# Failure Time Models

Prior to explaining the workings of the *failCompare p*ackage 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).

## Theoretical Basis

We consider time to failure to be a nonnegative random variable with an underlying probability distribution. This distribution can be transformed into a monotonically decreasing function from 1 to 0 defined over time as . In particular if the probability distribution of failure time has a continuous distribution , then the probability of failing prior to time is calculated based on the cumulative distribution function, :

|  |  |
| --- | --- |
|  | (1) |

The probability of not having failed by time is then:

|  |  |
| --- | --- |
|  | (2) |

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

|  |  |
| --- | --- |
|  | (3) |

The hazard function describes how prone a subject is to fail at time , 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.

## Failure Time Modeling in Practice

### Calculating the Sample Survival Function

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

|  |  |
| --- | --- |
|  | (4) |

where is the total number of observations and 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:

|  |  |
| --- | --- |
|  | (5) |

where  sample size,

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).

### 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 The simplest choice of failure time model is based on the exponential distribution with parameter and density function:

|  |  |
| --- | --- |
|  | (6) |

and therefore the survival distribution is

|  |  |
| --- | --- |
|  | (7) |

Interestingly, terms in the hazard rate cancel resulting in a constant hazard function defined by (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 .

Histogram

Description automatically generated

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

A more flexible alternative 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 and scale parameter is:

|  |  |
| --- | --- |
|  | (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). A special case is which is equivalent to the exponential model described above.

Chart, histogram

Description automatically generated

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

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 and :

|  |  |
| --- | --- |
|  | (9) |

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

|  |  |
| --- | --- |
|  | (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 latter having a convenient closed-form definition for its survival function. The hazard function of the 2-parameter gamma model approaches an asymptote as . 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). Both models have four parameters, two in common with one another. The common parameter describes the average rate of vitality loss; describes the variability in the rate of loss among individuals. The Vitality 2009 model assumes an initial distribution of vitalities when , defined by parameter () and an exponential model of extrinsic mortality acting on the entire population, defined by . The Vitality 2013 model 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 and the magnitude of these events by .

**Table 1.** *failCompare’*s *d*efault 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 |

## 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 (AICc; Liang and Zou 2008). Considering these potential complications and initial testing that confirmed the incompatibility of likelihood formulations between vitality and other models, we decided that AIC was not suitable for the variety of models that we wished to compare.

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 AICc. This goodness-of-fit (GOF) statistic is the average squared deviation between the Kaplan-Meier function and the fitted model values of across the *n* observed failure times (Figure 4):

|  |  |
| --- | --- |
|  | (13) |

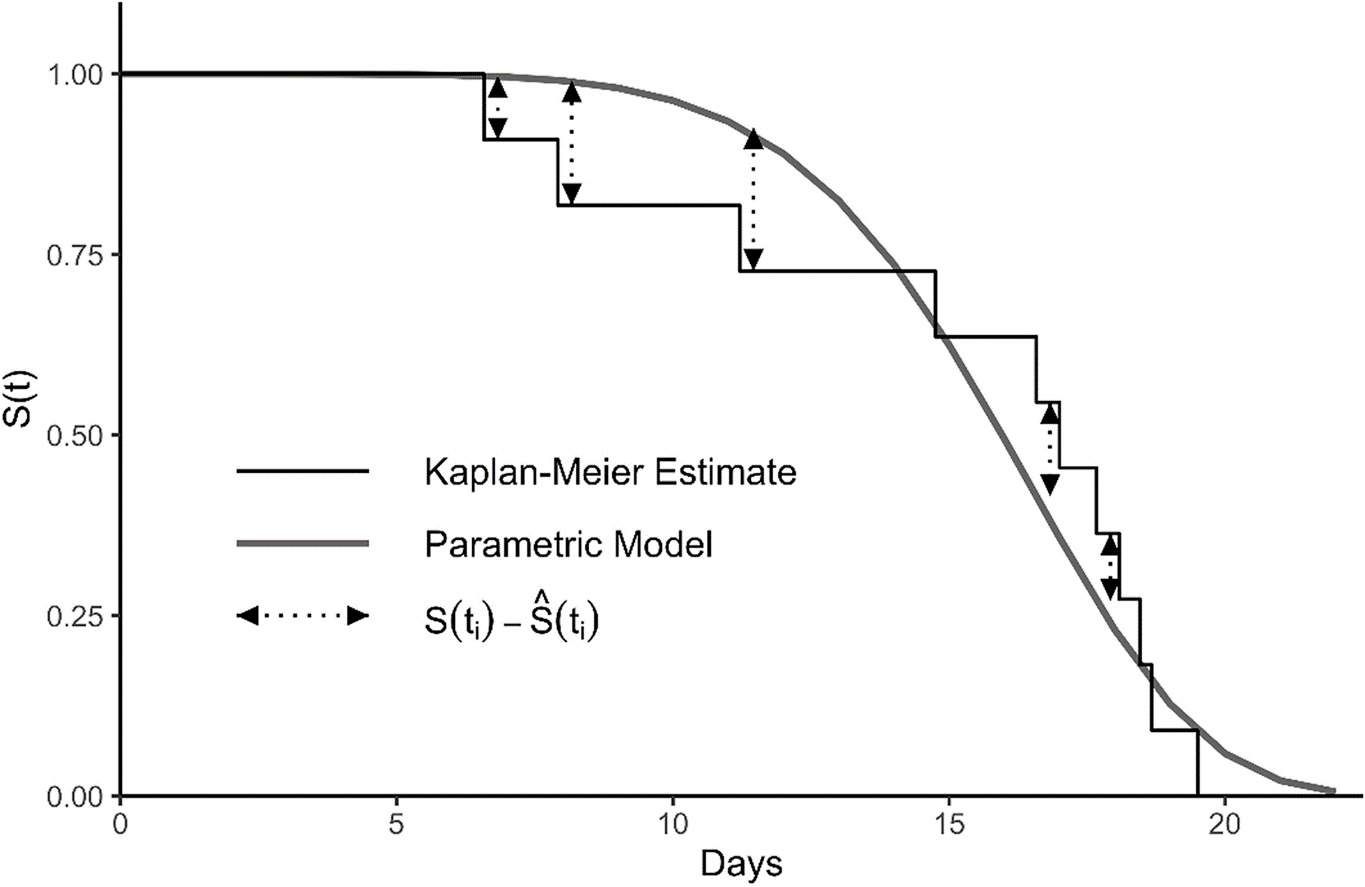
where survival value from parametric model at time *t* for the *i*th failure (,

K-M survival function estimate at time *t* for the *i*th failure (,

sample size (including censored observations)

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).

## 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 and anywhere along the fitted curve, i.e.,

|  |  |
| --- | --- |
| . | (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 (), and then repeatedly simulating datasets of length from the estimated distribution and calculating and storing simulated values to approximate the sampling distribution of the test statistic. After accumulating sufficiently large sample of values (e.g., 50,000), the P-value is computed as the proportion of simulated samples that exceed .

## Censored Data

In its current form, the package is equipped to handle two common forms of right-censored data. The two types of censoring accommodated by the software are (1) singly right censoring, where all subjects beyond a specific time-point are considered removed; and (2) progressively 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.

Chart

Description automatically generated

**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, with the dotted sections representing periods when an individual’s status was unobserved. Closed points at the end of lines represent observed failure times, and open points the time that at which 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 () for a model with observed failure times for all study subjects relies on the density function of the model’s distribution, and is simply:

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 type of information is:

Where is number of study subjects with observed failure times and 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

where denotes potentially differing times that which observations each of the observation were censored. It should be noted that this approach assumes an independent censoring mechanism.

# Using failCompare

….

## 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 that indicates if individual times describe when the subject failed or the when they were censored. See Example 3 below for a demonstration on how to account for these two forms of censoring using *failCompare*’s model-fitting and plottingfunctions.

…

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

We begin our description of *failCompare* using an example data set that comes with the package.

#### Loading data

Below we use the data() command loads the example data set of acoustic tag failure times called “sockeye”. The failure time of tags in days is the only variable in this data set, which we will store in the vector taglife.

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

Individual observations are displayed 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 |

#### Initial data visualization

We begin this example by computing the sample survival fraction and visualizing the data using basic R plotting tools. The code creates the following objects: n\_obs for storing the integer value of the number of rows in the data, and S\_t for storing the values of the sample survival function for each observation. Because there are censored observations in the data set we may compute the sample survival function using equation 4 with the function fc\_surv.

> 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.

#### Fitting a 2-parameter Weibull model

The first step is to use the function fc\_fit() with a 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 names corresponding to the nine model options 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 .

> plot(weib\_mod)

Chart

Description automatically generated

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)

Chart, diagram, histogram

Description automatically generated

The Weibull model appears to be a pretty good fit for the data, though 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.

#### Fitting a 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 rror of the 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)

Chart, line chart, histogram

Description automatically generated 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 next rank the performance of these two models using the GOF metric to help answer this question.

#### Creating a failure model 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

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)

Failure model list object  
  
Contains the following 2 models:   
 weibull ; vitality.ku  
  
\*use this object to compare models using the function: fc\_rank()

Chart

Description automatically generated

#### Ranking failure time models

Executing fc\_rank() prints a table ranking the models in the list, and creates a new model list object with the GOF ranking information. Printing fmods\_R displays the same information again.

> 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 stop here.

#### Ranking the full set of parametric models

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()

Executing fc\_rank 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 2 47 0.0011  
2 vitality.4p 0.05297191 50 3 46 0.0012  
3 gompertz 0.17255657 50 2 47 0.0037  
4 weibull 0.18241840 50 2 47 0.0039  
5 weibull3 0.18241771 50 3 46 0.0040  
6 gengamma 0.21987838 50 4 45 0.0049  
7 llogis 0.24616132 50 4 45 0.0055  
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 vitality.ku model beat out the 2-parameter Weibull model that we considered before.

Running ranked model list inside the plot() function displays up to three models at a time with the ranking in parentheses. 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")

Graphical user interface, chart, application, line chart

Description automatically generated

Graphical user interface, chart, application, line chart

Description automatically generated

**Figure 6.** The top ranked model all fit 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 “shoulder” of the curve, so it would appear to be the best choice.

## 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() concerns assessing the general lack-of-fit of a model, as opposed to a relative measure of model performance like the Skalski and Whitlock (2020) metric.

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() (with nothing inside of it). The example dataset for this task is titled “chinook”, and similar to example 1, it contains data describing the time until acoustic tag deactivation without any 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 ...

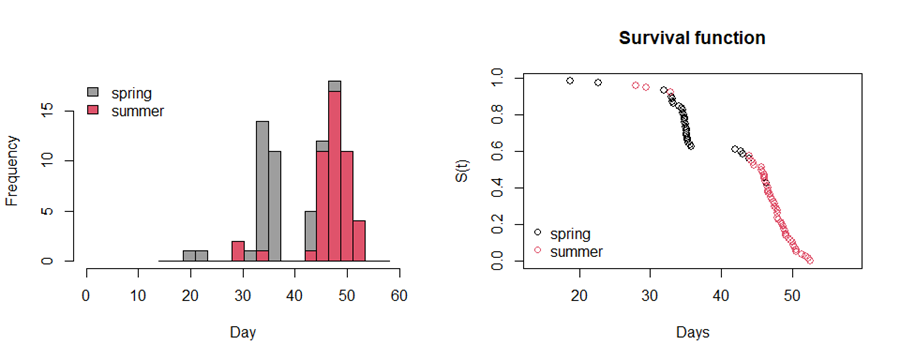
This reveals that we are working with a dataset with 83 observations. This time there is a second column identifying the group to which each tag belongs, a factor variable called “season” consisting of two groups: “spring” and “summer”. A frequency table for this second column further indicates that there are 33 tags in the spring group and 50 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 will visualize the data using a histogram and a plot. We enter additional argument for group that will distinguish 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?

### Log-rank test for comparing groups

Fortunately, *failCompare* includes function for performing a log-rank test, which tests the null hypothesis that two groups of observations share the same distribution. A log-rank test can be performed by providing the name of dataframe and specifying the failure time (time) and a categorical grouping variable (group). The test is carried out using the *survival* package, assuming a 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. The P-value for the test statistic is extremely small, indicating that there is significant evidence to reject the null hypothesis that the failure time distribution is the same for the two seasons.

We will now proceed by fitting separate models to the spring and summer groups, which we denote with the suffix \_SPR and \_SUM, respectively.

# Subsetting the groups  
> chn\_SPR=subset(chinook,season=="spring")  
> chn\_SUM=subset(chinook,season=="summer")

### 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 data set allows us to examine the quality of the fit. We can add a main title to each of the plots by providing character string for the optional argument main.

> 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); lognormal(7); llogis(8); gengamma(9)

Chart, histogram

Description automatically generated

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 as to the acceptability of the model, but it would be ideal to use a statistical test to confirm this.

### Kolmogorov-Smirnov test (simulation-based)

We will 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 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 should see a histogram of the sampling distribution of the test statistic D and 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.

> S\_chn\_SPR=fc\_surv(chn\_SPR$days)  
> fc\_test(times = chn\_SPR$days,model = "llogis",iters = 100000)

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

Chart, histogram

Description automatically generated

The P-value below 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, and it is reasonable to assume that there are no appropriate parametric models for describing the data, it may be best to avoid interpreting a parametric model and proceed with interpreting the nonparametric K-M model.

### Choosing 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().

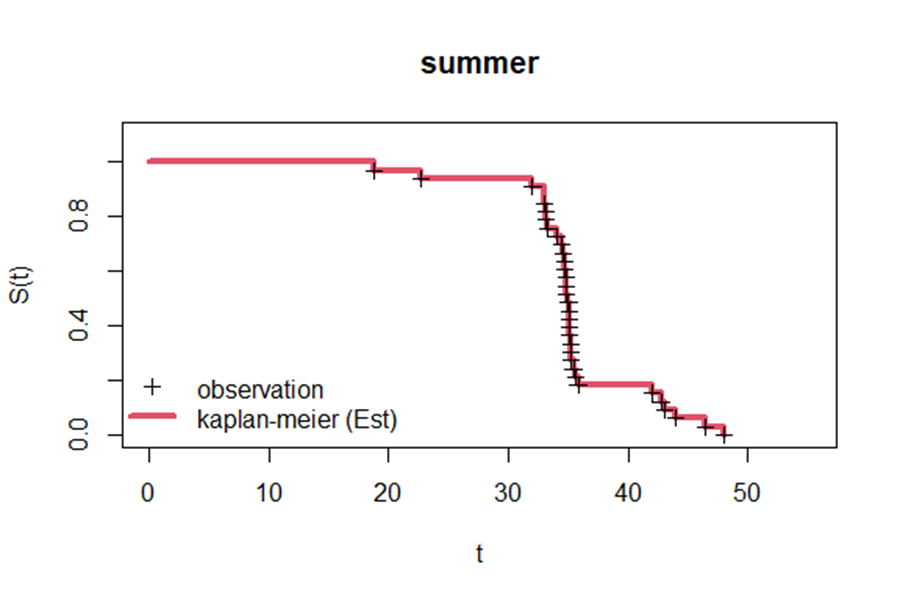
chnSPR\_KM=fc\_select(mod\_ls = chnSPR\_mods\_R,model = "kaplan-meier")

Printing the K-M model returns the estimates and confidence intervals for all the unique failure times 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="summer")



## 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 their 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” at some point in the study and their failure time could no longer be recorded (Type II censoring). Refer to the section above entitled “Censored Observations” for a more detailed explanation of these mechanisms and how they are dealt with during model estimation.

### 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 thirty days, at which point a portion of the study subjects were still alive.

We will load the data set in as before using data() as before. 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 vector of 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 will illustrate the consequences of failing to properly address censored observations. First, treating the censored observations as we would normal failure times ignores the fact that these subjects could have survived longer, and artificially lowers the height of survival function. The second option of simply omitting the censored fish would 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.

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).

Chart, scatter chart

Description automatically generated

Code for the left plot:

# survival function (treating censored observations regular data)  
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, this is used to indicate that the observations 30 are right-censored and not actual failure times.

Finally we will 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 3) and the GOF metric will be based on the distance between the K-M estimates and the parametric models’ survival function, both 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 ) are denoted with the grayed-out symbol at .

