



Python libraries for estimation of Weibull parameters

▼ You can see this file by checking out the link below

👉 [Python libraries for estimation of Weibull parameters](#)

Libraries that I used

- pandas → for data preprocessing
- surpyval → for reliability analysis
- reliability → for reliability analysis

Pandas

- Pandas is a software library in Python for data manipulation and analysis. It supports dataframe to manipulate a database table.
- You can use Series and DataFrame with Pandas.










SurPyval

<https://surpyval.readthedocs.io/en/latest/index.html>

<https://surpyval.readthedocs.io/en/latest/>

- Reliability and survival analysis library in Python
- This library can estimate parameters and print a plot.

SurPyval Modeling Methods

 Method	 Para/Non-Para	 Observed	 Censored	 Truncated
Maximum Likelihood (MLE)	Parametric		Yes	Yes
Probability Plotting (MPP)	Parametric		Yes	Limited
Mean Square Error (MSE)	Parametric		Yes	Limited
Method of Moments (MOM)	Parametric		No	No

Aa Method	▼ Para/Non-Para	☑ Observed	☰ Censored	☰ Truncated
<u>Maximum Product Spacing (MPS)</u>	Parametric	☑	Yes	No (planned)
<u>Kaplan-Meier</u>	Non-Parametric	☑	Right only	Left only
<u>Nelson-Aalen</u>	Non-Parametric	☑	Right only	Left only
<u>Fleming-Harrington</u>	Non-Parametric	☑	Right only	Left only
<u>Turnbull</u>	Non-Parametric	☑	Yes	Yes

- In specific distribution, 'MLE' is default method, but you can use other methods.
- SurPyval attempts to make parameter estimation possible for any distribution with arbitrary combinations of observations, censoring, and truncation.
- You can use various parametric and non-parametric estimation methods.

Caution when you use the SurPyval

- The censored code is 1, and the failure code is 0 in Surpyval library → Therefore, you should be careful when you preprocess data.
- In minitab, you can set the censored code, and the failure code as you want.
- In Reliability, you can set censored values, failure values (not the censored code)
 - However, You should convert a DataFrame to a list.



How to use

- **Jupyter notebook, colab, vscode, etc.**

```
# Install the library
```

```
pip install surpyval
```

```
# estimation of paramters by using the weibull distribution of surpyval
```

```
import surpyval as surv
```

```
result = surv.Weibull.fit([uptimes(722, 0 ...)], [censored code(0/1)], how = 'what modeling methods? ex)MLE')
```

```
# print a plot
```

```
result.plot()
```

```
# MTTF
result.mean()


# For more details, refer to my python code.
```

Reliability

<https://reliability.readthedocs.io/en/latest/>

- *reliability* is a Python library for reliability engineering and survival analysis.
- It significantly extends the functionality of `scipy.stats`.
- Plot is more intuitive than `SurPyval`.

Supported Distributions

<u>Aa</u> Parametric Models	 Fitting a specific distribution to data
<u>Weibull Distribution</u>	Fit_Weibull_2P
<u>Exponential Distribution</u>	Fit_Exponential_1P
<u>Normal Distribution</u>	Fit_Normal_2P
<u>Lognormal Distribution</u>	Fit_Lognormal_2P
<u>Gamma Distribution</u>	Fit_Gamma_2P
<u>Beta Distribution</u>	Fit_Beta_2P
<u>Loglogistic Distribution</u>	Fit_Loglogistic_2P
<u>Gumbel Distribution</u>	Fit_Gumbel_2P
<u>Location shifting the distributions</u>	

Caution when you use Reliability

- You can set failure values and censored values, but each information must be separated.
- In the case of Excel data, it is possible to import and convert your data by using the reliability.
→ https://reliability.readthedocs.io/en/latest/Converting_data_between_different_formats.html



How to use

```
# Install the library
```

```
pip install reliability
```

```
# estimation of parameters and print a plot
```

```
from reliability.Fitters import Fit_Weibull_2P
from reliability.Probability_plotting import Weibull_probability_plot
from reliability.Distributions import Weibull_Distribution
import matplotlib.pyplot as plt
```

```
# you should convert a DataFrame to list !
```

```
result = Fit_Weibull_2P(failure = [failure uptimes(722,0...)], right_censored = [censored uptimes(3, 500...)])
```

```
# print a plot
```

```
Weibull_probability_plot(failures = list_fail, right_censored = list_censored)
plt.show()
```

```
# MTTF
```

```
MTTF = Weibull_Distribution(alpha = result.alpha, beta = result.beta).mean
```

Minitab vs Python Library's Results with sample data

	Python-SurPyval	Python-Reliability	Minitab
shape parameter	0.47707	0.47707	0.47707
scale parameter	788024.148981	788021	788024
MTTF	1725283.34555	1725275.94047	1725283

- The results between the minitab and SurPyval are the most similar.
- You can check this results by checking my codes at github.

Comparison of each program's result

- The following data is randomly generated.

Python - SurPyval

<u>Aa</u> Data	# Shape Parameter	# Scale Parameter	# MTTF	# Probability of Failure	# Reliability	# Number of Suspension	# Number of Fail
<u>Sample Data(1)</u>	0.48137	1197531.18433	2575438.40219	0.02794	0.97206	1129	44

<u>Aa</u> Data	# Shape Parameter	# Scale Parameter	# MTTF	# Probability of Failure	# Reliability	# Number of Suspension	≡ Number of Fail
<u>Sample Data(2).</u>	0.47368	1396342.62614	3101719.32397	0.02751	0.97249	1130	43
<u>Sample Data(3).</u>	0.47877	1302376.87262	2831225.08271	0.02737	0.97263	1147	43

Python - Reliability

<u>Aa</u> Period	# Shape Parameter	# Scale Parameter	# MTTF	# Probability of Failure	# Reliability	# Number of Suspension	≡ Number of Fail
<u>Sample Data(1).</u>	0.48139	1197182.60554	2574489.25144	0.02794	0.97206	1129	44
<u>Sample Data(2).</u>	0.47369	1396107.83235	3101057.05628	0.02751	0.97249	1130	43
<u>Sample Data(3).</u>	0.47877	1302438.1842	2831393.55533	0.02737	0.97263	1147	43

Minitab

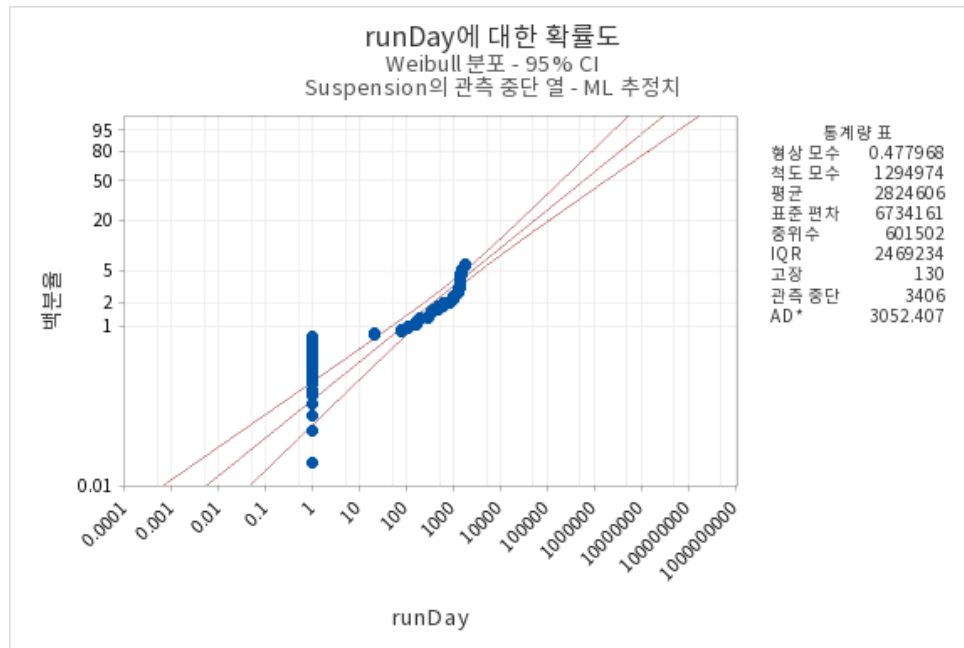
<u>Aa</u> Period	# Shape Parameter	# Scale Parameter	# MTTF	# Probability of Failure	# Reliability	# Number of Suspension	≡ Number of Fail
<u>Sample Data(1).</u>	0.48137	1197531	2575438	0.02794	0.97206	1129	44
<u>Sample Data(2).</u>	0.47368	1396343	3101719	0.02751	0.97249	1130	43
<u>Sample Data(3).</u>	0.47877	1302377	2831225	0.02737	0.97263	1147	43

Weibull++

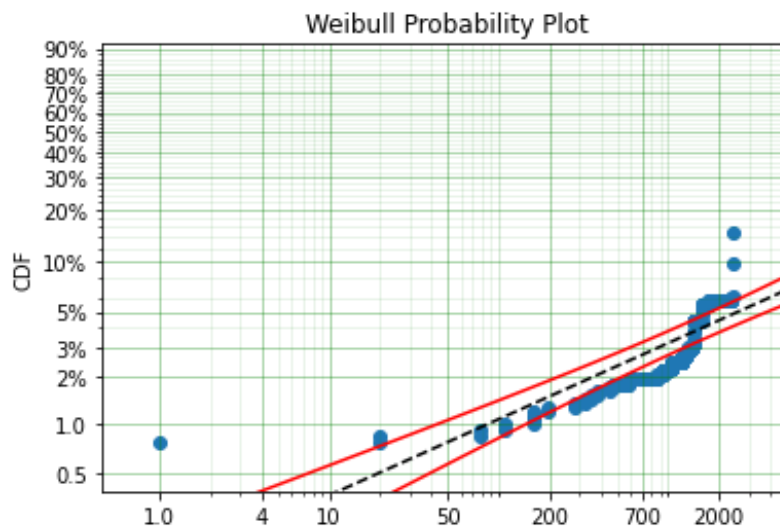
<u>Aa</u> Period	# Shape Parameter	# Scale Parameter	# MTTF	# Probability of Failure	# Reliability	# Number of Suspension	≡ Number of Fail
<u>Sample Data(1).</u>	0.48	1196754	2573277	0.0279	0.9721	1129	44
<u>Sample Data(2).</u>	0.47	1398258	3107226	0.0275	0.9725	1130	43
<u>Sample Data(3).</u>	0.48	1301719	2829379	0.0274	0.9726	1147	43

Comparing each plot

Minitab's Plot

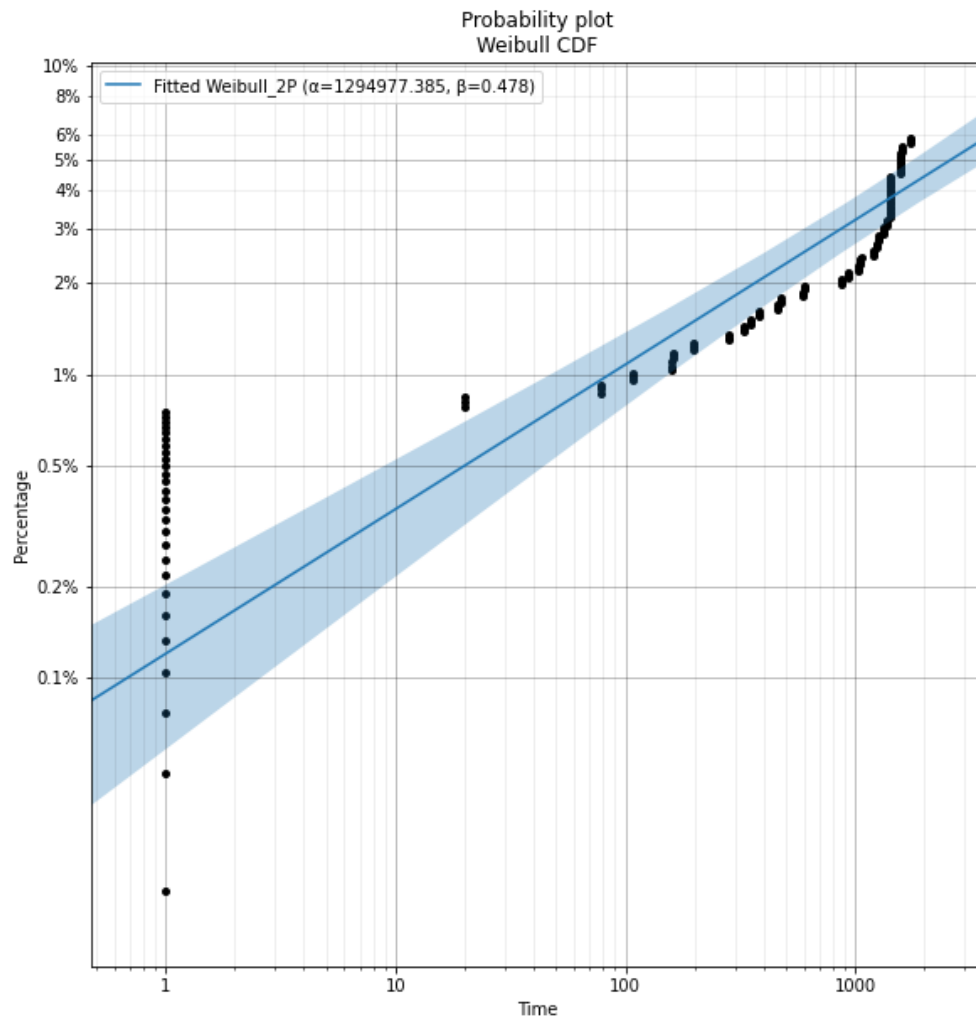


Python - Surpyval's PLOT



- You can save the plot, but you can't modify it.

Python - Reliability's PLOT



- Plot can be modified and saved.
- An intuitive graph

Using Weibayese, not the weibull distribution

- Weibayese small sample, zero and sudden death failure test and data analysis techniques in order to mathematically and graphically quantify the risk associated with assuming an incorrect value of the Weibull shape parameter, beta (beta).
- It should be used considering that it may be somewhat inaccurate.

The library I used

- Weibull library

- It is necessary to convert a DataFrame to a list.
- <https://weibull.readthedocs.io/en/v0.0.11/>



How to use

```
# Install the library
```

```
pip install weibull
```

```
# estimation of parameters and print a plot
```

```
import weibull
```

```
# you can designate a confidence level and beta
```

```
result = weibull.Weibayes(data = [uptimes], confidence_level= 0.95, beta = 1)
```

```
# print a plot
```

```
result.plot()
```

Python - Weibull's Plot (Weibayes)

