The Journal of Open Source Software

SurPyval: Survival Analysis with Python

Derryn Knife[1]

1 Independent researcher

DOI: 10.21105/joss.03484

Software

- Review
- Repository
- Archive

Editor: Dan Foreman-Mackey

Reviewers:

- @CamDavidsonPilon
- @MatthewReid854

Submitted: 13 June 2021
Published: 18 July 2021

Summary

Survival analysis is being increasingly used by scientists, data scientists, engineers, econometricians, and many more professionals to solve their problems. Survival analysis is a unique set of tools that are used to estimate either the time to an event or the chance of an event happening. That is, survival analysis allows you to estimate how long something is likely to last or what risk there is of some event happening in the future. This is vital for fields such as the medical sciences where we need to know how long someone with a particular diagnosis might live or if a treatment or intervention is successful at prolonging life. In engineering it is useful to understand the risk that fielded equipment might fail. For an insurance company it is necessary to help price policies and in economics it is useful for estimating the durations of recessions or the time to the next recession. Survival analysis in these examples encounter interesting kinds of data, for example, engineers conducting life testing may have components that do not fail during the observation period, or that might fail between two inspections. In this case the data is said to be censored. In medical trials you might have subjects enter a trial later than other subjects while insurance claims are only lodged above the excess value on the policy. In these cases the data is said to be truncated. These considerations are unique to survival analysis and are critical to handle correctly to make appropriate predictions or find significant differences.

SurPyval is designed to be pure Python to make installation and maintenance simple. Further, *SurPyval* is a flexible and robust survival analysis package that can take as input an arbitrary combination of observed, censored, and truncated data over a wide number of distributions and their variations. For this reason *SurPyval* is likely to be of interest to a wide field of analysts in broad industries including finance, insurance, engineering, medical science, agricultural science, economics, and many others.

Statement of need

SurPyval fills a gap in the Python ecosystem of survival analysis. Other survival analysis packages, e.g. *lifelines* (Davidson-Pilon, 2019) and *reliability* (Reid, 2021) offer excellent methods for most applications. SurPyval has the advantage of being able to handle arbitrary combinations of observed values, censoring and truncation. Further, *scipy* (Jones et al., 2001) has yet to implement some basic features of survival analysis; concretely, it does not handle censored data. Therefore, there is a gap in the Python ecosystem for a package that is flexible to accomodate any arbitrary combination of observed failures (or deaths); left, right, or interval censored; and left or right truncated data with a single format. Another powerful feature of *SurPyval* is that it lets users select an estimation method for their circumstances. Maximum Likelihood Estimation (MLE) is used in most other applications, but *SurPyval* also implements Maximum Product of Spacing, Method of Moments, Probability Plotting, Mean Square Error, and Expectation-Maximisation. This variety of estimation methods gives users of SurPyval greater flexibility in their choice of which parameter estimation technique to use

when fitting distributions to their data. Commercial packages are well developed but can be expensive. R is excellent for survival analysis but many analysts now use Python as is explained in the *lifelines* paper. Therefore there is a need to have another flexible and open source python package to do survival analysis.

Features

SurPyval is grouped into two modules, these are parametric and non-parametric modules. For the parametric module *SurPyval* offers several methods to estimate parameters; these are Maximum Likelihood Estimation (MLE), Mean Square Error (MSE), Method of Probability Plotting (MPP), Maximum Product of Spacing (MPS), Method of Moments (MOM), and Expectation-Maximisation (EM). The EM is only used for mixture models.

For the Non-Parametric estimation *SurPyval* can estimate the survival distribution using either the Kaplan-Meier (Kaplan & Meier, 1958), Nelson-Aalen (Nelson, 1969); (Aalen, 1978), Fleming-Harrington (Fleming & Harrington, 1984), or the Turnbull (Turnbull, 1976) estimators. Support for data types and estimation methods can be seen in the table below.

Method	Para/Non-Para	Observed	Censored	Truncated
MLE	Parametric	Yes	Yes	Yes
MPP	Parametric	Yes	Yes	Limited
MSE	Parametric	Yes	Yes	Limited
MOM	Parametric	Yes	No	No
MPS	Parametric	Yes	Yes	No
Kaplan-Meier	Non-Parametric	Yes	Right only	Left only
Nelson-Aalen	Non-Parametric	Yes	Right only	Left only
Fleming-Harrington	Non-Parametric	Yes	Right only	Left only
Turnbull	Non-Parametric	Yes	Yes	Yes

SurPyval achieves this flexibility with a simple API. *SurPyval* uses a data input API, the 'xcnt' format, that can be used to define any arbitrarty combination of censored or truncated data. 'x' is the value at the observation, 'c' is the censoring flag, 'n' is the counts, and 't' is the truncation values. *SurPyval* uses the convention for the censor flag where -1 is left censored, 0 is an observed value, 1 is right censored, and 2 is intervally censored. Utilities have also been created to help users transform their data into the 'xcnt' format if they have it in another format. For example, a lot of survival data is provided in an observered and suspended format, this is where you have a list of the failure times and a list of the suspended times. E.g. Failures of [1, 2, 3, 4, 5] and suspended times of [1, 2, 3]. *SurPyval* refers to this format as the 'fs' format.

For Non-Parametric analysis *SurPyval* takes as input the 'xcnt' format, but the KM, NA, and FH estimators are calculated using the 'xrd' format. This format takes 'x' as the value of the observation, 'r' as the count of items at risk at each 'x,' and 'd' is the number of deaths/failures at each time 'x.' Once data is in this format it is possible to compute the KM, NA, or FH estimators. For the Turnbull estimator the values of 'x,' 'r,' and 'd' are computed using Turnbull's algorithm; 'x,' 'r,' and 'd' estimates account for truncation and censoring. These values can then be used with the KM, NA, or FH estimator to get an estimate of the distribution.

Maximum Likelihood Estimation can be used for any arbitrary combination of censoring and truncation. The Probability Plotting and Mean Square Error methods can be used with arbitrarily censored data and limited truncation. Specifically, these methods are limited if the maximum and minimum of the observed data are truncated observations. This is because the

Turnbull Non-Parametric Maximum Likelihood Estimator (NPMLE) cannot assume the shape of the distribution and therefore cannot be used to estimate by how much the highest and lowest values are truncated. The Maximum Product of Spacing estimation can be used with censored observations. The Method of Moment estimation can only be used with observed data, i.e. no censoring or truncation.

SurPyval uses *scipy* for numerical optimisation but also aims to imitate as close as possible the API for parameter estimation, specifically, the use of the `fit()` method. The main difference between *scipy* and *SurPyval* is that *SurPyval* returns an object. The intent of this is to capture the distribution in an object for subsequent use. This could be used in Monte Carlo simulations using the `random()` method or it could be used in applications like *reliability* for interval optimisations.

Unlike other survival analysis packages *SurPyval* allows users fix any parameter with any distribution. This is similar to *scipy* which allows the location, shape, and scale parameters to be fixed. In *SurPyval* this is done using the `fixed` keyword with a dictionary of the name and value of the fixed parameter and value.

Optimisations

SurPyval, inspired by *lifelines*, uses *autograd* (Maclaurin et al., 2015) autodifferentiation to calculate the jacobians and hessians needed for optimisations in parametric analysis. Additionally, SurPyval uses lessons from deep learning to improve the stability of estimation. Concretely, SurPyval uses a modified ELU function (Clevert et al., 2015) to transform bounded parameters to be unbounded. For example, the alpha parameter for a Weibull distribution is supported on the half-real line, (0, Inf). Using the modified ELU function the input to the optimizer is transformed to be supported over the full real line (-Inf, Inf) but then will transform this value to a positive number when calculating the objective function. This reduces the risk of optimisations failing because the numeric gradient might 'overshoot' and hit a bound therefore produce undefined results which in turn causes the autodifferentiation to fail. The ELU is also useful for autodifferentiation because it is continuously differentiable which eliminates discrete jumps in the gradient. Another advantage of this improvement is that it can be used to robustly estimate offsets, i.e. the 'gamma' parameter, for half real-line supported distribtuions. For example, *SurPyval* can be used to estimate the parameters of the four parameter Exponentiated-Weibull distribution, which is a feature that is absent from other currently available survival packages.

Another optimisation used by *SurPyval* is the use of good initial approximations for parameter initialisation. Probability plotting methods do not require initial estimates of the parameters; which is in contrast to estimates using optimizers. Further, optimisation results are very sensitive to the initial estimates, if the initial estimate is too far from the actual result it can yield incredulous results. As such *SurPyval* uses either probability plotting estimates or estimates using transformed data with another distribution to do the initial esimate. Combining the use of autogradients, bound transformations, and close initial approximations, *SurPyval* is a stable software for estimating parameters for statistical distributions.

Examples

Some examples of the API and how flexible it can be. Firstly, a simple estimate from random data:

```
from surpyval import Weibull
import numpy as np
```

```
np.random.seed(10)
# Weibull parameters
alpha = 10
beta  = 2

# Random samples
N = 30

x = Weibull.random(N, alpha, beta)

model = Weibull.fit(x)
print(model)
```

```
Parametric Surpyval model with Weibull distribution fitted by MLE yielding
parameters [9.71565772 2.33944554]
```

Using offsets with SurPyval is a simple change by setting the offset parameter to True. Using data from Weibull's paper (Weibull & others, 1951) which introduced the wide applicability of the distribution to survival analysis, we can get a three parameter Weibull distribution:

```
from surpyval import Weibull
from surpyval.datasets import BoforsSteel

data = BoforsSteel.df

x = data['x']
n = data['n']

model = Weibull.fit(x=x, n=n, offset=True)
model.plot()
print(model)
```

```
Offset Parametric Surpyval model with Weibull distribution fitted by MLE yielding
parameters [7.14192522 2.6204524 ] with offset of 39.76562962867473
```

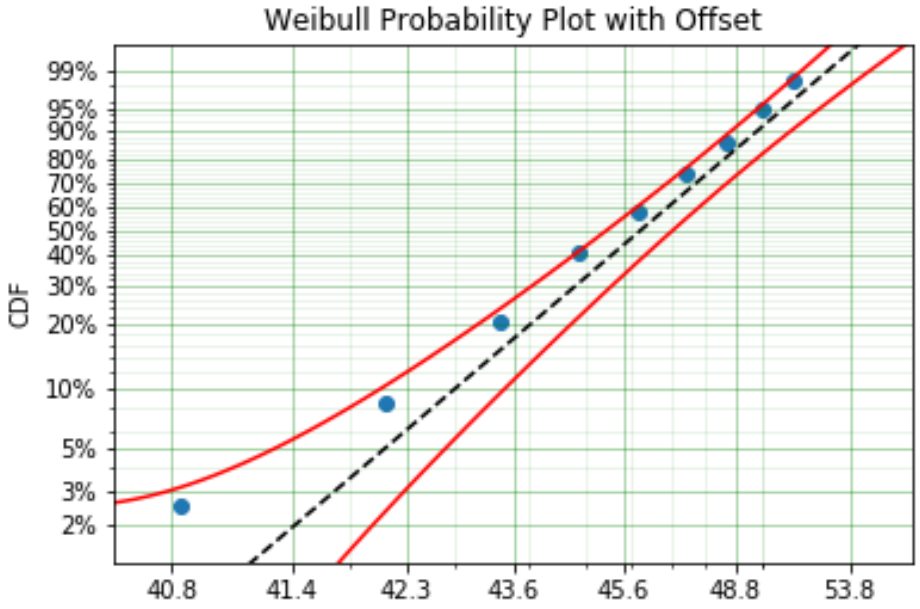

Figure 1: Weibull Data and Distribution

There are more examples of the flexible API in the main documentation.

References

Aalen, O. (1978). Nonparametric inference for a family of counting processes. *The Annals of Statistics*, 701–726. https://doi.org/10.1214/aos/1176344247

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv Preprint arXiv:1511.07289*.

Davidson-Pilon, C. (2019). Lifelines: Survival analysis in python. *Journal of Open Source Software*, *4*(40), 1317. https://doi.org/10.21105/joss.01317

Fleming, T. R., & Harrington, D. P. (1984). Nonparametric estimation of the survival distribution in censored data. *Communications in Statistics-Theory and Methods*, *13*(20), 2469–2486.

Jones, E., Oliphant, T., Peterson, P., & others. (2001). *SciPy: Open source scientific tools for python*.

Kaplan, E. L., & Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, *53*(282), 457–481.

Maclaurin, D., Duvenaud, D., & Adams, R. P. (2015). Autograd: Effortless gradients in numpy. *ICML 2015 AutoML Workshop*, *238*, 5.

Nelson, W. (1969). Hazard plotting for incomplete failure data. *Journal of Quality Technology*, *1*(1), 27–52. https://doi.org/10.1080/00224065.1969.11980344

Reid, M. (2021). *MatthewReid854/reliability: v0.5.7* (Version v0.5.7) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.5030847

Turnbull, B. W. (1976). The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society: Series B (Methodological)*, *38*(3), 290–295. https://doi.org/10.1111/j.2517-6161.1976.tb01597.x

Weibull, W., & others. (1951). A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, *18*(3), 293–297. https://doi.org/10.1115/1.4010337