hSSALT Rpackage: simple heterogeneous SSALT with exponential life distribution based on CE assumption

Yao Lu

November 27, 2023

Contents

1 Set-up

Consider a simple step-stress lifetime test with only two stress levels ($x_1 < x_2$). n identical units are placed under the experimental condition.
Censoring types:

- Type-I: the stress level changes from x_1 to x_2 at a pre-specified time-point τ_1. The experiment continues until a pre-fixed time-point τ_2 (Time censoring).
- Type-II: the stress level changes from x_1 to x_2 at a pre-specified time-point τ. The experiment continues until r (usually $r < n$) failures are observed (Failure censoring).

Model assumptions:

- Assumption 1: Units behave homogeneously under the first stress level, but differ into two subgroups under the second stress level. That is to say, heterogeneity is observed.
- Assumption 2: Lifetime under x_1 and x_2 follows an exponential distribution with parameter $\theta_1, \theta_{2j} > 0,\ j = 1, 2$. Failures are observed continuously (exponential distribution, Type-I or Type-II censoring) or internally (geometric distribution, only with Type-I censoring).
- Assumption 3: Data come from a cumulative exposure (CE) model. The CE model ensures the continuity of the distribution function in the two stress levels. It assumes that the remaining life depends only on the current cumulative exposure the units have experienced, with no memory on how this exposure was accumulated.

The CDF of the lifetime of a hSSALT model with exponential distribution is given by:

$$G^*(t) = \begin{cases} 1 - e^{-t/\theta_1}, & t < \tau \\ 1 - pe^{-(t-\tau)/\theta_{21} - \tau/\theta_1} - (1-p)e^{-(t-\tau)/\theta_{22} - \tau/\theta_1}, & t \geqslant \tau \end{cases} \tag{1}$$

The PDF is given by

$$g^*(t) = \begin{cases} \frac{1}{\theta_1} e^{-t/\theta_1}, & t < \tau \\ p\frac{1}{\theta_{21}} e^{-(t-\tau)/\theta_{21} - \tau/\theta_1} + (1-p)\frac{1}{\theta_{22}} e^{-(t-\tau)/\theta_{22} - \tau/\theta_1}, & t \geqslant \tau \end{cases} \tag{2}$$

Let $t_{1:n} < t_{2:n} < \cdots < t_{n:n}$ denote the ordered failure times. The log-likelihood in the continuous scenario:
Type-I censoring:

$$\begin{aligned} l(\theta_1, p, \theta_{21}, \theta_{22} \mid \boldsymbol{t}) \propto -n_1 \log\theta_1 - \frac{\sum_{i=1}^{n_1} t_{i:n}}{\theta_1} - \frac{(n-n_1)\tau_1}{\theta_1} + \\ \sum_{i=n_1+1}^{N} \log\left(\frac{p}{\theta_{21}} e^{-(t_{i:n}-\tau_1)/\theta_{21}} + \frac{1-p}{\theta_{22}} e^{-(t_{i:n}-\tau_1)/\theta_{22}}\right) + \\ (n-N)\log(pe^{-(\tau_2-\tau_1)/\theta_{21}} + (1-p)e^{-(\tau_2-\tau_1)/\theta_{22}}) \end{aligned} \tag{3}$$

Type-II censoring:

$$\begin{aligned} l(\theta_1, \theta_{21}, \theta_{22}, p \mid \boldsymbol{t}) \propto -n_1 \log\theta_1 - \frac{\sum_{i=1}^{n_1} t_{i:n}}{\theta_1} - \frac{(n-n_1)\tau}{\theta_1} + \\ \sum_{i=n_1+1}^{r} \log\left(p\frac{1}{\theta_{21}} e^{-(t_{i:n}-\tau)/\theta_{21}} + (1-p)\frac{1}{\theta_{22}} e^{-(t_{i:n}-\tau)/\theta_{22}}\right) + \\ (n-r)\log(pe^{-(t_{r:n}-\tau)/\theta_{21}} + (1-p)e^{-(t_{r:n}-\tau)/\theta_{22}}) \end{aligned} \tag{4}$$

The log-likelihood in the interval scenario:
Type-I censoring:

$$\begin{aligned} l(\boldsymbol{n}) \propto \sum_{j=1}^{q_1} n_{1j} \log\left(e^{-\frac{\tau_{1,j-1}}{\theta_1}} - e^{-\frac{\tau_{1,j}}{\theta_1}}\right) + \sum_{j=1}^{q_2} \log\left[p\left(e^{-\frac{\tau_{2,j-1}-\tau_1}{\theta_{21}}} - e^{-\frac{\tau_{2,j}-\tau_1}{\theta_{21}}}\right) + (1-p)\left(e^{-\frac{\tau_{2,j-1}-\tau_1}{\theta_{22}}} - e^{-\frac{\tau_{2,j}-\tau_1}{\theta_{22}}}\right)\right] \\ - \frac{\tau_1(n-n_1)}{\theta_1} + (n-n_f)\log\left(pe^{-\frac{\tau_2-\tau_1}{\theta_{21}}} + (1-p)e^{-\frac{\tau_2-\tau_1}{\theta_{22}}}\right) \end{aligned} \tag{5}$$

Reparameterization to a geometric mixture is needed in the interval monitoring case. Details are not provided here.

2 Functions in the pacakge

2.1 rhSSALT

Simulate a simple hSSALT dataset with exponential (continuous) or geometric (interval) distribution.
I will be responsible for this function.

rhSSALT(n, censoring, τ, r= NULL, monitoring = "continuous", Δ= NULL, θ_1, θ_{21}, θ_{22}, p):

- n: sample size, an integer.
- censoring: censoring: 1 or 2 (Type-I or Type-II). By default is 1.
- monitoring: continuous or interval, by default is continuous. In terms of interval monitoring, only equally-spaced inspection is supported.
- Δ: if interval monitoring, interval length, $\Delta > 0$, a positive numeric value. By default is NULL.
- θ_1: exponential parameter under x_1, a numeric value.
- θ_{21}: exponential parameter of one group under x_2, a numeric value.
- θ_{22}: exponential parameter of the second group under x_2, a numeric value.
- p: mixture proportion, a numeric value.

2.2 MLEhSSALT

Provide point estimation of a simple hSSALT model with exponential (continuous) or geometric (interval) distribution.
You will be responsible for this function since the key point is the EM algorithm in both scenarios.

MLEhSSALT(data, n, censoring, τ, monitoring= "continuous", Δ= NULL, θ_{21}^0, θ_{22}^0, p^0, maxit = 1000, tol = 1e-8, language = "CPP", parallel = FALSE, ncores):

- data: sample, a vector. The given data should be a censored vector with observations less than or equal to n. Thus, when censoring is 2, the length of data is r.
- n: sample size, an integer.
- censoring: 1 or 2 (Type-I or Type-II). By default is 1.
- τ: If censoring is 1, τ is a vector with length 2; if censoring is 2, τ is a positive numeric value. In the code, in order to keep consistency for two censoring methods, max(data) can be considered as tau2 when censoring is 2.
- monitoring: continuous or interval, by default is continuous. In terms of interval monitoring, only equally-spaced inspection is supported.
- Δ: if interval monitoring, interval length, $\Delta > 0$, a positive numeric value. By default is NULL. Please add one check if tau/Delta are not two integers, return an error "Invalid argument 'delta', Only equally-spaced inspection is supported".
- θ_{21}^0: initial value of θ_{21} for the EM algorithm, can be both a numeric value or a vector of values. Add one message that with a vector of initial values, the optimal one with the largest log-likelihood is returned as the MLE.
- θ_{22}^0: initial value of θ_{22} for the EM algorithm, can be both a numeric value or a vector of values. Please also add a check to ensure three vectors of initial parameters have the same length.
- p^0: initial value of mixture proportion p, can be both a numeric value or a vector of values.
- maxit: The maximum number of iterations allowed, an integer. By default is 1000. N in our current version
- tol: Tolerance limit for declaring algorithm convergence based on the change between two consecutive iterations. By default is 1e-8.
- language: R or CPP. By default is CPP.
- parallel: support parallel computation for multiple initial values, a logical value. By default is FALSE.
- ncores: the number of cores that are used in parallelization, an integer.

```
MLEhSSALT <- function(data, n, censoring = 1, tau, monitoring = "continuous",
delta = NULL, theta21, theta22, p, maxit = 1000, tol = 1e-8, language = "CPP",
parallel = FALSE, ncores){
  ###Some examples of checking the input of given parameters
  if (!is.vector(data)) {
    stop("Invalid argument 'data'! The type of 'data' should be a vector")
  }
  if (any(data < 0) || !is.numeric(data)) {
    stop("Invalid argument 'data'! The values of 'data' should be positive numbers")
  }
  ###Check the input of parameter n
  if (n < 0 || !is.numeric(n)) {
    stop("Invalid argument 'n'! The value of 'n' should be a positive number")
  }
  if (n < length(data)) {
    stop("Invalid argument 'n'! The value of 'n' should be larger than the
    length of data")
  }
  ###Check the input of parameter censoring
  if ((censoring == 1) + (censoring == 2) < 1) {
    censoring <- 1
    warning("Invalid argument 'censoring'! censoring is either 1 or 2.
    censoring = 1 is used instead")
  }

  ...

  if(monitoring == "continuous"){
    MLE_Exp(data, n, censoring, tau, theta21, theta22, p, maxit, tol, language,
    parallel, ncores)
  }else{
    MLE_Geo(data, n, censoring, tau, delta, theta21, theta22, p, maxit, tol, language,
    parallel, ncores)
  }
}
```

Two basic functions are needed here:

- MLE_Exp(data, n, censoring, τ, θ_{21}^0, θ_{22}^0, p^0, maxit, tol, language)
- MLE_Geo(data, n, censoring, τ, monitoring, Δ, θ_{21}^0, θ_{22}^0, p^0, maxit, tol, language)

The result should be a list to summarize all information. The MLE under x_1 should also be provided. This can be easily computed with a formula. Besides, the number of failures should be checked in advance to ensure the validity of the MLEs. For example, at least one observation under x_1 and at least 3 observations under x_2.

2.3 CIhSSALT

Provide interval estimation of a simple hSSALT model with exponential (continuous) or geometric (interval) distribution.
The key point for the two bootstrap methods is the EM algorithm. This function will be done after the first two functions.

Idea 1:

CIhSSALT(data, n, censoring, τ, monitoring= "continuous", Δ= NULL, CImethod, α, B, θ_{21}^0, θ_{22}^0, p^0, maxit = 1000, tol = 1e-8, language = "CPP", parallel = FALSE, ncores)

- data: sample, a vector.
- n: sample size, an integer.
- censoring: 1 or 2 (Type-I or Type-II). By default is 1.
- monitoring: continuous or interval, by default is continuous. In terms of interval monitoring, only equally-spaced inspection is supported.
- Δ: if interval monitoring, interval length, $\Delta > 0$, a positive numeric value. By default is NULL.
- CImethod: asymptotic, bootstrap percential, bootstrap BCa. If continuous, all are available; if interval, bootstrap BCa is not available. By default is asymptotic.

- α: given significant level. By default alpha = 0.05.
- B: size of bootstrap repetitions, an integer, by default is 1000.
- θ_{21}^0: initial value of θ_{21} for the EM algorithm, a numeric value.
- θ_{22}^0: initial value of θ_{22} for the EM algorithm, a numeric value.
- p^0: initial value of mixture proportion p, a numeric value.
- maxit: The maximum number of iterations allowed, an integer. Only for bootstrap methods. By default is 1000.
- tol: Tolerance limit for declaring algorithm convergence based on the change between two consecutive iterations. Only for bootstrap methods. By default is 1e-8.
- language: R or CPP. Only for bootstrap methods. By default is CPP.
- parallel: support parallel computation, a logical value. Only for bootstrap methods. By default is FALSE.
- ncores: the number of cores that are used in parallelization, an integer.

```
CIhSSALT <- function(data, n, censoring = 1, tau, monitoring = "continuous",
delta = NULL, CImethod = "asymptotic", alpha = 0.05, B = 1000,
theta21, theta22, p, maxit, tol, language = "CPP", parallel = FALSE, ncores){
  ###Check the input of parameter CImethod
  if(!(CImethod=="asymptotic" || CImethod=="percentile" || CImethod=="bca")){
    CImethod <- "asymptotic"
    warning("'CImethod' should be 'asymptotic', 'percentile' or 'bca'!
    The default value of 'asymptotic' is used instead")
  }
  ###3 methods
  if(CImethod == "asymptotic"){
    CIsay_hSSALT(data, n, censoring, tau, monitoring, delta, alpha,
    theta21, theta22, p, maxit, tol, language, parallel, ncores)
  }else{
    if(CImethod == "percentile"){
      CIbs_hSSALT(data, n, censoring, tau, monitoring, delta, alpha, B,
      theta21, theta22, p, maxit, tol, language, parallel, ncores)
    }else{
      CIbca_hSSALT(data, n, censoring,  tau, monitoring, delta, alpha, B,
      theta21, theta22, p, maxit, tol, language, parallel, ncores)
    }
  }
}
```

Three basic functions are needed here:

- CIsay_hSSALT(data, n, censoring, τ, monitoring, Δ, α, θ_{21}^0, θ_{22}^0, p^0, maxit, tol, language, parallel, ncores)
- CIbs_hSSALT(data, n, censoring, τ, monitoring, Δ, α, B, θ_{21}^0, θ_{22}^0, p^0, maxit, tol, language, parallel, ncores)
- CIbca_hSSALT(data, n, censoring, τ, monitoring, Δ, α, B, θ_{21}^0, θ_{22}^0, p^0, maxit, tol, language, parallel, ncores)

Idea 2:

CIhSSALT(...)

The input is an MLEhSSALT object. That is to say, man should first run the MLEhSSALT() to obtain the point estimation, and then the results can be extracted as the input here. The advantage is less input parameters should be provided here. The disadvantage is less flexibility. To be decided.

Information about Asymptotic CIs:

Approximate CIs are constructed by the asymptotic normality property of MLEs. The needed pivotal quantities for $100(1-\alpha)\%$ CIs of θ_1, θ_{21}, θ_{22} and p are $[\hat{\theta}_1 - E(\hat{\theta}_1)]/\sqrt{V_{\hat{\theta}_1}}$, $[\hat{\theta}_{21} - E(\hat{\theta}_{21})]/\sqrt{V_{\hat{\theta}_{21}}}$, $[\hat{\theta}_{22} - E(\hat{\theta}_{22})]/\sqrt{V_{\hat{\theta}_{22}}}$, $[\hat{p} - E(\hat{p})]/\sqrt{V_{\hat{p}}}$, where the variances can be obtained through the inverse of the observed Fisher information matrix.

Type II continuous: MLE of θ_1 is biased, the two-sided $100(1-\alpha)\%$ confidence interval for θ_1 is $\left[\hat{\theta}_1 - \left(\sum_{j=1}^{r-1}\sum_{k=0}^{j} c_{j,k}\tau_{j,k}\right) \pm z_{1-\frac{\alpha}{2}}\sqrt{V_{\hat{\theta}_1}}\right]$, where $\tau_{j,k} = (\tau/j)(n-j+k)$, $c_{j,k} = \frac{(-1)^k}{\sum_{i=1}^{r-1} p_i}\binom{n}{j}\binom{j}{k}e^{-(\tau/\theta_1)(n-j+k)}$, $p_i = \binom{n}{i}\left(1-e^{-(\tau/\theta_1)}\right)^i e^{-(\tau/\theta_1)(n-i)}$.

Type I continuous: MLE of θ_1 is biased, the two-sided $100(1-\alpha)\%$ confidence interval for θ_1 is $\left[\hat{\theta}_1 - c_n\left(\sum_{j=1}^{n-1}\sum_{k=0}^{j} c_{j,k}\tau_{j,k}\right) \pm z_{1-\frac{\alpha}{2}}\sqrt{V_{\hat{\theta}_1}}\right]$, where $c_n = \frac{1}{1-(1-p_1)^n-(1-p_2)^n+p_3^n}$ with $p_1 = G^*(\tau_1) = 1-e^{-\frac{\tau_1}{\theta_1}}$, $p_2 = G^*(\tau_2) - G^*(\tau_1) = e^{-\frac{\tau_1}{\theta_1}}\left\{1 - pe^{-(\tau_2-\tau_1)/\theta_{21}} - (1-p)e^{-(\tau_2-\tau_1)/\theta_{22}}\right\}$, $p_3 = 1-p_1-p_2$, $\tau_{j,k} = (\tau_1/j)(n-j+k)$, $c_{j,k} = (-1)^k\binom{n}{j}\binom{j}{k}\left\{(1-p_1)^{n-j} - p_3^{n-j}\right\}(1-p_1)^k$.

The two-sided $100(1-\alpha)\%$ CIs for the remaining three parameters are $\left[\hat{\theta}_{21} \pm z_{1-\frac{\alpha}{2}}\sqrt{V_{\hat{\theta}_{21}}}\right]$, $\left[\hat{\theta}_{22} \pm z_{1-\frac{\alpha}{2}}\sqrt{V_{\hat{\theta}_{22}}}\right]$, $\left[\hat{p} \pm z_{1-\frac{\alpha}{2}}\sqrt{V_{\hat{p}}}\right]$.

Information about Bootstrap CIs:

A two-sided $100(1-\alpha)\%$ percentile bootstrap CI for a specific parameter is obtained simply by $(\hat{\eta_l}^{[\alpha B/2]}, \hat{\eta_l}^{[(1-\alpha)B/2]})$. A two-sided $100(1-\alpha)\%$ BCa CI of η_l is given by (η_l^L, η_l^U), where $\eta_l^L = \hat{\eta_l}^{[B\alpha_{1l}]}$, $\eta_l^U = \hat{\eta_l}^{[B\alpha_{2l}]}$ with

$$\alpha_{1l} = \Phi\left(\hat{z}_{0l} + \frac{\hat{z}_{0l} + z_{\alpha/2}}{1 - \hat{a}_l\left(\hat{z}_{0l} + z_{\alpha/2}\right)}\right),$$

and

$$\alpha_{2l} = \Phi\left(\hat{z}_{0l} + \frac{\hat{z}_{0l} + z_{1-\alpha/2}}{1 - \hat{a}_l\left(\hat{z}_{0l} + z_{1-\alpha/2}\right)}\right).$$

Here, $\Phi(\cdot)$ is the standard normal CDF and the value of the bias correction $\hat{z}_{0l}$ is computed by

$$\hat{z}_{0l} = \Phi^{-1}\left(\frac{\#(\hat{\eta_l}^{(b)} < \hat{\eta_l})}{B}\right), \quad l = 1, \dots, 4, b = 1, \dots, B.$$

Further, choose the good estimate of the acceleration $\hat{a}_l$ with

$$\hat{a}_l = \frac{\sum_{i=1}^{r}\left(\hat{\eta}_{l(\cdot)} - \hat{\eta}_{l(i)}\right)^3}{6\left[\sum_{i=1}^{r}\left(\hat{\eta}_{l(\cdot)} - \hat{\eta}_{l(i)}\right)^2\right]^{\frac{3}{2}}}, \quad l = 1, \dots, 4,$$

where $\hat{\eta}_{l(i)}$ is the jackknife estimate for η_l based on the original sample, i.e., the MLE of parameter η_l with the i-th observation deleted, and $\hat{\eta}_{l(\cdot)}$ is the mean of the r jackknife estimates for l-th parameter. This can only be applied when at least two observations at each stress level of the initial sample.

Your tasks until our next meeting at 17:00 on Nov.30, 2023:

- **Tiny adjustments in R version for the output name**
- **Add the posterior probability distribution in the CPP version (I have adjusted in the R version)**

- **Please check the CPP version of the R package Rmpfr with high-precision computation if possible, I use mpfr() and chooseMpfr().**
- **Please adjust the BCa method in CIhSSALT(), formulate the input parameters of the help functions in a good manner and test**

2.4 plot

Provide the CDF plot based on the data and the estimates obtained by MLEhSSALT(). Need to define the output of MLEhSSALT() as an object. Then plot is created based on this object. Simply extract the useful information from the output list.

2.5 Heterogeneity test for an hSSALT model

Provide a heterogeneity test for a simple hSSALT model. The result can be linked to the package SSALT (in progress) later. The methodology in this part hasn't been developed.