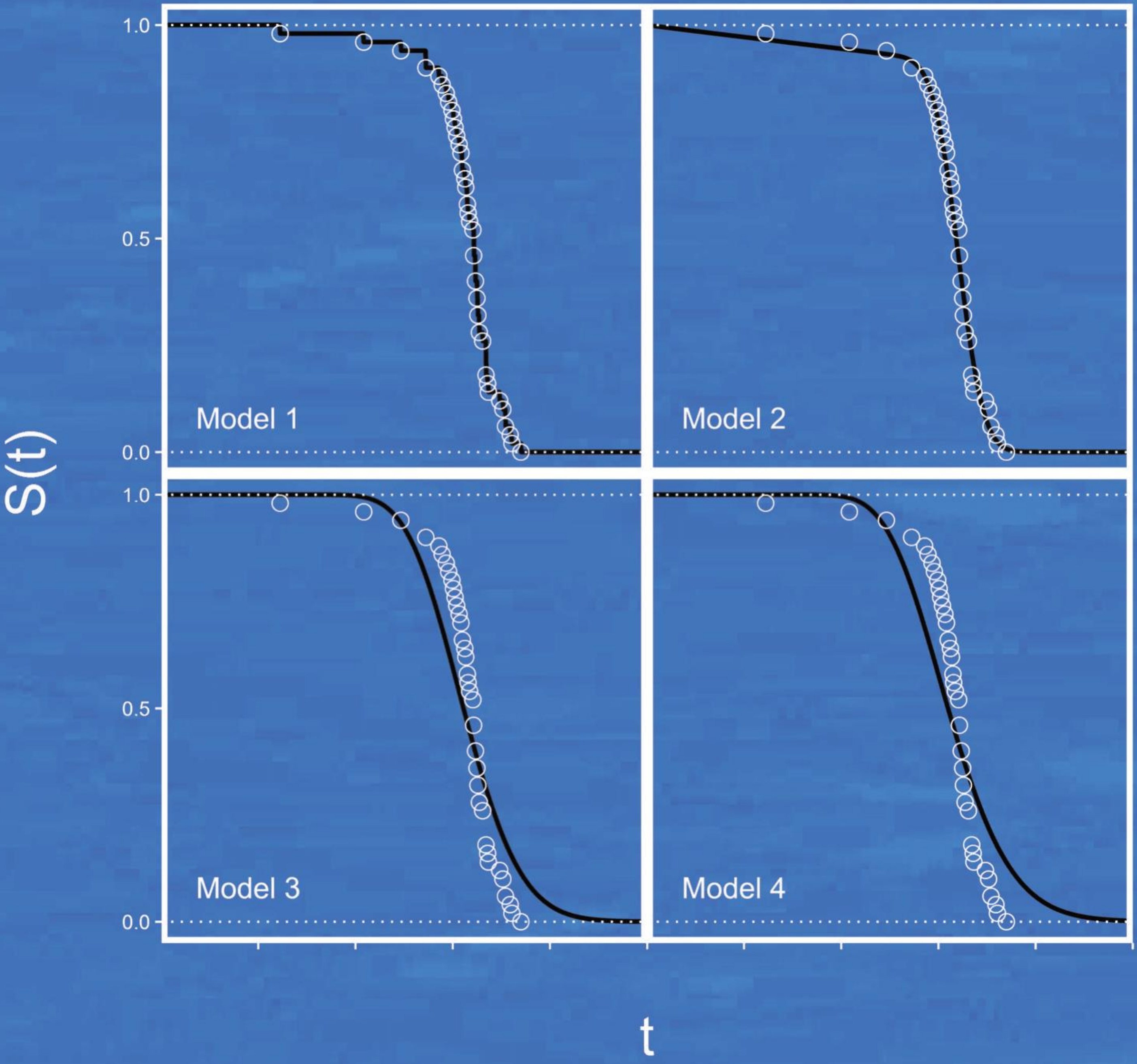
Columbia Basin Research R Package Series

failCOMPARE USER'S MANUAL

R PACKAGE FOR FAILURE TIME MODEL FITTING AND EVAUATION



Columbia Basin Research
School of Aquatic & Fishery Sciences
University of Washington
1325 4th Avenue, Suite 1515
Seattle, WA 98101
www.cbr.washington.edu

UNIVERSITY *of* WASHINGTON

failCompare R package

Failure Time Model Fitting and Comparisons

Steven L. Whitlock, Rebecca A. Buchanan, and Rich Townsend

Columbia Basin Research

School of Aquatic and Fishery Sciences

University of Washington 1325 4th Avenue, Suite 1515

Seattle, WA 98101-2509

Version: 1.0

Release date: November 30, 2021

Table of Contents

TABLE OF CONTENTS ..... **I**

1 INTRODUCTION **1**

2 FAILURE TIME MODELS **2**

2.1 Theoretical Basis 2
2.2 Failure Time Modeling in Practice 3
2.2.1 Calculating the Sample Survival Function 3
2.2.2 Parametric Models of Failure Time 4
2.3 Comparing Models 6
2.4 Lack-of-Fit Testing 7
2.5 Censored Data 8

3 USING *FAILCOMPARE* **9**

3.1 Overview of Functions 9
3.2 Relation to Other Software 10
3.3 Example Data Sets 10
3.4 Handling Censoring 10
3.4.1 Downloading the failCompare package 11
3.5 Getting Started 11
3.5.1 Installation within R 11
3.5.2 Loading the failCompare package 12
3.5.3 Example data 12
3.6 Example 1: Fitting, Visualizing, and Ranking Alternative Failure Time Models 12
3.6.1 Preparation 12
3.6.2 Fitting individual models 13
3.6.3 Model comparisons 17
3.6.4 Simultaneous model fitting 18
3.7 Example 2: Testing Methods for Failure Time Models 21
3.7.1 Log-rank test for comparing groups 22
3.7.2 Ranking models separately for two groups 23
3.7.3 Kolmogorov-Smirnov test (simulation-based) 24
3.7.4 Selecting the Kaplan-Meier model from a model list object 26
3.8 Example 3: Working with Censored Data 28
3.8.1 Type I censoring 28
3.8.2 Type II censoring 31

4 APPENDIX **34**

4.1 Parametric Model Descriptions 34
4.1.1 Models Based on Probability Distributions 34
4.1.2 Vitality Models 43

5 REFERENCES **48**

1 Introduction

Many scientific investigations are concerned with measuring or modeling the length of time before an event occurs. Such data may be referred to as “time to event data,” “failure time data,” or “survival data,” depending on the field. Efforts to analyze these types of data have spawned an array of statistical analysis methods owing to the variety of data-generating processes and special study design limitations across fields. For example, the practice of reliability modeling concerns measuring the time until system or equipment failure (Blischke and Murthy 2011) and may be used to determine when maintenance or replacement is advisable. The related field of survival analysis concerns measurement of time until death while also accounting for study subjects that survive past the end of the study (Hosmer et al. 2008). In fish and wildlife studies, it is sometimes necessary to model time to battery failure for biotelemetry transmitters (“tags”). Analysis tools developed for one application are often transferable to other fields. For example, the Weibull model, originally developed for reliability assessment (Leemis 1995), was later found useful for estimating the time interval between wildfires (McCarthy et al. 2001). However, the overwhelming number of alternative methods and models can make it difficult to decide on the most appropriate analysis tool, particularly for those working in fields without established analysis conventions.

A typical analysis need in these fields is to model the time to event (e.g., failure) or model the probable status of an individual at a particular time (e.g., probability of survival to a specific time). A diversity of statistical failure time models have been developed for this purpose, including but not limited to the Weibull, Gompertz, log-normal, log-logistic, gamma, and vitality models (Li and Anderson 2009, 2013). Selecting among the possible models can be challenging because it requires both fitting multiple models and ranking their relative fit. Models that arise from unrelated statistical families can cause additional difficulties in ranking. Skalski and Whitlock (2020) developed a novel performance metric for ranking alternative models that is valid even when comparing models from different statistical families and also presented a general lack-of-fit test applicable to different failure time models. Although these methods were presented in Skalski and Whitlock (2020), until now implementing them has required the user to draw on multiple software packages and to write their own code.

The *failCompare* R package was developed to easily implement the model fitting, ranking, and lack-of-fit tests for nine different failure time models as presented in Skalski and Whitlock (2020). We selected the open-source programming language and coding environment R (R Core Team 2020) as the platform for this tool because it is the most common programing environment used in fisheries and the broader ecological research community (Lai et al. 2019; Figure 1). It is also multiplatform and can be easily run using the Windows, Mac, or Linux operating systems. We have designed the package to require a minimal level of programming skill for users to operate. The output from the *failCompare* package may be used independently or as input to the *cbrATLAS* R package (in development), which provides estimation of survival from active tag studies in fish and wildlife investigations. The *failCompare* package focuses on investigations that measure failure time by monitoring study subjects over a defined period and recording the duration until failure and is not intended for analysis of life tables.

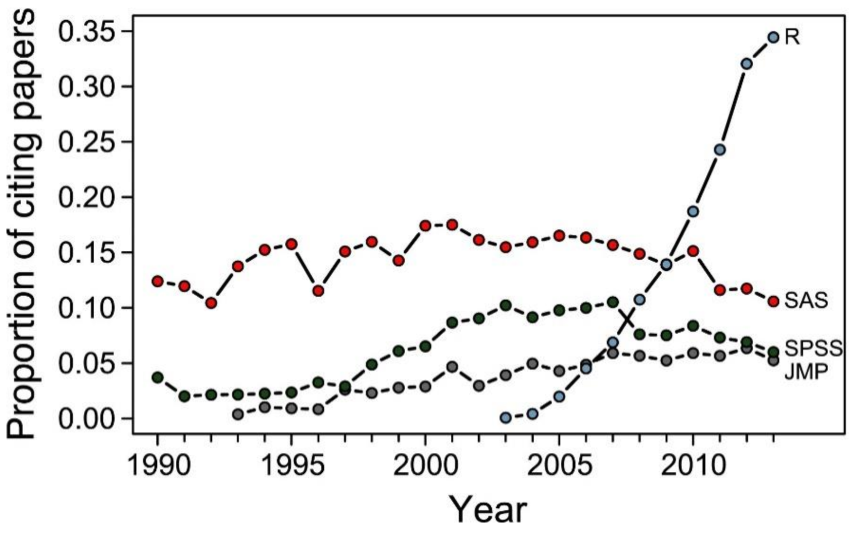


Figure 1. The four most commonly cited statistical analysis programs in Ecological Society of America Journals from 1990–2013 (Touchon and McCoy 2016).

2 Failure Time Models

Prior to explaining the workings of the failCompare package we lay a theoretical foundation for failure-time models in general. Although not exhaustive, this description of principles and techniques provides an appropriate background for understanding the tools in the package. This section also establishes consistent terminology and notation. With some minor exceptions, we use the language of "failure" instead of "survival" because it applies to a broader set of data types (e.g., death is frequently defined as a type of failure).

2.1 Theoretical Basis

We consider time to failure to be a nonnegative random variable T with an underlying probability distribution. This distribution can be transformed into a monotonically decreasing function from 1 to 0 defined over time (t) as $P(T < t)$. In particular if the probability distribution of failure time has a continuous distribution $f(t)$, then the probability of failing prior to time t is calculated as the value of the cumulative distribution function, $F(t)$:

$$F(t) = \int_0^t f(t)\,dt. \tag{1}$$

The probability of not having failed by time t is then:

$$S(t) = 1 - F(t), \tag{2}$$

known as the survival function. Another useful means of characterizing the failure process is the hazard function, defined as:

$$h(t) = \frac{f(t)}{S(t)}. \tag{3}$$

The hazard function describes how prone a subject is to fail at time t, conditional on not having failed up until that point. The shape of this function can be especially helpful for interpreting how the relative risk of failure changes over time. The probability distribution, survival, and hazard functions are mathematically equivalent, so knowledge of any one can be used to derive the others.

2.2 Failure Time Modeling in Practice

2.2.1 Calculating the Sample Survival Function

The initial step in analyzing failure time data is to transform observed data into a discrete sample survival function $\hat{S}(t)$. If failure of all subjects occurs within the observation period and the failure time is known precisely, then $\hat{S}(t)$ is defined as a discontinuous function that steps downward at each unique failure time increment:

$$\hat{S}(t) = \frac{\text{Count of observations} > \text{t}}{n}, \tag{4}$$

where n is the total number of observations and $\hat{S}(t)$ has a constant value between observed failure times. When a portion of the study subjects are removed prematurely, or the investigation concludes prior to the failure of all subjects, then this estimator is unsuitable. This is because although some of the study subjects do not have a known failure time, these incomplete records still provide information about the minimum time the subjects persisted without failure. This type of data complication is known as censoring, specifically “right censoring.” Another possibility is “interval censoring,” which occurs when some study subjects are missing observed failure times but were nevertheless known to have failed between monitoring points. For more information on censoring, see the “censored data” section below.

When either of these forms of censoring are present the product-limit estimate of sample survival probability should be used:

$$\hat{S}(t) = \prod_{t(i)<t} \left(\frac{n-i}{n-i+1}\right) \tag{5}$$

where $n =$ sample size,
$i =$ number of failures before time t.

This is known as the Kaplan-Meier survival estimate (“K-M estimate” hereafter) and is distinctive in that it is based on a maximum likelihood approach but does not actually assume a particular probability distribution (i.e., nonparametric; Kaplan and Meier 1958). The estimator can accommodate censored observations because estimates are based on a chain of conditional probabilities rather than the fraction of the total subjects yet to have failed. These Kaplan-Meier estimates serve a descriptive purpose and may be used to compare multiple groups in a study. They also form a basis for examining the fit of parametric models (Kalbfleisch and Prentice 2011).

2.2.2 Parametric Models of Failure Time

There are many reasons why one would want to characterize a failure-time process using a parametric distribution. If the model fits the data well, then the parameter estimates can provide a succinct description of how likely failure is to occur or how the failure rate may change over time and/or across multiple groups. Arguably, a smooth function better resembles the underlying failure-time process for a population, as opposed to the K-M survival function where the position of downward steps is necessarily defined by particular observations.

In this section, we survey the failure time models discussed in Skalski and Whitlock (2020). All of the following models assume that $t \geq 0$. The simplest choice of failure time model is based on the exponential distribution with parameter λ and density function:

$$f(t) = \frac{1}{\lambda} e^{-\left(\frac{t}{\lambda}\right)}, \qquad \lambda > 0 \tag{6}$$

and therefore the survival function is

$$S(t) = e^{-\frac{t}{\lambda}}. \tag{7}$$

Interestingly, terms in the hazard rate cancel resulting in a constant hazard function defined by $1/\lambda$ (Figure 2). Two noticeable features of this model are that failures occur immediately and the instantaneous risk of failure at any moment is unaffected by the passage of time ("memoryless"). The exponential model is therefore inappropriate for data sets in which failures are accumulated at a variable rate or with a noticeable delay after $t = 0$.

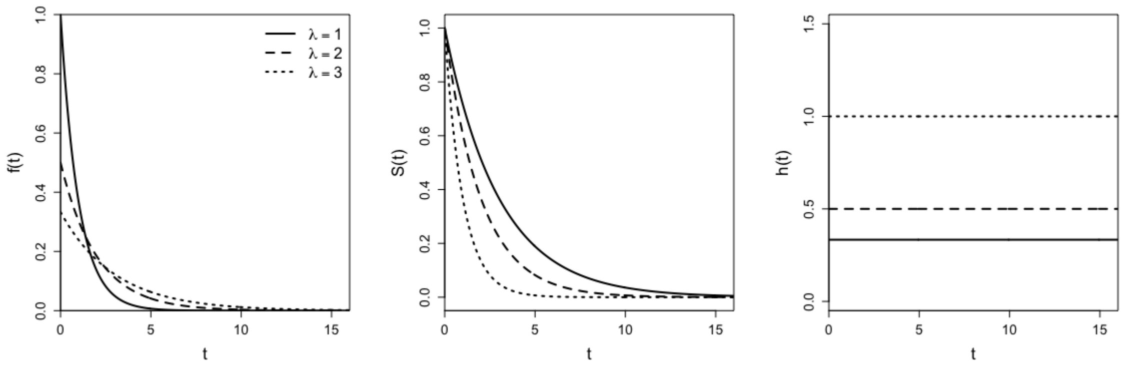


Figure 2. Exponential failure time model density (left), survival (middle), and hazard (right) functions with three different values of parameter λ.

A more flexible model is the Weibull model, of which the exponential model is a special case. The survival function of the 2-parameter Weibull model with shape parameter λ and scale parameter β is:

$$S(t) = e^{-\left(\frac{t}{\lambda}\right)^{\beta}}, \qquad \lambda, \beta > 0. \tag{8}$$

The probability density of the Weibull model can reach an apex, meaning that rate of failures will accelerate or decelerate over time (Figure 3). The exponential model arises in cases where $\beta = 1$.

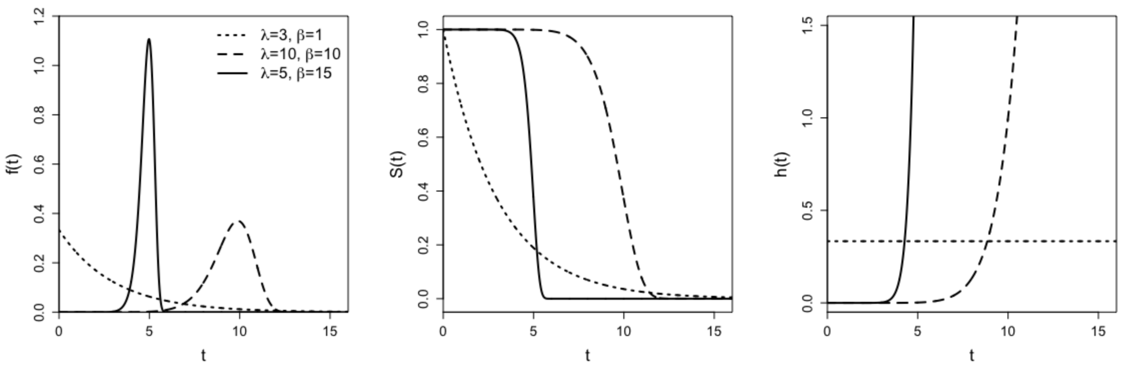


Figure 3. Weibull failure time model density (left), survival (middle), and hazard (right) functions with three different values for shape (β) and scale parameters (λ).

There are several other models based on familiar probability distributions, which are intuitive extensions of simpler models or special cases of more general distributions. We briefly introduce these alternative models here and provide more complete definitions in the appendix (Table 1). The Gompertz model and 3-parameter Weibull model are extensions of the two distributions discussed above. The Gompertz model extends the exponential model by defining the hazard rate as a log-linear function of parameters γ and b:

$$h(t) = e^{\gamma + bt}. \tag{9}$$

The 2-parameter Weibull model can be further extended to a 3-parameter version:

$$S(t) = e^{-\left(\frac{t-\gamma}{\lambda}\right)^{\beta}}, \qquad \gamma > 0 \tag{10}$$

with a threshold (or "guarantee") parameter γ defining an initial "failure free" portion of the curve.

Also included in the *failCompare* package are the log-normal, log-logistic, and 2-parameter and 3-parameter ("generalized") gamma models. Log-normal and log-logistic models are well suited to failure processes that have an initially increasing and then decreasing hazard function, with the log-logistic model having a convenient closed-form definition for its survival function. The hazard function of the 2-parameter gamma model approaches an asymptote as $t \to \infty$. The 3-parameter gamma distribution is the most flexible model described so far and incorporates the exponential, Weibull, and 2-parameter gamma as special cases (Kalbfleisch and Prentice 2011). All the above-mentioned models except the Gompertz and 3-parameter Weibull represent special cases of the generalized F distribution (not implemented in *failCompare*). See Kalbfleisch and Prentice (2011) for a full description.

Among the nine models compared in Skalski and Whitlock (2020) and implemented in *failCompare*, the vitality models are distinct because they do not represent an adoption of a familiar probability distribution for descriptive purposes (Anderson 1992). Rather, the vitality survival functions were derived by explicitly considering the two processes that contribute to the death of organisms: (1) stochastic decline in vitality over a lifetime (intrinsic) and (2) chance external events that cause early deaths (extrinsic). Under these models, death occurs at the time when an individual's vitality, engaging in a random walk downward, crosses the zero line. The two versions of the vitality model included in the *failCompare* package are the Vitality 2009 model (Li and Anderson 2009) and the Vitality 2013 model (Li and Anderson 2013). See the appendix subsection on vitality models for more information.

Table 1. The *failCompare* package's default failure time models and their basic characteristics. Detailed descriptions of these models are available in the appendix.

Model	**Family**	**# Parameters**
2-parameter Weibull	Generalized F	2
3-parameter Weibull	Other	3
Gompertz	Other	2
Log-normal	Generalized F	2
Log-logistic	Generalized F	2
Gamma	Generalized F	2
Generalized Gamma	Generalized F	3
Vitality 2009	Vitality	4
Vitality 2009	Vitality	4

2.3 Comparing Models

Alternative models represent various tradeoffs in terms of flexibility, ease of fitting, and interpretation. Models with a greater number of parameters are more flexible but may overfit the data and are prone to estimation problems with small data sets (e.g., failed convergence and inability to compute the standard errors of parameters).

When faced with this type of model comparison problem, a commonly used criterion for model selection is the Akaike Information Criterion (AIC; Burnham and Anderson 2007). However, some types of models cannot be compared with this criterion because of differences in how the likelihoods are formulated (Burnham and Anderson 2007, p. 307). Moreover there are differing opinions on how censored observations should be counted when calculating the second order AIC criterion, which includes sample size n (AIC_c; Liang and Zou 2008). Considering these potential complications and initial testing that confirmed the incompatibility of likelihood formulations between vitality and other models, AIC did not appear to be a suitable criterion comparing the variety of models supported by the package.

Instead, we created a more intuitive performance measure based on the squared difference between empirical K-M estimates of the survival function and parametric model predictions, adjusted by a penalty for complexity that incorporates the sample size and number of parameters in a manner similar to AIC$_c$. This goodness-of-fit (GOF) statistic is the average squared deviation between the Kaplan-Meier function and the fitted model values of $S(t)$ across the n observed failure times (Figure 4):

$$GOF = \frac{\sum_{i=1}^{n}\left(\hat{S}(t_i) - S(t_i)\right)^2}{(n - p - 1)} \tag{13}$$

where $\hat{S}(t_i)$ = survival value from parametric model at time *t* for the *i*th failure ($i = 1, \dots, n$),
$S(t_i)$ = K-M survival function estimate at time *t* for the *i*th failure ($i = 1, \dots, n$),
n = sample size (including censored observations)
p = number of fitted model parameters.

The model with the smallest GOF value within a set of candidates is considered most suitable.

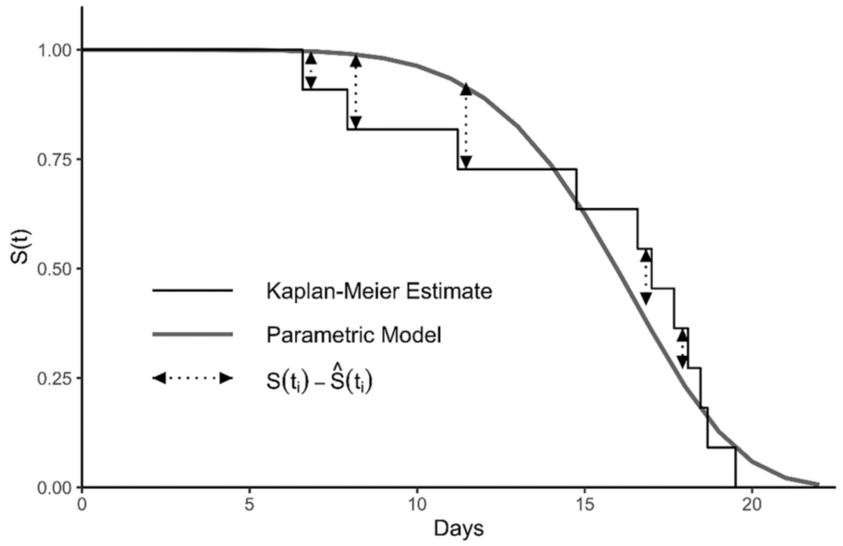


Figure 4. Fitted parametric survival function and observed deviations in survival values at the time of a failure event. The deviation in survival values is calculated at each time step in the K-M curve. Reproduced from Skalski and Whitlock (2020).

2.4 Lack-of-Fit Testing

The GOF metric provides a measure of relative goodness-of-fit to compare alternative models but does not indicate the suitability of the model in general. A clear choice for testing the suitability of the estimated survival function is to apply the one-sample Kolmogorov Smirnov test (Sokal and

Rohlf 1995). This test is a common nonparametric method for comparing empirical versus theoretical cumulative distribution functions. For the failure time model evaluation, the test statistic is the absolute value of the largest discrepancy between $\hat{S}(t_i)$ and $S(t_i)$ anywhere along the fitted curve, i.e.,

$$D = \underset{i=1,\ldots,n}{\text{MAX}} \left|\hat{S}(t_i) - S(t_i)\right| . \tag{12}$$

However, the critical region of the test is not valid for cases in which the model parameters are estimated from data, as in the application here, rather than specified *a priori*. Instead we used the simulation-based testing procedure described in Lilliefors (1967). This process involves first computing the observed test statistic (D_{obs}) using equation (12) and then repeatedly simulating datasets of length n from the estimated distribution and calculating and storing simulated values (D_{sim}) to approximate the sampling distribution of the test statistic. After accumulating sufficiently large sample of D values (e.g., 50,000), the P-value is computed as the proportion of simulated samples that exceed D_{obs}.

2.5 Censored Data

In its current form, the package is equipped to handle two common forms of right-censored data. Interval-censored data and left-censored data are not supported. The two types of censoring accommodated by the software are: Type I, single right censoring, where all subjects beyond a specific time-point are considered removed, and Type II, progressive right censoring where some study subjects are censored prior to termination of the study (Lee and Wang 2003). See example 3 for a demonstration on how to specify censored data types.

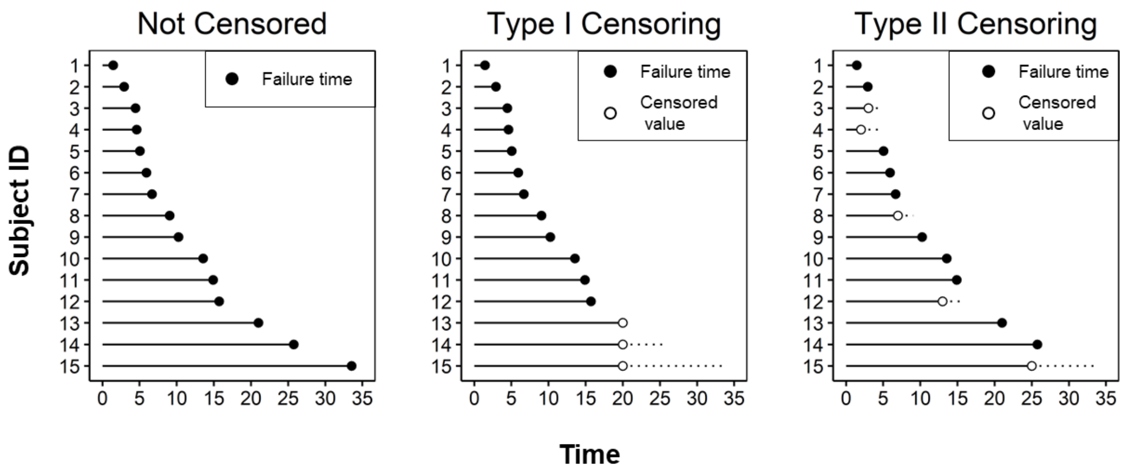


Figure 5. Illustration of a small failure time data without censored observations (left), and alternate versions with Type I (middle) and Type II censoring (right). Horizontal line segments represent the hypothetical lifespans of 15 study subjects ordered by their longevity; dotted sections represent periods when an individual’s status was unobserved. Closed points at the end of lines represent observed failure times, and open points represent the time where the observation was censored.

Fitting a parametric model to data with right censoring requires modification of the likelihood function, which is maximized in obtaining parameter estimates. The likelihood value (L) for a model with observed failure times for all n study subjects relies on the density function of the model's distribution, and is simply:

$$L = \prod_{i=1}^{n} f(t_i).$$

Because right-censored observations imply only that the study subject lasted a minimum duration without failure, these observations need to be handled separately in the likelihood by evoking the survival function based on the same underlying parameters. For the Type I case, the likelihood combining the two types of information (pre-censoring and post-censoring) is:

$$L \propto \left[\prod_{i=1}^{r} f(t_i)\right] \cdot S(t_{end})^{n-r},$$

where r is number of study subjects with observed failure times and t_{end} is the time at which the study was terminated. A more general version of the likelihood which encompasses the Type II (progressive) censoring case is

$$L \propto \prod_{i=1}^{r} f(t_i) \prod_{i=1}^{n-r} S(t_i^{+})$$

where t_i^{+} denotes potentially differing times where each of the $n-r$ observations was censored. It should be noted that this approach assumes an independent censoring mechanism.

3 Using *failCompare*

The focus of this package is statistical modeling of failure time data resulting from monitoring study subjects over a defined period. In this section, we introduce the package's functions, provide instructions on package installation, and work through several examples.

3.1 Overview of Functions

The *failCompare* package includes functions for several stages of analysis of failure time data: fitting the failure time models, ranking the models, and assessing lack-of-fit. There are several functions that can be used to estimate the sample survival function and for preliminary plotting (fc_surv() and fc_plot()). Individual models or sets of models are created using the fc_fit() function, which has two required arguments: (1) the data set and (2) a text string of one or more model names. This function fits the specified model(s) and stores the information in an object. Additional information on these models can be obtained by calling the failure model object inside the generic summary() function and the model fit can be visualized using plot(). The second stage of analysis ranks models using the GOF statistic via the function fc_rank(). Convenience functions are provided to assist the users by combining singular models (fc_combine()) into model lists for ranking or by selecting a single model (fc_select()) from a list. Finally, the fc_test() function performs the lack-of-fit test described above for any of the default models.

3.2 Relation to Other Software

The *failCompare* package depends on code from several other R packages: *survival (*Therneau and Grambsch 2000), *flexsurv* (Jackson 2016), and the *vitality* package. The vitality package was developed in connection with members of Columbia Basin Research (Passolt et al. 2018). Documentation and source code for these packages are available at:

https://cran.r-project.org/web/packages/survival/index.html

https://cran.r-project.org/web/packages/flexsurv/index.html

https://cran.r-project.org/web/packages/vitality/index.html

In the past Columbia Basin Research has provided a tool for adjusting survival models using tag life in the desktop application ATLAS (http://www.cbr.washington.edu/analysis/apps/atlas). The package *failCompare* provides the same tag-life fitting functions as ATLAS in addition to offering a greater selection of failure-time models and additional model ranking and testing capabilities. The *failCompare* package stands alone but is also designed to interface with the forthcoming *cbrATLAS R* package, which will be a script-based version of the desktop application.

3.3 Example Data Sets

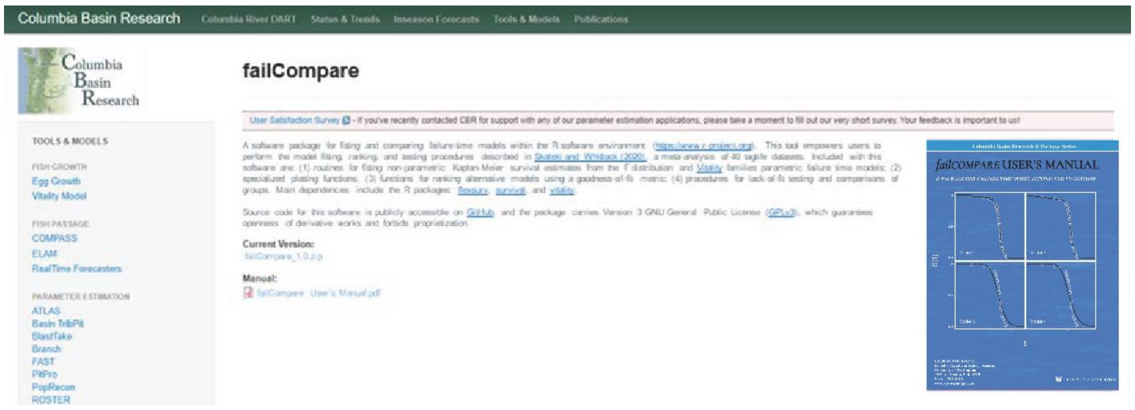
Most examples we provide here concern modeling the time until electronic tag failure. Accurately modeling the tag-failure process is a critical step in many fish and wildlife studies based on telemetry tags (i.e., transmitters). In these studies, acoustic tags are activated and implanted in fish so their movement and survival can be monitored as they move past receiver stations. Tags implanted in living study subjects eventually fail because of battery discharge, and sometimes prematurely due to manufacturing defects. Investigators need an estimate of the longevity of these tags to account for the tag-failure process in survival models. For this reason, a representative sample of the batch of tags used in the study is held out and used to measure the tag failure rate as a function of time in what is known as a tag life study (Townsend et al. 2006). The tags in the tag life study are configured the same as those implanted in fish, placed in water with a similar thermal environment, and monitored continuously by one or more nearby receivers.

3.4 Handling Censoring

In its current form (version 1.0), the *failCompare* package is equipped to handle two common forms of right-censored data: Type I and Type II, see the “Censored Data” section above for technical details. Type I censoring is conveniently handled by specifying the value after which all observations will be considered censored (rc.value, i.e., identifying the minimum failure time of the observations). Type II censoring requires specifying a censorID argument the same length as the time argument the censor ID argument indicates whether individual times describe the subject failure time or the subject censoring time. See Example 3 below for a demonstration on how to account for these two forms of censoring using *failCompare*'s model-fitting and plotting functions.

3.4.1 Downloading the *failCompare* package

To download the package, navigate to the Columbia Basin Research website and click on the hyperlink for the latest version at: http://www.cbr.washington.edu/analysis/apps/failCompare (as seen below). There is no need to unzip the compressed folder after downloading it to your hard drive.



failCompare

User Satisfaction Survey - If you've recently contacted CBR for support with any of our parameter estimation applications, please take a moment to fill out our very short survey. Your feedback is important to us!

A software package for fitting and comparing failure-time models within the R software environment (https://www.r-project.org). This tool empowers users to perform the model fitting, ranking, and testing procedures described in Skalski and Whitlock (2020), a meta-analysis of 40 taglife datasets. Included with this software are: (1) routines for fitting non-parametric Kaplan-Meier survival estimates from the F distribution and Vitality families parametric failure time models; (2) specialized plotting functions; (3) functions for ranking alternative models using a goodness-of-fit metric; (4) procedures for lack-of-fit testing and comparisons of groups. Main dependences include the R packages: flexsurv, survival, and vitality.

Source code for this software is publicly accessible on GitHub, and the package carries Version 3 GNU General Public License (GPLv3), which guarantees openness of derivative works and forbids proprietization.

Current Version:
failCompare_1.0.zip

Manual:
failCompare User's Manual.pdf

3.5 Getting Started

We assume that the user has previously installed program R and is familiar with basic operations (e.g., importing data, basic plotting). Program R can be downloaded freely at: https://cran.r-project.org. There are many freely accessible instructional books and online resources that explain the use of R. The user might also want to consider installing RStudio (https://www.rstudio.com), a free program that serves as wrapper for the basic R graphical user interface with a built-in text editor.

3.5.1 Installation within R

To install from within the base R graphical interface, select the option "install package(s) from local files…" from the "Packages" dropdown menu and then navigate to the compressed folder that was downloaded. Alternatively, to install *failCompare* from within RStudio, select the "Packages" tab and click the "Install" button. From there, choose the "Package Archive File (.zip;tar.gz)" and then navigate to the compressed folder that was downloaded, select it, and click "install"

3.5.2 Loading the *failCompare* package

Once the package is installed, it must be loaded into the working environment before it can be used. The installation needs to be performed only once, but the package must be loaded during each R session using the library() command:

```
> library(failCompare)
```

3.5.3 Example data

Four example data sets are provided with the *failCompare* package and are used here. Two of the example data sets contain failure time data of microacoustic transmitters ("tags") used in studies of migration survival of juvenile salmon. The third and fourth example data sets contain survival times of fish in a toxicology study and rats in a cancer study. The data sets are named "sockeye," "chinook," "trout," and "pike."

3.6 Example 1: Fitting, Visualizing, and Ranking Alternative Failure Time Models

Our first example uses the "sockeye" data set that comes with the package.

3.6.1 Preparation

3.6.1.1 Loading data

Below we use the data() command to load the example data set "sockeye." The data set contains only one variable, days, which identifies the failure times of acoustic tags in a tag life study. The $ is used to extract the values in a labeled column. We will store this variable in the vector object taglife.

```
> data(sockeye)
> taglife=sockeye$days   # vector of tag failure times
```

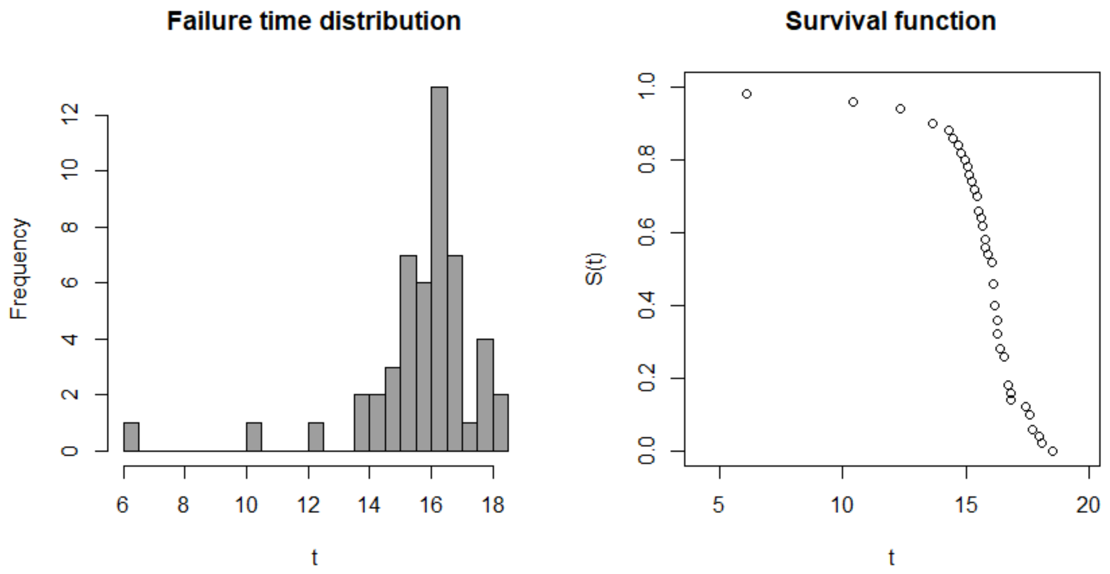
Individual observations are displayed in the table below.

6.12	14.29	15.04	15.50	15.75	16.08	16.25	16.37	16.71	17.71
10.42	14.46	15.12	15.50	15.79	16.08	16.25	16.54	16.79	17.71
12.33	14.67	15.21	15.62	15.87	16.17	16.29	16.71	16.83	17.96
13.62	14.79	15.33	15.67	16.04	16.17	16.29	16.71	17.42	18.04
13.62	14.96	15.42	15.75	16.08	16.17	16.37	16.71	17.58	18.50

3.6.1.2 Initial data visualization

We begin this example by computing the sample survival fraction and visualizing the data using basic R plotting tools. We will start by computing the sample survival function using the function fc_surv(). Because there are no censored observations in the data set we can compute the sample survival function $\hat{S}(t)$ using equation 4. The function fc_surv() displays both a histogram of the failure times and the sample survival function over time (t).

```
> S=fc_surv(taglife)

> fc_plot(time=taglife,surv=S)
```



There is one failure relatively early (~6 days) in the study and a peak between 15–20 days. The gradual decrease in survival preceding a cascade of failures indicates that a model with a variable hazard rate function is suitable. We start by fitting a 2-parameter Weibull model.

3.6.2 Fitting individual models

3.6.2.1 2-parameter Weibull model

The first step is to use the function fc_fit() with the vector of failure times for the time argument and "weibull" for the model argument.

```
> weib_mod=fc_fit(time=taglife,model="weibull")
```

Output produced by the function is stored in the object weib_mod. The accepted in the model argument are listed in the appendix of this manual or can be found in the help documentation for the function (accessible by entering ?fc_fit in the R console).

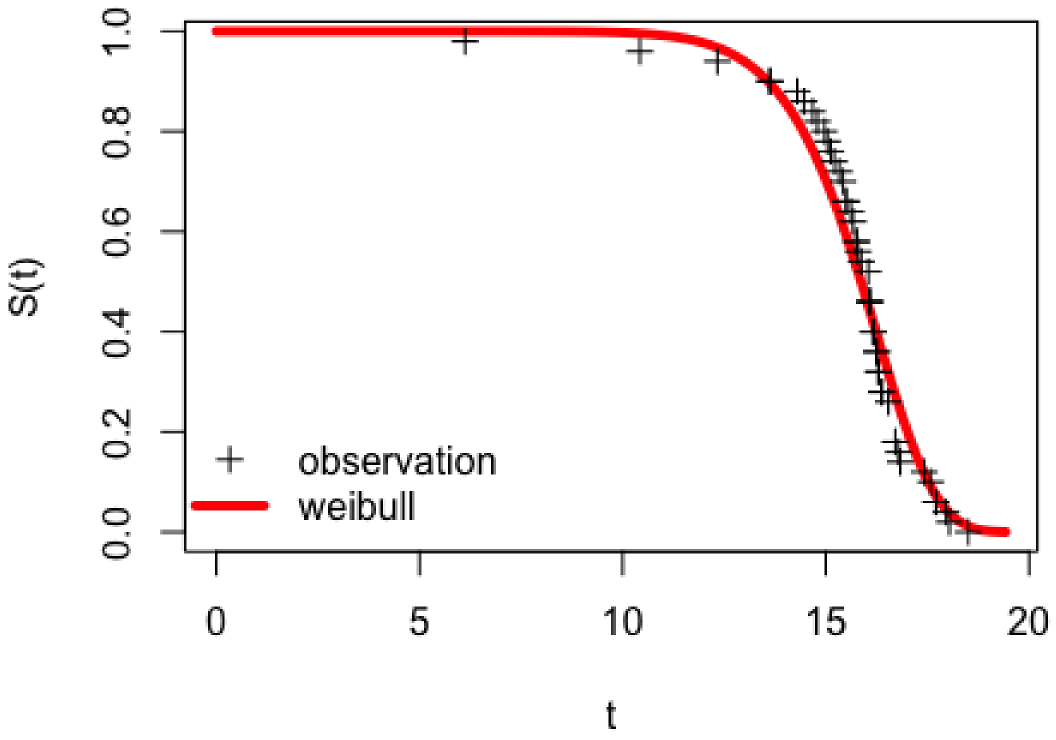
Entering weib_mod into the console prints some of the output, including parameter estimates and standard errors.

```
> weib_mod
```

```
weibull failure model object

Parameter estimates:
           est        se
shape 12.08980 1.3689716
scale 16.35115 0.1986745
```

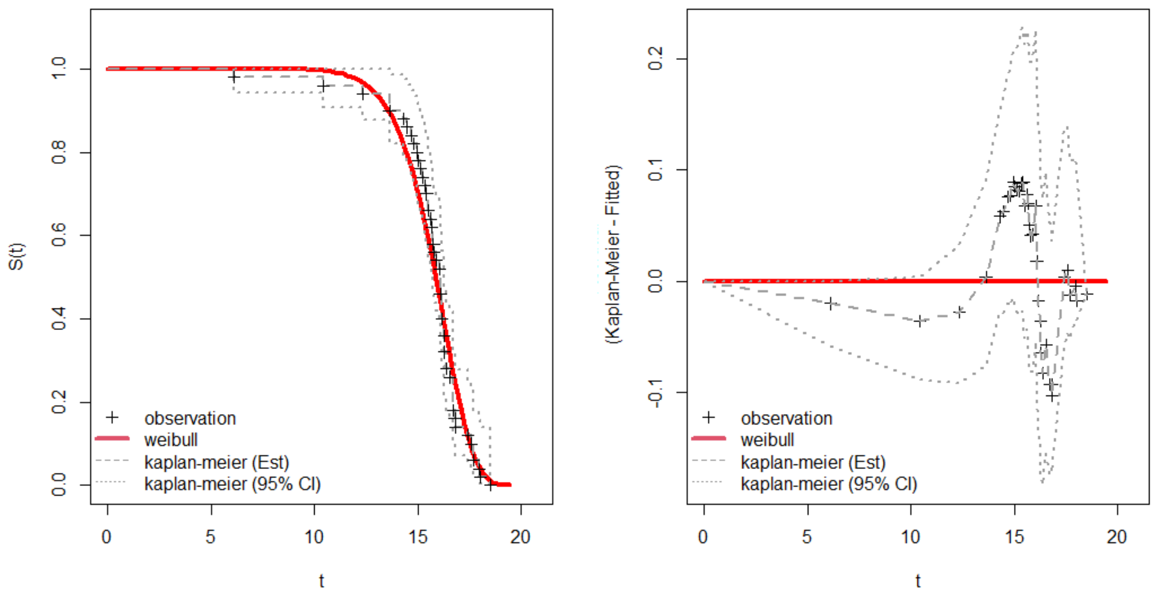
Placing the model object inside the plot() function produces a scatterplot of the fitted Weibull model (red line) versus the observations (“+” symbols) that reflect downward steps in $\hat{S}(t)$.

```
> plot(weib_mod)
```



Dashed lines between the K-M estimates can be added by including the argument km=TRUE, and confidence intervals for the K-M model can be added with the argument km.ci=TRUE (see below). An alternative plot may be created with the argument type="resid", showing the difference between the parametric model and the K-M model (i.e., “residuals at each observed failure time”). The residual option can be helpful for determining when the parametric model estimates are above or below the observations. For example, the plot on the right shows that the Weibull model overestimates survival in the first half of the study.

```
> plot(weib_mod,km = TRUE,km.ci = TRUE)
> plot(weib_mod,type = "resid",km = TRUE,km.ci = TRUE)
```



The Weibull model appears to be a pretty good fit for the data, although the fitted line does weave through the data over time. There may be a better alternative, so next we will try fitting the Vitality 2009 model.

3.6.2.2 Fitting the Vitality 2009 model

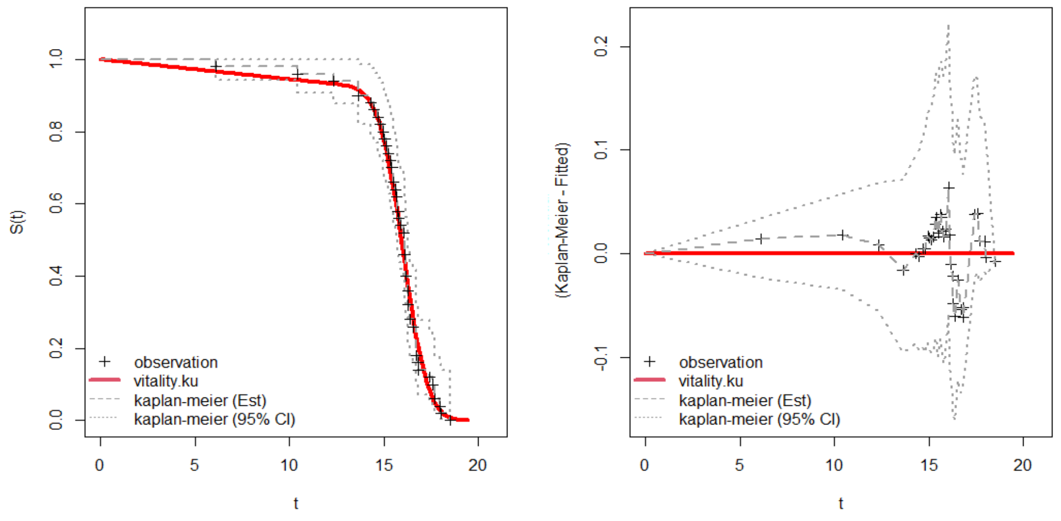
The Vitality 2009 model is estimated using fc_fit() in much the same way as before but now with "vitality.ku" entered into the model argument.

```
> vit09_mod=fc_fit(time=taglife,model="vitality.ku")
> vit09_mod

vitality.ku failure model object

Parameter estimates:
          est         se
r 6.2346e-02 0.0046263
s 8.5906e-08       NaN
k 5.7006e-03 0.0231610
u 6.4295e-02 0.0609970
```

Printing vit09_mod displays the four parameter estimates for the model. We see that the standard error of the *s* parameter could not be estimated.

We will now visualize the model fit as before.

```
> plot(vit09_mod,km = T,km.ci = T)
> plot(vit09_mod,type = "resid",km = T,km.ci = T)
```



This model appears to conform to the data slightly better than the Weibull model. However, it is unclear whether the Vitality 2009 model is actually more parsimonious (i.e., does the improvement in model fit warrant a model with two additional parameters?). We will rank the performance of these two models using the GOF metric to help answer this question. First we will predict tag survival for each model.

3.6.2.3 Obtaining predicted survival from models using fc_pred()

Once a model object is created, you can use it to predict the survival fraction at a given time. This is accomplished by passing the model object and the time for which we would like survival predicted to ofc_pre. For example, we will predict the survival fraction at 6, 12, 15, and 18 days under each of the models estimated above.

```
> pred_times=c(6,12,15,18) # days whose survival we wish to predict

# Weibull model predictions
> pred_s_weib=fc_pred(mod_obj=weib_mod,times=pred_times)
> pred_s_weib
[1] 0.99999455 0.97653702 0.70293295 0.04097672

# Vitality model predictions
> pred_s_vit=fc_pred(mod_obj=vit09_mod,times=pred_times)
> pred_s_vit
[1] 0.96637473 0.93383827 0.77416444 0.02585396
```

We can see from these predictions that the survival estimates are lower for the vitality model in all but the 15-day estimate.

3.6.3 Model comparisons

3.6.3.1 Combing failure models in a list

The models need to be placed in a list before they can be ranked. There are two options for doing this: combining the two models into a list() object and running fc_combine, or rerunning fc_fit while specifying a vector of more than one model name using the combine function c().

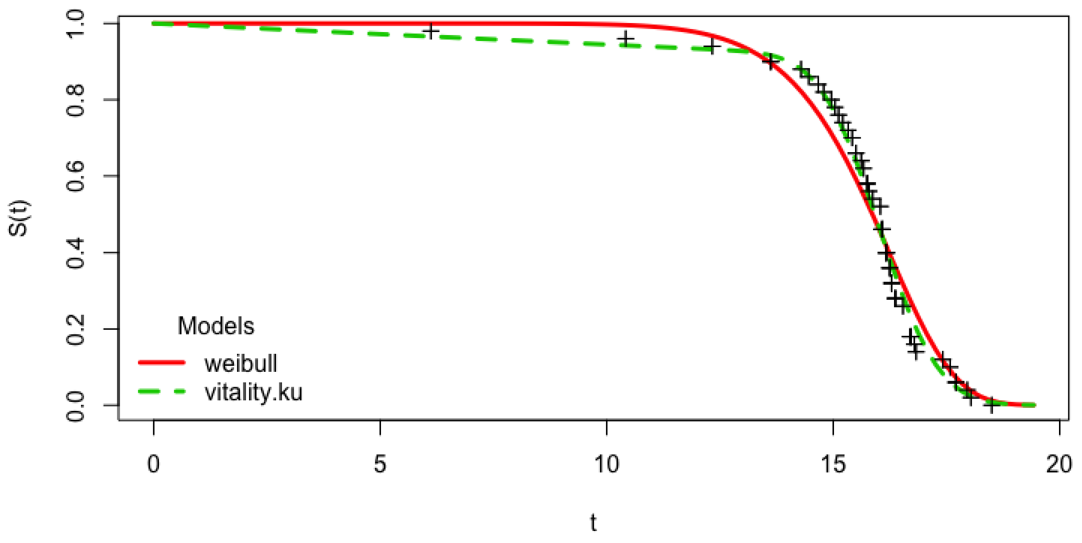
```
> fmods=fc_combine(list(weib_mod,vit09_mod)) # option 1

# OR

> fmods=fc_fit(taglife,c("weibull","vitality.ku")) # option 2
```

Printing fmods provides a short description of the models in the list and a message indicating that the model list can be ranked. Placing the model list object inside summary() provides details on the parameter estimates for each model.

```
> fmods
Failure model list object

Contains the following 2 models:
 weibull ; vitality.ku

*use this object to compare models using the function: fc_rank()
```

Plotting fmods produces a survival function plot with the raw K-M estimates and the two models labeled. Dashed lines connecting K-M estimates can be added with the argument km=T.

```
> plot(fmods)
```



3.6.3.2 Ranking failure time models based on GOF measure

Executing fc_rank()prints a table ranking the models in the list, and creates a new model list object with the GOF ranking information. The fmods_R object stores this information and it can be printed by the user at any time.

```
> fmods_R=fc_rank(fmods)

Candidate models ranked by goodness of fit measure:

        model    SSE_KM  n npars denom    GOF
1 vitality.ku 0.0502537 50     2    47 0.0011
2     weibull 0.1824184 50     4    45 0.0041
```

The Vitality 2009 model has the lowest GOF score, and therefore ranks above the 2-parameter Weibull. If these were the only two models being considered, then we would select the Vitality 2009 model and stop here.

The GOF measure is used to compare and rank models based on their fit to the data. It is a relative measure, so it is possible for a model to have a high GOF rank compared to the other models but to still be a poor fit for the data; this happens if none of the models fit well. The overall fit of a model to the data is assessed in a lack-of-fit test as demonstrated in Example 2 (Section 3.7).

3.6.4 Simultaneous model fitting

A shortcut for fitting all nine parametric models in *failCompare* is to type "all" into the model argument of fc_fit(). Printing fmods_all tells us the nine models were fit.

```
> fmods_all=fc_fit(taglife,model="all")

Fitting all available parametric survival models
fmods_all
Failure model list object

Contains the following 9 models:
 weibull ; weibull3 ; gompertz ; gamma ; lognormal ; llogis ; gengamma
; vitality.ku ; vitality.4p

*use this object to compare models using the function: fc_rank()
```

3.6.4.1 Ranking the full set of parametric models

Executing fc_rank on fmods_all returns a table ranking all nine models and stores all the information in a ranked model list object.

```
> fmods_all_R=fc_rank(fmods_all)


Candidate models ranked by goodness of fit measure:

        model     SSE_KM  n npars denom    GOF
1 vitality.ku 0.05025370 50     4    45 0.0011
2 vitality.4p 0.05297191 50     4    45 0.0012
3    gompertz 0.17255657 50     2    47 0.0037
4     weibull 0.18201868 50     2    47 0.0039
5    weibull3 0.18241771 50     3    46 0.0040
6    gengamma 0.21987838 50     3    46 0.0048
7      llogis 0.24616132 50     2    47 0.0052
8       gamma 0.74189324 50     2    47 0.0158
9   lognormal 0.89427673 50     2    47 0.0190
```

The top ranking model is labeled `vitality.ku` which corresponds to the Vitality 2009 model. We can further see that the `vitality.4p` (Vitality 2013) model is ranked a close second and that the `Gompertz` model beat out the 2-parameter Weibull model that we considered before.

Placing a ranked model list inside the `plot()` function displays up to three models at a time; model rank is shown in parentheses after the model name. The three top-ranking models are displayed by default, but specific subsets can also be obtained by supplying a vector of up to three names to the `model` argument. Below is a default plot and a plots of sets of three models that shared the middle and lowest rankings (top right and bottom left).

```
> plot(fmods_all_R,main="top ranked")

> plot(fmods_all_R,model=c("weibull","weibull3","gengamma"),
        main="middle ranked")

> plot(fmods_all_R,model=c("llogis","lognormal","gamma"),
        main="bottom ranked")
```

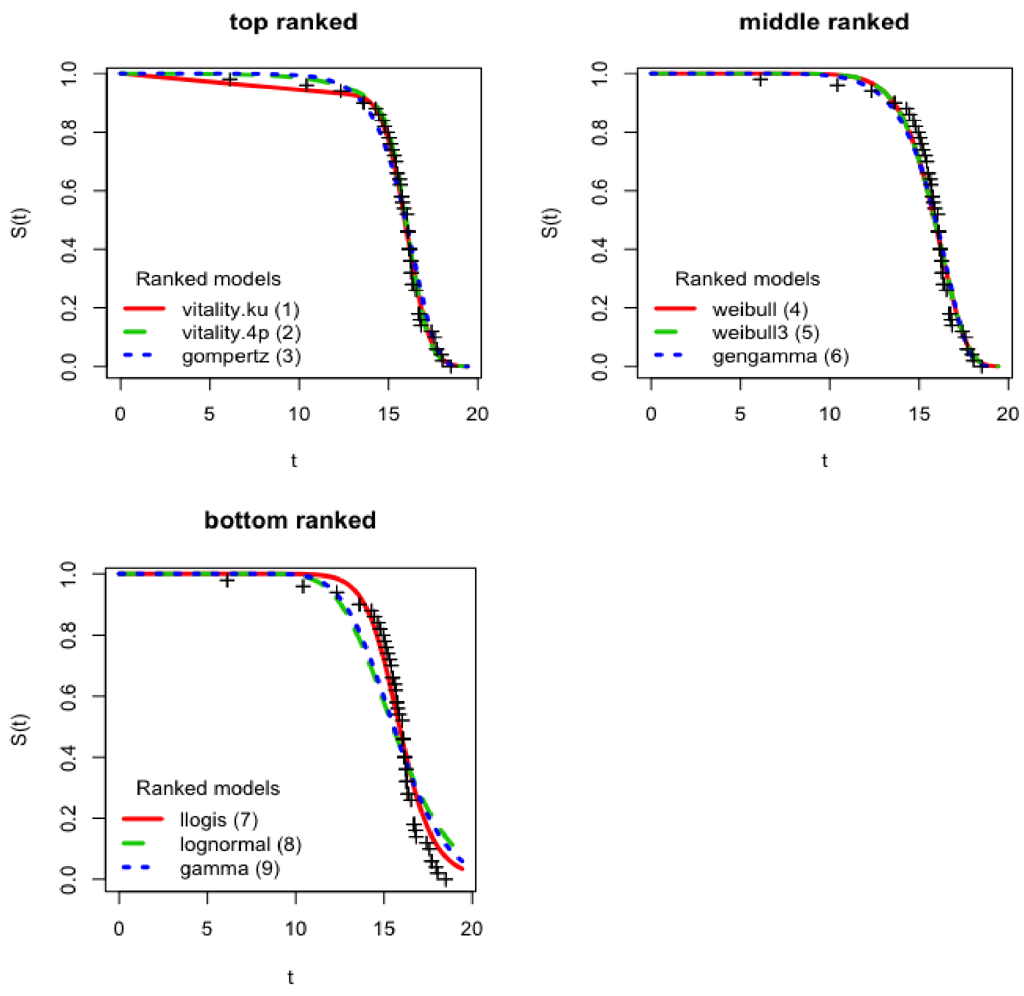


Figure 6. Plots of the top-ranked, middle-ranked, and bottom-ranked three models for the “sockeye” example data set. The top ranked model all fit the data well, especially after the 15-day mark, whereas the poorest ranking models depart from the data significantly. The Vitality 2009 model is unique in that it is the only model with an initial linear decline in survival that intersects with early failures in the “shoulder” of the curve, so it appears to be the best choice.

3.7 Example 2: Testing Methods for Failure Time Models

In this example, we demonstrate the testing procedures contained within the *failCompare* package. The first of these, fc_diff(), is used to test for differences among groups in a combined failure-time data set. The second test, fc_test(), assesses the general lack-of-fit of a model; this is an absolute measure that is relevant to a single model, unlike the relative measure of model performance (i.e., GOF measure; Skalski and Whitlock 2020).

If you are continuing to this example from the first it may be wise to restart R and reload the *failCompare* package. This will prevent any confusion resulting from accidentally accessing previously defined objects in the environment. Alternatively, you can remove all objects in the workspace by running the command: rm(list=ls()) (with nothing inside of it). The example dataset for this task is titled "chinook." Similar to example 1, it contains data describing the time until acoustic tag deactivation without censoring. As before, we can load the dataframe into the R environment by using data(). The structure of the data is examined below using the function str().

```
> data(chinook)
> str(chinook)

'data.frame': 80 obs. of  2 variables:
 $ days  : num  18.7 22.7 33.3 34.1 34.4 ...
 $ season: Factor w/ 2 levels "spring","summer": 1 1 1 1 1 1 1 1 1 1 ..
.
```

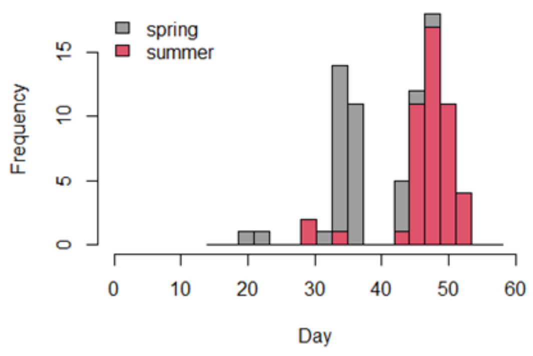
This reveals that we are working with a dataset with 80 observations. This time there is a second column identifying the group to which each tag belongs, a factor variable called "season" consisting of two groups: "spring" and "summer." A frequency table for this second column further indicates that there are 33 tags in the spring group and 47 tags in the summer group.

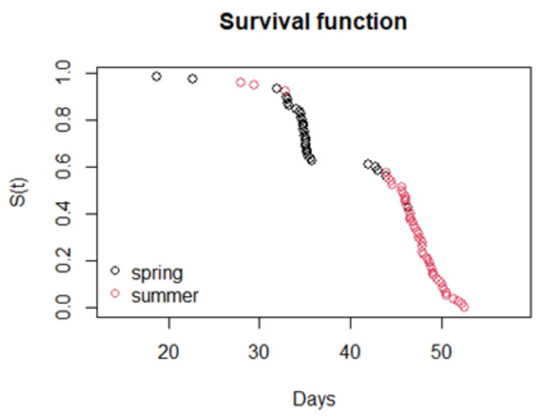
```
> table(chinook$season)

spring summer
    33     47
```

We would like to know whether it is most appropriate to pool all tags when modeling tag failure or model each group separately.

To begin, we visualize the data using a histogram and a plot, notice the additional argument for group that distinguishes between groups in the plots.

```
> surv=fc_surv(chinook$days)
> fc_plot(chinook$days,surv,group = chinook$season)
```





These plots show a bimodal distribution of the combined failure times and indicate that the spring tag failure times are earlier on average. The survival function is also far from smooth, particularly before day 45, when the majority of "spring" group of tags failed. Given that the failure time distribution appears to differ by season it may not be justified to pool them, but how do we know for sure?

3.7.1 Log-rank test for comparing groups

Fortunately, *failCompare* includes a function for performing a log-rank test, which tests the null hypothesis that two groups of observations arise from the same distribution. A log-rank test can be performed by providing the name of the dataframe and specifying the failure time (time) and a categorical grouping variable (group). The test is carried out using the *survival* package, assuming a χ^2 distribution for the test statistic.

```
> fc_diff(data=chinook,time="days",group="season")

Call:
survdiff(formula = f1, data = data)

               N Observed Expected (O-E)^2/E (O-E)^2/V
season=spring 33       33     10.9     44.96      63.9
season=summer 47       47     69.1      7.08      63.9

Chisq= 63.9  on 1 degrees of freedom, p= 1e-15
```

The output shows the observed (O) versus the expected (E) values and the test statistic Chisq. The P-value for the test statistic is extremely small (le-15), indicating that there is significant evidence to reject the null hypothesis that the failure time distribution is the same for the two seasons.

We now proceed by fitting separate models to the spring and summer groups, which we denote with the suffix _SPR and _SUM, respectively. First, we subset the data into groups, and then we fit the value and rank all models for all groups.

```
# Subsetting the groups
> chn_SPR=subset(chinook,season=="spring")
> chn_SUM=subset(chinook,season=="summer")
```

3.7.2 Ranking models separately for two groups

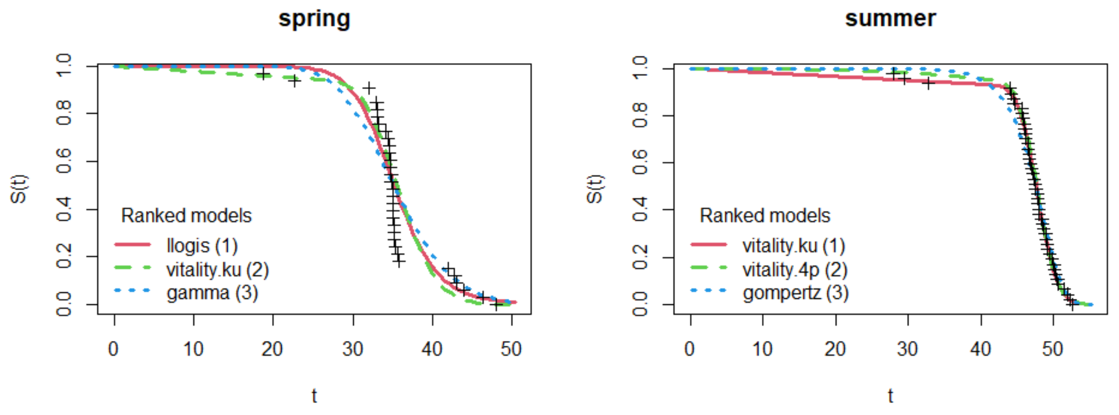
Next, we fit the nine different default models in *failCompare* to each dataset separately by entering "all" into the model argument

```
# Ranking of models for season="spring"
> chnSPR_mods=fc_fit(time=c(chn_SPR$days,42,42.4),model="all")

Fitting all available parametric survival models

> chnSPR_mods_R=fc_rank(chnSPR_mods)

Candidate models ranked by goodness of fit measure:
        model    SSE_KM  n npars denom    GOF
1      llogis 0.5148720 33     2    30 0.0172
2 vitality.ku 0.5827913 33     4    28 0.0208
3       gamma 0.6258051 33     2    30 0.0209
4   lognormal 0.6398952 33     2    30 0.0213
5    gengamma 0.6537550 33     3    29 0.0225
6 vitality.4p 0.6359227 33     4    28 0.0227
7    weibull3 0.6832296 33     3    29 0.0236
8     weibull 0.7103504 33     2    30 0.0237
9    gompertz 0.7849153 33     2    30 0.0262

# Ranking of models for season="summer"
> chnSUM_mods=fc_fit(time=chn_SUM$days,model="all")

Fitting all available parametric survival models

> chnSUM_mods_R=fc_rank(chnSUM_mods)

Candidate models ranked by goodness of fit measure:
        model     SSE_KM  n npars denom    GOF
1 vitality.ku 0.01928644 47     4    42 0.0005
2 vitality.4p 0.03373248 47     4    42 0.0008
3    gompertz 0.12668419 47     2    44 0.0029
4     weibull 0.16703594 47     2    44 0.0038
5    weibull3 0.16704176 47     3    43 0.0039
6    gengamma 0.19689311 47     3    43 0.0046
7      llogis 0.23086554 47     2    44 0.0052
8       gamma 0.86607690 47     2    44 0.0197
9   lognormal 0.96823470 47     2    44 0.0220
```

The rankings based on the GOF metric indicate that the log-logistic (llogis) and the Vitality 2009 model (vitality.ku) provide the best fits to the spring and summer groups, respectively.

Plotting the three top-ranking models for each season allows us to examine the quality of the fit. We can add a main title to each of the plots by providing a character string for the optional argument main. The function also prints a message reminding the user of the other models in the list that they are not seeing in the plot.

```
> plot(chnSPR_mods_R,main="spring")

Additional models with rankings: weibull(4); weibull3(5); gompertz(6);
lognormal(7); gengamma(8); vitality.4p(9)

> plot(chnSUM_mods_R,main="summer")

Additional models with rankings: weibull(4); weibull3(5); gamma(6); log
normal(7); llogis(8); gengamma(9)
```



Looking first at the spring data, we see that although the log-logistic model is the highest ranking, it does not appear to fit the spring group all that well. Conversely, the Vitality 2009 model (vitality.ku) fits the summer data quite well, and even the second and third ranked models are competitive. We will select the vitality 2009 model from among the list of candidates using the fc_select() function and then print parameter estimates.

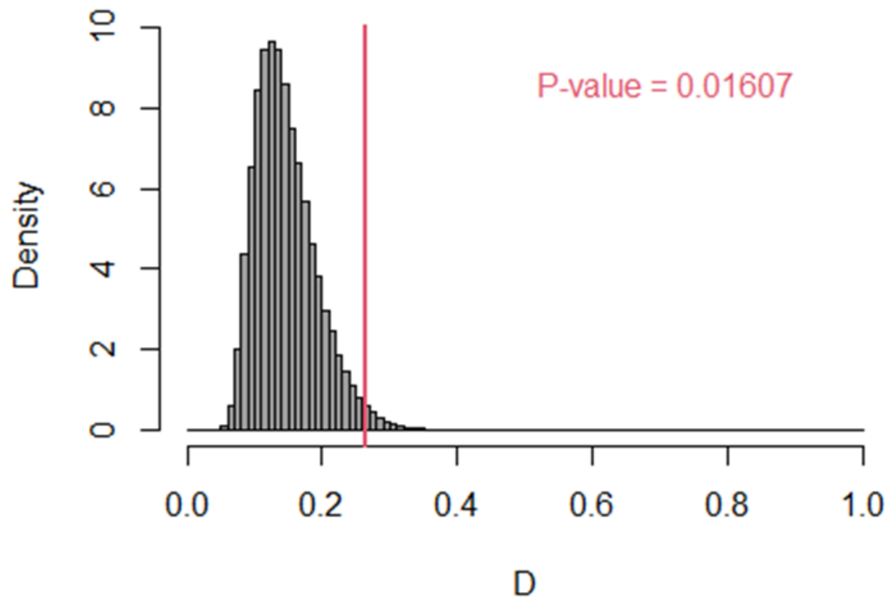
```
> chnSUM_vit09=fc_select(mod_ls = chnSUM_mods_R,model = "vitality.ku")
> chnSUM_vit09

vitality.ku failure model object

Parameter estimates:
         est         se
1 2.0898e-02 0.00098417
2 6.3562e-03 0.00493800
3 1.7174e-03 0.00665980
4 1.5304e-06 0.00243200
```

The obviously poor fit to the spring data should make us skeptical of the validity of the model, but it would be good to use a statistical test to confirm this.

3.7.3 Kolmogorov-Smirnov test (simulation-based)

We now examine the general lack-of-fit of the top model using a simulation-based Kolmogorov-Smirnov test. This testing method was described in Lilliefors (1967) and was used to compare all nine default models in *failCompare* in Skalski and Whitlock (2020). The test works by fitting a parametric model to the data, calculating a test statistic by which to compare data to the failure time model used to fit it, and then simulating random data sets of the same size and recomputing the test statistic to approximate the sampling distribution. Provided that enough samples are drawn, we can accurately approximate a P-value for the test by determining the proportion of the sampling distribution that is larger (i.e., more extreme) than the observed statistic. We perform this test using the command fc_test with the model argument matching the name of the top-ranking model for this population of tags, and the number of iterations (iters) set to 100,000. It may take up to a minute for the following code to run. If the argument plot=TRUE is included, you a histogram of the sampling distribution of the test statistic (D) is displayed, with a vertical red line denoting the observed value. The p-value of the test will be printed in the console and on the right side of the plot.

```
> fc_test(times = chn_SPR$days,model = "llogis",iters = 100000,plot=TRUE)

Results of a one-sample Kolmogorov-Smirnov test based on a simulation

model =  llogis

iterations =  1e+05

observed test statistic
 D[obs] = 0.2648246

p-value =  0.01607
```



The P-value below $\alpha = 0.05$ indicates that the observed data do not adhere to log-logistic model. It should be noted that the P-value may change slightly from run-to-run because of the inherent randomness of the bootstrap. The default number of iterations is 50,000. If the histogram of the sample statistic is not smooth and/or the P-value varies significantly from run to run, the number of iterations should be increased.

Because the log-logistic model was the best model available, it is reasonable to assume that there are no appropriate parametric models for describing the data; thus, it may be best to avoid interpreting a parametric model and proceed with interpreting the nonparametric K-M model (Section 3.11).

3.7.4 Selecting the Kaplan-Meier model from a model list object

An option for selecting the K-M model exists in *failCompare* if the user enters `kaplan-meier` into the `model` argument of `fc_select()`.

```
> chnSPR_KM=fc_select(mod_ls = chnSPR_mods_R,model = "kaplan-meier")
```

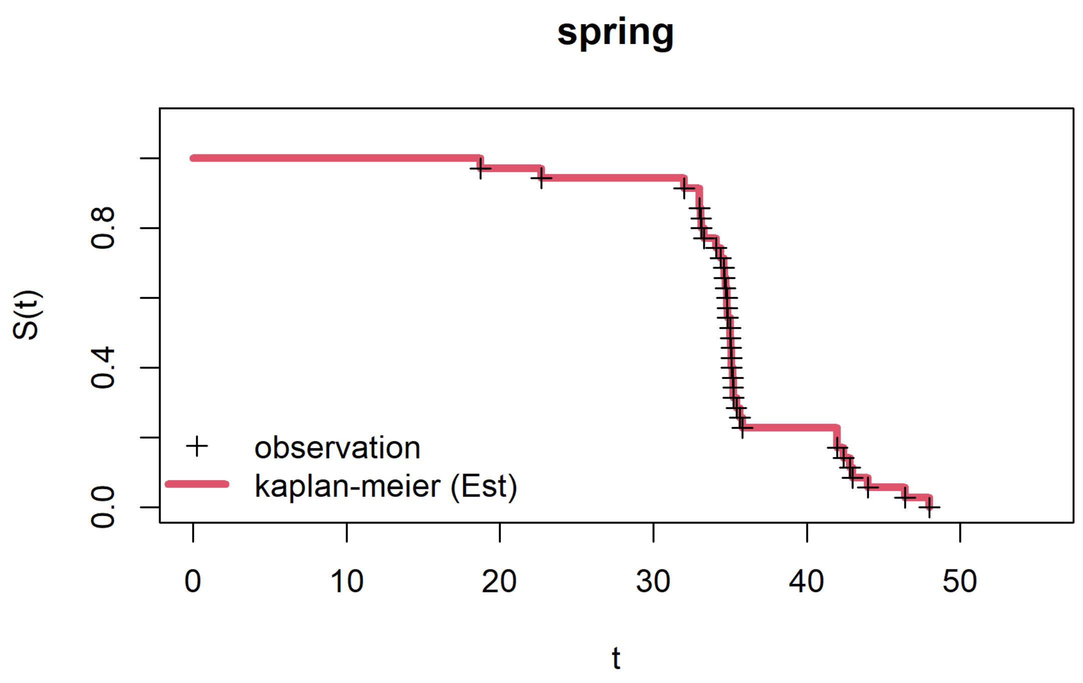
Printing the K-M model returns the estimates and confidence intervals for the cumulative survival to each unique failure time in the data set.

```
> chnSPR_KM


Kaplan-Meier estimates for increments between failure times
    time        est         lcl       ucl
1   0.00 1.00000000 1.000000000 1.0000000
2  18.74 0.97142857 0.917774047 1.0000000
3  22.70 0.94285714 0.869010963 1.0000000
4  32.00 0.91428571 0.826091339 1.0000000
5  33.00 0.85714286 0.748711879 0.9812772
6  33.11 0.82857143 0.712664580 0.9633292
7  33.13 0.80000000 0.677876107 0.9441253
8  33.30 0.77142857 0.644136245 0.9238760
9  34.08 0.74285714 0.611299632 0.9027271
10 34.40 0.71428571 0.579261536 0.8807836
11 34.58 0.68571429 0.547944329 0.8581238
12 34.64 0.65714286 0.517289404 0.8348068
13 34.70 0.62857143 0.487252103 0.8108781
14 34.80 0.60000000 0.457798406 0.7863723
15 34.82 0.57142857 0.428902724 0.7613162
16 34.83 0.54285714 0.400546403 0.7357297
17 35.00 0.51428571 0.372716736 0.7096268
18 35.03 0.48571429 0.345406318 0.6830169
19 35.07 0.45714286 0.318612700 0.6559048
20 35.08 0.42857143 0.292338278 0.6282909
21 35.10 0.40000000 0.266590415 0.6001716
22 35.18 0.37142857 0.241381805 0.5715393
23 35.20 0.34285714 0.216731133 0.5423818
24 35.21 0.31428571 0.192664100 0.5126825
25 35.41 0.28571429 0.169214967 0.4824198
26 35.64 0.25714286 0.146428847 0.4515671
27 35.80 0.22857143 0.124365165 0.4200927
28 42.00 0.17142857 0.082749766 0.3551400
29 42.40 0.14285714 0.063455758 0.3216125
30 42.80 0.11428571 0.045441103 0.2874319
```

```
31 43.00 0.08571429 0.029049945 0.2529071
32 44.00 0.05714286 0.014877206 0.2194838
33 46.40 0.02857143 0.004139792 0.1971902
34 48.00 0.00000000 0.000000000 0.0000000
```

Plotting the K-M model produces a plot similar to the one above only with discontinuous steps corresponding to the observed failure time.

```
plot(chnSPR_KM,main="spring")
```



3.8 Example 3: Working with Censored Data

Censored data occur when not all study subjects have an observed failure time. All that is known about the censored observations is that the subject did not fail for at least a certain duration. If these observations are not handled correctly, then the survival function could be severely biased. In the following two examples, we illustrate approaches for handling right censoring, which is the most common data complication in failure time studies.

Right-censoring occurs when the initiation time for study subjects is known, but the failure time of some subjects is not. This occurs because the study concludes before all possible failures have occurred (Type I censoring) or because individual subjects dropped out before the end of the study and their failure time could not be recorded (Type II censoring). Refer to the section above titled "Censored Observations" for a more detailed explanation of these mechanisms and how they are dealt with during model estimation.

3.8.1 Type I censoring

The following example concerns monitoring the mortality of fish that are exposed to gas supersaturation. Gas supersaturation occurs when water contains an overabundance of dissolved gas, a state that sometimes occurs at the outflow of dams and may be lethal to fish (Weitkamp and Katz 1980). This study is an example of Type I censoring because the study was terminated after 30 days, at which point a portion of the study subjects were still alive.

We load the data set in as before using `data()`. Printing the dataframe reveals that 9 of the 35 observations (~26%) all have a value of 30 and `TRUE` under the column titled `censored`.

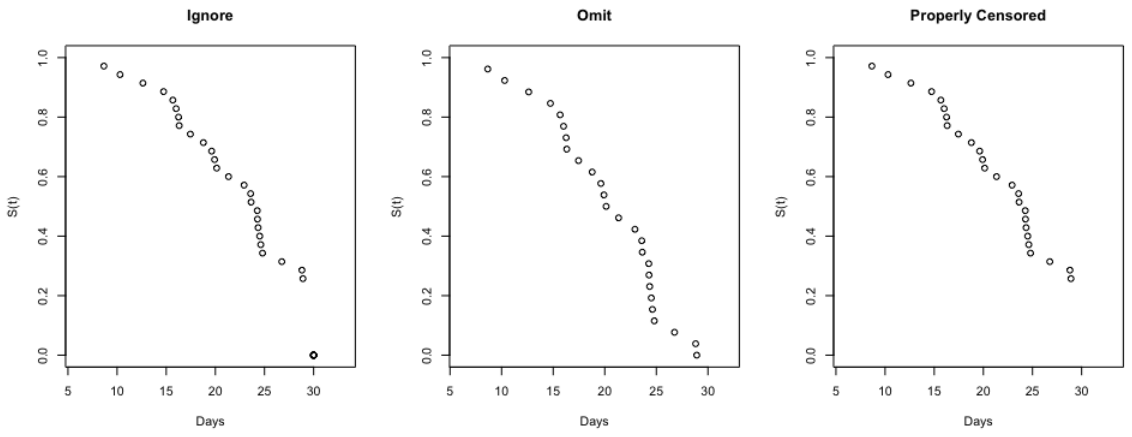
```
> data(trout)

#printing the dataframe
> trout
                                        (output continued)
 days  censored                          …
 8.66      FALSE                         24.29      FALSE
 10.3      FALSE                         24.35      FALSE
 12.63     FALSE                         24.51      FALSE
 14.73     FALSE                         24.62      FALSE
 15.67     FALSE                         24.8       FALSE
 16.01     FALSE                         26.76      FALSE
 16.26     FALSE                         8.81       FALSE
 16.32     FALSE                         8.91       FALSE
 17.46     FALSE                            30      TRUE
 18.78     FALSE                            30      TRUE
 19.62     FALSE                            30      TRUE
 19.92     FALSE                            30      TRUE
 20.13     FALSE                            30      TRUE
 21.34     FALSE                            30      TRUE
 22.93     FALSE                            30      TRUE
 23.59     FALSE                            30      TRUE
 23.64     FALSE                            30      TRUE
 24.26     FALSE
```

We will save the failure times (observed mortalities in this case) in a vector object for convenience.

```
# saving a vector of the failure times for convenience
> mort_day=trout$days
```

Next, we illustrate the consequences of failing to properly address censored observations. First, treating the censored observations as actual failure times ignores the fact that these subjects could have survived longer and artificially lowers the height of survival function. That is, it indicates a more intense mortality process than is appropriate. The second option of simply omitting the censored fish causes the survival function to drop even more abruptly. We illustrate these two incorrect approaches in the left and middle plots below. The plot on the right shows a survival function where the censoring is handled correctly; note that the estimated survival functions s(t) do not extend to 0 in the right plot.

These plots were created using the following code, which showcases some new arguments for the functions fc_surv(), fc_plot(), and fc_fit() (used in previous examples).



Code for the left plot:

```
# survival function (treating censored observations as actual failure times)
> unadj_S=fc_surv(time = mort_day)
> fc_plot(time = mort_day,
      surv = unadj_S,
      hist=F,
      main="Ignore")
```

Code for the middle plot:

```
# survival function (ignoring censoring)
> mort_day_sub30=mort_day[mort_day<30] # subsetting
> omit_S=fc_surv(time = mort_day_sub30)
> fc_plot(time = mort_day_sub30,
      surv = omit_S,
      hist=F,
      main="Omit")
```

Code for the right plot:

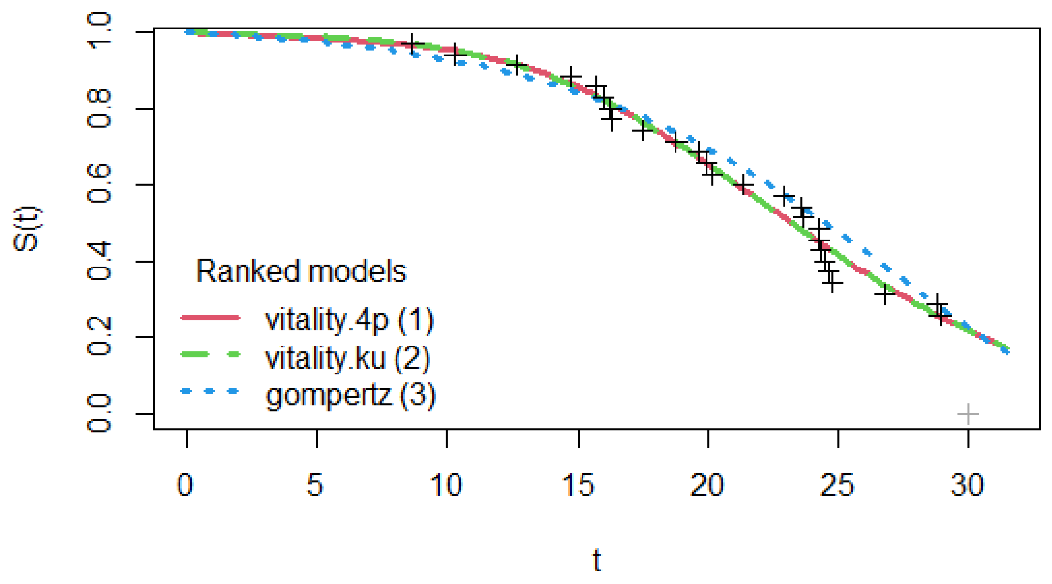
```
# survival function (with right censoring)
> cens_S=fc_surv(time = mort_day,
             rc.value = 30)
> fc_plot(time = mort_day,
       surv = cens_S,
       hist=F,
       main="Properly Censored")
```

Within fc_plot(), we use the argument hist=FALSE to prevent histograms of failure time from being created (as in example 2), and the argument main overrides the default main title for each plot. The object mort_day_sub30 represents a subset of the failure times that includes only those below the study's conclusion at 30 days. The object cens_S is created using fc_surv() as above but with the argument rc.value=30, used to indicate that the observations ≥ 30 are right-censored and not actual failure times.

Finally, we use the same rc.value=30 argument inside the fc_fit function to fit all available *failCompare* models, and then rank the list of models using fc_rank. Importantly, the sample survival function estimates are the K-S estimates based on the calculation in equation 5 (page 4) and the GOF metric is based on the distance between the K-M estimates and the parametric model survival functions, all of which account for the censoring.

```
# Fitting models
> trout_mods=fc_fit(mort_day,rc.value = 30,model="all")

# Ranking models
> trout_mods_R=fc_rank(trout_mods)

Candidate models ranked by goodness of fit measure:

         model    SSE_KM  n npars denom    GOF
1 vitality.4p 0.4512892 35     2    32 0.0141
2 vitality.ku 0.4524401 35     4    30 0.0151
3     weibull 0.5548681 35     2    32 0.0173
4    gompertz 0.5428142 35     3    31 0.0175
5       gamma 0.6020101 35     2    32 0.0188
6      llogis 0.6131972 35     2    32 0.0192
7    gengamma 0.6073838 35     4    30 0.0202
8   lognormal 0.6641553 35     2    32 0.0208
```

From this, we can see that the Vitality 2013 model (vitality.4p) is top ranking model, followed closely by the Vitality 2009 model (vitality.ku). The plot below shows the fit of the top-performing models. Notice that the survival functions do not reach 0 within the study period, indicating that fitted models and K-M estimates are accounting for right-censoring. Censored times (at or beyond $t = 30$) are denoted with the grayed-out symbol at $s(t) = 0$.



3.8.2 Type II censoring

Our final censoring example describes the case in which study subjects drop out while the investigation is ongoing rather than only at the end of the study (e.g., Type II censoring). This example data set can be loaded into R by running data(pike). Despite the "fishy" name, this data set is actually named for the author of a study on cancer deaths in rats and is given as an example in Lee and Wang (2003).

We begin by loading and viewing the data.

```
> data(pike)
> pike                                  (output continued)
   days death                              days death
1   142     1                           12  233     1
2   156     1                           13  239     1
3   173     1                           14  240     1
4   198     1                           15  261     1
5   204     0                           16  280     1
6   205     1                           17  280     1
7   232     1                           18  296     1
8   232     1                           19  296     1
9   233     1                           20  323     1
10  233     1                           21  344     0
11  233     1
```

There are 21 study subjects, the failure times are recorded in days, and the column titled "death" is an indicator variable with observed deaths denoted by 1 and censored observations by 0. Unlike in the previous example, the two censored observations do not both fall at the end of the study: one is at day 204 and another is at 344. In this case, we must account for the censoring using the censorID argument in the function fc_fit. For convenience we first define two vectors as inputs associated with the two columns days and death.

```
mort=pike$days  # represents observed mortality or censoring
death=pike$death # zeroes indicate censoring
```

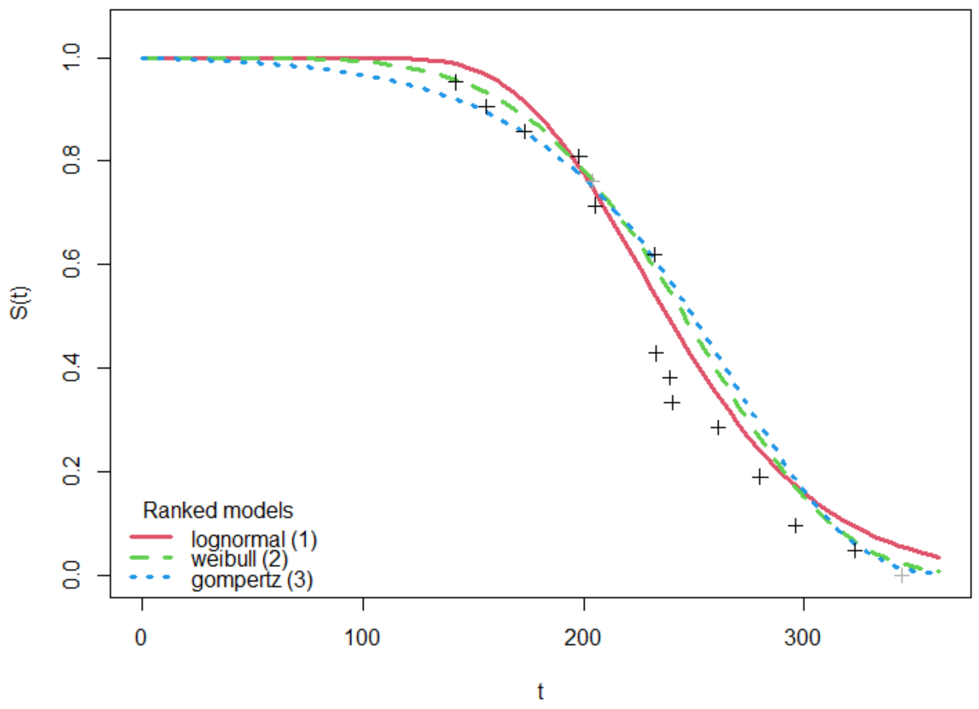
The original study modeled the death of rats using a Weibull model with Type II censoring. Here, we will compare three of our parametric models: the 2-parameter Weibull, Gompertz, and the lognormal model. We do this using fc_fit this time listing the models we want to fit in a character string.

```
> pike_mods=fc_fit(time=mort,censorID=death,model=c("lognormal", "weibull","gompertz"))
```

After defining the model list, we rank the model fit using fc_rank.

```
> pike_mods_ranked=fc_rank(pike_mods)

Candidate models ranked by goodness of fit measure:

      model    SSE_KM  n npars denom    GOF
1 lognormal 0.1259917 21     2    18 0.0070
2   weibull 0.2127602 21     2    18 0.0118
3  gompertz 0.2734661 21     2    18 0.0152

> plot(pike_mods_ranked)
```

We see from this ranking that the lognormal model outperforms the alternatives. Also, notice that the two grayed out "+" signs correspond to the censored observations at 204 and 344 days.



We will now select and summarize the lognormal model.

```
> pike_lnrm_mod=fc_select(pike_mods_ranked,model = "lognormal")
> summary(pike_lnrm_mod)
Summary of lognormal failure model object

Call:
flexsurv::flexsurvreg(formula = survival::Surv(time = y, event = non_ce
n) ~
    1, dist = model[i])

Estimates:
         est     L95%    U95%    se
meanlog  5.4727  5.3722  5.5732  0.0513
sdlog    0.2309  0.1675  0.3181  0.0378

N = 21,  Events: 19,  Censored: 2
Total time at risk: 5033
Log-likelihood = -104.2402, df = 2
AIC = 212.4804
```

4 Appendix

4.1 Parametric Model Descriptions

The following sections summarize distinguishing characteristics and key functions (density, survival, and hazard) of the nine parametric failure time models contained within *failCompare* (version 1.0). Information on the R package used to fit the model internally and a mapping of parameter names to equations are also provided.

4.1.1 Models Based on Probability Distributions

Failure time models organized under this subheading are similar in that they are all based on positive continuous distributions describing the time until failure from which the survival function is derived using their cumulative distribution function ($S(t) = 1 - F(t)$). Almost all models described here belong to the generalized F distribution family. This group of models is in contrast to the Vitality family of models, which have “evolving” density functions (Section 4.1.2).

4.1.1.1 Weibull model (2-parameter)

Label in *failCompare*: "weibull"

Model fitting package: *flexsurv*

Parameter definitions: shape = β ; scale = λ

Description:
Log-linear hazard function describing accelerating (or decelerating) failures over time. Also provides the benefit of closed-form estimates for the three defining functions. Reduces to exponential distribution with scale parameter λ when $\beta = 0$.

Density function:

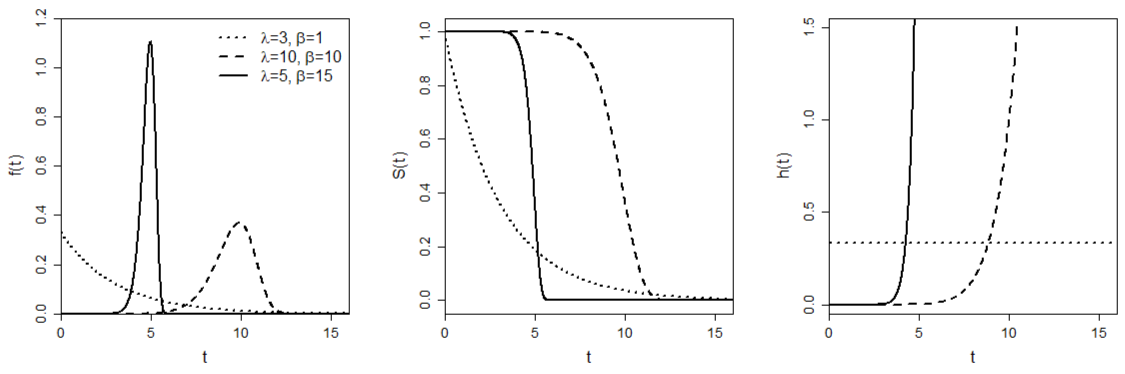
$$f(t) = \frac{\beta}{\lambda}\left(\frac{t}{\lambda}\right)^{\beta-1} e^{-\left(\frac{t}{\lambda}\right)^{\beta}}$$

Survival function:

$$S(t) = e^{-\left(\frac{t}{\lambda}\right)^{\beta}}$$

Hazard function:

$$h(t) = \frac{\beta}{\lambda}\left(\frac{t}{\lambda}\right)^{\beta-1}$$



4.1.1.2 Weibull model (3-parameter)

Label in *failCompare*: weibull3

Model fitting package: Specific to *failCompare*

Parameter definitions: shape = β ; scale = λ ; thrsh = γ

Description:
Equivalent to 2-parameter Weibull, but with a threshold parameter (γ) which defines an initial "failure-free" interval and may shift the distribution in time. Reverts to 2-parameter Weibull when $\gamma = 0$ and to an exponential when $\beta = \gamma = 0$.

Density function:

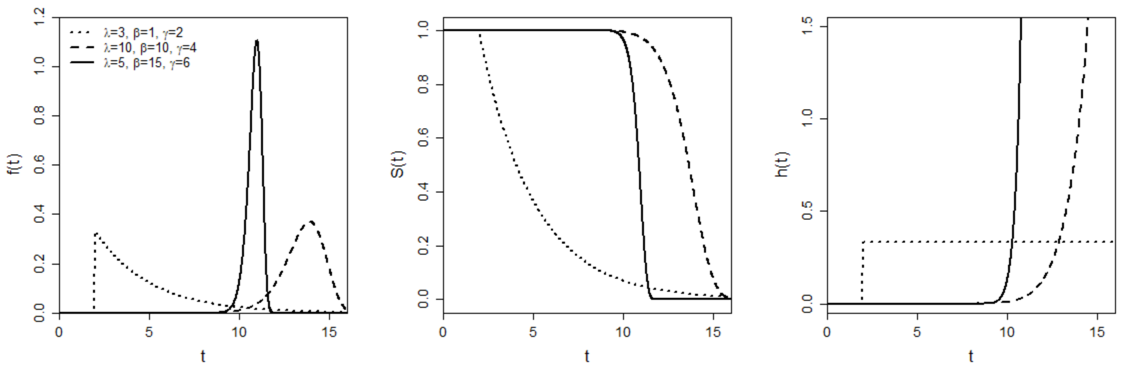
$$f(t) = \frac{\beta}{\lambda}\left(\frac{t-\gamma}{\lambda}\right)^{\beta-1} e^{-\left(\frac{t-\gamma}{\lambda}\right)^{\beta}}$$

Survival function:

$$S(t) = e^{-\left(\frac{t-\gamma}{\lambda}\right)^{\beta}}$$

Hazard function:

$$h(t) = \frac{\beta}{\lambda}\left(\frac{t-\gamma}{\lambda}\right)^{\beta-1}$$



4.1.1.3 Gompertz model

Label in *failCompare*: "gompertz"

Model fitting package: *flexsurv*

Parameter definitions: shape = a ; rate = b

Description:
Log-linear hazard function that allows more rapid acceleration of failures, relative to the 2- and 3-parameter Weibull. Reverts to exponential model when $a = 0$. Note that $a = e^{\gamma}$ in equation (9).

Density function

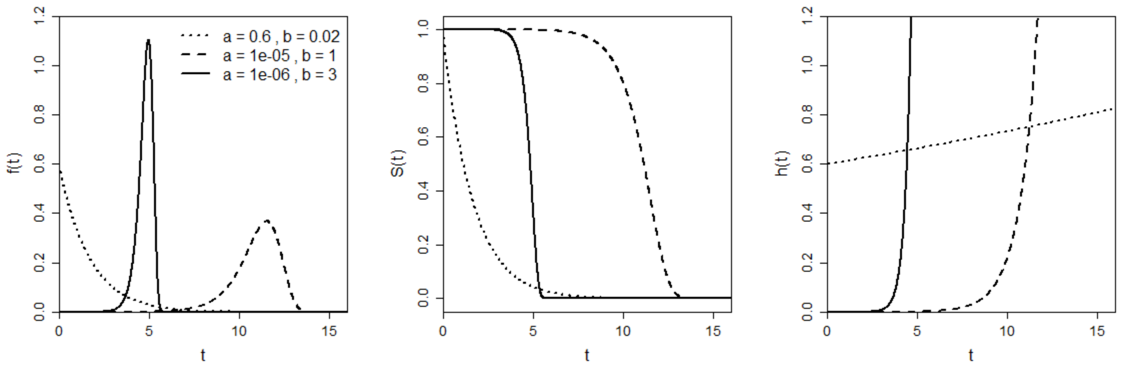
$$f(t) = ae^{bt} \cdot exp\left[(-a/b) \cdot \left(e^{bt} - 1\right)\right]$$

Survival function

$$S(t) = exp[(-a/b)(e^{bt} - 1)]$$

Hazard Function

$$h(t) = ae^{bt}$$



4.1.1.4 Log-normal model

Label in *failCompare*: "lognormal"

Model fitting package: *flexsurv*

Parameter definitions: meanlog = μ ; sdlog = σ

Description:
Hazard function that increases from the origin (0,0) and can reach a peak and decline at different rates.

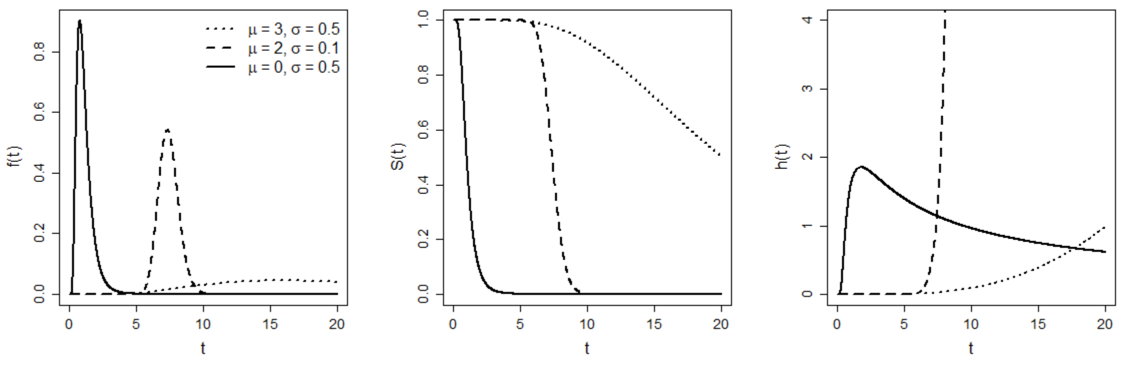
$$f(t) = \frac{1}{t\sigma\sqrt{2\pi}} e^{\left(-\frac{1}{2\sigma^2}[\log(t)-\mu]^2\right)}$$

Survival function

$$S(t) = \frac{1}{\sigma\sqrt{2\pi}} \int_t^\infty \frac{1}{t} e^{\left(-\frac{1}{2\sigma^2}[\log(t)-\mu]^2\right)} dt = 1 - \Phi[(log(t) - \mu)/\sigma]$$

Hazard Function
A simplified form of the hazard function is calculated by making the following substitution:
$a = e^{-\mu}$, which yields:

$$h(t) = \frac{\frac{1}{t\sigma\sqrt{2\pi}} e^{-\frac{log(at)^2}{2\sigma^2}}}{1 - \Phi(log(at/\sigma))}$$



4.1.1.5 Log-logistic model

Label in *failCompare*: "llogis"

Model fitting package: *flexsurv*

Parameter definitions: shape = a ; scale = b

Description:
Similar to the log-normal model, but with thicker tails in the density function. Benefit of having a closed-form for the three distribution functions.

Density function

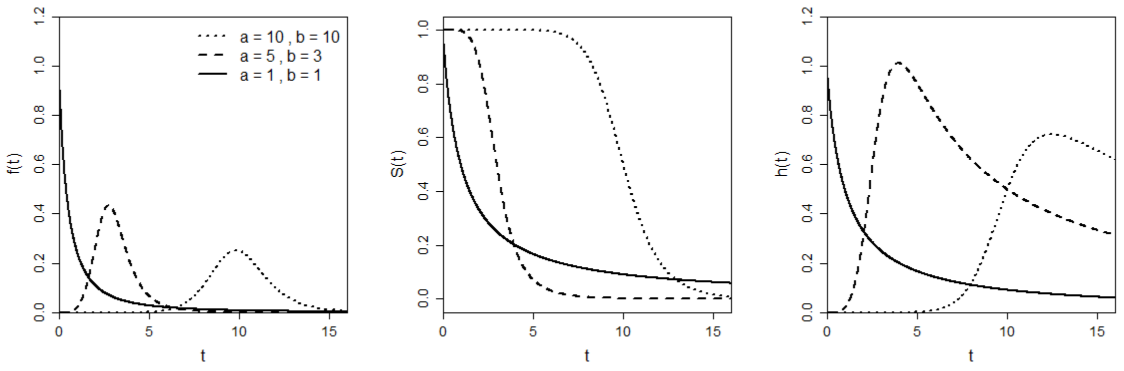
$$f(t) = \frac{(a/b)(t/b)^{a-1}}{[1 + (t/b)^a\ ]^2}$$

Survival function

$$S(t) = \frac{1}{1 + (t/b)^a}$$

Hazard function

$$h(t) = \frac{(a/b)(t/b)^{a-1}}{1 + (t/b)^a}$$



4.1.1.6 Gamma model (2-parameter)

Label in *failCompare*: "gamma"

Model fitting package: *flexsurv*

Parameter definitions: shape = a ; scale = s

<u>Description:</u>
Hazard function may decrease or increase to approach a particular value (e.g., three of the four hazard functions depicted below approach the value 1/2). This a and s parameterization below is consitent with the dgamma() documentation in the base R *stats* package. An alternate parameterization replaces scale with rate = $1/s$

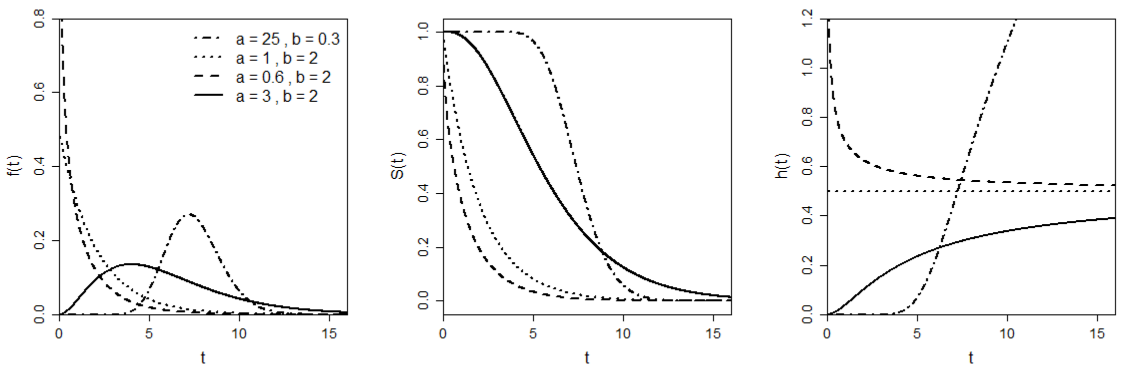
<u>Density function</u>

$$f(t) = \frac{t^{a-1}e^{-t/s}}{\Gamma(a)s^a}$$

<u>Survival function</u>

$$S(t) = 1 - \int_0^t \frac{t^{a-1}e^{-t/s}}{\Gamma(a)s^a} dt$$

<u>Hazard function</u>
No closed-form version exists.



4.1.1.7 Generalized gamma model (3-parameter)

Label in *failCompare*: "gengamma"

Model fitting package: *flexsurv*

Parameter definitions: mu = μ ; sigma = σ ; Q = Q

Description:
The additional parameter allows for greater kurtosis and can produce hazard functions that initially increase and then decrease and vice versa. This distribution can also take on a “bathtub-shaped” hazard function that is useful for characterizing populations with high risk of mortality early and late in life. *failCompare* implements the *flexsurv* version of the generalized gamma described by Prentice (1974) which estimates parameters using a log-gamma distribution with parameters: μ, σ, Q. Below we describe the distributions using the more recognizable parameterization with parameters a and s as in the 2-parameter gamma model above but with the additional parameter b.

Conversion from the dgamma() to Prentice (1974)

$$\mu = log(s) + \frac{log(a)}{b}$$

$$\sigma = \frac{1}{(b\sqrt{a})}$$

$$Q = \frac{1}{\sqrt{a}}$$

Conversion from Prentice (1974) to dgamma()

$$a = \left(\frac{1}{Q}\right)^2, b = \frac{Q}{\sigma}\ , s = exp(\mu - \frac{log(a)}{b})$$

Density function

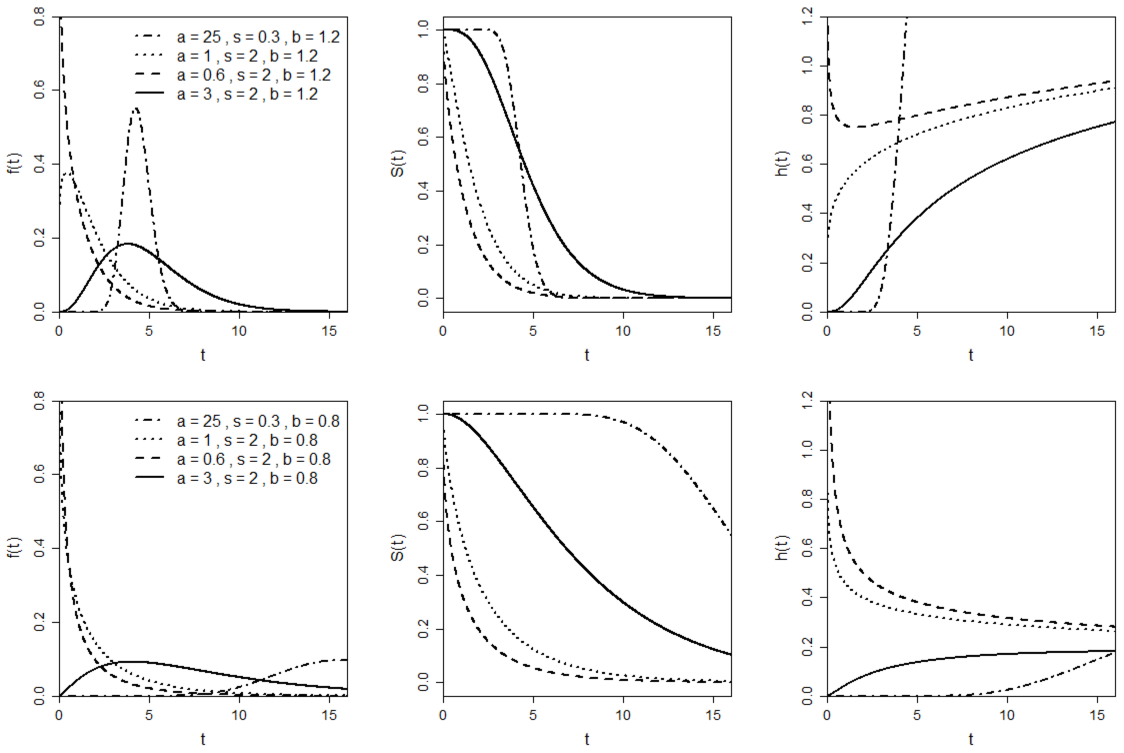
$$f(t) = \frac{bt^{ba-1}e^{-(t/s)^b}}{\Gamma(a)s^{ab}}$$

Survival function

$$S(t) = 1 - \int_0^t \frac{bt^{ba-1}e^{-(t/s)^b}}{\Gamma(a)s^{ab}}dt$$

Hazard function
No closed-form version exists.

Plots below are made with identical parameter values as the 2-parameter gamma distribution example above, which is a special cased of the generalized gamma with $b = 1$, but with the b parameter value increased (top row) and decreased (bottom row) by 0.2 to show this parameter’s effect on the shape of the functions.


a = 25 , s = 0.3 , b = 1.2
a = 1 , s = 2 , b = 1.2
a = 0.6 , s = 2 , b = 1.2
a = 3 , s = 2 , b = 1.2
a = 25 , s = 0.3 , b = 0.8
a = 1 , s = 2 , b = 0.8
a = 0.6 , s = 2 , b = 0.8
a = 3 , s = 2 , b = 0.8
f(t)
S(t)
h(t)
t

4.1.2 Vitality Models

The two vitality models in *failCompare* require greater elaboration compared to other models because they combine two survival processes (intrinsic and extrinsic). The Vitality models have "evolving" density functions produced by a stochastic model of vitality that declines over a lifetime along with extrinsic forces that can cause accidental mortality. Both models have four parameters, two in common with one another. The common parameter r describes the average rate of vitality loss; s describes the variability in the rate of loss among individuals. The Vitality 2009 model (Li and Anderson 2009) assumes an initial distribution of vitalities when $t = 0$, defined by parameter u, and an exponential model of extrinsic mortality acting on the entire population, defined by k. The Vitality 2013 model (Li and Anderson 2013) assumes an extrinsic failure process consisting of challenge events that occur throughout the lifetime of the population and which preferentially eliminate low vitality individuals. The frequency of challenges is defined by λ and the magnitude of these events by β.

4.1.2.1 Vitality 2009

Label in *failCompare*: "vitality.ku"

Model fitting package: *vitality*

Parameter definitions:
r = r ; s = s ; k = k ; u = u

intrinsic
r = rate of vitality loss, s = vitality spread over time, u = CV of initial vitality at $t = 0$
extrinsic
k = accidental failure rate

Description
Distinguishing features of the Vitality 2009 model are that three parameters (r, s, and u) define the intrinsic failure process, whereas only a single parameter (k) defines an independent extrinsic (accidental) morality rate. Below we denote the intrinsic and extrinsic survival components with subscripts I and E, respecively. The time to first passage of the zero boundary follows an inverse-Gaussian distribution, and the intrinsic survival function for the model is obtained by integrating the time to first passage over the values of a normal distribution of initial vitality surrounding 1.0:

Survival function (intrinsic)

$$S_I(t) = 1 - \int_{-\infty}^{\infty} f_I\,(t|v_0)p(v_0)\,dv_0$$

where $p(v_0)$ is the initial vitality distribution with a mean of 1.0 and coefficient of variation u. This function is simplified as:

$$S_I(t) = \left[\Phi\left(\frac{1-rt}{\sqrt{u^2+s^2t}}\right) - exp\left(\frac{2u^2r^2}{s^4} + \frac{2r}{s^2}\right)\Phi\left(-\frac{1+rt+\frac{2u^2r}{s^2}}{\sqrt{u^2+s^2t}}\right)\right]$$

where r is rate of vitality loss, s is the spread of the vitality loss, and u defines the coefficient of variation (CV) of the initial vitality distribution.

This underlying stochastic process produces a density function that evolves over time, beginning with a half-normal distribution with a vitality value near 1 then shifting to a normal distribution as vitality deceases. Over time, the distribution becomes more asymmetric resembling a gamma-like distribution with the probability density mass increasingly accumulating near the zero boundary. The intrinsic survival function of the Vitality 2009 model can be approximated integrating the initial vitality distribution and the stochastic Wiener process.

Evolving density function

$$f_I(v|t) = \frac{exp\left(-\frac{(v-1+rt)^2}{2(u^2+s^2t)}\right)\left[1-exp\left(-\frac{2v(ru^2+s^2)}{s^2(u^2+s^2t)}\right)\right]}{\sqrt{2\pi(u^2+s^2t)}}$$

where v is the relative density of the vitality curve at time t.

The extrinsic component of the model is much simpler, and is defined as an exponential survival distribution with parameter k (where $k = \frac{1}{\lambda}$ in equation 6):

Survival function (extrinsic)

$$S_E(t) = e^{-kt}$$

Survival function (combined)

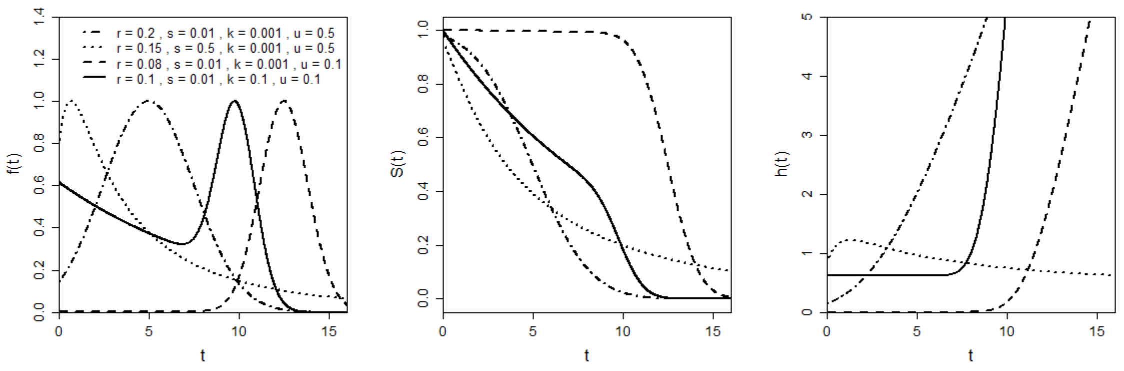
The combined survival function is the joint probability of having a vitality value greater than zero and not suffering extrinsic (accidental) mortality:

$$S(t) = S_i(t) \cdot S_E(t)$$

Hazard function

The hazard rates for the two processes are additive, and the exponential hazard rate is simply k, owing to the "memoryless" property of the extrinsic distribution.

$$h(t) = \frac{f_I(t)}{S(t)} + k$$


r = 0.2 , s = 0.01 , k = 0.001 , u = 0.5
r = 0.15 , s = 0.5 , k = 0.001 , u = 0.5
r = 0.08 , s = 0.01 , k = 0.001 , u = 0.1
r = 0.1 , s = 0.01 , k = 0.1 , u = 0.1
f(t)
S(t)
h(t)
t

4.1.2.2 Vitality 2013

Label in *failCompare*: "vitality.ku"

Model fitting package: *vitality*

Parameter definitions:
r = r ; s = s ; lambda = λ ; beta = β

intrinsic
r = rate of vitality loss, s = vitality spread over time
extrinsic
λ = frequency of random survival challenges, β = magnitude of challenges

Description
Parameters r and s correspond to the definitions given above for the Vitality 2009 model. A distinguishing characteristic of the 2013 model is that there are 2 parameters that characterize extrinsic mortality: the frequency of random survival challenges (λ) and the magnitude of challenges. Whether an individual dies given a challenge depends on its vitality value relative to the challenge magnitude. This has the effect of preferentially eliminating individuals that happen to have lower vitality at a given time. The challenge frequency is Poisson distributed with parameter λ, and the distribution of challenge magnitudes is exponentially distributed with scale parameter β (corresponding to λ in equation 6). Like the Vitality 2009 version, this model is extremely flexible and can capture a minority of earlier failures prior to the main decline. This model is uniquely able to fit failure time models seemingly without a right tail on the survival function.

Hazard function (Intrinsic)

$$h_I(t) = \frac{t^{-3/2} e^{(1-rt)^2/2s^2 t}}{s\sqrt{2\pi}\left(\Phi\left[\frac{1-rt}{s\sqrt{t}}\right] - e^{2r/s^2} \cdot \Phi\left[-\frac{1+rt}{s\sqrt{t}}\right]\right)}$$

Hazard function (Extrinsic)

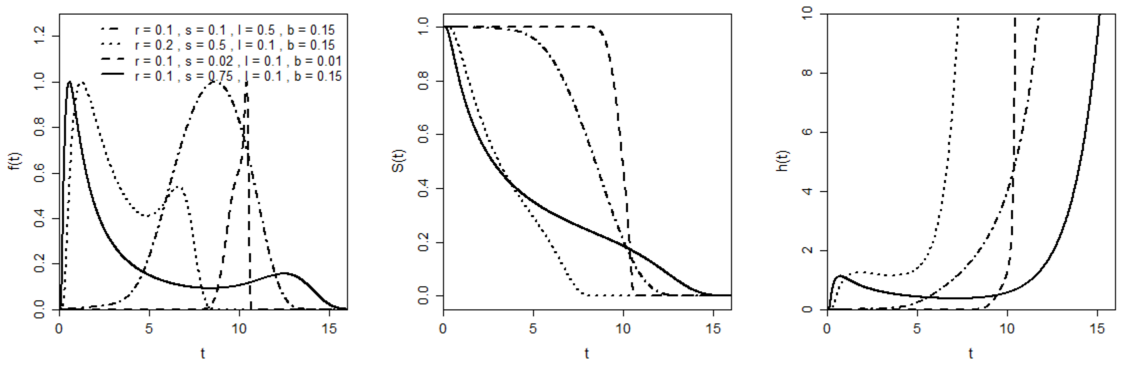
$$h_E(t) = \lambda\, e^{-(1-rt)/\beta}$$

Hazard function (Combined)

$$h(t) = h_I(t) + h_E(t)$$

Survival function (Combined)

$$S(t) = \left[\Phi\left(\frac{1-rt}{s\sqrt{t}}\right) - exp\left(\frac{2r}{s^2}\right)\Phi\left(-\frac{1+rt}{s\sqrt{t}}\right)\right] exp\left[\frac{\lambda\beta}{r} e^{-\frac{1}{\beta}}\left(e^{\frac{rt}{\beta}} - 1\right)\right]$$


r = 0.1 , s = 0.1 , l = 0.5 , b = 0.15
r = 0.2 , s = 0.5 , l = 0.1 , b = 0.15
r = 0.1 , s = 0.02 , l = 0.1 , b = 0.01
r = 0.1 , s = 0.75 , l = 0.1 , b = 0.15
f(t)
S(t)
h(t)
t
t
t

5 References

Anderson, J. J. 1992. A vitality-based stochastic model for organism survival. Pages 256–277 Individual-based models and approaches in ecology. Editor: DeAngelis, D.L. CRC press Boca Raton, FL.

Blischke, W. R., and D. N. P. Murthy. 2011. Reliability: Modeling, Prediction, and Optimization. John Wiley & Sons.

Burnham, K. P., and D. R. Anderson. 2007. Model selection and multimodel inference: a practical information-theoretic approach. Springer Science & Business Media.

Hosmer, D. W., S. Lemeshow, and S. May. 2008. Applied survival analysis: regression modeling of time-to-event data. John Wiley & Sons, Hoboken, N.J.

Jackson, C. H. 2016. flexsurv: a platform for parametric survival modeling in R. Journal of Statistical Software 70. Europe PMC Funders.

Kalbfleisch, J. D., and R. L. Prentice. 2011. The statistical analysis of failure time data. John Wiley & Sons.

Kaplan, E. L., and P. Meier. 1958. Nonparametric estimation from incomplete observations. Journal of the American statistical association 53(282):457–481. Taylor & Francis.

Lee, E. T., and J. Wang. 2003. Statistical methods for survival data analysis. John Wiley & Sons.

Leemis, L. M. 1995. Reliability: probabilistic models and statistical methods. Prentice Hall, Englewood Cliffs, N.J.

Li, T., and J. J. Anderson. 2009. The vitality model: A way to understand population survival and demographic heterogeneity. Theoretical Population Biology 76(2):118–131. Elsevier.

Li, T., and J. J. Anderson. 2013. Shaping human mortality patterns through intrinsic and extrinsic vitality processes. Demographic research 28:341–372.

Liang, H., and G. Zou. 2008. Improved AIC Selection Strategy for Survival Analysis. Computational statistics & data analysis 52(5):2538–2548.

Lilliefors, H. W. 1967. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. Journal of the American statistical Association 62(318):399–402. Taylor & Francis.

McCarthy, M. A., A. M. Gill, and R. A. Bradstock. 2001. Theoretical fire-interval distributions. International Journal of Wildland Fire 10(1):73.

Passolt, G., J. J. Anderson, T. Li, D. H. Salinger, and D. Sharrow J. 2018. vitality: Fitting Routines for the Vitality Family of Mortality Models.

Prentice, R. L. 1974. A Log Gamma Model and Its Maximum Likelihood Estimation. Biometrika 61(3):539–544.

Skalski, J. R., and S. L. Whitlock. 2020. Vitality models found useful in modeling tag-failure times in acoustic-tag survival studies. Animal Biotelemetry 8(1):1–10. BioMed Central.

Sokal, R. R., and F. J. Rohlf. 1995. Biometry, 3rd edition. W.H. Freeman and Company, New York.

Therneau, T. M., and P. M. Grambsch. 2000. Modeling Survival Data: Extending the Cox Model. Springer Science & Business Media.

Touchon, J. C., and M. W. McCoy. 2016. The mismatch between current statistical practice and doctoral training in ecology. Ecosphere 7(8):e01394.

Weitkamp, D. E., and M. Katz. 1980. A review of dissolved gas supersaturation literature. Transactions of the American Fisheries Society 109(6):659–702. Taylor & Francis.