[**https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6**](https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6)

**Survival Analysis in Python**

A Quick Guide to The Weibull Analysis

The Weibull Analysis is very popular among reliability engineers due to its flexibility and straightforwardness. This guide will demonstrate basic concepts of the Weibull Analysis with sample code. In order to conduct the Weibull Analysis, we will be using the open source Python package [***predictr***](https://pypi.org/project/predictr/)*.*



Image by Author

**Table of contents**

1. [Installing and Using ***predictr***](https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6#3bf2)
2. [The Weibull Plot](https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6#2931)
3. [Parameter Estimation: MRR vs. MLE](https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6#4837)
4. [Confidence Intervals](https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6#c6b3)
5. [Bias-Corrections](https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6#1829)
6. [Comprehensive Plots](https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6#cc4a)
7. [Conclusion](https://towardsdatascience.com/survival-analysis-in-python-a-quick-guide-to-the-weibull-analysis-5babd4f137f6#d5c4)

**Installing and Using *predictr***

You need to have Python 3 installed (version >3.5). If you’re new to Python, just download [anaconda](https://www.anaconda.com/) and set up a virtual environment according to the [anaconda documentation](https://docs.anaconda.com/), e.g. paste this code into terminal (macOS, Linux) and command (Windows), respectively:

conda create -n my\_env python=3.10

This code creates a new virtual environment called *my\_env* with Python 3.10. Feel free to change the name and Python version.

The next step involves the installation predictr using [pip](https://pip.pypa.io/en/stable/) in terminal (or command):

conda activate my\_env  
pip install predictr

In order to use predictr in your IDE or text editor of choice, just import the *predictr* module in your python file:

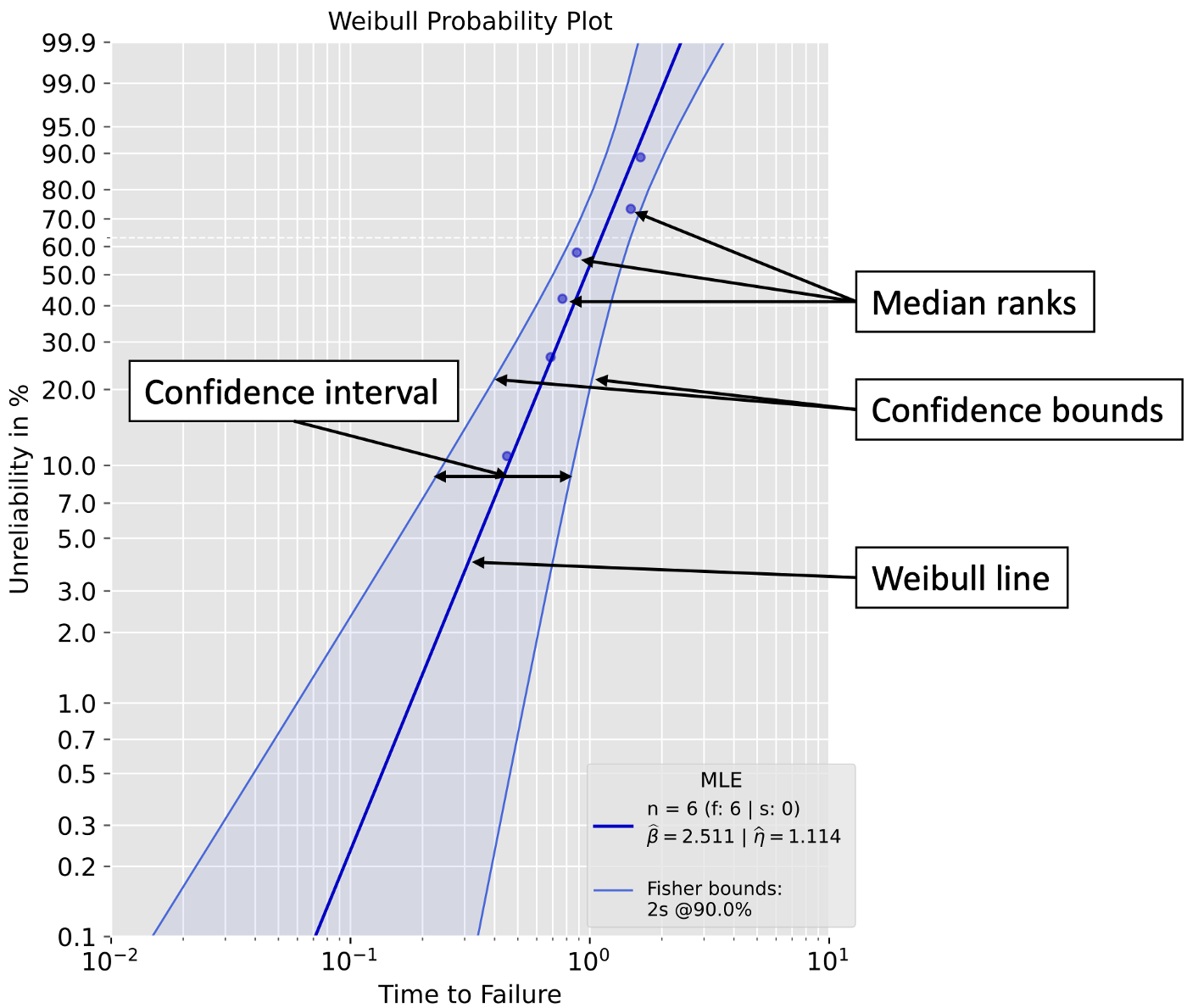
import predictr

[***predictr***](https://pypi.org/project/predictr/) has two classes: *Analysis* for Weibull analyses, and *PlotAll* for detailed plots. For a comprehensive documentation of *predictr* with many examples check out the [official documentation](https://tvtoglu.github.io/predictr/).

**The Weibull Plot**

Probability plots allow to grasp an idea about the present data and compare regression lines, i.e. failure modes and failure data, with each other. In Weibull Analysis the plot is called Weibull Probability Plot. It is essential to understand the plot. Usually, the plot consists of…

* a double-logarithmic y-axis (unreliability),
* a logarithmic x-axis (time to failure, e.g. number of cycles),
* a Weibull line (parameter estimation of the Weibull shape and scale parameter) and median ranks of the given data,
* and confidence bounds (one-sided or two-sided).



Weibull Probability Plot (Image by Author)

The legend is optional, however it is recommend to show information like sample size n (=number of failures f + number of suspensions s), the parameter estimation method that is being used (Maximum Likelihood Estimation (MLE) or Median Rank Regression (MRR) or other), the actual estimated Weibull parameters (β and η), which confidence bounds method is being used (Fisher Bounds, Likelihood Ratio Bounds, Bootstrap Bounds, Beta-Binomial Bounds, Monte-Carlo Pivotal Bounds, …), and the confidence level.

**Parameter Estimation: MRR vs. MLE**

Both MLE and MRR can be used to estimate the Weibull shape and scale parameter. In this tutorial, we consider the Weibull location parameter to be zero, i.e. a two-parameter Weibull distribution:

* The shape parameter β represents the slope of the Weibull line and describes the failure mode (-> the famous *bathtub curve*)
* The scale parameter η is defined as the x-axis value for an unreliability of 63.2 %

Let’s assume we gathered following, type-II right-censored data from testing:

1. Failures: 0.4508831, 0.68564703, 0.76826143, 0.88231395, 1.48287253, 1.62876357 (6 failures in total)
2. Suspensions: 1.62876357, 1.62876357, 1.62876357, 1.62876357 (4 suspensions in total)

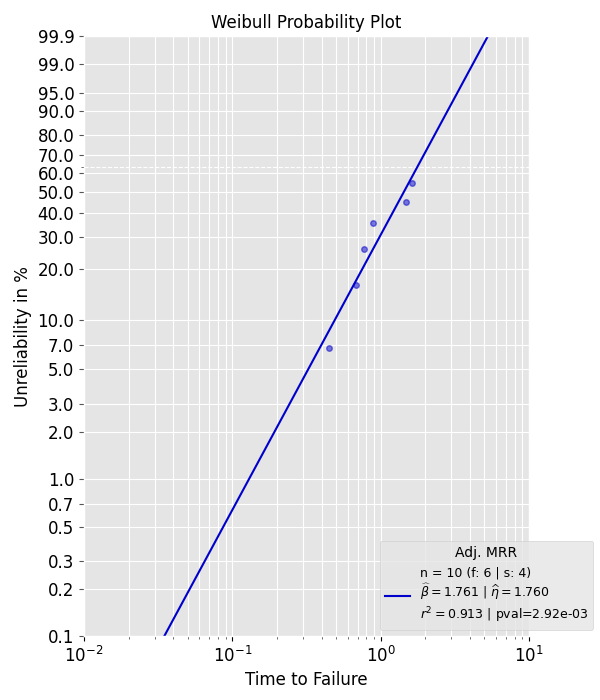
Our data is censored, and therefore we have to deal with suspensions. Suspensions are units that did not fail during testing. MRR and MLE handle this information differently.

**MRR**

The Medan Rank Regression uses so called median ranks and the method of least squares in order to determine the Weibull parameters. Median ranks are estimated unreliability (or reliability) values for each failure (censored times cannot be considered by MRR, only the total number of suspensions). More precisely, MRR estimates are based on the median ranks of the individual failure time and not the actual failure time values.

We will use the Analysis class in *predictr*in order to conduct the Weibull Analysis.

from predictr import Analysis# Data from testing  
# Failures and suspensions are lists containing the valuesfailures = [0.4508831, 0.68564703, 0.76826143, 0.88231395, 1.48287253, 1.62876357]  
suspensions = [1.62876357, 1.62876357, 1.62876357, 1.62876357]# Weibull Analysis  
x = Analysis(*df*=failures, ds=suspensions, *show*=True)  
x.mrr()

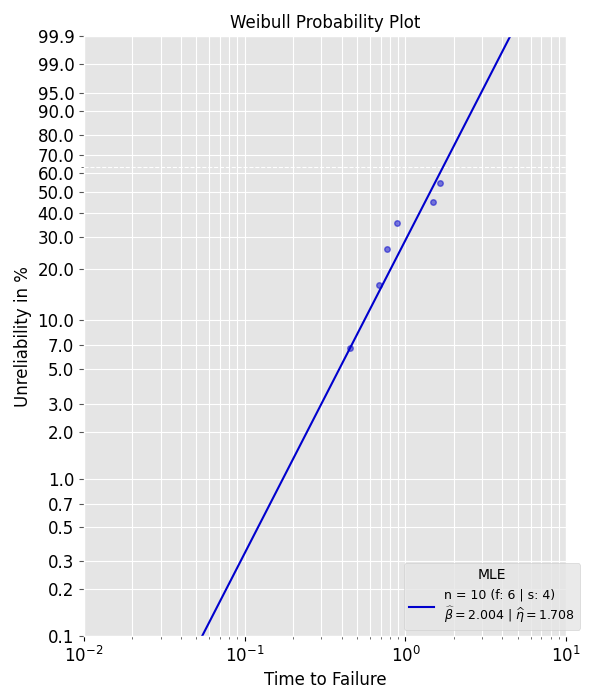


MRR for given data (Image by Author)

**MLE**

In contrast to the MRR, the MLE considers the actual failure and suspension times. Increasing the number of suspensions mainly increases the Weibull scale parameter, the shape parameter estimate does not significantly change. We perform the MLE for the same data (Note: *Analysis* is already imported):

# Data from testing  
# Failures and suspensions are lists containing the valuesfailures = [0.4508831, 0.68564703, 0.76826143, 0.88231395, 1.48287253, 1.62876357]  
suspensions = [1.62876357, 1.62876357, 1.62876357, 1.62876357]# Weibull Analysis  
x = Analysis(*df*=failures, ds=suspensions, *show*=True)  
x.mle()



MLE for given data (Image by Author)

Statistical values are stored as so called attributes of the objects in *Analysis.*Hence, we can print and/or save them. Check out the official documentation of predictr to have an overview of all object attributes.

# We are using f-strings  
# x and y are the names of the class objects we created (see code above)  
# beta and eta are the attributes we want to access and print  
# Just type *object*.*attribute to access them  
# e.g. for the object x type the following: x.beta or x.eta*print(f'MRR: beta={x.beta:2f}, eta={x.eta:2f}\nMLE: beta={y.beta:2f}, eta={y.eta:2f}\n')# Output  
>>> MRR: beta=1.760834, eta=1.759760  
 MLE: beta=2.003876, eta=1.707592

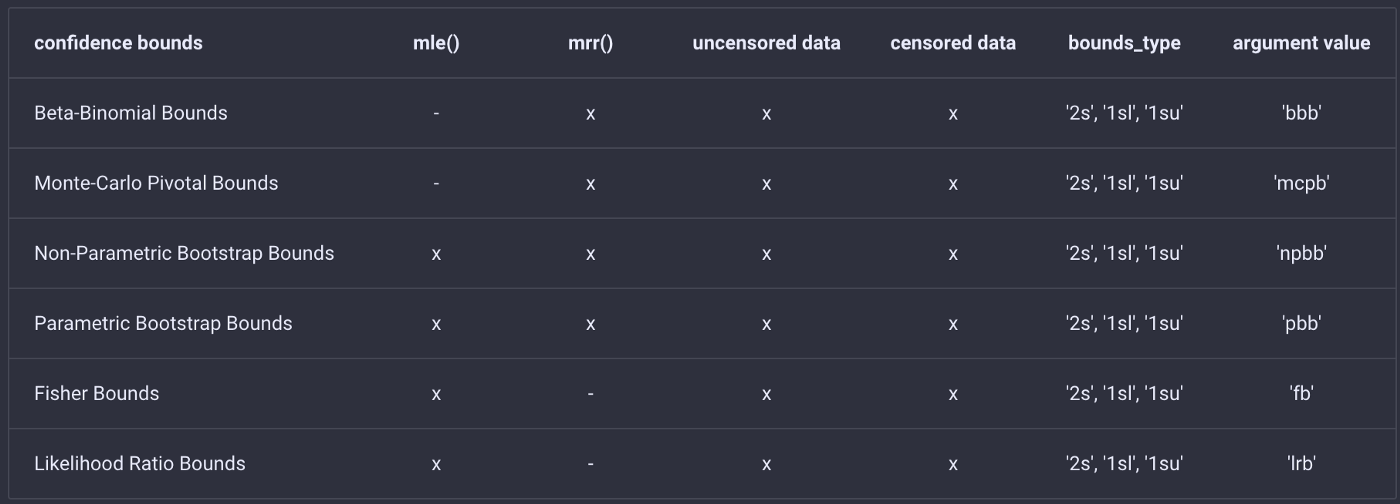
**Confidence Intervals**

Solely using point estimates is risky, especially if you only have a small number of failed units from testing. Assuming that estimated sample statistics, e.g. the Weibull parameters, are close or even equal to the population statistics will likely result in a false sense of security. It is **generally recommended**to use confidence bounds methods in your Weibull Analysis. By using confidence intervals, we can assume with a certain confidence that the actual true population (or ground truth) Weibull line lies within this interval. Hence, we are less likely to overestimate our system’s reliability. A typical confidence interval is 90%, meaning the lower confidence bound **could** be set to 5% and the upper to 95%. The bounds could also be set to 1% and 91%. As you can see, the interval is only defined by the difference between the lower and upper bounds and don‘t have to be symmetrical.

It is important to note that there are two kinds of bounds:

1. Bounds for a fixed unreliability/reliability value (e.g. R(t)=80%) on the the time-to-failure axis, e.g. lower bound: 5000 hours and upper bound: 7000 hours.
2. Bounds for a fixed time-to-failure (e.g. t= 6000 hours) value on the the unreliabilty/reliability axis, e.g. lower bound: R(t)=20% and upper bound: R(t)=38%.

There are plenty confidence bounds methods to chose from. I will soon publish a follow-up medium article about which method to chose in which situation. The following table lists the supported confidence bounds methods by *predictr*.

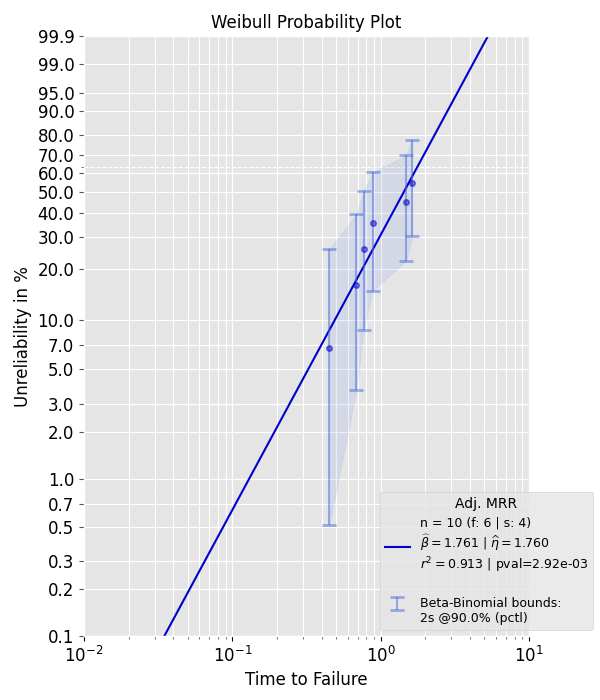


Overview of supported confidence bounds methods in predictr (Image by Author)

You can choose between two-sided (2s) and and one-sided (1sl: one-sided lower; 1su: one-sided upper) confidence bounds. All methods except Beta-Binomial bounds use bounds for a fixed unreliability/reliability value. By changing the argument values, you can customize your Weibull analysis.

Let’s use Beta-Binomial bounds for the same data which we already used:

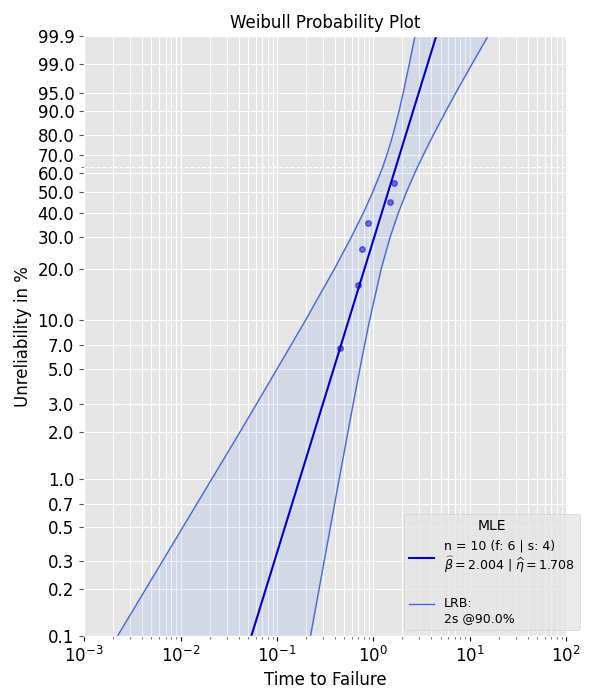
from predictr import Analysis# Data from testing  
failures = [0.4508831, 0.68564703, 0.76826143, 0.88231395, 1.48287253, 1.62876357]  
suspensions = [1.62876357, 1.62876357, 1.62876357, 1.62876357]# Weibull Analysis with two-sided bounds and a plot  
x = Analysis(*df*=failures, *ds*=suspensions, *show*=True, *bounds*='bbb', *bounds\_type*='2s')  
x.mrr()



MRR with Beta-Binomial Bounds (Image by Author)

Now, let’s conduct an MLE and use two-sided Likelihood Ratio bounds for the same data:

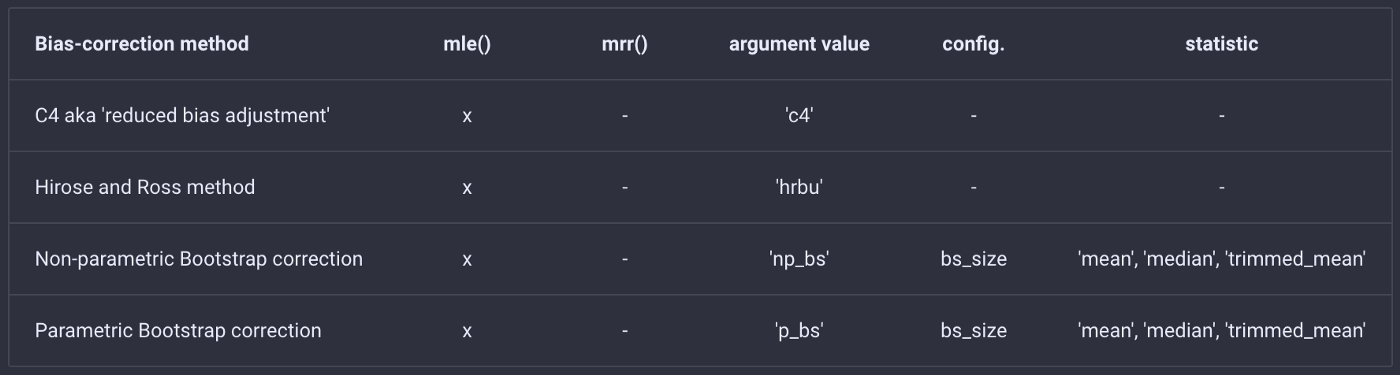
from predictr import Analysis# Data from testing  
failures = [0.4508831, 0.68564703, 0.76826143, 0.88231395, 1.48287253, 1.62876357]  
suspensions = [1.62876357, 1.62876357, 1.62876357, 1.62876357]# Weibull Analysis  
x = Analysis(*df*=failures, *ds*=suspensions, *show*=True, *bounds*='lrb', *bounds\_type*='2s')  
x.mle()



MLE with Beta-Binomial Bounds (Image by Author)

**Bias-Corrections**

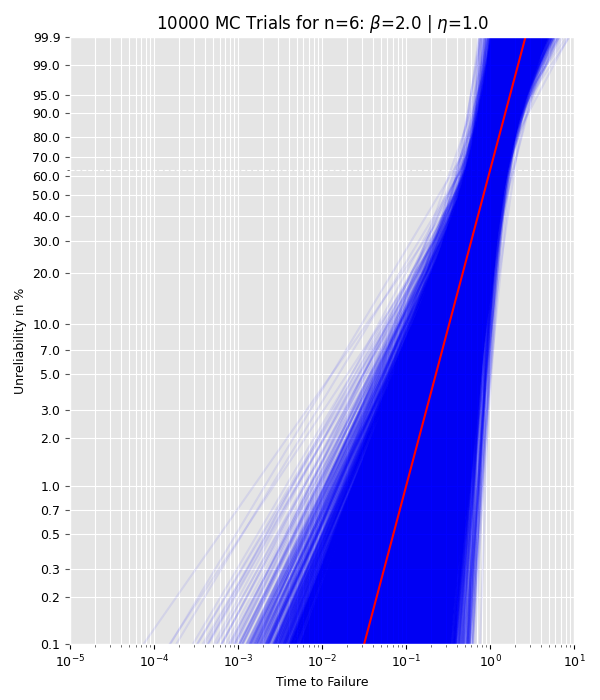
Small sample sizes or number of failures result in biased Weibull parameter estimates. There are no clearly defined hard limits for small or big enough sample sizes. Simulation data shows that sample sizes equal or greater than 20 tend to result in significantly more accurate estimates, no matter which parameter estimation and confidence bounds method one uses. But in practice, reliability engineers often have to deal with much smaller sample sizes. Therefore, the use of bias-correction methods is quite common. *Predictr*supportsfollowing bias-correction methods:



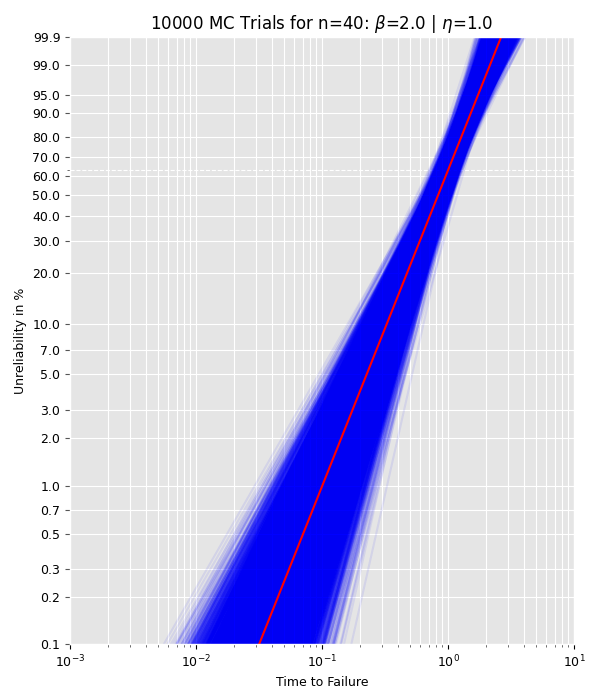
Overview of supported bias-correction methods in predictr (Image by Author)

*Bias-corrections influence the estimation of Weibull parameters as well as the confidence bounds. Accurate estimates of Weibull parameters using bias-correction methods do not automatically result in more accurate confidence bounds! Not all confidence bounds are equally sensitive to bias-corrections. For more information on this topic, you can check out my publications on bias-corrections¹².*

To better understand what the effects of biased estimates are and how bias-corrections work, let’s conduct a Monte-Carlo (MC) study. We will repeatedly draw **random samples**(sample size n=6, uncensored) from a predetermined Weibull distribution (β =2 and η=1 aka our ground truth) and conduct a Weibull Analysis for each of them. For each sample, the resulting Weibull line will be drawn in the Weibull probabilty plot. The number of MC trials is set to 10,000. All randomly drawn samples are represented by blue lines, whereas the ground truth color is set to red.



As can be seen from the plot, the estimated Weibull parameters vary quite a lot for n=6 although the samples were drawn from the same Weibull distribution. This shows how small sample sizes could yield biased estimates. Increasing the sample size to 40 decreases the bias of the estimates (drawn Weibull lines are generally closer to the ground truth). This is expected, since the MLE is asymptotically unbiased.



The plots below show histograms for all 10,000 estimated Weibull parameters. For small sample sizes, the shape parameter tends to be overestimated and is not symmetrically distributed (in contrast to the scale parameters). That is the reason why nearly all bias-correction methods solely focus on the shape parameter and try to decrease it. Most bias-corrections derive a correction factor from the difference between the ground truth and sample mean (or sample median) from MC studys. Keep in mind that these bias-corrections could falsely adjust estimates downwards when the actual estimate is already underestimated. But in general, bias-correction methods should work the way they are intended to.

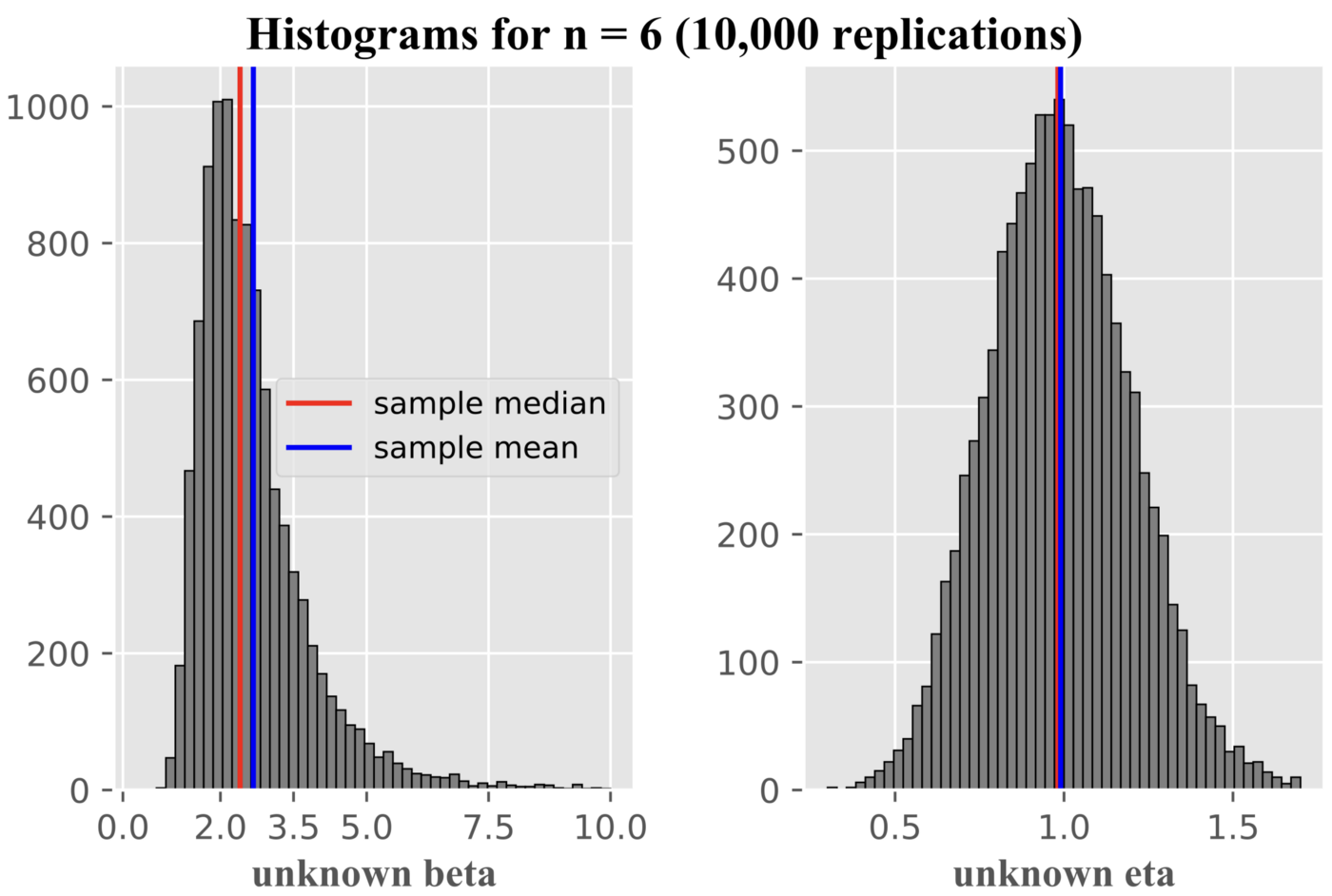


Image by Author

In *predictr*, the *bcm*argument set the bias-correction. We will draw one random uncensored sample from a two-parameter Weibull distribution and will apply a bias-correction to the estimates.

# Needed imports  
from scipy.stats import weibull\_min  
from predictr import Analysis  
import numpy as np# Draw one random sample with a set seed for reproducibility np.random.seed*(seed*=42)  
sample = np.sort(weibull\_min.rvs(2, *loc* = 0, *scale* = 1, *size* = 4))x = Analysis(*df*=sample, *bcm*='c4', *bounds*='fb', *show*=True)  
x.mle()

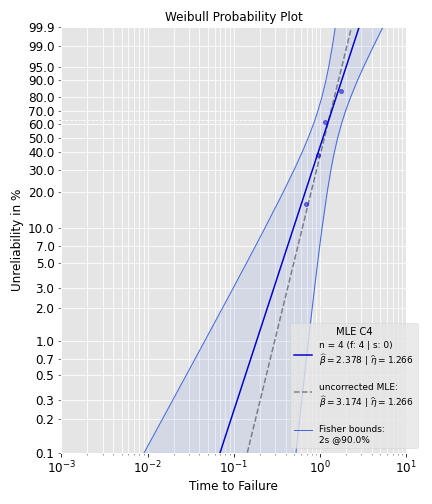
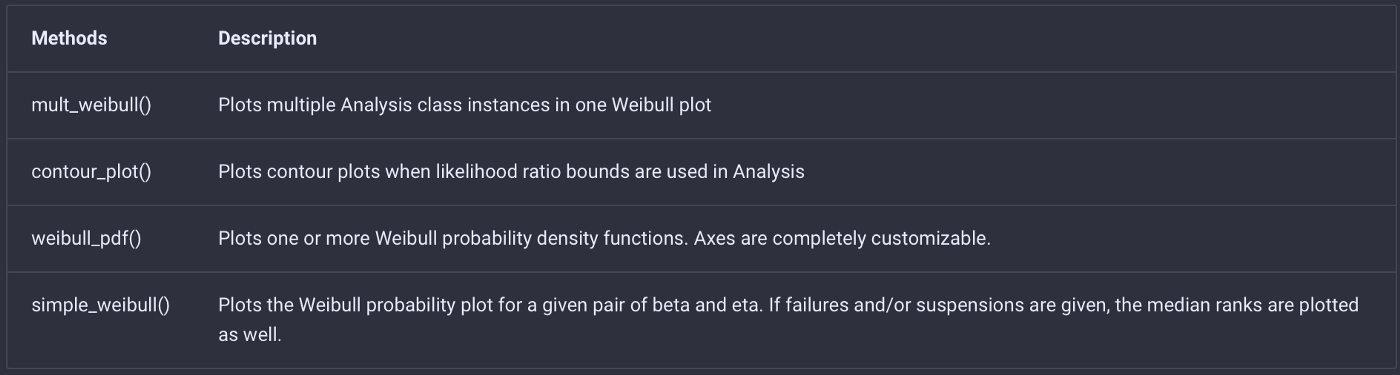


Image by Author

The legend shows the estimates for the uncorrected MLE (dashed line). Using the C4 correction, the corrected shape parameter estimate is 2.4, which is closer to the ground truth value of 2.0. Try out other bias-corrections methods in *predictr* and compare the results!

**Comprehensive Plots**

*PlotAll* is another class in *predictr*that let’s you create and save insightful plots. It uses the object and its attributes created in*Analysis.*Following methods are currently integrated in *PlotAll*:



Methods in PlotAll (Image by Author)

In order to compare two or more designs (prototypes in this example), you can use the *mult\_weibull* method in *PlotAll:*

from predictr import Analysis, PlotAll# Create new objects, e.g. name them prototype\_a and prototype\_b  
failures\_a = [0.30481336314657737, 0.5793918872111126, 0.633217732127894, 0.7576700925659532, 0.8394342818048925, 0.9118100898948334, 1.0110147142055477, 1.0180126386295232, 1.3201853093496474, 1.492172669340363]prototype\_a = Analysis(*df*=failures\_a, *bounds*='lrb', *bounds\_type*='2s')  
prototype\_a.mle()failures\_b = [1.8506941739639076, 2.2685555679846954, 2.380993183650987, 2.642404955035375, 2.777082863078587, 2.89527127055147, 2.9099992138728927, 3.1425481097241, 3.3758727398694406, 3.8274990886889997]prototype\_b = Analysis(*df*=failures\_b, *bounds*='pbb', *bounds\_type*='2s')  
prototype\_b.mle()# Create dictionary with Analysis objects  
# Keys will be used in figure legend. Name them as you please.objects = {fr'proto\_a: *$\w*idehat*\b*eta*$*={prototype\_a.beta:4f} | *$\w*idehat\eta*$*={prototype\_a.eta:4f}': prototype\_a, fr'proto\_b: *$\w*idehat*\b*eta*$*={prototype\_b.beta:4f} | *$\w*idehat\eta*$*={prototype\_b.eta:4f}': prototype\_b}# Use mult\_weibull() method  
PlotAll(objects).mult\_weibull()

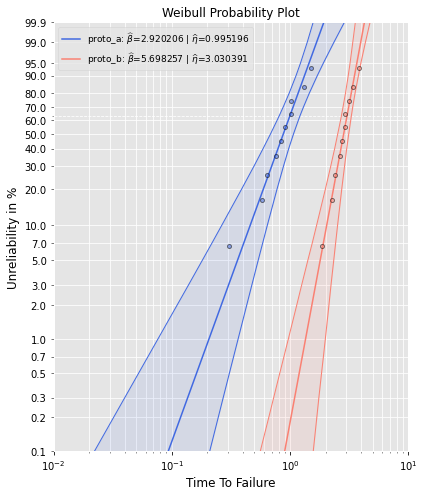


Image by Author

In order to draw density functions, use the *weibull\_pdf* method:

from predictr import Analysis, PlotAll# Use analysis for the parameter estimation  
failures1 = [3, 3, 3, 3, 3, 3, 4, 4, 9]  
failures2 = [3, 3, 5, 6, 6, 4, 9]  
failures3 = [5, 6, 6, 6, 7, 9]a = Analysis(df=failures1, bounds='lrb', bounds\_type='2s', show = False, unit= 'min')  
a.mle()b = Analysis(df=failures1, ds = failures2, bounds='fb', bounds\_type='2s', show = False, unit= 'min')  
b.mle()c = Analysis(df=failures3, bounds='lrb', bcm='hrbu', bounds\_type='2s', show = False, unit= 'min')  
c.mle()# Use weibull\_pdf method in PlotAll to plot the Weibull pdfs  
# beta contains the Weibull shape parameters, which were estimated using Analysis class. Do the same for the Weibull scale parameter eta.  
# Cusomize the path directory in order to use this code  
PlotAll().weibull\_pdf(beta = [a.beta, b.beta, c.beta], eta = [a.eta, b.eta, c.eta], linestyle=['-', '--', ':'], labels = ['A', 'B', 'C'], x\_bounds=[0, 20, 100], plot\_title = 'Comparison of three Prototypes', x\_label='Time to Failure', y\_label='Density Function', save=False, color=['black', 'black', 'black'])

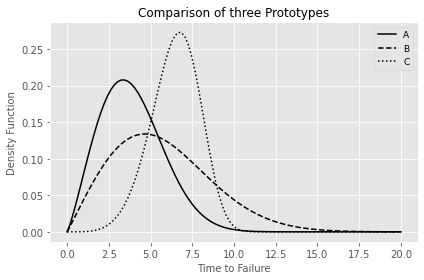


Image by Author

Please check the [official documentation](https://tvtoglu.github.io/predictr/classes/) for more examples and a detailed description of the code.

**Conclusion**

You’re now able to conduct your own Weibull Analyses with knowledge of fundamentalstatistical concepts using [predictr](https://pypi.org/project/predictr/). Try out different combinations of testing data, parameter estimation, confidence bounds, and bias-correction in order to get a feel for mutual interdependencies.

**References**

1. T. Tevetoglu and B. Bertsche, “On the Coverage Probability of Bias-Corrected Confidence Bounds,” *2020 Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM)*, 2020, pp. 1–6, doi: 10.1109/APARM49247.2020.9209464.
2. T. Tevetoglu and B. Bertsche, “*Bias Corrected Weibull Parameter Estimation and Impact on Confidence Bounds”*. Esrel2020-PSAM15, 2020 doi: 10.3850/978–981–14–8593–0\_3925-cd.