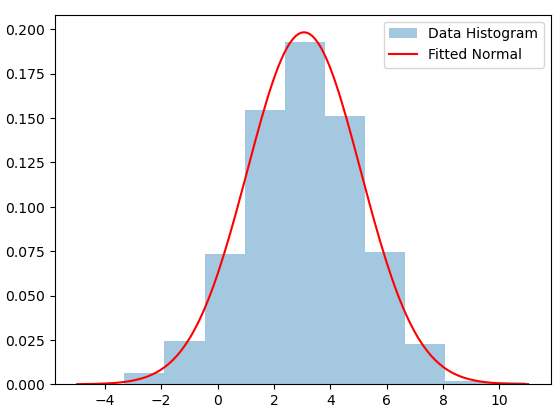
gof\_test = gof.Gof(dist\_type='normal', par=(mu,sigma))

gof\_test.gof(data)

The fitted MLE parameter is 3.0661 2.0122.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: ['Uniform', 'Normal']



*Figure 3: Fitted Normal distribution versus generated data histogram*

Test Statistics: 4.1215; Critical Value: 5.9915

Accept H0 that the distribution is a good fit at the given significance level.

True

# perform Goodness of Fit test for uniform

# first fit data to uniform

model = dft.Fitting(data, dist\_type='uniform')

gof\_test = gof.Gof(dist\_type='uniform', par=model.fit())

gof\_test.gof(data)

Test Statistics: 1726.9527; Critical Value: 5.9915

Reject H0, the distribution is NOT a good fit at this significance level.

False

In this example, we generated synthetic data from a Normal distribution with a mean (μ) of 3 and a standard deviation (σ) of 2. We then used DistFit to estimate the MLE estimates of the parameters, explore the possible distributions, and visually inspect the fitted Normal distribution.

The estimated MLE parameters for the Normal distribution are approximately 3.0661 for μ, and 2.0122 for the standard deviation (σ),both aligned with the parameters of the underlying distribution of the observed data.

DistFit explores and suggests possible distributions for the given dataset based on the MLE parameters. In this example, it suggests that the data could be fitted to either a uniform or a Normal distribution and then, by conducting goodness of fit tests for these two distribution types, it finds Normal distribution as a good fit with test statistics value of 4.1215 (less than critical value of 5.99) while Uniform distribution with test statistics value of 1726.96 (larger than critical value) is rejected as a good fit.

* + 1. **Weibull Distribution**

The Weibull distribution models reliability and life data. It is applicable in reliability engineering to schedule periodic maintenance and model the probability of failure, with the ability to represent different failure patterns.

where a is the shape parameter and b is the scale parameter.

The following example demonstrates the usage of the DistFit package for fitting a Weibull distribution to a generated dataset (from a Weibull distribution) and performing a Goodness of Fit (GoF) test.

import DistFit.distfit as dft

import DistFit.datagen as dgn

import DistFit.gof as gof

# Generate data from a weibull distribution

data\_dist = dgn.Datagen(dist\_type='weibull', row\_count=2000, par=(2.1,4.3), seed=1)

data = data\_dist.data\_generation()

# Estimate MLE parameters using DistFit

model = dft.Fitting(data, dist\_type='weibull')

a, b = model.fit()

print("The fitted MLE parameter is %(m).4f %(s).4f." %{'m':a, 's':b})

# Explore possible distributions using DistFit

possible\_distributions = model.guess\_distributions()

# Visualize the fitted uniform distribution

model.weibull\_plot((a,b))

# Perform Goodness of Fit test using DistFit

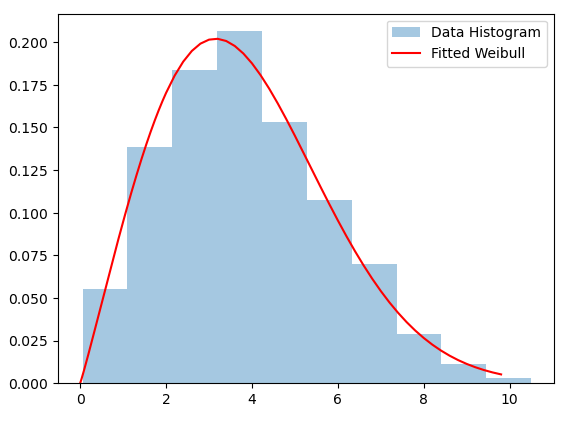
gof\_test = gof.Gof(dist\_type='weibull', par=(a,b))

gof\_test.gof(data)

The fitted MLE parameter is 2.0678 4.3409.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: ['Uniform', 'Exponential', 'Normal', 'Weibull', 'Gamma']



*Figure 4: Fitted Weibull distribution versus generated data histogram*

Test Statistics: 3.4686; Critical Value: 5.9915

Accept H0 that the distribution is a good fit at the given significance level.

True

# perform Goodness of Fit test for all selected distribution types:

dist\_types=[x.lower() for x in possible\_distributions]

for dt in dist\_types:

print('\nfor ',dt, ' distribution type:')

model = dft.Fitting(data, dist\_type=dt)

gof\_test = gof.Gof(dist\_type=dt, par=model.fit())

gof\_test.gof(data)

for uniform distribution type:

Test Statistics: 1029.1273; Critical Value: 5.9915

Reject H0, the distribution is NOT a good fit at this significance level.

for exponential distribution type:

Test Statistics: 876.7024; Critical Value: 7.8147

Reject H0, the distribution is NOT a good fit at this significance level.

for normal distribution type:

Test Statistics: 74.5496; Critical Value: 5.9915

Reject H0, the distribution is NOT a good fit at this significance level.

for weibull distribution type:

Test Statistics: 3.4686; Critical Value: 5.9915

Accept H0 that the distribution is a good fit at the given significance level.

for gamma distribution type:

Test Statistics: 43.2718; Critical Value: 5.9915

Reject H0, the distribution is NOT a good fit at this significance level.

In this case, the estimated MLE parameters for the Weibull distribution are the shape parameter (a) with value of approximately 2.0678, and the scale parameter (b) with value of approximately 4.3409. The estimated parameters are successfully aligning with true parameters of the underlying data distribution.

DistFit explores and suggests possible distributions for the given dataset based on the MLE parameters. In this example, it suggests that the data could be fitted to a variety of distributions, including Uniform, Exponential, Normal, Weibull, and Gamma.

In the subsequent analysis, upon conducting goodness-of-fit tests across all the suggested distribution types, it is evident that only the Weibull distribution, with a test statistics value of 3.47 below the critical value of 5.99, is a good fit. Conversely, the remaining distribution types yield test statistics exceeding the critical value, leading to the rejection of the null hypothesis for these distributions.

* + 1. **Gamma Distribution**

The Gamma distribution is a versatile distribution that generalizes exponential, chi-squared, and Erlang distributions. It is applicable in scenarios requiring the modeling of waiting times, sums of exponential random variables, and various other statistical applications. It is usually used to model time to failure of electrical components.

Where is the shape parameter, is the scale parameter, and is the Gamma function.

The lookup method is a utility for linear interpolation based on a look-up table. It is used in the Gamma\_fit method to find from a precomputed table based on the sample mean. DistFit estimates the  and parameters through the MLE method.

The following code shows how to use DistFit to fit data to a gamma distribution.

import DistFit.distfit as dft

import DistFit.datagen as dgn

import DistFit.gof as gof

# Generate data from a Gamma distribution

data\_dist = dgn.Datagen(dist\_type='gamma', row\_count=5000, par=(3.1,5.6), seed=1)

data = data\_dist.data\_generation()

# Estimate MLE parameters using DistFit

model = dft.Fitting(data, dist\_type='gamma')

a, b = model.fit()

print("The fitted MLE parameter is %(a).4f %(b).4f." %{'a':a, 'b':b})

# Explore possible distributions using DistFit

possible\_distributions = model.guess\_distributions()

# Visualize the fitted uniform distribution

model.gamma\_plot((a,b))

# Perform Goodness of Fit test using DistFit

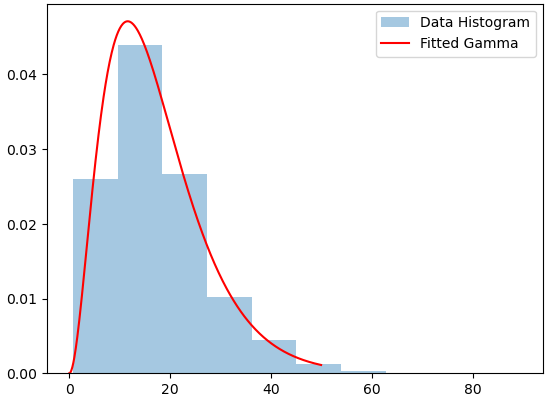
gof\_test = gof.Gof(dist\_type='gamma', par=(a,b))

gof\_test.gof(data, k=5)

The fitted MLE parameter is 3.0262 5.7138.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: ['Uniform', 'Exponential', 'Weibull', 'Gamma']



*Figure 5: Fitted Gamma distribution versus generated data histogram*

Test Statistics: 1.7092; Critical Value: 3.8415

Accept H0 that the distribution is a good fit at the given significance level.

True

In this example, the data are radomly generated from a Gamma distribution with given parameters (α=3.1, β=5.6) using Datagen.

The estimated MLE parameters for the Gamma distribution are, approximately 3.0262 for the shape parameter α, and 5.7138 for the scale parameter β.

The GoF test assesses the agreement between the observed data and the fitted Gamma distribution. In this instance, the result is True, indicating the acceptance of the null hypothesis (H0). Consequently, at the designated significance level, it implies that the Gamma distribution is deemed a suitable fit for the observed data.

* 1. **Discrete Distributions**
     1. **Binomial Distribution**

The Binomial distribution extends the Bernoulli distribution to multiple independent trials. It models the number of successes in a fixed number of trials, denoted by *n*.

DistFit estimates the probability of success, *p*, through MLE, when *n* is known and specified by the user. It is Applicable in scenarios where there are a fixed number of independent trials, such as the number of successful attempts in a series of experiments. The following is an example demonstrating the application of the code for Binomial data.

**import DistFit.distfit as dft**

**import DistFit.datagen as dgn**

**import DistFit.gof as gof**

**# Generate data from a binomial distribution**

data\_dist = dgn.Datagen(dist\_type='binomial', row\_count=2000, par=(10, 0.7), seed=1)

data = data\_dist.data\_generation()

**# Estimate MLE parameters using DistFit**

**model = dft.Fitting(data, dist\_type='binomial', n=10)**

**(n,p) = model.fit()**

**print("The fitted MLE parameters are %(n)d %(p).4f." %{'n':n, 'p':p})**

**# Explore possible distributions using DistFit**

**possible\_distributions = model.guess\_distributions()**

**# Visualize the fitted uniform distribution**

**model.binomial\_plot((n,p))**

**# Perform Goodness of Fit test using DistFit**

**gof\_test = gof.Gof(dist\_type='binomial', par=(n,p))**

**gof\_test.gof(data, k=10, alfa=0.05)**

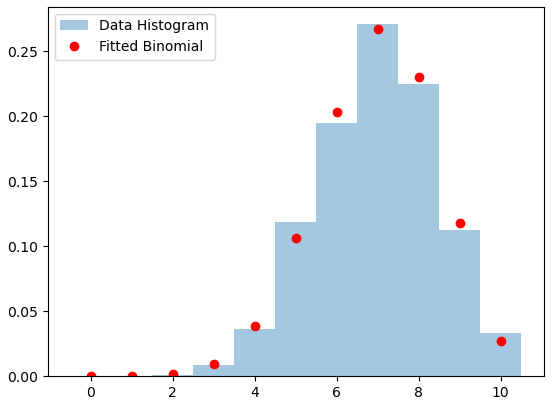
Note: When estimating p with very rare events and a small n, using MLE estimator leads to p=0 which sometimes is unrealistic and undesirable. In such cases, use alternative estimators.

The fitted MLE parameters are 10 0.6968.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: ['Binomial', 'Geometric', 'Poisson']

If data are binomial, n is at least 10



*Figure 6: Fitted Binomial distribution versus generated data histogram*

Test Statistics: 8.3381; Critical Value: 14.0671

Accept H0 that the distribution is a good fit at the given significance level.

True

This code generates a dataset with 2000 observations following a Binomial distribution with parameters *n*=10 and *p*=0.7. It then uses the DistFit package to estimate the MLE parameters for a Binomial distribution based on the generated data.

The fitted MLE parameter is determined to be p=0.697, note that n was predefined by the user. The analysis considers a limited set of distributions in the library, and in this case, the possible distributions for the data include 'Binomial', 'Geometric', and 'Poisson'. Additionally, a note emphasizes that if the data is Binomial, n should be at least 10.

A Goodness of Fit (GoF) test is performed, comparing the observed data to the fitted Binomial distribution. The test statistic is calculated as 8.3381, and the critical value at a significance level of 0.05 is 14.0671. The result of the test is True, indicating acceptance of the null hypothesis (H0) of a good fit at the specified significance level. Therefore, Binomial distribution is considered a good fit for the observed data.

A potential issue with MLE estimation of binomial is when dealing with very rare events and a small *n*, leading to an estimated probability (*p*) of 0, which might be unrealistic. In such cases, alternative estimators are recommended.

Next, we run the GoF test for the other two possible distributions, Geometric and Poisson. Both tests are rejected, indicating not a good fit.

# perform Goodness of Fit test for geometric

# first fit data to geometric

model = dft.Fitting(data, dist\_type='geometric')

gof\_test = gof.Gof(dist\_type='geometric', par=model.fit())

gof\_test.gof(data)

Test Statistics: 4039.7864; Critical Value: 7.8147

Reject H0, the distribution is NOT a good fit at this significance level.

False

# perform Goodness of Fit test for poisson

# first fit data to poisson

model = dft.Fitting(data, dist\_type='poisson')

gof\_test = gof.Gof(dist\_type='poisson', par=model.fit())

gof\_test.gof(data)

Test Statistics: 669.7525; Critical Value: 7.8147

Reject H0, the distribution is NOT a good fit at this significance level.

False

For the Geometric distribution, the test statistic is 4039.7864, and the critical value at a significance level of 0.05 is 7.8147. The result of the test is False, indicating rejection of the null hypothesis (H0) at 0.05 significance level. Therefore, the Geometric distribution is deemed not a good fit for the observed data.

Similarly, for the Poisson distribution, the test statistic is 669.7525, and the critical value at a significance level of 0.05 is 7.8147. The result of the test is False, indicating rejection of the null hypothesis (H0) at the specified significance level. Consequently, the Poisson distribution is also considered not a good fit for the observed data.

These results suggest that, among the considered distributions (Binomial, Geometric, and Poisson), only the Binomial distribution is a good fit for the generated dataset based on the Chi-squared GoF tests.

* + 1. **Bernoulli Distribution**

The Bernoulli distribution models binary outcomes, such as success or failure in a single trial.

DistFit estimates the probability of success, denoted as *p*, through the MLE method. This distribution is particularly useful for scenarios where there are only two possible outcomes. It commonly models scenarios like coin flips or determining success/failure in a single trial. The following code is an example of using the DistFit package for data generated from a Bernoulli distribution.

Import DistFit.distfit as dft

import DistFit.datagen as dgn

import DistFit.gof as gof

# Generate data from a Bernoulli distribution

data\_dist = dgn.Datagen(dist\_type=’binomial’, row\_count=2000, par=(1, 0.3), seed=1)

data = data\_dist.data\_generation()

# Estimate MLE parameters using DistFit

model = dft.Fitting(data, dist\_type=’bernoulli’)

fitted\_mle\_param = model.fit()

print(“The fitted MLE parameter is %.4f.” %fitted\_mle\_param)

# Explore possible distributions using DistFit

possible\_distributions = model.guess\_distributions()

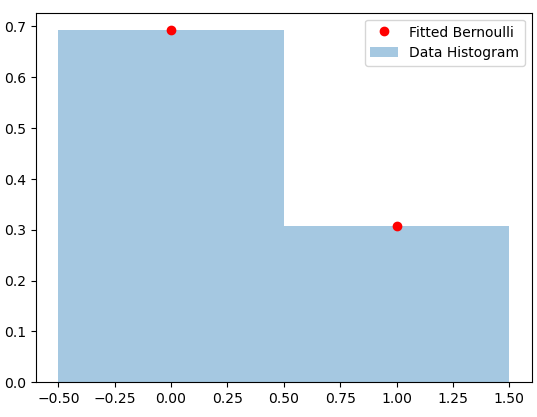
# Visualize the fitted Bernoulli distribution

model.bernoulli\_plot(fitted\_mle\_param)

The fitted MLE parameter is 0.3080.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: [‘Bernoulli’]



*Figure 7: Fitted Bernoulli distribution versus generated data histogram*

Goodness of fit is not valid for Bernoulli distribution, therefore if you try running Gof for a Bernoulli a Value Error is raised. Due to the fact that data are {0,1} only, the suggested distribution in Bernoulli only.

* + 1. **Geometric Distribution**

The Geometric distribution models the number of Bernoulli trials needed to achieve the first success. It is suitable for scenarios where the interest lies in the number of trials required for a specific outcome.

DistFit estimates the probability of success, 'p', through MLE. A complete example is demonstrated in the following code.

import DistFit.distfit as dft

import DistFit.datagen as dgn

import DistFit.gof as gof

# Generate data from a geometric distribution

data\_dist = dgn.Datagen(dist\_type='geometric', row\_count=2000, par=(.42), seed=1)

data = data\_dist.data\_generation()

# Estimate MLE parameters using DistFit

model = dft.Fitting(data, dist\_type='geometric')

fitted\_mle\_param = model.fit()

print("The fitted MLE parameter is %.4f." %fitted\_mle\_param)

# Explore possible distributions using DistFit

possible\_distributions = model.guess\_distributions()

# Visualize the fitted geometric distribution

model.geometric\_plot(fitted\_mle\_param)

# Perform Goodness of Fit test using DistFit

gof\_test = gof.Gof(dist\_type='geometric', par=(fitted\_mle\_param))

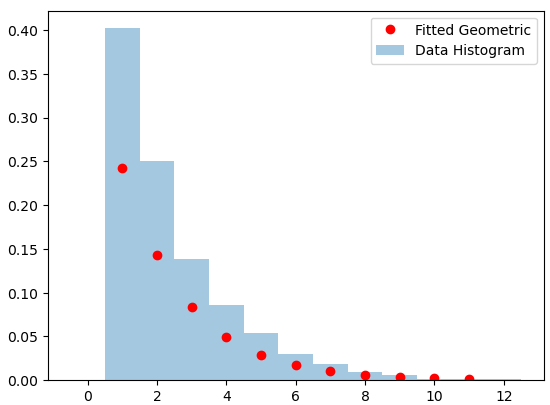
gof\_test.gof(data, k=10)

The fitted MLE parameter is 0.4114.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: ['Binomial', 'Geometric', 'Poisson']

If data are binomial, n is at least 12



*Figure 8: Fitted Geometric distribution versus generated data histogram*

Test Statistics: 12.4406; Critical Value: 15.5073

Accept H0 that the distribution is a good fit at the given significance level.

True

The code generates a dataset of size 2000 from a Geometric distribution with a probability parameter of 0.42. DistFit is used to estimate the parameter for the Geometric distribution using MLE. The fitted MLE parameter is 0.41, indicating the estimated probability parameter for the Geometric distribution. The GoF test statistics value is 12.44 which is less than critical value. Therefore, in this case, the null hypothesis (H0) is accepted, indicating that, at the 0.05 significance level, the Geometric distribution is considered a good fit for the observed data.

* + 1. **Poisson Distribution**

The Poisson distribution models the number of events occurring within fixed intervals. DistFit estimates the rate parameter, through the MLE method. It is applicable when dealing with count data in scenarios like the number of emails received in an hour or the number of customers arriving for a service.

The following code is an example of using the DistFit package and performing a goodness-of-fit analysis for data generated from a Poisson distribution.

import DistFit.distfit as dft

import DistFit.datagen as dgn

import DistFit.gof as gof

# Generate data from a Poisson distribution

data\_dist = dgn.Datagen(dist\_type='poisson', row\_count=2000, par=(15), seed=1)

data = data\_dist.data\_generation()

# Estimate MLE parameters using DistFit

model = dft.Fitting(data, dist\_type='poisson')

fitted\_param = model.fit()

print("The fitted MLE parameter is %.4f." %fitted\_param)

# Explore possible distributions using DistFit

possible\_distributions = model.guess\_distributions()

# Visualize the fitted uniform distribution

model.poisson\_plot(fitted\_param)

# Perform Goodness of Fit test using DistFit

gof\_test = gof.Gof(dist\_type='poisson', par=(fitted\_param))

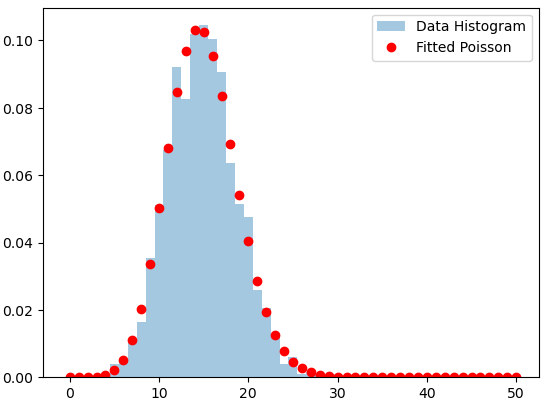
gof\_test.gof(data)

The fitted MLE parameter is 14.8965.

Note: only a limited number of distributions are considered in this library.

The possible distributions for the data are: ['Binomial', 'Geometric', 'Poisson']

If data are binomial, n is at least 33



*Figure 9: Fitted Gamma distribution versus generated data histogram*

Test Statistics: 2.3115; Critical Value: 7.8147

Accept H0 that the distribution is a good fit at the given significance level.

True

This code generates a dataset of size 2000 from a Poisson distribution with a mean parameter of 15. The fitted MLE parameter is 14.9, indicating the estimated mean parameter for the Poisson distribution. The GoF test statistics value is 2.31 which is less than critical value. Therefore, in this case, the null hypothesis (H0) is accepted, indicating that, at 0.05 significance level, the Poisson distribution is considered a good fit for the observed data.

A goodness of Fit test is also conducted for Geometric distribution, which is one of the possible distributions for our data.

import DistFit.distfit as dft

# perform Goodness of Fit test for geometric

# first fit data to geometric

model = dft.Fitting(data, dist\_type='geometric')

gof\_test = gof.Gof(dist\_type='geometric', par=model.fit())

gof\_test.gof(data, k=10)

Test Statistics: 3493.8377; Critical Value: 15.5073

Reject H0, the distribution is NOT a good fit at this significance level.

False

The results indicate that, based on the statistical test, the Geometric distribution is not a good fit for the data, which aligns with our expectations.

1. **Requirements**

In conjunction with the indispensable modules (datagen, distfit, and gof), DistFit mandates the installation of specific Python libraries to guarantee the efficient and error-free execution of the package. These prerequisites encompass a suite of essential tools and utilities tailored to augment the package's overall functionality. Therefore, it is recommended to ensure that the following dependencies are installed on your Python environment:

* **Python version**: DistFit is designed to seamlessly integrate with Python 3 versions. It is crucial to have Python installed, and for optimal compatibility and feature support, consider using the latest stable release.
* **Numpy**: At the heart of numerical operations in Python, NumPy is an indispensable library for array manipulations and mathematical computations. DistFit leverages NumPy's capabilities to enhance the efficiency of data processing and analysis.
* **Scipy**: An extension of NumPy, SciPy extends the functionality of scientific computing in Python. DistFit relies on SciPy for its comprehensive statistical distributions and advanced hypothesis testing.
* **Pandas**: Pandas offers a powerful framework for data manipulation and analysis. DistFit takes advantage of Pandas to seamlessly handle tabular data structures, providing flexibility in data input and manipulation.
* **Matplotlib**: Matplotlib stands out as a versatile plotting library that facilitates the creation of visually appealing and informative graphs. DistFit integrates Matplotlib to generate graphical representations of distribution fitting outcomes, enhancing the interpretability of results.
* **Statsmodels**: DistFit incorporates Statsmodels, a statistical modeling library, to enable advanced statistical analyses and diagnostics. This ensures a robust and comprehensive approach to exploring data distributions.
* **Scipy.stats**: The Scipy.stats module is a valuable resource for an extensive range of statistical functions. DistFit makes use of Scipy.stats to access various probability distributions and perform essential statistical calculations.
* **Math**: Math module allows user to perform mathematical functions. DisFit uses it to calculate some advanced mathematical functions such as Gamma function.
* **Bisect**: This module provides support for maintaining a list in sorted order without having to sort the list after each insertion. DisFit uses this function in lookup calculations for Gamma distribution.

1. **Challenges and Future Updates**

While the DistFit Python package offers a robust methodology for fitting data to various probability distributions, like any software project, it may encounter challenges that is worth to state in this article. Some potential future updates can help mitigate the challenges associated with the DistFit project. Here are some challenges that might be addressed through future enhancements to the project:

* **Distribution Selection**: Determining the appropriate probability distribution for a given dataset can be subjective and complex. Providing additional guidance or automated methods for users to assist in selecting suitable distributions can be helpful.
* **Handling Large Datasets**: Processing large datasets may lead to performance issues or memory constraints. For future updates, we can plan to optimize algorithms and explore the integration of parallel processing techniques. These enhancements aim to significantly improve the package's performance, particularly when dealing with larger datasets. The optimization efforts will focus on refining computational efficiency, and parallel processing strategies will be investigated to leverage the capabilities of multi-core systems, ensuring faster and more scalable data fitting processes.
* **Model Interpretability**: Interpreting the results of fitted models, especially for complex distributions, can be challenging for non-expert users. Adding more examples, and visualizations can aid users in interpreting the results of the fitted models.
* **User Interface Design**: If the package is intended for a broad audience, creating an intuitive and user-friendly interface might be challenging. We can plan to develop a user-friendly interface, possibly through a graphical user interface (GUI) or interactive visualization tools, to make the package accessible to a wider audience.
* **Maintenance and Updates**: Keeping the package up-to-date with the latest versions of dependencies and addressing issues reported by users can be challenging. We will Establish a maintenance plan, encourage user feedback, and actively address reported issues to ensure the package remains reliable and relevant.
* **Documentation and Educational Resources**: Users may face challenges understanding the concepts of probability distributions and how to use the package effectively. We will improve our work with provide more documentation, tutorials, and educational resources to assist users in understanding both the statistical concepts and the practical usage of the package.

1. **Conclusion**

The methodology provides a systematic and versatile approach to fitting data to a range of distributions, empowering users with tools for exploratory data analysis and statistical modeling. The visualizations help in understanding how well the fitted distributions align with the observed data. DistFit is not only as a distribution-fitting tool but as a comprehensive guide to the complicated world of probability distributions. By navigating through each distribution type and their parameters, users can harness the full potential of DistFit.

The gof module, or Goodness of Fit, is a critical component that enables users to assess the suitability of the fitted distribution. Through statistical tests and comparison with critical values, the module helps users make informed decisions about accepting or rejecting the null hypothesis, indicating whether the chosen distribution is a good fit for the observed data or not. Note that the GoF test used in this library is approximate and if the test statistics is close to the critical value, caution should be exercised.

While DistFit offers a powerful and convenient tool for distribution analysis, it is essential to note that the results are contingent on the assumptions made during the modeling process. Users should exercise caution and consider the specific characteristics of their datasets when interpreting the outcomes.

In summary, DistFit provides a valuable resource for researchers, data scientists, and statisticians seeking a comprehensive and user-friendly solution for probability distribution fitting in Python. Through the intricate details provided in this paper, users can navigate the package with confidence, harnessing the power of DistFit to gain valuable insights into the underlying nature of their data.

1. **References**

[Bienvenu](https://blog.joffreybvn.be/?source=post_page-----f7f618719e14--------------------------------), J., (2020), Deep dive: Create and publish your first Python library

Law, A. M. (2015). Simulation modeling and analysis (5th Edition). New York: Mcgraw-Hill

NumPy documentation, version 1.26, <https://numpy.org/doc/stable/index.html>