ReliabilitySupportFns

R.J.Marriott

2019 8 1

```
# Custom written functions to assist with Reliability analyses

# R.J. Marriott. 29 June 2016. (Version 2.0)

##################

# Calculate probability of failure:

Calc.Unreliability.w2p <- function(beta,eta,time){
  Unreliability <- (1-exp(-((time/eta)^beta)))
  return(Unreliability)
}

##################

# Calculate warranty time for target reliability:

Calc.Warranty.w2p <- function(beta,eta,Rval){
  time=eta*((-log(Rval))^(1/beta))
  return(time)
}

##################

# NormalDist.rrx method:

xmax <- function(x){
  # this function gets upper limit to x axis for the plot
  # to replicate the Weibull++ plot format.
  # x is the max of x
  x.max <- ceiling(x/(10^(nchar(as.character(x))-1))) *
    (10^(nchar(as.character(x))-1))
  return(x.max)
}

####

# Extract Expectation and variance from Weibull distribution:

Weibull.2p.Expectation <- function(eta,beta){
  # In R, eta=scale and beta=shape parameters
  # of built-in functions.
  Expectn <- eta * gamma(1 + 1/beta)
  return(Expectn)
}

Weibull.2p.Var <- function(eta,beta){
```

```
  # In R, eta=scale and beta=shape parameters
  # of built-in functions.
  b <- eta
  a <- beta
  Var.W2p <- b^2* (gamma(1 + 2/a) - (gamma(1 + 1/a))^2)
  return(Var.W2p)
}

###################

# For constructing Weibull plots:

F0inv    <- function (p) log(qweibull(p, 1, 1))
# This is a fancy way to give you the y axis scale by setting
# shape and scale parameters to 1, p= median rank.

Weibull.2p.plot <- function(x,y){
  # x = time; y = median ranks.
  ticks    <- c(seq(0.01,0.09,0.01),(1:9)/10,seq(0.91,0.99,0.01))
  xticks <- round(exp(seq(0.1,log(xmax(max(x))),
                          length.out=5)),0)
  y.trans <- F0inv(y)
  plot(x,y.trans,xlim=c(exp(0.1),xmax(max(x))),
       ylim=F0inv(c(0.01,0.99)),log="x",axes=F)
  axis(1,at=xticks)
  axis(2,at=F0inv(ticks),labels=ticks)
  abline(h=F0inv(ticks),col="lightgray")
}

###########

# Produce failure rate plot: 2-parameter Weibull.

failure.rate.w2p <- function(beta,eta,time){
  r <- (beta/eta) *
    (time/eta)^(beta-1)
  return(r)
}

hazard.plot.w2p <- function(beta,eta,time,line.colour,nincr=500){
  max.time <- max(time,na.rm=F)
  t <- seq(0,max.time,length.out=nincr)
  r <- numeric(length(t))
  for(i in 1:length(t)){
    r[i] <- failure.rate.w2p(beta,eta,t[i])
  }
  plot(t,r,type='l',bty='l',
       col=line.colour,lwd=2,
       main="",xlab="Time",
       ylab="Failure rate",
       las=1,adj=0.5,
       cex.axis=0.85,cex.lab=1.2)
}
```

```
##########

# Produce Reliability plot.

Reliability.w2p <- function(beta,eta,time){
  R <- exp(-(time/eta)^beta)
  return(R)
}

Reliability.plot.w2p <- function(beta,eta,time,line.colour,nincr=500){
  max.time <- max(time,na.rm=F)
  t <- seq(0,max.time,length.out=nincr)
  R <- numeric(length(t))
  for(i in 1:length(t)){
    R[i] <- Reliability.w2p(beta,eta,t[i])
  }
  plot(t,R,type='l',bty='l',
       col=line.colour,lwd=2,
       main="",xlab="Time",
       ylab="Reliability",
       las=1,adj=0.5,
       cex.lab=1.2,cex.axis=0.85)
}

####

Plot.Observations <- function(reliability.data,Ntotal=-999){
  # Specify Ntotal if plotting a subset of the data.
  dat <- reliability.data

  fail <- sort(dat[dat$event==1,"time"])
  # i.e., if there are censored observations:
  if(sum(dat$event)<nrow(dat)){
    cens <- sort(dat[dat$event==0,"time"])
    dat2 <- data.frame(fail=fail,cens=NA)
    fail.df <- data.frame(fail=NA,cens=cens)
    dat2 <- rbind(dat2,fail.df)
    time.index <- apply(dat2,1,sum,na.rm=T)
    dat3 <- dat2[order(time.index),]
    par(yaxt="n")
    barplot(t(as.matrix(dat3[nrow(dat3):1,])),
            beside=T,horiz=T,col=c("black","red"),border=NA,
            xlab="time") # as per Fig 1.5 in Meeker & Escobar.
    plot.dat <- barplot(t(as.matrix(dat3[nrow(dat3):1,])),
                        beside=T,horiz=T,border=NA,plot=F)
    par(yaxt="s")
    N.dataset <- ifelse(Ntotal==-999,nrow(dat),Ntotal)
    Start.y <- N.dataset - nrow(dat) + 1
    Difference <- abs(nrow(dat)-max(c(seq(1,nrow(dat3),by=5))))
    axis(2,at=plot.dat[1,c(seq(1,nrow(dat3),by=5))] + ceiling(plot.dat[2,Difference]),
         labels=rev(seq(Start.y,N.dataset,by=5)),
         las=1,adj=0.5,cex.axis=0.85)
  } else{
```

```
    barplot(t(as.matrix(fail[length(fail):1])),col="black",
            horiz=T,border=NA,
            xlab="time")
    plot.dat <- barplot(t(as.matrix(fail[length(fail):1])),
                        horiz=T,border=NA,plot=F)
    Difference <- abs(length(fail)-max(c(seq(1,length(fail),by=5))))
    axis(2,at=plot.dat[seq(1,length(fail),by=5)] + ceiling(plot.dat[2,Difference]),
         labels=rev(seq(1,length(fail),by=5)),
         las=1,adj=0.5,cex.axis=0.85)
  }
  legend("topright",
         legend=c("Failure","Suspension"),
         pch=15,
         col=c("black","red"),
         bty='n',
         horiz=F,xpd=NA,pt.cex=1.5)
  mtext("Time-ranked observation",side=2,line=2.5,adj=0.5)
}

###

Calculate.Fhat <- function(reliability.data){
  # reliability.data has columns "time" and "event"={1,0}
  dat1 <- reliability.data
  n <- nrow(dat1)
  dat1$suspensions <- 1 - dat1$event
  dat2 <- aggregate(dat1[,2:3],list(time=dat1$time),sum)
  names(dat2)[2:3] <- c("dj","rj")
  dat2$sum_dj <- cumsum(dat2$dj)
  dat2$sum_rj <- cumsum(dat2$rj)
  dat2$nj <- 0 # set empty numeric vector
  m <- nrow(dat2)
  attach(dat2)
  dat2$nj[1] <- n
  for(i in 2:m){
    dat2$nj[i] <- n - sum_dj[i-1] - sum_rj[i-1]
  }
  detach(dat2)
  dat2$pj <- dat2$dj / dat2$nj
  dat2$qj <- 1 - dat2$pj
  dat2$Shat <- cumprod(dat2$qj)
  dat2$Fhat <- 1 - dat2$Shat
  dat3 <- dat2[dat2$dj>0,colnames(dat2) %in% c("time","Fhat")==T]
  return(dat3)
}

Calculate.a_b <- function(reliability.data){
  # reliability.data has columns "time" and "event"={1,0}
  dat1 <- reliability.data
  n <- nrow(dat1)
  dat1$suspensions <- 1 - dat1$event
  dat2 <- aggregate(dat1[,2:3],list(time=dat1$time),sum)
  names(dat2)[2:3] <- c("dj","rj")
```

```
  dat2$sum_dj <- cumsum(dat2$dj)
  dat2$sum_rj <- cumsum(dat2$rj)
  dat2$nj <- 0 # set empty numeric vector
  m <- nrow(dat2)
  attach(dat2)
  dat2$nj[1] <- n
  for(i in 2:m){
    dat2$nj[i] <- n - sum_dj[i-1] - sum_rj[i-1]
  }
  detach(dat2)
  dat2$sigma_j <- dat2$dj / (dat2$nj*(dat2$nj - dat2$dj))
  cumsum.sigma <- cumsum(dat2$sigma_j)
  dat2$sigmahat <- 0 # set empty numeric vector
  for(i in 2:m){
    dat2$sigmahat[i] <- n * cumsum.sigma[i-1]
  }
  dat2$Khat <- dat2$sigmahat / (1 + dat2$sigmahat)
  dat3 <- dat2[,colnames(dat2) %in% c("time","sigmahat","Khat")==T]
  dat4 <- dat3[dat3$Khat> 0 & dat3$Khat <1, ] # Can't be zero or 1.
  a <- min(dat4$Khat); b <- max(dat4$Khat)
  result <- list(a=a, b=b)
  return(result)
}

Calculate.e_val <- function(ab.obj){
  a <- ab.obj$a; b <- ab.obj$b
  # e_alpha is a global object created in this file (lookup table from Meeker & Escobar)
  (e_val <- e_alpha[e_alpha$a==e_alpha$a[which.min(abs(a - e_alpha$a))] &
                      e_alpha$b==e_alpha$b[which.min(abs(b - e_alpha$b))],
                    "cl_0.95"])
  return(e_val)
}

Calc.95.simultaneous.CI <- function(reliability.data,e_val){
  print(c("Adjustments to 95 % simultaneous confidence bounds to account for"))
  print(c("non-increasing values follow method of Meeker & Escobar (1998)"))
  dat1 <- reliability.data
  n <- nrow(dat1)
  dat1$suspensions <- 1 - dat1$event
  dat2 <- aggregate(dat1[,2:3],list(time=dat1$time),sum)
  names(dat2)[2:3] <- c("dj","rj")
  dat2$sum_dj <- cumsum(dat2$dj)
  dat2$sum_rj <- cumsum(dat2$rj)
  dat2$nj <- 0 # set empty numeric vector
  m <- nrow(dat2)
  attach(dat2)
  dat2$nj[1] <- n
  for(i in 2:m){
    dat2$nj[i] <- n - sum_dj[i-1] - sum_rj[i-1]
  }
  detach(dat2)
  dat2$pj <- dat2$dj / dat2$nj
  dat2$qj <- 1 - dat2$pj
```

```
  dat2$Shat <- cumprod(dat2$qj)
  dat2$Fhat <- 1 - dat2$Shat
  dat3 <- dat2[dat2$dj>0,]
  dat3$se.summation <- dat3$pj / (dat3$nj*(1-dat3$pj))
  sum.term <- cumsum(dat3$se.summation)
  attach(dat3)
  dat3$se <- sqrt( (Shat)^2 * sum.term )
  detach(dat3)
  dat3$w <- exp((e_val*dat3$se) / (dat3$Fhat*(1-dat3$Fhat)))
  attach(dat3)
  dat3$loUnadj <- Fhat / (Fhat + (1-Fhat)*w)
  dat3$hiUnadj <- Fhat / (Fhat + (1-Fhat)/w)
  detach(dat3)
  dat4 <- dat3[is.nan(dat3$se)==F,]
  lo.NonDecreasing <- ifelse(sum(diff(dat4$loUnadj)<0)==0,"No","Yes")
  hi.NonDecreasing <- ifelse(sum(diff(dat4$hiUnadj)<0)==0,"No","Yes")
  if(lo.NonDecreasing=="Yes"){
    Max.lo <- max(dat3$loUnadj,na.rm=T)
    dat3$lo=dat3$loUnadj
    dat3$lo[which.max(dat3$loUnadj):length(dat3$loUnadj)] <- Max.lo
  }else{
    dat3$lo=dat3$loUnadj
  }
  if(hi.NonDecreasing=="Yes"){
    Min.hi <- min(dat3$hiUnadj,na.rm=T)
    dat3$hi=dat3$hiUnadj
    dat3$hi[1:which.min(dat3$hiUnadj)] <- Min.hi
  }else{
    dat3$hi=dat3$hiUnadj
  }
  return(dat3[dat3$Fhat>0 & dat3$Fhat <1,
              colnames(dat3)%in%c("time","Fhat","lo","hi")])
}

####

# Reference table for e_{a,b,1-alpha/2} factors from Table 3.5
# of Meeker & Escobar (1998)
e_alpha <- data.frame(
  a = c(rep(c(0.005,0.01,0.05,0.1),4)),
  b = c(rep(c(0.995,0.99,0.95,0.9),each=4)),
  cl_0.95 = c(3.36,3.34,3.28,3.25,
              3.34,3.31,3.25,3.21,
              3.28,3.25,3.16,3.11,
              3.25,3.21,3.11,3.06)
)
####

# Functions to produce probability plots in Step 1 of analysis:

Normal.probability.plot <- function(x,y,gridlines=F,
                                    label.individual.axes=T){
  # x = time; y = F(t).
```

```
  x <- x[y > 0 & y <1] # Can't be plotted on probability paper.
  y <- y[y > 0 & y <1]
  y.trans <- qnorm(y)
  yticks <- seq(round(min(y.trans),2),round(max(y.trans),2),
                length.out=5)
  xticks <- round(seq(0,xmax(max(x)),length.out=5),0)
  if(label.individual.axes==T){X.label <- "Time"
  } else{
    X.label <- ""
  }
  if(label.individual.axes==T){Y.label <- "Unreliability, F(t)=1-R(t)"
  } else{
    Y.label <- ""
  }
  plot(x,y.trans,xlim=c(0,xmax(max(x))),
       ylim=c(round(min(y.trans),2),round(max(y.trans),2)),
       axes=F,pch=16,adj=0.5,
       main="Normal",xlab=X.label,ylab=Y.label)
  axis(1,at=xticks)
  axis(2,at=yticks,labels=sprintf("%.2f",round(pnorm(yticks),2)),
       las=1,adj=0.5)
  if(gridlines==T){
    abline(h=yticks,col="lightgray")
    abline(v=xticks,col="lightgray")
  }
}

Lognormal.probability.plot <- function(x,y,gridlines=F,
                                       label.individual.axes=T){
  # x = time; y = F(t).
  x <- x[y > 0 & y <1] # Can't be plotted on probability paper.
  y <- y[y > 0 & y <1]
  x.trans <- log(x)
  y.trans <- qnorm(y)
  yticks <- seq(round(min(y.trans),2),round(max(y.trans),2),
                length.out=5)
  xticks <- round(seq(floor(min(x.trans)),ceiling(max(x.trans)),
                      length.out=5),0)
  if(label.individual.axes==T){X.label <- "Time"
  } else{
    X.label <- ""
  }
  if(label.individual.axes==T){Y.label <- "Unreliability, F(t)=1-R(t)"
  } else{
    Y.label <- ""
  }
  plot(x.trans,y.trans,xlim=c(floor(min(x.trans)),ceiling(max(x.trans))),
       ylim=c(round(min(y.trans),2),round(max(y.trans),2)),
       axes=F,pch=16,adj=0.5,
       main="Lognormal",xlab=X.label,ylab=Y.label)
  axis(1,at=xticks,labels=floor(exp(xticks)))
  axis(2,at=yticks,labels=sprintf("%.2f",round(pnorm(yticks),2)),
       las=1,adj=0.5)
```

```
  if(gridlines==T){
    abline(h=yticks,col="lightgray")
    abline(v=xticks,col="lightgray")
  }
}

add95CIs.Lognormal <- function(CL.data){
  # CL.data is the data frame generated from using Calc.95.simultaneous.CI()
  time <- CL.data$time
  lo <- CL.data$lo # the lower 95% simultaneous confidence limit
  hi <- CL.data$hi # the upper 95% simultaneous confidence limit
  points(log(time),qnorm(lo),pch="-",lwd=2,cex=1.2)
  points(log(time),qnorm(hi),pch="-",lwd=2,cex=1.2)
}

Weibull.backtrans.Y <- function(y){1-(1/exp(exp(y)))}

Weibull.probability.plot <- function(x,y,gridlines=F,
                                     label.individual.axes=T){
  # x = time; y = F(t).
  x <- x[y > 0 & y <1] # Can't be plotted on probability paper.
  y <- y[y > 0 & y <1]
  x.trans <- log(x)
  y.trans <- log(-log(1-y))
  yticks <- seq(round(min(y.trans),2),round(max(y.trans),2),
                length.out=5)
  xticks <- round(seq(floor(min(x.trans)),ceiling(max(x.trans)),
                      length.out=5),0)
  if(label.individual.axes==T){X.label <- "Time"
  } else{
    X.label <- ""
  }
  if(label.individual.axes==T){Y.label <- "Unreliability, F(t)=1-R(t)"
  } else{
    Y.label <- ""
  }
  plot(x.trans,y.trans,xlim=c(floor(min(x.trans)),ceiling(max(x.trans))),
       ylim=c(round(min(y.trans),2),round(max(y.trans),2)),
       axes=F,pch=16,adj=0.5,
       main="Weibull",xlab=X.label, ylab=Y.label)
  axis(1,at=xticks, labels=round(exp(xticks),0))
  axis(2,at=yticks,labels=sprintf("%.2f",round(Weibull.backtrans.Y(yticks),2)),
       las=1,adj=0.5)
  if(gridlines==T){
    abline(h=yticks,col="lightgray")
    abline(v=xticks,col="lightgray")
  }
}

add95CIs.Weibull <- function(CL.data){
  # CL.data is the data frame generated from using Calc.95.simultaneous.CI()
  time <- CL.data$time
  lo <- CL.data$lo # the lower 95% simultaneous confidence limit
```

```
  hi <- CL.data$hi # the upper 95% simultaneous confidence limit
  points(log(time),log(-log(1-lo)),pch="-",lwd=2,cex=1.2)
  points(log(time),log(-log(1-hi)),pch="-",lwd=2,cex=1.2)
}

Exponential.backtrans.Y <- function(y){1-(1/exp(y))}

Exponential.probability.plot <- function(x,y,gridlines=F,
                                         label.individual.axes=T){
  # x = time; y = F(t).
  x <- x[y > 0 & y <1] # Can't be plotted on probability paper.
  y <- y[y > 0 & y <1]
  y.trans <- -log(1-y)
  yticks <- seq(round(min(y.trans),2),
                round(max(y.trans),2),
                length.out=5)
  xticks <- round(seq(0,xmax(max(x)),length.out=5),0)
  if(label.individual.axes==T){X.label <- "Time"
  } else{
    X.label <- ""
  }
  if(label.individual.axes==T){Y.label <- "Unreliability, F(t)=1-R(t)"
  } else{
    Y.label <- ""
  }
  plot(x,y.trans,xlim=c(0,xmax(max(x))),
       ylim=c(round(min(y.trans),2),
              round(max(y.trans),2)),
       axes=F,pch=16,adj=0.5,
       main="Exponential",xlab = X.label, ylab = Y.label)
  axis(1,at=xticks)
  axis(2,at=yticks,labels=sprintf("%.2f",round(Exponential.backtrans.Y(yticks),2)),
       las=1,adj=0.5)
  if(gridlines==T){
    abline(h=yticks,col="lightgray")
    abline(v=xticks,col="lightgray")
  }
}

add95CIs.Exponential <- function(CL.data){
  # CL.data is the data frame generated from using Calc.95.simultaneous.CI()
  time <- CL.data$time
  lo <- CL.data$lo # the lower 95% simultaneous confidence limit
  hi <- CL.data$hi # the upper 95% simultaneous confidence limit
  points(time,-log(1-lo),pch="-",lwd=2,cex=1.2)
  points(time,-log(1-hi),pch="-",lwd=2,cex=1.2)
}

Probability.Plots <- function(reliability.data,gridlines=F,
                              label.individual.axes=F,dist="All"){
  dat <- reliability.data
  Fhat <- Calculate.Fhat(dat) # $time, $Fhat
  e_val <- Calculate.e_val(Calculate.a_b(dat))
```

```
  simult.CLs <- Calc.95.simultaneous.CI(dat,e_val=e_val) # $time, $lo, $hi
  if(dist=="All"){
    par(mfrow=c(2,2), mar=c(3, 3, 1, 0.25) + 0.1,cex.axis=0.75,
        oma=c(0,0,0,0),mgp=c(3,1,0),xpd=T,mai=c(0.75,0.75,0.2,0.2))
    # Plot for Weibull distbn.
    Weibull.probability.plot(Fhat$time,Fhat$Fhat,gridlines,
                             label.individual.axes)
    add95CIs.Weibull(simult.CLs)
    # Plot for Lognormal distbn.
    Lognormal.probability.plot(Fhat$time,Fhat$Fhat,gridlines,
                               label.individual.axes)
    add95CIs.Lognormal(simult.CLs)
    # Plot for Normal distbn.
    Normal.probability.plot(Fhat$time,Fhat$Fhat,gridlines,
                            label.individual.axes)
    points(simult.CLs$time,qnorm(simult.CLs$lo),pch="-",lwd=2,cex=1.2)
    points(simult.CLs$time,qnorm(simult.CLs$hi),pch="-",lwd=2,cex=1.2)
    # Plot for Exponential cdf.
    Exponential.probability.plot(Fhat$time,Fhat$Fhat,gridlines,
                                 label.individual.axes)
    add95CIs.Exponential(simult.CLs)
    mtext("Time",side=1,line=-1.5,adj=0.5,outer=T,font=2)
    mtext("Unreliability, F(t)=1-R(t)",side=2,line=-1.5,adj=0.5,
          outer=T,font=2)
  }
  else if(dist=="Weibull"){
    par(mar= c(5, 5, 4, 1) + 0.1,font.lab=2,cex.axis=0.8,cex.lab=1.1,
        cex.main=1.3)
    Weibull.probability.plot(Fhat$time,Fhat$Fhat,gridlines,T)
    add95CIs.Weibull(simult.CLs)
  }
  else if(dist=="Normal"){
    par(mar= c(5, 5, 4, 1) + 0.1,font.lab=2,cex.axis=0.8,cex.lab=1.1,
        cex.main=1.3)
    Normal.probability.plot(Fhat$time,Fhat$Fhat,gridlines,T)
    points(simult.CLs$time,qnorm(simult.CLs$lo),pch="-",lwd=2,cex=1.2)
    points(simult.CLs$time,qnorm(simult.CLs$hi),pch="-",lwd=2,cex=1.2)
  }
  else if(dist=="Lognormal"){
    par(mar= c(5, 5, 4, 1) + 0.1,font.lab=2,cex.axis=0.8,cex.lab=1.1,
        cex.main=1.3)
    Lognormal.probability.plot(Fhat$time,Fhat$Fhat,gridlines,T)
    add95CIs.Lognormal(simult.CLs)
  }
  else{
    par(mar= c(5, 5, 4, 1) + 0.1,font.lab=2,cex.axis=0.8,cex.lab=1.1,
        cex.main=1.3)
    Exponential.probability.plot(Fhat$time,Fhat$Fhat,gridlines,T)
    add95CIs.Exponential(simult.CLs)
  }
  par(mfrow=c(1,1),mar= c(5, 4, 4, 2) + 0.1,cex.axis=1,xpd=F,
      oma=c(0,0,0,0),lheight=1,mgp=c(3,1,0),mai=c(1.02,0.82,0.82,0.42)) # return defaults.
}
```

```
###

# Bias-adjusted non-parametric bootstrapping to estimate approx 95% CIs for MTTF.

# "Bias-corrected percentile" method e.g., Section 13.7, Meeker & Escobar.

# NB. Jeng and Meeker (1998 in Meeker & Escobar 1998) demonstrated that
# simulation-based methods, for most situations, provide important
# improvements over normal-approximation methods.

MTTF.boot.percentile.adj <- function(data,i){
  d <- data[i,]
  mod <- Lifedata.MLE(Surv(time,event)~1,
                      d,
                      dist="weibull")
  beta.b <- 1/unname(exp(mod$coef[2]))
  eta.b <- unname(exp(mod$coef[1]))
  MTTF <- Weibull.2p.Expectation(eta=eta.b,
                                 beta=beta.b)
  return(MTTF)
}

###

# For calculating joint confidence region:

sev.pdf <- function(z){exp(z - exp(z))}
sev.cdf <- function(z){1-exp(-exp(z))}

loglik.sev <- function(data,mu,sigma){
  t <- data$time
  ll.vec <- numeric(nrow(data))
  delta <- data$event
  t.lnorm <- (log(t)-mu)/sigma
  for(i in 1:nrow(data)){
    ll.vec[i] <- (delta[i] * log(1 / (sigma*t[i]))) +
      (delta[i] *
         log(sev.pdf(t.lnorm[i]))) +
      ((1-delta[i]) *
         log(1 - sev.cdf(t.lnorm[i])))
  }
  loglik <- sum(ll.vec)
  return(loglik)
}

contour.val <- function(dataset,mu.input,sigma.input,maximum.loglik){
  pchisq(q=-2*(loglik.sev(dataset,mu.input,sigma.input) -
                 maximum.loglik),df = 2)
}

Get.contour.plot.data <- function(data,fitted.2parameter.Weibull.model,steps){
  mod <- fitted.2parameter.Weibull.model # 2 parameter Weibull model fitted using SPREDA
  mu.MLE <- unname(mod$coef[1])
```

```
  sigma.MLE <- unname(exp(mod$coef[2]))
  Max.loglik <- loglik.sev(data,mu=mu.MLE,sigma=sigma.MLE)
  # Use 95% CIs for MLEs to determine input range for parameters.
  beta.95cl_hi <- 1 / (summary(mod)$coefmat["sigma","95% Lower"])
  beta.95cl_lo <- 1 / (summary(mod)$coefmat["sigma","95% Upper"])
  eta.95cl_lo <- exp(summary(mod)$coefmat["(Intercept)","95% Lower"])
  eta.95cl_hi <- exp(summary(mod)$coefmat["(Intercept)","95% Upper"])
  # Input ranges (steps)
  Betas <- seq(0.9*beta.95cl_lo,1.1*beta.95cl_hi,length.out=steps)
  Etas <- seq(0.9*eta.95cl_lo,1.1*eta.95cl_hi,length.out=steps)
  Sigmas <- 1 / Betas
  Mus <- log(Etas)
  ContourVals.df <- data.frame(Mus=rep(Mus,steps),
                               Sigmas=rep(Sigmas,each=steps))
  for(i in 1:nrow(ContourVals.df)){
    ContourVals.df$z[i] <- contour.val(data,ContourVals.df$Mus[i],
                                       ContourVals.df$Sigmas[i],Max.loglik)
  }
  ContourVals.df$Eta <- exp(ContourVals.df$Mus)
  ContourVals.df$Beta <- 1 / ContourVals.df$Sigmas
  return(ContourVals.df)
  # z vector is the probability that Chi-square stat with 2d.f. is less than
  # or equal to -2log likelihood ratio, for each input pair of Betas,Etas. Uses
  # the large sample Chi-square approximation for the distribution of the log-
  # likelihood ratio statistic.
}

Weibull.Confidence.Region <- function(data,model,probability,
                                      title,show.contour.labels,
                                      steps=100,html="No"){
  # requires lattice package.
  # Implemented for the fit of the 2-parameter Weibull model only.
  beta.MLE <- 1 / unname(exp(model$coef[2]))
  eta.MLE <- unname(exp(model$coef[1]))
  reliability.data <- data
  fitted.Weibull.model <- model
  label.show <- as.logical(show.contour.labels)
  Contour.df <- Get.contour.plot.data(reliability.data,
                                      fitted.Weibull.model,steps)
  if(html=="Yes"){ # this right align is not consistent for html generation.
    title.settings <- list(
      par.main.text = list(font = 2,
                           just = "right",
                           x = grid::unit(170, "mm")))
  } else{ # Use default settings.
    title.settings <- list(par.main.text=trellis.par.get("par.main.text"))
  }
  contourplot(z ~ Eta * Beta, data=Contour.df,
              at = probability, labels=label.show,
              panel=function(data, ...){
                panel.contourplot(...)
                panel.points(x=eta.MLE,y=beta.MLE,pch=19,col='black')
              },
```

```
              par.settings=title.settings,
              main=title)
}

###########

# Some functions used for internal referencing in R markdown:
# chunkref <- local({
# function(chunklabel) {
# sprintf('[%s](#%s)', chunklabel, chunklabel )
# }
# })

# secref <- local({
# function(seclabel) {
# sprintf('[%s](#%s)', seclabel, seclabel )
# }
# })

# pgref <- local({
# function(n)
# sprintf('[Page-%i](#Page-%i)', n, n)
# })

# sec <- local({
# function(seclabel) {
# sprintf('# <a name="%s"/> %s', seclabel, seclabel )
# }
# })

# pgcount <- local({
# pg <- 0
# function(inc=T) {
# if( inc ) { pg <<- pg + 1 }
# return( pg )
# }
# })

# pganchor <- local({
# function(doLabel=T) {
# if( doLabel) {
# sprintf('\n-----\nPage-%i\n<a name="Page-%i"/>\n', pgcount(inc=F), pgcount() )
# } else {
# sprintf('\n<a name="Page-%i"/>\n', pgcount() )
# }
# }
# })

# knit_hooks$set( anchor = function(before, options, envir) {
# if ( before ) {
# sprintf('<a name="%s"/>\n', options$label )
# }
# })
```

```
# knit_hooks$set( echo.label = function(before, options, envir) {
# if ( before ) {
# sprintf('> %s', options$label )
# }
# })

# knit_hooks$set( pgbreak = function(before, options, envir) {
# if ( !before ) {
# pganchor();
# }
# })
```