## Example 1: Fitting a Failure-Time Model to Data

#### Loading failCompare package

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

library(failCompare)

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

#### Loading data

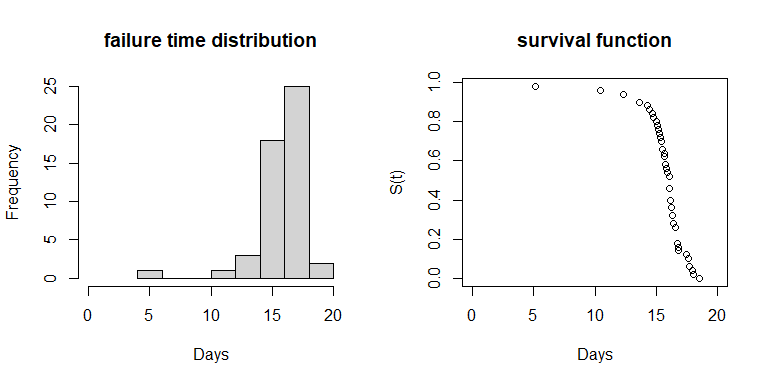
The data() command loads the example data set of acoustic tag failure times called “sockeye”.

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

#### Initial data visualization

Basic ploting features in R can be used to examine this distribution and compute the sample survival fraction. The following code creates an object storing the integer value of the number of rows in the data n\_obs. Because there are no censored observations in the data set we may compute the sample survival function . We then store the value in the object S\_t and plot the data using a histogram and scatterplot.

n\_obs=nrow(sockeye) # count number of observations  
  
S\_t=c() # empty vector  
for(i in 1:n\_obs){  
 n\_greater\_t=sum(taglife>taglife[i]) # temporarily store count of failures after t[i]  
 S\_t[i]=n\_greater\_t/n\_obs  
}  
  
# Histogram   
hist(x=taglife,  
 xlim=c(0,20), # scale set at zero  
 main="failure time distribution",xlab="Days")  
  
# Sample survival function  
plot(x=taglife,y=S\_t,  
 xlim=c(0,20), # scale set at zero  
 main="survival function",  
 xlab="Days",ylab="S(t)")

 There is one failure relatively early (5 days) in the study and a peak between 15-20 days. The gradual decrease in survival preceeding 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.

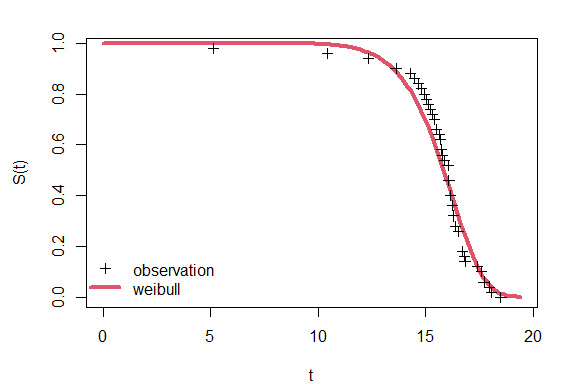
#### Fitting a 2-parameter Weibull model

The first step is to use the function fc\_fit() with the vector of failure times for the time argument and “weibull” for the model argument. Output is then stored in the object we have decided to label weib\_mod. The names corresponding to the nine model options are listed in the appendix of this report or can be found in the help documentation for the function (accessible by entering ?fc\_fit).

Entering weib\_mod into the console prints some of the output, including parameter estimates and standard errors. Placing the model object inside plot() function produces a scatterplot of the fitted Weibull model (red line) versus .

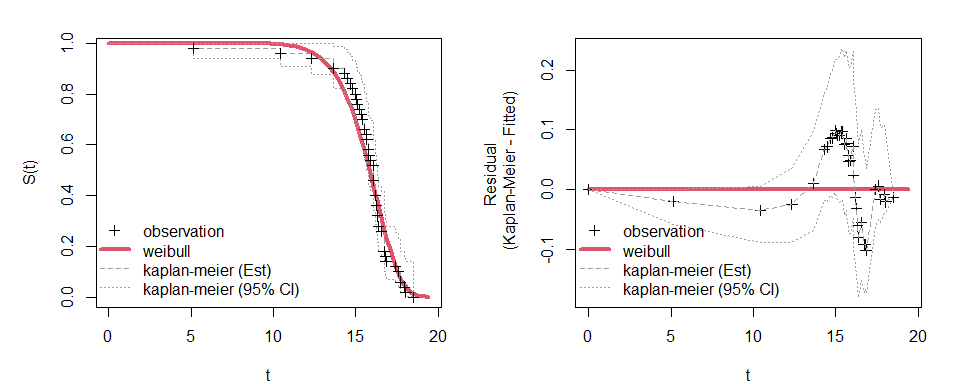
weib\_mod  
plot(weib\_mod)

> weibull failure model object   
>   
> Parameter estimates:  
> est se  
> shape 11.76181 1.3428043  
> scale 16.33821 0.2035392



Discontinuous lines for K-M estimates can be added by including the argument km=TRUE, and confidence intervals for the K-M model can also be added with the argument km.ci=TRUE. An alterntaive plot showing the the difference between the parametric model and the K-M model (“residual”) is created with the argument type="resid".

plot(weib\_mod,km = TRUE,km.ci = TRUE)  
plot(weib\_mod,type = "resid",km = TRUE,km.ci = TRUE)



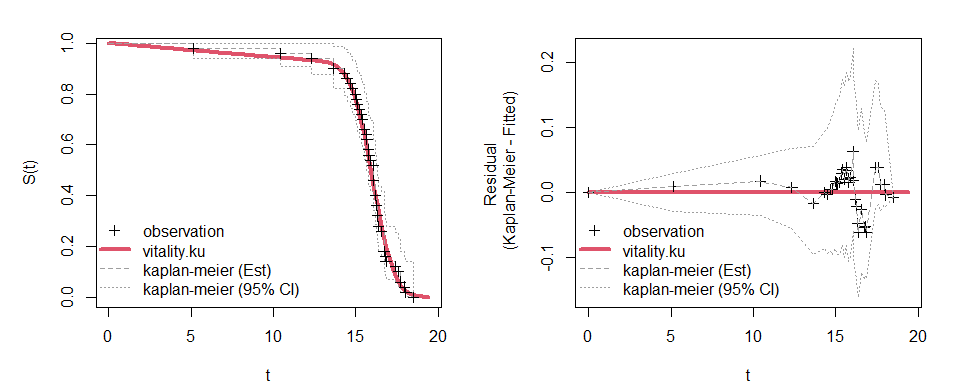
The Weibull model appears to be a pretty good fit for the data, but there may be a better alternative. Next we will try the Vitality 2009 model.

#### Fitting a Vitality 2009 model

This model is estimated in fc\_fit() in much the same way as before only with “vitality.ku” entered into the model argument

vit09\_mod=fc\_fit(time=taglife,model="vitality.ku")  
vit09\_mod  
plot(vit09\_mod,km = T,km.ci = T)  
plot(vit09\_mod,type = "resid",km = T,km.ci = T)

> vitality.ku failure model object   
>   
> Parameter estimates:  
> est se  
> r 6.2345e-02 0.0046264  
> s 5.0524e-08 NaN  
> k 5.7114e-03 0.0232010  
> u 6.4287e-02 0.0610020

 This model appears to conform to the data slightly better, particularly for earlier failure times. However, it is unclear whether the 4-parameter vitality model is more parsimonious. We will next rank the performance of these two models using the GOF metric.

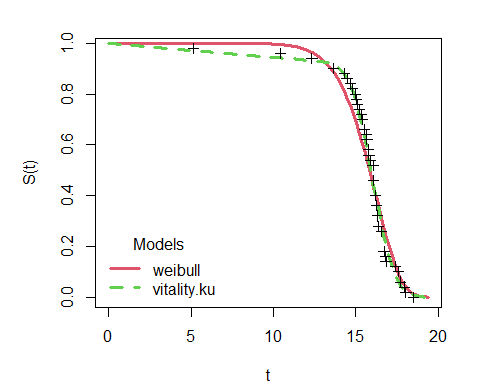
#### Ranking two alternative models

The models need to be placed in a list before they can be ranked. This can be done by rerunning fc\_fit and specifying a vector of more than one model name.

fmods=fc\_combine(list(weib\_mod,vit09\_mod))  
fmods  
  
fmods=fc\_fit(taglife,c("weibull","vitality.ku"))  
fmods  
  
plot(fmods)

printing fmods provides a brief description of the models in the list and placing the object inside summary() provides details on the parameter estimates for each model. Placing fmods inside plot() produces a survival function plot with the raw K-M estimates and the two models labeled. The argument km=T may also be added to see the horizontal lines connecting K-M estimates.

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



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

fmods\_ranked=fc\_rank(fmods)  
> Candidate models ranked by goodness of fit measure:  
>   
> model SSE\_KM n npars denom GOF  
> 1 vitality.ku 0.05014571 50 4 45 0.0011  
> 2 weibull 0.20326340 50 2 47 0.0043

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.

#### Fitting and ranking the full set of parametric models

A shortcut for fitting all nine parametric models in failCompare is to use enter “all” into the model argument.

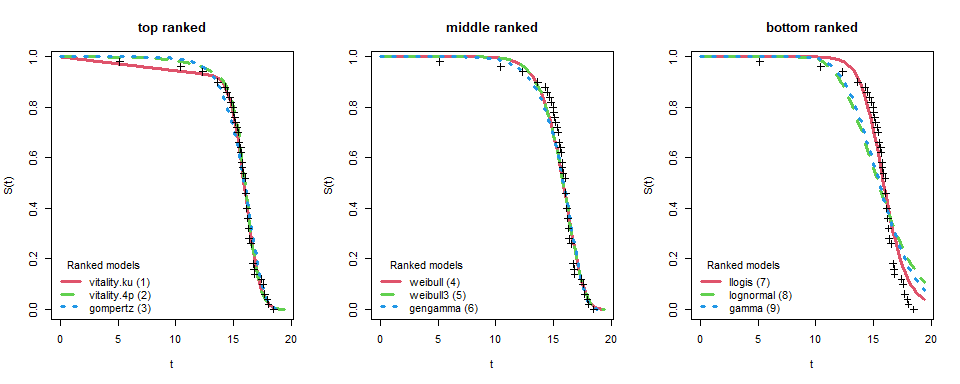
fmods\_all=fc\_fit(taglife,model="all")  
> Fitting all default parametric survival models  
fmods\_all  
> Failure model list object  
>   
> Contains the following 9 models:   
> weibull ; weibull3 ; gompertz ; gamma ; lognormal ; llogis ; gengamma ; vitality.ku ; vitality.4p  
>   
> \*this object can used to compare model fit using the function: fc\_rank()  
  
fmods\_all\_ranked=fc\_rank(fmods\_all)  
> Candidate models ranked by goodness of fit measure:  
>   
> model SSE\_KM n npars denom GOF  
> 1 vitality.ku 0.05014571 50 4 45 0.0011  
> 2 vitality.4p 0.05261863 50 4 45 0.0012  
> 3 gompertz 0.17543074 50 2 47 0.0037  
> 4 weibull 0.20326340 50 2 47 0.0043  
> 5 weibull3 0.20326526 50 3 46 0.0044  
> 6 gengamma 0.24224782 50 3 46 0.0053  
> 7 llogis 0.28433110 50 2 47 0.0060  
> 8 gamma 0.87624305 50 2 47 0.0186  
> 9 lognormal 1.07096950 50 2 47 0.0228

Printing the model list object tells us that all nine models were fit, and fc\_rank returns a table raning all nine models. The the top ranking model remains “vitatlity.ku” which corresponds to the Vitality 2009 model, but we can see that the vitality.4p model is ranked a close second, and that the gompertz 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 model names to the “model” arguement. Below is a default plot and a plot of the three models that fit the data the poorest, according to the GOF measure.

plot(fmods\_all\_ranked,main="top ranked")  
plot(fmods\_all\_ranked,model=c("vitality.ku","lognormal","gamma"),main="bottom ranked")

par(mfrow=c(1,3))  
plot(fmods\_all\_ranked,main="top ranked")  
plot(fmods\_all\_ranked,model=c("weibull","weibull3","gengamma"),main="middle ranked")  
plot(fmods\_all\_ranked,model=c("llogis","lognormal","gamma"),main="bottom ranked")



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 intial linear decline in survival that intersects with early failures in “shoulder” of the curve, so it would appear to be the best choice.

The very latest failCompare package can be download and install the package in one line if you already have the primary R developer tools installed.

vit09\_mod$par\_tab  
  
my\_seq=seq(0,20,0.01)  
  
plot(my\_seq,fc\_pred(my\_seq,vit09\_mod$par\_tab[,1],model = "vitality.ku"))  
lines(my\_seq,exp(-6.2345e-02\*my\_seq))  
par\_kset0=c(vit09\_mod$par\_tab[1:2,1],0,vit09\_mod$par\_tab[3,1])  
lines(my\_seq,fc\_pred(my\_seq,par\_kset0,model = "vitality.ku"))  
  
)