REIDAR - an S-plus Library for Analysing Data from Repairable Systems

Georg Elvebakk

1 INTRODUCTION

REIDAR (REpaIrable systems Data Analysis Routines) is an *S-plus* library of functions, i.e. a collection of functions available after entering the library. *S-plus* itself is an environment for statistical and graphical data analysis and programming. It is an enhanced version of the high level language *S* and is distributed by Mathsoft, Inc. Especially in the statistics field *S-plus* have become sort of standard tool, new methods are often accompanied with an publicly accessible *S-plus* implementation.

The library REIDAR includes functions for parametric modelling, model checking, testing and nonparametric/graphical techniques. Some of them are standard methods/models for repairable systems, others are developed as part of research at the Department of Mathematical Sciences, Norwegian University of Technology and Science, Trondheim. These are functions which are not standard in S-plus, which hopefully will be useful when analysing repairable systems data. The purpose of REIDAR is therefore to make these functions easily available. This report is a manual and documentation of the contents of REIDAR, including some of the mathematical and computational aspects.

1.1 Development of REIDAR

REIDAR was started as a student project for Ole Kristian Lundal (Lundal, 1994), supervised by professor Bo Lindqvist and Georg Elvebakk. Originally the intension was to implement some of the existing graphical analysing techniques for data from repairable systems. Later, REIDAR was extended to include methods developed for parametric model estimation and model checking in general heterogeneous trend-renewal processes (Elvebakk, Lindqvist & Heggland, 1999). REIDAR also includes some tests for trend for repairable systems, several standard tests and also some new tests developed by Kvaløy & Lindqvist (1998) and Elvebakk (1998). Several students have worked on models that can now be seen as subsets of the REIDAR model class, in particular (Tveit, 1993), (Heggland, 1994) and (Brandt, Korsaksel & Rosland, 1995).

REIDAR may thus be said to be an S-plus implementation of methods for analysing data from repairable systems. But in addition to standard techniques it also includes several new models and methods developed at the Department of Mathematical Sciences, NTNU, Trondheim.

The work on REIDAR was supported by "the Growth Point Centre in Safety and Reliability" at NTNU/SINTEF with a grant in 1996.

Contents

2 Documentation

2.1 Notation

Consider a repairable system, started at time 0, observed in the interval $[a, b]$. The successive failure times are denoted $T_1, T_2, \ldots, T_n$, where T_1 is the time of the first recorded failure. If $T_n = b$ we say that the system is *failure truncated*, otherwise it is *time truncated*.

The interfailure times or times between failures are denoted $X_1, X_2, \ldots$ thus

$$X_i = T_i - T_{i-1}, \quad i = 1, 2 \ldots$$

where it is defined that $T_0 = a$.

Another way of representing the same information is by the number of failures at any given time, thus:

$$N(t) = \text{number of failures in } \ (a, t]$$

Also define $\mathcal{F}_{t-}$, the *history* of the failure process up to, but not including, time t. For the models considered in REIDAR $\mathcal{F}_{t-}$ can be thought of as generated by $\{N(s) \ : \ 0 \leq s < t\}$.

For datasets consisting of several repairable systems system number j is observed in $[a_j, b_j]$ and failure times recorded at $T_{j1}, T_{j2}, \ldots, T_{jn_j}$.

2.2 Data Input Functions

Functions for reading data from a file into a format used by the other functions in the REIDAR library. The function *rs.read.frame* reads data from a file into a data frame. The function *is.rsframe* checks whether an object is an *rsframe*.

rs.read.frame(file, time.scale)

Required arguments

file: the name of the file to be read.

Optional arguments

time.scale: the time unit used.

Output

A data frame object (called an *rsframe*, "repairable systems data frame") with vectors:

start: a vector of times of observation start for each system.

stop: a vector of times of observation stop for each system.

number: a vector of number of failures for each system.

failures: a list of vectors containing the failure times for each system.

The data frame may have an attribute *time.scale*.

Purpose

This is a function for reading repairable systems data from file to the format used in REIDAR.

Details

The file to be read must be as follows: Each line contains data from one system in the following order: number of failures (n), observation start (a), observation stop (b), failure times (T_i).

The optional argument *time.scale* sets the "time" unit used in the observations.

It is of course possible to construct an *rsframe* without using this function. Other vectors may also be added to this structure.

****

rs.is.rsframe(x, warn=T)

Required arguments

x: an S-plus object.

Optional arguments

warn: if *warn* is TRUE warnings will be given.

Output

TRUE or FALSE, additional warnings may be given.

Purpose

This is a function for checking whether x is or is not an *rsframe*. May also return warnings.

Details

To check whether or not x is an *rsframe*, the function checks if x is a data frame with vectors named *start*, *stop*, *number* and *failures*.

If the optional argument *warn* is TRUE, warnings will be returned.

****

2.3 HTRP Models

The largest part of REIDAR is a collection of functions for analysing HTRP (heterogeneous time renewal process) models. These include functions for simulating, (*rs.htrp.sim*), estimating model parameters (*rs.htrp.mle*), model plotting (*rs.htro.plot*), estimating residual processes (*rs.htrp.res*) and pp-plotting of residual data (*rs.htrp.pp*).

A HTRP model (Elvebakk *et al.*, 1999) is a model for multi-system recurrence data, for example failure data from a number of cars, where the age or mileage or operating time etc. is recorded for every failure. For a dataset with only one system, a TRP (trend renewal process) model is used. A TRP model is a generalization of a non-homogeneous Poisson process (NHPP) and a renewal process (RP), having both as special cases. The following well-known property of an NHPP process helps to illustrate how the TRP model is constructed:

If $T_1, T_2, \ldots$ is an NHPP with intensity function $\lambda(t)$, then the time-transformed process $\Lambda(T_1), \Lambda(T_2), \ldots$ is a homogeneous Poisson process (HPP) with expected time between failures equal to 1.

The trend-renewal process (TRP) is defined simply by allowing the HPP of the NHPP definition above to be any renewal process. So the intensity of a TRP process is specified by the NHPP intensity function $\lambda(t)$ (from now on referred to as a trend function or transformation) and the distribution of the interfailure times in the renewal process, $F(x)$. To avoid ambiguities in the models $F(x)$ is restricted to be a distribution with expectation 1. The intensity given the history up to time t for a TRP process is then:

$$\lambda(t|\mathcal{F}_{t-}) = z(\Lambda(t) - \Lambda(T_{N(t-)}))\lambda(t) \tag{1}$$

where z(t) is the hazard rate corresponding to F. Thus the intensity is a function of the time t and the transformed time since last failure $\Lambda(t) - \Lambda(T_{N(t-)})$. For an NHPP model the hazard is always 1, so the intensity given history becomes:

$$\lambda(t|\mathcal{F}_{t-}) = \lambda(t) \tag{2}$$

For an RP model $\Lambda(t) = \lambda$, giving:

$$\lambda(t|\mathcal{F}_{t-}) = z(t - T_{N(t-)})\lambda \tag{3}$$

In case of several systems the HTRP model allows modelling of unobserved heterogeneity between systems by modifying the trend function with a multiplicative factor; for system j:

$$\lambda_j(t) = a_j\lambda(t) \tag{4}$$

where a_j is an unobservable random variable that takes values independently across systems according to a common probability distribution, H. For convenience H is assumed to have expectation 1.

HTRP Model Notation

The notation HTRP$(F, \lambda(\cdot), H)$ will be used for a HTRP model with distribution F for the underlying renewal process, H for the heterogeneity distribution and a trend function described by $\lambda(\cdot)$. In practice also the sub-models of the HTRP model will be of interest. These include the HPP, RP, NHPP and TRP, but also the heterogeneous versions of these models, which are denoted by putting an H in front.

From the full model HTRP$(F, \lambda(\cdot), H)$, there are seven sub-models:

HNHPP$(\lambda(\cdot), H)$ = HTRP$(exp, \lambda(\cdot), H)$

HRP(F, λ, H) = HTRP(F, λ, H)

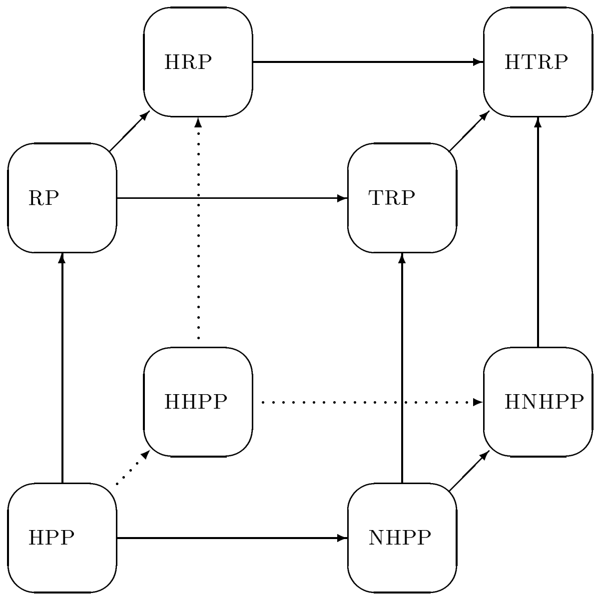

Figure 1: The model cube.

HHPP(λ, H) = HTRP(exp, λ, H)

TRP$(F, \lambda(\cdot))$ = HTRP$(F, \lambda(\cdot)$,1)

NHPP$(\lambda(\cdot))$ = HTRP$(exp, \lambda(\cdot), 1)$

RP(F, λ) = HTRP$(F, \lambda$,1)

HPP(λ) = HTRP$(exp, \lambda, 1)$

Here *exp* means the exponential distribution with parameter 1, while 1 means the distribution degenerate at 1. Note the difference between the function $\lambda(\cdot)$ and the constant λ.

The eight models are depicted in Figure 1, where each vertex represents a model, and the lines connection them corresponds to changing one of the "coordinates" in the HTRP notation. Going to the right here represents introducing a trend, going up introduces a non-Poisson model, and going backwards (inwards) introduces heterogeneity in the model.

Computational Implementation

Functions for the HTRP$(F, \lambda(\cdot), H)$ models include functions for simulating, likelihood optimization, plotting, model checking, etc. Efforts have been made to write these in a general way, meaning that all implemented options for the distribution of F are exchangeable. The same is true for $\Lambda(\cdot)$ and H. This also makes it easier to later add other functions or distributions to REIDAR.

2.3.1 Simulating from an HTRP Model

For simulations from TRP models we take advantage of the fact that:

If $S_1, S_2, \ldots$ is an RP with interfailure times distributed according to F (RP(F,1)), then the time-transformed process $\Lambda^{-1}(S_1), \Lambda^{-1}(S_2), \ldots$ is a trend renewal process (TRP($F, \lambda(\cdot)$)).

As always, F is supposed to have expectation 1.

Implementation Details

The function *rs.htrp.sim* simulates from an HTRP model. Several options for trend function (transformation) and renewal and heterogeneity distributions are implemented. (Listed later in this section and in Section 2.3.6.) With the model and model parameters set the simulation algorithm for the simulation goes as follows. We assume here a heterogeneous model with m systems, failure truncated, with $n_1, n_2, \ldots, n_m$ number of failures respectively, all observed from time 0.

1. Draw $a_1, a_2, \ldots, a_m$ from the heterogeneity distribution, H.
2. For systems $j = 1, 2, \ldots, m$ draw $X_{j1}, \ldots, X_{jn_j}$ from the renewal distribution, F. The arrival times of the renewal process are the cumulative interarrival times:
$$S_{ji} = \sum_{k=1}^{i} X_{jk}$$
3. Transform the arrival times of the renewal process with the inverse of the trend function. For systems $j = 1, 2, \ldots, m$:
$$T_{ji} = \Lambda_j^{-1}(S_{ji}) = \Lambda^{-1}(S_{ji}/a_j) \tag{5}$$

For models without heterogeneity $a_j = 1, \forall j$. Time truncated models are simply simulated until a given censoring time.

For (heterogeneous) TRP models it is always assumed that a renewal takes place at the time of observation start.

rs.htrp.sim(model, par, end, start=0, sysnum=1, timetrunc=T)

Required arguments

model: a list specifying a model. The list should include component names *renewal*, *transformation* and *heterogeneity*. See the details section for further details.

par: a vector or list of model parameters.

end: a vector of endpoints (stopping times), or just a single point if only 1 system is simulated. If *timetrunc* = F the value(s) of *end* is interpreted as number of failures. If *sysnum* > 1 and length(*end*) = 1 all systems are assumed to have the same endpoint.

Optional arguments

start: a vector of observation starting points (starting times), or just a single point if only 1 system is simulated. If *sysnum* > 1 and length(*start*) = 1 all systems are assumed to have the same starting point. Default is 0.

sysnum: number of systems to be simulated, default is 1.

timetrunc: if TRUE the function simulates until a given time, else until a given number of failures. If *sysnum* > 1 and length(*timetrunc*) = 1 all systems are assumed to have the same truncation type.

Output

A data frame object (called an *rsframe*, repairable systems data frame) with *names*: *start*, *stop*, *number* and *failures*. The vector *number* is a vector of number of failures for each system. The vector *start* is a vector of starting times (for observation) for each system. The vector *stop* is a vector of stopping times (for observation) for each system. The list *failures* is a list of vectors containing the failure times for each system. Other *names* and other *attributes* may be added later to the frame.

Purpose

To simulate data from an HTRP model as specified in the arguments *model* and *par*, and return the result as an *rsframe*.

Details

Heterogeneous trend renewal processes (HTRP) are a class of models including homogeneous Poisson processes (HPP), non-homogeneous Poisson processes (NHPP), renewal processes (RP), trend renewal process (TRP) and heterogeneous versions of these models. We build this class of models by starting with a renewal process (includes the HPP). The arrival times of this process can then be transformed by a trend function, the result is a so-called trend renewal process. (If we started with an HPP with expectation 1 we would get an NHPP.) In case of multiple systems (processes) heterogeneity may be added to the model. The trend function of each process is then multiplied by a factor distributed according to a specified distribution. So these models can be described in three parts: the renewal distribution, the transformation (trend) function, and the heterogeneity distribution. To avoid ambiguities the distribution of the interarrival times of the renewal process and the heterogeneity factor are restricted to have expectation 1.

RENEWAL: For the distribution of the interarrival times in the renewal process, F, the following distributions are implemented:

exponential: 0-parameter exponential distribution with expectation 1.

$$\begin{array}{l} f(x) = e^{-x} \\ E(X) = 1 \quad Var(X) = 1 \end{array} \tag{6}$$

Weibull: 1-parameter Weibull distribution with expectation 1.

$$\begin{array}{ll} f(x|\beta) = \frac{(\Gamma(\beta+1))^{\frac{1}{\beta}} x^{\frac{1}{\beta}-1} e^{(-(\Gamma(\beta+1)x)^{\frac{1}{\beta}})}}{\beta}, & \beta > 0 \\ E(X) = 1 \quad Var(X) = \frac{\Gamma(2\beta+1)}{\Gamma^2(\beta+1)} - 1 & \end{array} \tag{7}$$

For $\beta = 1$ this is the exponential(1) distribution. For $\beta > 1$ the variance is greater than exponential, smaller for $\beta < 1$.

gamma: 1-parameter gamma distribution with expectation 1.

$$\begin{array}{ll} f(x|\gamma) = \frac{\gamma^{-\frac{1}{\gamma}} x^{\frac{1}{\gamma}-1} e^{-\frac{x}{\gamma}}}{\Gamma(\frac{1}{\gamma})}, & \gamma > 0 \\ E(X) = 1 \qquad Var(X) = \gamma & \end{array} \tag{8}$$

For $\gamma = 1$ this is the exponential(1) distribution.

bimodal-exponential: A 2-parameter distribution, the bimodal exponential distribution with expectation 1.

$$\begin{array}{l} f(x|p,q) = p\alpha_1 e^{-\alpha_1 x} + (1-p)\alpha_2 e^{-\alpha_2 x}, \qquad 0 < p, q < 1 \\ \text{where} \quad \alpha_1 = \frac{p(q-1)+1}{q} \quad \text{and} \quad \alpha_2 = p(q-1)+1 \\ E(X) = 1 \qquad Var(X) = (p-p^2)(\alpha_1^{-2} + \alpha_2^{-2} - 2(\alpha_1\alpha_2)^{-1}) \end{array} \tag{9}$$

Remark: A bimodal exponential RP is an RP were the times between arrivals with probability p and $(1-p)$ are exponentially distributed with parameters α_1 and α_2 respectively. The parameters are restricted such that the expectation will always be 1. To do that a new parameter is introduced $q = \frac{\alpha_2}{\alpha_1}$, where $\alpha_2 < \alpha_1$.

TRANSFORMATION: The trend transformation function can either follow a homogeneous (constant trend) model, a power law model, a log linear model, a mixed log linear power law model or a linear model. They correspond to intensities for well-known NHPP models and have the following trend functions:

homogeneous:

$$\lambda(t|a) = a \qquad a > 0 \tag{10}$$

power-law:

$$\lambda(t|a,b) = abt^{b-1} \qquad a, b > 0 \tag{11}$$

log-linear:

$$\lambda(t|a,c) = ae^{ct} \qquad a > 0 \qquad -\infty < c < \infty \tag{12}$$

log-linear-power-law:

$$\lambda(t|a,b,c) = abt^{b-1}e^{ct} \qquad a, b > 0 \qquad -\infty < c < \infty \tag{13}$$

linear:

$$\lambda(t|d,e) = (d+et) \vee 0, \qquad -\infty < d, e < \infty \tag{14}$$

Remark: For the log linear and log linear power law models $c < 0$ may lead to infinite failure times.

HETEROGENEITY: To add heterogeneity to the model the trend transformation functions for each system is multiplied with a factor a_j:

$$\lambda_j(t) = a_j \lambda(t), \qquad j = 1, \ldots, \text{sysnum} \;\; (m)$$

where a_j is distributed according to one of the following distributions:

Weibull: 1-parameter Weibull distribution with expectation 1.

$$f(a|\beta) = \frac{(\Gamma(\beta+1))^{\frac{1}{\beta}} a^{\frac{1}{\beta}-1} e^{(-(\Gamma(\beta+1)a)^{\frac{1}{\beta}})}}{\beta}, \qquad \beta \geq 0 \tag{15}$$

gamma: 1-parameter gamma distribution with expectation 1.

$$f(a|\gamma) = \frac{\gamma^{-\frac{1}{\gamma}} a^{\frac{1}{\gamma}-1} e^{-\frac{a}{\gamma}}}{\Gamma(\frac{1}{\gamma})}, \qquad \gamma \geq 0 \tag{16}$$

Remark: These distributions are the same gamma and Weibull distributions used for the renewal process.

SPECIFYING A MODEL:
A list specifying the model must be given as input to the *model* argument. The list should include the component names *renewal*, *transformation* and *heterogeneity*. An example:

> my.mod <- list(renewal=”exponential”,transformation=”homogeneous”,heterogeneity=”no”)

This specifies homogeneous Poisson process (HPP(λ)) model. Notice that another way do to this would be to use an (unrestricted) 1-parameter exponential distribution for the renewal process and no transformation. This is, however, an illegal model specification within this framework.

Another more general RP model is the gamma distributed renewal function, RP($gamma, \lambda$):

> my.mod <- list(renewal=”gamma”,transformation=”homogeneous”,heterogeneity=”no”)

For an NHPP model choose a non-homogeneous transformation function, for example the power law, NHPP(*power law*):

> my.mod <- list(renewal=”exponential”,transformation=”power-law”,heterogeneity=”no”)

A TRP model: the gamma distributed renewal function with a power law transformation function, TRP(*gamma*,*power law*):

> my.mod <- list(renewal=”gamma”,transformation=”power-law”,heterogeneity=”no”)

For multi-system datasets heterogeneity may be added to any of the TRP models (or its sub-models, NHPP, RP, HPP) by choosing for example a gamma distribution for the heterogeneity factor, HTRP(*gamma*,*power law*,*gamma*):

> my.mod <- list(renewal=”gamma”,transformation=”power-law”,heterogeneity=”gamma”)

The HPP model is of course a special case of both RP and NHPP models, RP and NHPP models are special cases of TRP models, and non-heterogeneous models are special cases of heterogeneous models. So the heterogeneous TRP (HTRP) model includes all the other models.

MODEL PARAMETERS:
Model parameters are chosen via the *par* argument. It should be either a vector with number of elements corresponding to the number of parameters in that particular model, or a list like the *model* argument. For example the TRP model:

> my.mod <- list(renewal=”gamma”,transformation=”power-law”,heterogeneity=”no”)

```
> my.par1 <- c(1,2,0.75)
> my.par2 <- list(renewal=c(1),transformation=c(2,0.75),heterogeneity=NULL)
```

Both *my.par1* and *my.par2* are valid input to the argument *par*. In the parameter vector the renewal distribution parameters should come first, followed by the transformation parameters and the heterogeneity parameters. Within a distribution or a trend function the parameters should follow alphabetical order.

The names and number of parameters for the different distributions and transformation (trend) functions:

Name	Number of parameters	
exponential:	0	
gamma:	1	(γ)
Weibull:	1	(β)
bimodal-exponential:	2	(p, q)
homogeneous:	1	(a)
power-law:	2	(a, b)
log-linear:	2	(a, c)
log-linear-power-law:	3	(a, b, c)
linear:	2	(a, d)

So to simulate data from a model with parameters as specified above:

```
> my.data <- rs.htrp.sim(model=my.mod,par=my.par1,end=c(10),sysnum=1,timetrunc=T)
```

This will be a one-system dataset, following a TRP($gamma, power\ law$) model, observed from time 0, time truncated at time 10.

WARNINGS:
For the log linear and log linear power law models $c < 0$ may lead to infinite failure times.

For (heterogeneous) TRP models it is always assumed that a renewal takes place at the time of observation start.

****

2.3.2 Parameter Estimation in an HTRP Model

A general expression for the likelihood function of a (single) counting process observed in the interval $[0,\tau]$ given observations $T_1, T_2, \ldots, T_n$ is (Andersen, Borgan, Gill & Keiding, 1993):

$$L = \left\{ \prod_{i=1}^{n} \lambda(T_i|\mathcal{F}_{T_i-}) \right\} exp\left(-\int_0^{\tau} \lambda(u|\mathcal{F}_{u-})du \right) \tag{17}$$

For a TRP model (1) is substituted into (17), giving the likelihood:

$$L = \left\{ \prod_{i=1}^{n} z(\Lambda(T_i) - \Lambda(T_{i-1}))\lambda(T_i) \right\} exp\left(-\int_0^{\tau} z(\Lambda(u) - \Lambda(T_{N(u-)}))\lambda(u)du \right) \tag{18}$$

If f is the intensity function corresponding to F the likelihood can be written as:

$$L = \left\{ \prod_{i=1}^{n} f(\Lambda(T_i) - \Lambda(T_{i-1}))\lambda(T_i) \right\} (1 - F(\Lambda(\tau) - \Lambda(T_n))) \tag{19}$$

This latter form of the likelihood function is used in the function *rs.htrp.mle*.

For m systems, we assume independence and models that are identical up to a heterogeneity factor a_j. The full likelihood becomes:

$$\begin{aligned} L &= \prod_{j=1}^{m} L_j \qquad (20) \\ &= \prod_{j=1}^{m} \left\{ \prod_{i=1}^{n_j} f(a_j(\Lambda(T_i) - \Lambda(T_{i-1})))a_j\lambda(T_i) \right\} (1 - F(a_j(\Lambda(\tau) - \Lambda(T_{n_j})))) \end{aligned}$$

Since a_j is unobservable L_j is now obtained as the expected value of L_j with respect to a_j. With h being the density function corresponding to H the full likelihood is then:

$$\begin{aligned} L &= \prod_{j=1}^{m} \int L_j dH(a_j) \qquad (21) \\ &= \prod_{j=1}^{m} \int \left\{ \prod_{i=1}^{n_j} f(a_j(\Lambda(T_i) - \Lambda(T_{i-1})))a_j\lambda(T_i) \right\} (1 - F(a_j(\Lambda(\tau) - \Lambda(T_{n_j}))))h(a_j)da_j \end{aligned}$$

Implementation Details

The function *rs.htrp.mle* does maximum likelihood estimation in an HTRP model. The options for trend function (NHPP intensity) and renewal and heterogeneity distributions are the same as for *rs.htrp.sim*. The parametric optimization is done by maximising the log-likelihood, the logarithm of (21). This is either done by the standard *S-plus* function *nlminb*, a quasi-Newton optimizer, or by a REIDAR function *amoebamin*, a modification of a simplex routine from Press, Teukolsky, Vetterling & Flannery (1992). When using these numerical optimizers we have of course no guarantee that the real optimum is found, even when the procedure converges. Our experience is that for those functions currently implemented in REIDAR there seems to be few problems, but we cannot expect that to hold for every choice of functions and parameters.

The integral in (21) can be solved analytically for some choices of F, Λ and H, but generally, it cannot. So in REIDAR the likelihood function (21) is always computed by numerical integration for heterogeneous models. The numerical integration is based on the the following form of the integral:

$$\begin{aligned} \int L_j(a_j)dH(a_j) &= \int g(a_j)da_j \\ &= \int exp\{ \textstyle\sum_{i=1}^{n_j} log(f(a_j(\Lambda(T_i) - \Lambda(T_{i-1})))) + n_j \log(a_j) \\ &\quad + \textstyle\sum_{i=1}^{n_j} log(\lambda(T_i)) + log(1 - F(a_j(\Lambda(\tau) - \Lambda(T_{n_j})))) \qquad (22) \\ &\quad + log(h(a_j)) \} \, da_j \end{aligned}$$

The integrand g is a function of a_j, F, Λ and H. These functions, or rather $log(f)$, $log(\lambda)$, $log(h)$ and $log(1-F)$ are arguments to g. As the implementation of many of these function are themselves based on numerical methods the optimization procedure may be very time consuming for some choices of models. The functions currently implemented in REIDAR are listed in Section 2.3.6.

The default numerical routine used is the standard *S-plus* function *integrate* which uses an adaptive 15-point Gauss-Kronod quadrature. In our experience this function was faster than a simple Simpson's routine for comparable accuracy. In some cases, typically when the numerical integration routine returns 0 or ∞, the integration routine is aided by some simple measures. The algorithm goes as follows:

1. Find the value of a_j, $\hat{a_j}$ that maximizes $g(a_j)$ by minimising $-log(g(a_j))$ with *nlminb*.
2. Find the constant C such that $\frac{g(\hat{a_j})}{e^C} = 1 \Rightarrow C = log(g(\hat{a_j}))$
3. Find the interval where $\frac{g(\hat{a_j})}{e^C} = exp\{log(g(\hat{a_j})) - C\} > 0$.
4. Numerically integrate $\frac{g(\hat{a_j})}{e^C} = exp\{log(g(\hat{a_j})) - C\}$ by the function *integrate* with lower and upper integration limits.
5. $log(\int g(a_j)da_j)$ is then approximated by the numerically evaluated $log(\int \left(\frac{g(\hat{a_j})}{e^C}\right) da_j) + C$

For (heterogeneous) TRP models it is always assumed that a renewal takes place at the time of observation start.

rs.htrp.mle(model, data, start, lower, upper, print=0, amoeba=F, amoeba.tol=1e-10, amoeba.iter=1000, intern.scale=F, ...)

Required arguments

model: a list specifying a model. The list should include component names *renewal*, *transformation* and *heterogeneity*. See under details in Section 2.3.1 for further details.

data: counting process data on the form specified by the function *rs.read.frame*.

Optional arguments:

start: a vector of starting values for the parameters, if not supplied defaults will be set.

lower: a vector of lower bounds for the parameters. If this is not supplied default lower parameter bounds for each model is used.

upper: a vector of upper bounds for the parameters. If this is not supplied default upper parameter bounds for each model is used.

print: if FALSE (default) no intermediate results will be displayed. If TRUE parameter and function values will be printed for each function evaluation.

amoeba: if FALSE (default) *nlminb* is used for optimization. If TRUE a simplex optimizer is used.

amoeba.tol: absolute function convergence tolerance when *amoeba* is TRUE, default is 1e-10.

amoeba.iter: maximum number of iterations when *amoeba* is TRUE, default is 1000.

intern.scale: a simple scaling sometimes useful to obtain convergence. The default is FALSE.

...: additional arguments for *nlminb* (includes arguments to *nlminb.control*).

Output: A list with the following components:

model: same as the input model.

parameters: mle for the parameters.

loglik: the log likelihood of the model.

start: starting values for the model parameters.

lower: lower bounds for the parameters.

upper: upper bounds for the parameters.

intern.scale: if the argument *intern.scale* is TRUE, the start vector. Else a vector of 1s, length is number of parameters.

data: input data.

nlmin: a list of results from the optimization with components *message*, *grad.norm*, *iterations* and *f.evals*. For further information use *help(nlminb)*.

amoeba: a list of results from the optimization with components *message*, *iterations*, *f.evals* and *tol*.

call: the function call.

Purpose

To estimate parameters of a HTRP model of the data in *data*. The model is specified by the argument *model*. The result is returned as a *list*.

Details

For details on the choice and specification of models, see Section 2.3.1.

START PARAMETERS:
The function tries to choose reasonable start parameters, but these can also be supplied via the *start* argument. It should be either a vector with number of elements corresponding to the number of parameters in that particular model, or a list like the *model* argument. An example for a TRP model:

```
> my.mod <- list(renewal="gamma",transformation="power-law",heterogeneity="gamma")
> my.start1 <- c(1,2,0.75)
> data.mle <- rs.htrp.mle(model=my.mod,start=my.start1,data=data)
> my.start2 <- list(renewal=c(1),transformation=c(2,0.75),heterogeneity=NULL)
> data.mle <- rs.htrp.mle(model=my.mod,start=my.start2,data=data)
```

Both *my.start1* and *my.start2* are valid input to the argument *start*. In the vector, the renewal distribution parameters should come first, followed by the transformation parameters and the heterogeneity parameters. Within a distribution or a trend function the parameters should follow alphabetical order. The arguments *lower* and *upper* should be specified in the same manner.

The names and number of parameters for the different distributions and transformation functions:

Name	Number of parameters	
exponential:	0	
gamma:	1	(γ)
Weibull:	1	(β)
bimodal-exponential:	2	(p, q)
homogeneous:	1	(a)
power-law:	2	(a, b)
log-linear:	2	(a, c)
log-linear-power-law:	3	(a, b, c)
linear:	2	(a, d)

AVOIDING ERRORS:
If something goes wrong in the optimization here are some tips:

Try different start values. Parameter estimates from sub-models will often be a good starting point.

Restrict the parameters with the *lower* and *upper* arguments to keep the parameters away from the area where the errors occurred. It is also possible to keep a parameter constant using the *lower* and *upper* arguments.

Transform the input data, i.e. divide or multiply them with a constant. This means observing them on another time scale. About one event per system per time unit can often be recommended.

When the parameters vary a lot in size it may be advantageous to use the option *intern.scale* $= T$.

Try simplex optimization (*amoeba*) instead of *nlminb*.

WARNINGS:

Multiple events in the same system are not allowed. (Remedy: add or subtract a small value, or delete one event)

For (H)TRP models it is assumed that a renewal takes place at the time of observation start.

For large datasets (in particular those with many systems) some of the heterogeneous models will converge very slowly and use a lot of memory capacity. This is due to the use of numerical integration routines within the functions. This is especially true for models where the log linear power law transformation are used, which will often converge slowly even for non-heterogeneous models.

If you try to use a heterogeneous model for datasets that do not exhibit any heterogeneity, the parameters in the gamma and Weibull heterogeneity distributions will converge to 0. This may lead to problems for the optimization procedure, which should be monitored closely.

****

2.3.3 Plot of a fitted HTRP Model

Generally,the intensity of a HTRP model is given by:

$$\lambda_j(t|\mathcal{F}_{N(t-)}) = z(a_j(\Lambda(t) - \Lambda(t_{N(t-)})))a_j\lambda(t) \tag{23}$$

Here Λ is the trend function (NHPP intensity), z is the hazard function of F, the distribution of the interfailure times of the underlying renewal process, and a_j is the unobservable heterogeneity factor for system j.

The cumulative intensity is just the integral of the intensity:

$$\Lambda(t|\mathcal{F}_{N(t-)}) = \int_a^t \lambda_j(u|\mathcal{F}_{N(u-)})du \tag{24}$$

For heterogeneous models the trend function for system j is given by

$$\lambda_j(t) = a_j\lambda(t) \tag{25}$$

where the a_js are unobservable. To get estimates of the heterogeneity factors for individual systems we may use a Bayes estimator (Follmann & Goldberg, 1988), i.e. the expectation in the aposteriori distribution of a_j. The apriori distributions of a_j is $h(a_j)$ and the likelihood function of the TRP model (where a_j is the only unknown parameter) is $L_j(a_j)$ (21). We thus get the aposteriori distribution:

$$\Pi(a_j|T_{j1}, T_{j2}, \ldots, T_{jn_j}) \propto h(a_j)L_j(a_j) \tag{26}$$

Implementation Details

The form of the intensity function (23) implemented in REIDAR is

$$\begin{aligned}\lambda_j(t|\mathcal{F}_{N(t-)}) \quad = \quad & exp\{\, log(f(a_j(\Lambda(t) - \Lambda(t_{N(t-)})))) \\ & -log(1 - F(a_j(\Lambda(t) - \Lambda(t_{N(t-)})))) - log(a) - log(\lambda(t))\,\}\end{aligned}$$

where the functions $log(f)$, $log(1 - F)$, Λ, λ, the heterogeneity factor a_j, the fitted parameters and the dataset are arguments to the intensity function. The functions currently implemented in REIDAR are listed in Section 2.3.6.

For the cumulative intensity (24) a simple Simpson's integration routine is used to compute the integral.

The expectation of the aposteriori distribution (26) is found by numerically integrating

$$\frac{\int_0^\infty a_j h(a_j)L_j(a_j)da}{\int_0^\infty h(a_j)L_j(a_j)da} \tag{27}$$

As in the case of likelihood optimization (Section 2.3.2) the *S-plus* function *integrate* is employed for the integration. Also the integrals are re-written on the same form as in (22), and a similar algorithm is used to aid the integration routine.

rs.htrp.plot(model.fit, cum=T, splitplot=F, renew=T, hetero=T, add=F, resolution=100, show=T, intervals=1000,...)

Required arguments:

model.fit: output from *rs.model.mle*. (Only data, parameters and fitted model are needed.)

Optional arguments:

cum: if TRUE (default) the cumulative intensity function will be computed. If FALSE the intensity function will be computed.

splitplot: if FALSE (default) the intensities of different systems are plotted in the same plot. If TRUE individual intensity functions are plotted in individual plots.

renew: if TRUE (default) time since last failure is used to compute the intensity functions. If FALSE, the model is treated as if it was a (heterogeneous) NHPP model. (Only the trend function is plotted, the plot will be easier to read.)

hetero: if TRUE (default) the function tries to compute heterogeneity factors for the individual systems (only for heterogeneous models). If FALSE the model is treated as if it was a TRP model. (All heterogeneity factors are 1.)

add: if TRUE the computed function (or functions) is plotted in an already existing plot. If FALSE a new plot will be created.

resolution: the number of points for which the function is computed (and plotted). More points give a smoother plot, default is 100.

show: if TRUE (default) the intensity function will be plotted. If FALSE the function returns a list of vectors of times and the intensity function computed at these times, one for each system.

intervals: number of intervals used in a numerical Simpson's integration for the cumulative intensity function. More points gives a better estimate, default is 1000.

...: additional arguments to function *plot*.

Output

If *show* is FALSE a list with components:

x: a vector of times for which the (cumulative) intensity is computed.

y: the (cumulative) intensity computed at x.

If there are several systems a vector is computed for each system.

Purpose

To compute (and plot) the (cumulative) intensity of models fitted by the function *rs.htrp.mle*. The estimated model can then for instance be compared to the Nelson-Aalen estimator. To extend the usefulness of this function we have added the possibility to suppress certain aspects of the model, like heterogeneity and renewal.

Details

Computes and plots the (cumulative) intensity of a model specified in *model.fit* (including data).

Because the value of the heterogeneity factor a_j is given only in terms of a distribution, a Bayes estimate for this factor is computed for each system if *hetero* is TRUE.

If *hetero* is FALSE the models will be plotted as were they TRPs ($\text{HTRP}(F, \lambda(\cdot), 1)$), i.e. with the multiplicative heterogeneity factor a_j equal to 1 for all systems.

If *renew* is FALSE all the models will be plotted as were they (heterogeneous) NHPP ($\text{HTRP}(exp, \lambda(\cdot), H)$), i.e. without regard to the history of the individual systems.

Output is either a plot or a list of times with corresponding computed intensities. See *rs.htrp.mle* or *rs.htrp.sim* for more details concerning HTRP models.

****

2.3.4 Computing the Residual Process of a Fitted HTRP Model

The function *rs.htrp.res* computes the so-called residual process of a fitted model. This means transforming the observed failure times with the estimated cumulative intensity of the fitted model.

From the definition of a TRP process we have that if $T_1, T_2, \ldots$ are generated by a $\text{TRP}(F, \lambda(\cdot))$ model then

$$\Lambda(T_i) - \Lambda(T_{i-1}) \qquad i = 1, 2, \ldots \tag{28}$$

are i.i.d random variables distributed according to F. (For an NHPP F is exponential(1).) We also have that that if $T_1, T_2, \ldots$ are generated by a $\text{TRP}(F, \lambda(\cdot))$ model then

$$\Lambda(T_i|\mathcal{F}_{T_i-}) - \Lambda(T_{i-1}|\mathcal{F}_{T_{i-1}-}), \qquad i = 1, 2, \ldots \tag{29}$$

are i.i.d random variables from an exponential(1) distribution. This follows from general theory of point processes and is also seen directly from:

$$\begin{aligned}
\Lambda(T_i|\mathcal{F}_{T_i-}) - \Lambda(T_{i-1}|\mathcal{F}_{T_{i-1}-}) &= \int_{T_{i-1}}^{T_i} \lambda(t|\mathcal{F})dt \\
&= \int_{T_{i-1}}^{T_i} z(\Lambda(t) - \Lambda(T_{i-1}))\lambda(t)dt \\
&= \int_0^{\Lambda(T_i)-\Lambda(T_{i-1})} z(y)dy
\end{aligned} \tag{30}$$

which is exponential with parameter 1 since $\Lambda(T_i) - \Lambda(T_{i-1})$ has distribution F and $z(\cdot)$ is the corresponding hazard rate.

So either the E-residual process $\widehat{\Lambda}(T_i|\mathcal{F}_{T_i-})$ or the F-residual process $\widehat{\Lambda}(T_i)$ can be used to check for deviations from the model assumptions, and in fact they are equivalent. The so-called residuals are now the interarrival times of the residual processes. Typical things to check for would be the distribution of the residuals, (deterministic) trend, serial correlation or heterogeneity in the residual processes.

For heterogeneous processes the same approach is used as for the plotting function. The unobserved multiplicative heterogeneity factors, a_j, are estimated by the aposteriori expectation. Another possibility would be to let all systems have the same factor $a_j = 1$. Assuming the model was correct, the E-residual processes from different systems would then follow homogeneous Poisson processes (HPPs) with differing intensity parameter, $\text{HPP}(a_j)$.

Implementation Details See Section 2.3.3.

rs.htrp.res(model.fit, renew=T, hetero=T, ...)

Required arguments

model.fit: output from function **rs.htrp.mle**. (Only data, parameters and fitted model are needed.)

Optional arguments

renew: if FALSE the residuals from a (heterogeneous) TRP model are computed as were they from a (heterogeneous) NHPP model.

hetero: if FALSE the residuals from a heterogeneous model are computed as were they from a model without heterogeneity.

...: additional arguments to function *plot*.

Output:

An *rsframe* containing the residual process. The number of events will be the same as in the original systems.

Purpose

To compute the so called residual process of a model fitted by the function *rs.htrp.mle* as specified in *model.fit*. This process can then be examined to check if the model fits the data. To extend the usefulness of this function we have added the possibility to suppress certain aspects of the model, like heterogeneity and renewal.

Details

The function returns an *rsframe* containing the residual process(es) (i.e. transformed failure times). This residual dataset could then be examined by functions like *rs.tbf*, *rs.ttt*, *rs.kaplan.meier*, *rs.nelson.aalen*, *rs.intens.hist*, *rs.test.trend*, *rs.test.exp* and others for deviations from the model assumptions. It could also be modelled by a HTRP model.

If the model is correct the residual process should be a realization of an $\mathrm{HPP}(exp, 1)$, i.e. the times between arrivals should be exponentially distributed with expectation 1.

If the TRP model is correct and $renew = F$ the residual process will be a realization of an $\mathrm{RP}(F, 1)$, i.e. the times between arrivals should be distributed according to F, the distribution of the interarrival times of the underlying renewal process.

If the chosen heterogeneous model is correct and $hetero = F$ the residual processes will be realizations of $\mathrm{HPP}(a_j)$ with intensities differing by a multiplicative factor distributed according to the specified heterogeneity distribution.

The computation is done using the function *rs.htrp.plot*, see this for computational details.

****

2.3.5 Checking the Distribution of the Residuals of a Fitted HTRP Model

The residuals, i.e. the interfailure times of a residual process, are i.i.d random variables if the fitted model is correct. The E-residuals (29) are exponentially distributed with expectation 1, the F-residuals (28) are distributed according to the F, the distribution of the interarrival times of the underlying renewal process.

The function *rs.htrp.pp* plots

$$\left(\hat{F}(\underline{X}), \hat{F}_{EMP}(\underline{X})\right) \tag{31}$$

where $\hat{F}$ is the (estimated) distribution of the residuals if the model is correct and $\hat{F}_{EMP}$ is an empirical distribution function. $\underline{X}$ is here the vector of (uncensored) ordered residuals of the model.

Implementation Details

The options for the distribution of F are the same as the options for the distribution of the renewal process of the function *rs.htrp.mle* (see Section 2.3.2).

There are two empirical estimator of the distribution function implemented, the Kaplan-Meier estimator and the Nelson estimator (Section 2.4.5), both can handle censored data. For uncensored data the Kaplan-Meier estimator is just the usual empirical distribution function.

rs.htrp.pp(x, dist="exponential", par=c(1), emp="kaplan-meier", line=T, show=T, add=F, ...)

Required arguments

x either an rsframe (as specified in *rs.read.frame*) or a data vector which may have a names attribute indicating a censored observation (see *rs.tbf*). Data from an rsframe will be converted into times between failures.

Optional arguments

dist: the distribution of the residuals (usually) according to the fitted HTRP model. The implemented distributions are *exponential* (default), *gamma*, *weibull* and *bimodal-exponential*.

par: the parameter(s) of the fitted distribution.

emp: type of empirical distribution function estimator, *kaplan-meier* (default) or *nelson*.

line: if TRUE a straight line between (0,0) and (1,1) will be drawn in the plot.

show: if TRUE (default) the probability-probability plot is plotted. If FALSE the result is returned as a list of two vectors.

add: if TRUE the p-p plot is plotted in an already existing plot.

...: additional arguments to function *plot*.

Output:

If *show* is FALSE a list with components:

x: the estimated cumulative distribution (from the fitted model) at at the interarrival times.

y: the empirically estimated cumulative distribution at the interarrival times.

If *show* is TRUE a plot is created on the current graphics device.

Purpose

To plot the (parametrically) estimated (cumulative) distribution of the interarrival times of a residual process versus an empirical estimate of the distribution at the same interarrival times. The result may be used to check the model fit of an HTRP model fitted by *rs.htrp.mle* by checking whether the interfailure times of the residual process of the fitted model have the correct distribution.

Details

If the residuals are computed with the argument $renew = T$ in function *rs.htrp.res* and the model is correct the residuals should be exponentially distributed with expectation 1. Otherwise ($renew = F$) the residuals should follow the distribution of the underlying renewal process of the fitted HTRP model.

The implemented distributions are the same as the the renewal process distributions fitted in *rs.htrp.mle*: *exponential* with expectation 1 (default), 1-parameter *gamma* with expectation 1, 1-parameter *Weibull* with expectation 1 and 2-parameter *bimodal-exponential* with expectation 1. Arguments to this function are the name of the distribution and (for non-exponential distributions) the estimated parameter(s) from output of *rs.htrp.mle*.

For the empirical distribution function there are two options: the *Kaplan-Meier* estimator or the *Nelson* estimator.

The function returns a plot, if the values are close to the diagonal the fit is good.

****

2.3.6 Miscellaneous Functions

Mostly functions that are arguments to other functions, typically functions associated with a distribution or trend function. For example functions that simulate from or compute the probability density of a Weibull distribution.

The Exponential Distribution with Expectation 1

$$\begin{aligned} f(x) &= e^{-x} \\ E(X) &= 1 \qquad Var(X) = 1 \end{aligned}$$

lfexpo1(x, par)

Computes the log of the probability density of the distribution:

$$log(f(x)) = -x$$

lRexpo1(x, par)

Computes the log of the survival function of the distribution:

$$log(1 - F(x)) = -x$$

rexpo1(n, par)

Samples n observations from the distribution:

$$(-log(1 - U))$$

where U is $uniform(0, 1)$ distributed.

A Weibull Distribution with Expectation 1

$$\begin{aligned} f(x|\beta) &= \frac{(\Gamma(\beta+1))^{\frac{1}{\beta}} x^{\frac{1}{\beta}-1} e^{(-(\Gamma(\beta+1)x)^{\frac{1}{\beta}})}}{\beta}, \qquad \beta > 0 \\ E(X) &= 1 \quad Var(X) = \frac{\Gamma(2\beta+1)}{\Gamma^2(\beta+1)} - 1 \end{aligned}$$

lfweibull1(x, par)

Computes the log of the probability density of the distribution:

$$log(f(x|\beta)) = \frac{log\Gamma(\beta+1)}{\beta} + (\frac{1}{\beta} - 1)log(x) - (\Gamma(\beta+1)x)^{\frac{1}{\beta}} - log(\beta)$$

lRweibull1(x, par)

Computes the log of the survival function of the distribution:

$$log(1 - F(x|\beta)) = -(\Gamma(\beta+1)x)^{\frac{1}{\beta}}$$

qweibull1(u, par)

Computes a quantile of the distribution:

$$q(u|\beta) = \frac{(-log(1-u))^{\beta}}{\Gamma(\beta+1)}$$

rweibull1(n, par)

Samples n observations from the distribution by computing:

$$\frac{(-log(1-U))^{\beta}}{\Gamma(\beta+1)}$$

where U is $uniform(0,1)$ distributed.

A Gamma Distribution with Expectation 1

$$f(x|\gamma) = \frac{\gamma^{-\frac{1}{\gamma}} x^{\frac{1}{\gamma}-1} e^{-\frac{x}{\gamma}}}{\Gamma(\frac{1}{\gamma})}, \qquad \gamma > 0$$
$$E(X) = 1 \qquad Var(X) = \gamma$$

lfgamma1(x, par)

Computes the log of the probability density of the distribution:

$$log(f(x|\gamma)) = \frac{-log(\gamma)}{\gamma} + (\frac{1}{\gamma} - 1)log(x) - \frac{x}{\gamma} - log(\Gamma(\frac{1}{\gamma}))$$

lRgamma(x, par)

Computes the log of the survival function of the distribution. This is a function of the incomplete gamma function, computed by $log(1 - pgamma(\frac{x}{\gamma}, \frac{1}{\gamma}))$, or a continued fraction development (Abramowitz & Stegun, 1972) for added accuracy for certain parameter values.

$$log(1 - F(x|\gamma)) = log\left(\frac{\int_{\frac{x}{\gamma}}^{\infty} v^{\frac{1}{\gamma}-1} e^{-v} dv}{\Gamma(\frac{1}{\gamma})}\right)$$

qweibull1(u, par)

Computes the u-quantile of the distribution by solving for x:

$$u = \frac{\int_0^{\frac{x}{\gamma}} v^{\frac{1}{\gamma}-1} e^{-v} dv}{\Gamma(\frac{1}{\gamma})}$$

rgamma1(n, par)

Samples n observations from the distribution by solving

$$F(x|\gamma) - U = \frac{\int_0^{\frac{x}{\gamma}} v^{\frac{1}{\gamma}-1} e^{-v} dv}{\Gamma(\frac{1}{\gamma})} - U = 0$$

for x. Where U is $uniform(0, 1)$ distributed.

A Bimodal-Exponential Distribution with Expectation 1

$$f(x|p,q) = p\alpha_1 e^{-\alpha_1 x} + (1-p)\alpha_2 e^{-\alpha_2 x}, \qquad 0 < p,q < 1$$
$$\text{where} \quad \alpha_1 = \frac{p(q-1)+1}{q} \qquad \text{and} \qquad \alpha_2 = p(q-1)+1$$
$$E(X) = 1 \qquad Var(X) = (p-p^2)(\alpha_1^{-2} + \alpha_2^{-2} - 2(\alpha_1\alpha_2)^{-1})$$

lfbimodexpo1(x, par)

Computes the log of the probability intensity of the distribution.

$$log(f(x|p,q)) = log(p\alpha_1 e^{-\alpha_1 x} + (1-p)\alpha_2 e^{-\alpha_2 x})$$

lRbimodexpo1(x, par)

Computes the log of the survival function of the distribution.

$$log(1 - F(x|p,q)) = log(pe^{-\alpha_1 x} + (1-p)e^{-\alpha_2 x})$$

rbimodexpo1(n, par)

Samples n observations from the distribution by solving

$$F(x|p,q) - U = 1 - pe^{-\alpha_1 x} - (1-p)e^{-\alpha_2 x} - U = 0$$

for x, where U is $uniform(0,1)$ distributed.

The Homogeneous Trend (Transformation) Function

$$\lambda(t|a) = a, \qquad a > 0$$

<u>**lhomtrend(t, par)**</u>

Computes the logarithm of the intensity.

$$log(\lambda(t|a)) = log(a)$$

<u>**Homtrend(t1, t2, par)**</u>

Computes the difference in cumulative intensity at two points:

$$\Lambda(t_2|a) - \Lambda(t_1|a) = \int_{t_1}^{t_2} \lambda(u)du = a(t_2 - t_1)$$

<u>**Homtrend.inv(s, par)**</u>

Computes the inverse of the cumulative intensity at s.

$$\Lambda^{-1}(s|a) = \frac{s}{a}$$

The Power Law Trend (Transformation) Function

$$\lambda(t|a, b) = abt^{b-1}, \qquad a, b > 0$$

<u>**lpowlaw(t, par)**</u>

Computes the logarithm of the intensity.

$$log(\lambda(t|a, b)) = log(a) + log(b) + (b - 1)log(t)$$

<u>**Powlaw(t1, t2, par)**</u>

Computes the difference in cumulative intensity at two points:

$$\Lambda(t_2|a, b) - \Lambda(t_1|a, b) = a(t_2^b - t_1^b)$$

<u>**Powlaw.inv(s, par)**</u>

Computes the inverse of the cumulative intensity at s.

$$\Lambda^{-1}(s|a, b) = \left(\frac{s}{a}\right)^{\frac{1}{b}}$$

The Log Linear Trend (Transformation) Function

$$\lambda(t|a,c) = ae^{ct}, \qquad a > 0, \;\; -\infty < c < \infty$$

lloglin(t, par)

Computes the logarithm of the intensity.

$$log(\lambda(t|a,c)) = log(a) + ct$$

Loglin(t1, t2, par)

Computes the difference in cumulative intensity at two points:

$$\Lambda(t_2|a,c) - \Lambda(t_1|a,c) = \frac{a}{c}(e^{ct_2} - e^{ct_1})$$

Loglin.inv(s, par)

Computes the inverse of the cumulative intensity at s.

$$\Lambda^{-1}(s|a,c) = \frac{\log\left(\frac{sc}{a} + 1\right)}{c}$$

The Log Linear Power Law Trend (Transformation) Function

$$\lambda(t|a,b,c) = abt^{b-1}e^{ct}, \qquad a,b > 0, \;\; -\infty < c < \infty$$

llogpow(t, par)

Computes the logarithm of the intensity.

$$log(\lambda(t|a,b,c)) = log(a) + log(b) + (b-1)log(t) + ct$$

Logpow(t1, t2, par)

Computes the difference in cumulative intensity at two points:

$$\Lambda(t_2|a,b,c) - \Lambda(t_1|a,b,c) = \int_{t_1}^{t_2} abt^{b-1}e^{ct}dt$$

Can be solved with incomplete gamma functions for $c < 0$ (see *lRgamma* above), numerical integration and/or Taylor series for $c > 0$.

Logpow.inv(s, par)

Computes the inverse of the cumulative intensity at s by solving

$$\Lambda(t|a,b,c) - s = 0$$

for t.

The Linear Trend (Transformation) Function

$$\lambda(t|d,e) = (d + et) \vee 0, \qquad -\infty < d, e < \infty$$

llinear(t, par)

Computes the logarithm of the intensity.

$$log(\lambda(t|d,e)) = log(d + et)$$

Linear(t1, t2, par)

Computes the difference in cumulative intensity at two points:

$$\Lambda(t_2|d,e) - \Lambda(t_1|d,e) = d(t_2 - t_1) + e(t_2^2 - t_1^2)$$

Linear.inv(s, par)

Computes the inverse of the cumulative intensity at s.

$$\Lambda^{-1}(s|a,b) = \frac{-d + \sqrt{d^2 + 2es}}{e}$$

A Simpson's Numerical Integration Function

integrate.simp(f, lower, upper, n=100, ...)

Required arguments

f: the integrand.

Optional arguments

lower: lower integration limit, default is $-\infty$.

upper: upper integration limit, default is ∞.

n: the number of intervals is $2n$.

...: additional arguments to *f*.

Output

a list with the item *integral* containing the numerical approximation.

Purpose

To compute an approximation of the integral of *f(x,...)*, from *lower* to *upper*.

Details

Uses Simpson's method with $2n$ intervals. Additional arguments to *f* can be supplied via ....

****

A Simplex Optimizer Function

amoebamin(start, funk, ftol=1e-10, upper, lower, max.iter=1000, ...)

Required arguments

start: a vector of starting values for the parameters.

funk: the function to be minimized

lower: a vector of lower bounds for the parameters.

upper: a vector of upper bounds for the parameters.

Optional arguments

ftol: absolute function convergence tolerance.

max.iter: maximum number of iterations.

...: additional arguments to *funk*.

Output

A list with the following components:

parameters: final parameter values of the optimization.

loglik: the final value of *funk*.

lower: lower bounds for the parameters.

upper: upper bounds for the parameters.

nfunk: the number of function calls.

message: a statement of the reason for termination.

Purpose

To minimize a function in n dimensions.

Details

A simplex-type function minimizer in n dimensions. This function is a slightly modified S-plus translation of the function *amoeba* in Press *et al.* (1992). Given a vector of starting values (*start*) and *lower* and *upper* limits a minimum is found for the function *funk*. If the given limits are not $\pm\infty$ a simple internal transformation takes place. A start simplex is made by adding or subtraction 5% around the start point. Arguments can be transferred to *funk* via ....

amoebamin returns the result of the optimization, the number of iterations etc.

****

2.4 Graphical and Non-Parametric Functions

Includes non-parametric estimates of intensity and cumulative intensity in an NHPP model, the total time on trend transform for systems and i.i.d. data, the Kaplan-Meier and Nelson estimators for survival probability and some functions for detecting serial dependency in repairable systems data.

2.4.1 The Nelson-Aalen Estimator for Cumulative Intensity

The Nelson-Aalen estimator (Nelson, 1969),(Aalen, 1978) of cumulative intensity of a (multiple) counting process following a multiplicative intensity model (see Andersen *et al.* (1993)):

$$\hat{\Lambda}(t) = \sum_{T_{ji} \leq t} \frac{1}{Y(T_{ji})} \tag{32}$$

where $Y(T_{ji})$ is the number of processes (systems) operating immediately before time T_{ji}. $\hat{\Lambda}(t) = 0$ for $t \leq \min_{j,i} T_{ji}$.

rs.nelson.aalen(data, splitplot=F, show=T, add=F, ...)

Required arguments

data: counting process data on the form specified by the function *rs.read.frame*.

Optional arguments

splitplot: if TRUE the function computes and plots the estimator for each system in individual plots.

show: if TRUE (default) the estimator will be plotted. If FALSE a list giving failure times and number of systems alive will be returned.

add: if TRUE the estimator is plotted in an already existing plot.

...: additional arguments to *plot*.

Output

If *show* is FALSE a list with components:

failtimes: failure times.

numsys: number of systems alive at time of the failure.

cumintens: estimated cumulative intensity at time of the failure.

obsstart: time of observation start.

obsstop: time of observation stop.

If *splitplot* is TRUE these are lists of vectors, one for each system.
If *show* is TRUE the estimator is plotted.

Purpose

To give a non-parametric estimate of the cumulative intensity by the Nelson-Aalen estimator. The result may for example be compared to parametric models estimated by function *rs.htrp.mle*.

Details

Computes and plots the Nelson-Aalen estimator of cumulative intensity for counting process data on the form specified by the function *rs.read.frame*

If *splitplot* is TRUE the function computes and plots the estimator for each system in individual plots. If *add* is TRUE it will be plotted in an already existing plot. If *show* is FALSE it will not be plotted.

Output is a list or a plot of the estimated cumulative intensity function.

To get estimators for several individual systems in the same plot compute the estimator for one system at the time and use $add = T$.

****

2.4.2 A Histogram-Like Intensity Estimator

rs.intens.hist(data, nclass, breaks, splitplot=F, show=T, ...)

Required arguments

data: counting process data on the form specified by the function *rs.read.frame*.

Optional arguments

nclass: the number of classes in the histogram. The default is a number proportional to the logarithm of the number of failures in *data*.

breaks: vector of break points for the bars of the histogram.

splitplot: if TRUE the function computes (and plots) an intensity estimate for each system.

show: if TRUE the intensity histogram will be plotted.

...: additional arguments to *hist*

Output

If *show* is TRUE a vector containing the coordinate of the centre of each box is returned. if *show* is FALSE the function returns a list with components:

intensity: intensity in each class of the intensity histogram.

breaks: break points between intensity histogram classes.

If *splitplot* is TRUE *intensity* and *breaks* will be list of vectors, one for each system.

Purpose

Computes and plots a histogram-like intensity estimate for counting process data on the form specified by the function *rs.read.frame.*

Details

The function is based on the S-plus function *hist.* It computes a simple histogram-like intensity estimate by counting the number of events within each class and divide it by the length of the class. Several (parallel) systems will give a better estimate. There is no smoothing. See Lundal (1994) for details.

The number of classes (i.e. bars) in the *histogram* is given by *nclass* . *breaks* is a vector of break points. If *splitplot* is TRUE an individual estimate will be computed (and plotted) for each system. If *show* is FALSE a plot will not be created.

Output is a list or a plot of the estimated intensity function. The result may be compared to parametric models estimated by function *rs.htrp.mle.*

****

2.4.3 Functions for Detecting Serial Dependence

A graphical function for plotting X_i against X_{i-1} and a function that computes an estimate of the autocorrelation coefficient.

rs.lagplot(data, lag=1, ...)

Required arguments

data: counting process data on the form specified by the function *rs.read.frame*.

Optional arguments

lag: the lag k at which to plot X_i against X_{i+k}. Default is 1.

...: additional arguments to *plot*.

Output

A lagplot is created on the graphics device.

Purpose

To check for dependencies between interfailure times at *lag* 1, 2 etc. Can for example be used to examine the residual processes as part of a model checking procedure.

Details

Plots X_i against X_{i+k} for $k = lag$ for data given by *data*. For multi-system data $(X_{ji}, X_{j,i+k}),\ \ j = 1, \ldots, m$ are plotted in the same plot.

****

rs.acf(data, lag=2)

Required arguments

data: counting process data on the form specified by the function *rs.read.frame.*

Optional arguments

lag: the maximum lag, at which the autocorrelation coefficient is computed.

Output:

A list with the following components:

acf: vector of the autocorrelation coefficient computed at lags $1, \ldots, lag$.

stdev: vector of the estimated standard deviation for the computed coefficients.

sysnum: vector of the number of systems used at the specified lags.

Purpose

To check for serial correlation between interfailure times at lags $1, 2, \ldots$. Can for example be used to examine the residual processes in a model checking procedure.

Details

Computes autocorrelation coefficient at lag k for $k = 1, \ldots, lag$. Returns a list of autocorrelation coefficients and estimated standard deviation.

In case of multiple systems (processes) the autocorrelation coefficient is computed for each system and then added together. As the number of failures in the systems may differ we often cannot use all systems for a specified lag. Therefore the number of systems used in computing the coefficients are returned. The estimate of standard deviation is adjusted for this. The following expression is used to compute the autocorrelation coefficient for one system at lag k:

$$\widehat{\rho(k)} = \frac{n}{n-k} \frac{\sum_{i=1}^{n-k}(X_{i+k} - \overline{X})(X_i - \overline{X})}{\sum_{i=1}^{n}(X_i - \overline{X})^2} \tag{33}$$

$\widehat{\rho(k)}$ is unbiased but may be greater than ± 1. If there is no autocorrelation it is asymptotically normally distributed with expectation 0 and asymptotic variance $\frac{1}{n}$

For several systems the mean of the individual coefficients is returned. The number of systems may vary if the number of failures in each system are not the same.

For reference see Lundal (1994).

****

2.4.4 The Total Time on Test Transform for Systems

The TTT plot for repairable systems data was introduced by Barlow & Davis (1977) for the NHPP model. It is based on the total time on test transform, which is the total time, all systems added together, have been under observation until time t.

Assume that m independent NHPPs with common intensity function $\lambda(t)$ are under observation. The observation intervals $(a_j, b_j]$ are contained in the interval $(0, S]$. Let $N = \sum_{j=1}^{m} n_j$ and let the arrival times of the superimposed process be $S_1, S_2, \ldots S_N$ If we let $p(u)$ denote the number of systems under observation at time u, the total time on test from 0 to t is:

$$\mathcal{T}(t) = \int_0^t p(u)du \tag{34}$$

The scaled total time on test statistic is the total time on test at time t relative to the total time on test for the whole observation period:

$$\frac{\mathcal{T}(t)}{\mathcal{T}(S)} = \frac{\int_0^t p(u)du}{\int_0^S p(u)du} \tag{35}$$

The (scaled) TTT plot is the plot of the scaled total time on test statistic at the time of the k'th failure of the superposed process versus scaled failure number:

$$\left(\frac{\mathcal{T}(S_k)}{\mathcal{T}(S)}, \frac{k}{N}\right), \quad k = 1, \ldots, N \tag{36}$$

If all systems are observed for the same time period this TTT plot is just a scaled Nelson-Aalen plot with the axes interchanged.

If there is "no trend" (HPP) in the data the TTT plot will be located near the main diagonal. Different shapes of the TTT plot may give indications of different types of trend.

rs.ttt(data, splitplot=T, show=T, add=F, line=T, ...)

Required arguments

data: counting process data on the form specified by the function *rs.read.frame*.

Optional arguments

splitplot: if TRUE the function computes and plots an estimator for each system individually.

show: if TRUE (default) the estimator will be plotted. If FALSE a list giving the scaled TTT statistic and the scaled failure number at the times of failure will be returned. (Can be used to draw the TTT plot.)

add: if TRUE the TTT plot will be plotted in an already existing plot.

line: if TRUE a straight line between (0,0) and (1,1) will be drawn in the plot.

...: additional arguments to *plot*.

Output

If show is FALSE, a list with components:

x: (failure number)/(total number of failures).

y: the scaled TTT statistic.

If *splitplot* is TRUE x and y are lists of vectors, one for each system. If *show* is TRUE a plot is created on the current graphics device.

Purpose

To create the (scaled) TTT plot for repairable systems. The plot may be used to check for trends in the failure times. Also used for TTT trend tests in Section 2.5.1.

Details

Computes (and plots) the (scaled) TTT plot for repairable systems.

If *splitplot* is TRUE a plot is created for each system. If *add* is TRUE it will be plotted in an already existing plot. If *show* is FALSE output is a list that can be plotted.

****

2.4.5 Functions for Analysis of Interfailure Times

The function *rs.tbf* converts repairable systems data in the form of an *rsframe* into a vector of interfailure times. The function *rs.ttt.tbf* computes the (standard) total time on test plot for independent and identically distributed possibly censored data. The functions *rs.kaplan.meier* and *rs.nelson* compute the Kaplan-Meier and Nelson estimators of survival probability for i.i.d. data.

The TTT estimator for i.i.d data can be seen as a special case of the TTT estimator for systems data (34-36) where each "system" is observed only until the first failure. For exponential distribution the plot will be located near the diagonal. A concave plot indicates an increasing failure rate while a convex plot indicates a decreasing failure rate.

The Kaplan-Meier estimator, $\hat{R}_{KM}(x)$ is a non-parametric estimator of survival probability at time x. The result is a right continuous function where $\hat{R}_{KM}(0) = 1$. It is defined (Kaplan & Meier, 1958) as

$$\hat{R}_{KM}(x) = \prod_{k \in K_x} \frac{(n_k - 1)}{n_k} \tag{37}$$

where K_x is the set of all indices k where $X_{(k)} \leq x$, $X_{(k)}$ represents an uncensored interarrival time and n_k is the number of units under observation immediately before time $X_{(k)}$. If a censored interarrival time is the largest this will not be a proper distribution.

The Nelson estimator, $\hat{R}_N(x)$ is another non-parametric estimator of survival probability at time x. The result is again a right continuous function where $\hat{R}_N(0) = 1$. It is defined (Nelson, 1969) as

$$\hat{R}_N(x) = exp\left\{ -\sum_{k \in K_x} \frac{1}{n_k} \right\}. \tag{38}$$

This will not be a proper distribution as it will never be 0.

rs.tbf(data, addcen=F)

Required arguments

data: a data frame for counting process data as specified in *rs.read.frame*. May contain data from several systems. The systems can be time truncated (censored last interfailure time).

Optional arguments

addcen: if TRUE censored observations are added to the next uncensored observation. If FALSE (default) a *names* attribute is added indicating if an observation is censored (T) or not (F).

Output

A vector of times between failures. If *addcen* is FALSE, with a *names* attribute indicating whether it was censored or not.

Purpose

To compute a vector of interfailure times (times between failures) for repairable systems data in *rsframe* format.

Details

The single output vector contains interfailure times from all systems in *data*.

If *addcen* is TRUE censored observations are added to the next uncensored. (Or the first if the last observation is censored.) If the interfailure observations in *data* are independent and identically exponentially distributed the resulting dataset will have the same properties.

If *addcen* is FALSE the output vector will have a *names* attribute which indicates if an observation is censored (T) or not (F).

****

rs.ttt.tbf(x, add=F, show=T, line=T,...)

Required arguments

x: either counting process data on the form specified by the function *rs.read.frame* or a data vector which may have a names attribute indicating a censored observation (see *rs.tbf*). Data from an *rsframe* will be converted into times between failures.

Optional arguments

show: if TRUE (default) the estimator will be plotted. If FALSE a list giving the scaled TTT statistic and the scaled failure number at the times of failure will be returned. (Can be used to draw the TTT plot.)

add: if TRUE the TTT plot will be plotted in an already existing plot.

line: if TRUE a straight line between (0,0) and (1,1) will be drawn in the plot.

...: additional arguments to *plot*.

Output

If show is FALSE, a list with components:

x: (failure number)/(total number of failures).

y: the scaled TTT statistic.

If *show* is TRUE a plot is created on the current graphics device.

Purpose

To create the (scaled) TTT plot for possibly censored i.i.d data. The plot may be used to check the distribution of interfailure times, especially for a residual process (Section 2.3.4). Also used for tests for exponential distribution in Section 2.5.2.

Details

Computes (and plots) the (scaled) TTT plot.

If *add* is TRUE it will be plotted in an already existing plot. If *show* is FALSE output is a list that can be plotted.

****

rs.kaplan.meier(x, add=F, show=T, ...)

Required arguments

x: either counting process data on the form specified by the function *rs.read.frame* or a data vector which may have a names attribute indicating a censored observation (see *rs.tbf*). Data from an *rsframe* will be converted into times between failures.

Optional arguments

show: if TRUE (default) the Kaplan-Meier estimator will be plotted. If FALSE a list from which the estimator can be plotted is returned.

add: if TRUE the Kaplan-Meier estimator will be plotted in an already existing plot.

...: additional arguments to *plot*.

Output

If *show* is FALSE a list with the following components:

x: the sorted uncensored interfailure times.

y: the value of the estimator at x.

If *show* is TRUE a plot is created on the current graphics device.

Purpose

To compute the Kaplan-Meier estimator of survival probability for possibly censored data. In a repairable systems situation it may be employed to examine the distribution of the interfailure times, especially the so-called residuals, the interfailure times of a residual process. (Section 2.3.4.)

Details

Computes and plots the Kaplan-Meier estimator of survival probability. Input data is either an *rsframe* or a vector which may have a *names* attribute indicating a censored observation (see *rs.tbf*).

If *show* is TRUE the estimator will be plotted. If *add* is TRUE it will be plotted in an already existing plot. Output is a list or a plot of the Kaplan-Meier estimator.

****

rs.nelson(x, add=F, show=T, ...)

Required arguments

x: either counting process data on the form specified by the function *rs.read.frame* or a data vector which may have a names attribute indicating a censored observation (see *rs.tbf*). Data from an *rsframe* will be converted into times between failures.

Optional arguments

show: if TRUE (default) the Nelson estimator will be plotted. If FALSE a list from which the estimator can be plotted is returned.

add: if TRUE the Nelson estimator will be plotted in an already existing plot.

...: additional arguments to *plot*.

Output

If *show* is FALSE a list with the following components:

x: the sorted uncensored interfailure times.

y: the value of the estimator at x.

If *show* is TRUE a plot is created on the current graphics device.

Purpose

To compute the Nelson estimator of survival probability for possibly censored data. In a repairable systems situation it may be employed to examine the distribution of the interfailure times, especially the so-called residuals, the interfailure times of a residual process. (Section 2.3.4.)

Details

Computes and plots the Nelson estimator of survival probability. Input data is either an *rsframe* or a vector which may have a *names* attribute indicating a censored observation (see *rs.tbf*).

If *show* is TRUE the estimator will be plotted. If *add* is TRUE it will be plotted in an already existing plot. Output is a list or a plot of the Nelson estimator.

****

2.5 Tests

2.5.1 Trend Tests

The function *rs.test.trend* tests for trend in a dataset by several different tests. We say that there is a trend if the interfailure times of a repairable system tends to get either shorter or longer (monotonic trends), or they may follow any other non-stationary model (non-monotonic trends). A system has no trend if the marginal distributions of all interarrival times are identical. In the HTRP case a trend would mean a non-homogeneous trend function, $\lambda(t)$.

Trend tests can be categorized by their null and alternative hypotheses. Ascher & Feingold (1984) consider three different null hypotheses: a) homogeneous Poisson process (HPP), b) renewal process (RP) and c) general stationary sequence (GSS). The possible alternative hypotheses are split into two: I) Monotonic trend (MT), and II) Non-monotonic trend (NT). Many of the standard tests in the reliability field are tests for HPP against MT. A possible problem with these tests is a lack of robustness versus other stationary processes like renewal processes (Lindqvist, Kjønstad & Meland, 1994).

Tests for Trend in One System

Most tests for trend in repairable systems are based on the Poisson process assumption. Three such tests have been implemented in REIDAR, the Laplace test, the Anderson-Darling test and the so-called military handbook test. Most of the tests are defined for one system observed either until n failures, or until time b is reached. To get an efficient notation, we define $\hat{n}$ as

$$\hat{n} = \begin{cases} n & \text{if the process is time truncated} \\ n-1 & \text{if the process is failure truncated} \end{cases}$$

where for time truncated processes $n = N(b)$.

The Laplace Test

The null hypothesis of this test is HPP and the alternative is a NHPP with monotonic intensity function. It is well known (Bain, Engelhardt & Wright, 1985), (Cox & Lewis, 1966) that this test is an optimal test if the alternative hypothesis is an NHPP of the log-linear type. The test statistic is

$$L = \frac{\sum_{i=1}^{\hat{n}} T_i - \frac{1}{2}\hat{n}(b+a)}{\sqrt{\frac{1}{12}\hat{n}(b-a)^2}} \tag{39}$$

It is approximately standard normally distributed under the null hypothesis, this approximation is said to be very good (Ascher & Feingold, 1984) if $n \geq 3$. The null hypothesis is rejected for small or large values of L. If $L > 0$ ($L < 0$) it is an indication of an increasing (decreasing) trend.

The Military Handbook Test

The null hypothesis of this test is HPP and the alternative is an NHPP with monotonic trend. This test is an optimal test if the alternative hypothesis is a NHPP of the power-law type with increasing intensity (Bain *et al.*, 1985). The test statistic is

$$MH = 2\sum_{i=1}^{\hat{n}} \ln\left(\frac{b-a}{T_i - a}\right) \tag{40}$$

It is $\chi^2_{2(n-1)}$-distributed under the null hypothesis (*MIL-HDBK-189, Reliability Growth Management*, 1981). The null hypothesis is rejected for small or large values of Z. Large (small) MH is an indication of increasing (decreasing) trend.

The Anderson-Darling Test

This is a test studied in Kvaløy & Lindqvist (1998). The null hypothesis of this test is HPP and the alternative is a trend, possibly non-monotonic or any other deviation from a homogeneous Poisson process. The test statistic is

$$\begin{aligned} AD &= -\frac{1}{\hat{n}}\left[\sum_{i=1}^{\hat{n}}(2i-1)(\ln\left(\frac{T_i-a}{b-a}\right)\right. \\ &\quad \left. +\ln(1-\frac{T_{\hat{n}+1-i}-a}{b-a}))\right]-\hat{n} \end{aligned} \tag{41}$$

For the distribution of AD see Kvaløy & Lindqvist (1998). If AD is large the null hypothesis is rejected, a visual inspection of the TTT transform can give an indication of the type of trend.

For the last two tests the null hypothesis is a renewal process (RP):

The Lewis-Robinson Test

The null hypothesis of this test is RP and the alternative is a monotonic trend. It is a modification of the Laplace test proposed by Lewis & Robinson (1974) where L is divided by the estimated coefficient of variation of the interarrival times, $\widehat{CV(X)}$. The test statistic is thus

$$LR = \frac{L}{\widehat{CV(X)}}. \tag{42}$$

Under the RP assumption LR will be asymptotically standard normally distributed.

The Mann Test

This is a rank test developed by Mann (1945). The null hypothesis of this test is RP and the alternative is a monotonic trend, more exactly a sequence of independent random variables with monotonically increasing or decreasing expectation. The test statistic is computed by counting the number of reverse arrangements, M, among the interarrival times, $X_1, X_2, \ldots X_n$. A reverse arrangement exists if $X_i < X_k$ for $i < k$:

$$M = \sum_{i=1}^{n-1}\sum_{k=i+1}^{n} I(X_i < X_k) \tag{43}$$

M is approximately normally distributed for $n \geq 10$, there exists tables for $n < 10$. In REIDAR the p-values are always computed under the assumption of normality, to get more accurate estimates for small samples permutation methods may be employed.

Tests for Trend in Several Systems

In the case of several systems case there are at least two ways of generalising the Poisson tests. One possibility is to combine test statistics for each individual system into a combined trend test for the whole dataset. Note that for Poisson assumption tests we are then testing the null hypothesis of possibly different HPPs, i.e. each system may follow its own homogeneous Poisson process. In accordance with Kvaløy & Lindqvist (1998) these tests are called *combined tests.*

The other method consists of testing with the original single system tests for trend, but where the the original dataset are replaced by the TTT transformation . Unless all observation periods are equal, this corresponds to a null hypothesis of a common HPP for all the original systems. Again following Kvaløy & Lindqvist (1998) these tests are called *TTT tests.* Because the null hypothesis for TTT tests is more restrictive than for the combined tests, TTT tests are generally stronger.

Resampled Tests for Trend

For standard Poisson assumption tests for trend the rejection of H_0 does not necessarily indicate a trend, just that it is not an HPP. An RP or any other GSS may cause rejection, even if a trend is not present. This is especially true for the Anderson-Darling test which is constructed to detect deviations from the homogeneous Poisson process. In this situation tests like the Lewis-Robinson test and the non-parametric Mann test (where the null hypothesis is RP) can be used.

Another approach is to robustify the Poisson tests by resampling them under a more general null hypothesis (Elvebakk (1998) or Elvebakk (1999)). There are two resampling methods implemented in REIDAR, both dealing with the RP null hypothesis situation:

Permutation

This method can be used only for failure truncated systems. Resampling is performed by permuting the interfailure times of an observed system. For each new sample the test statistic is computed, and we thus get a distribution under the extended H_0 from which a p-value can be computed by standard Monte Carlo methods.

Bootstrap

For failure truncated systems this method is only a slight adjustment of the permutation method. A bootstrapped sample (with replacements) is used instead of a permuted sample. For time truncated systems resampling is performed by sampling from the empirical distribution of the interfailure times until the time of observation stop for the original data. Unless the resampled system contains the same number of failures as the original systems it is rejected. The Kaplan-Meier estimator is used as an empirical distribution.

Note that in case of several systems there is a difference in the resampling procedure for *combined* and *TTT tests*. For TTT tests interfailure times from all systems are pooled to construct our empirical distribution. For combined tests each system is resampled individually.

rs.test.trend(data, type="all", aggr=F, resample="no", resnum=1000)

Required Arguments

data: counting process data on the form specified by the function *rs.read.frame*.

type: vector indicating which tests should be used. If "laplace" is an element of *type* the function returns the p-value for the Laplace test, likewise for "military-handbook", "mann", "lewis-robinson" and "anderson-darling". If type="all" (default) p-values for all implemented test are returned.

aggr: if aggr = F (default) a test statistic is computed for each system which are then combined and a combined p-value is computed. If aggr = T the ttt-transformation of the dataset is used as the testing object.

resample: if resample = "no" standard no-resampling tests are performed. If resample = "permutation" a resampling test is performed based on permuted versions of the original dataset. If resample = "bootstrap" a resampling test is performed based on bootstrapped versions of the original dataset.

resnum : the number of resampling replications, default is 1000.

Output

A data frame with test statistics (+ degrees of freedom in case of chi-square distribution), p-values against trend and p-values against increasing trend (where possible) for the specified tests with the input dataset.

Purpose

To test a given dataset for trends in the interfailure times using one or more of the implemented trend tests. For datasets with several systems two different methods of generalising these tests are implemented. The test can be robustified against renewal processes by resampling.

Details

The function performs a trend test by any of the methods in the *type* argument.

If *aggr* is FALSE *combined* tests are used. Test statistics are computed for each system individually and then combined. Under H_0 each system must then follow a (possibly different) HPP. If *aggr* is TRUE *TTT* tests are used. The test statistics are then computed for the TTT transformation (*rs.ttt*) of the original dataset. Under H_0 each system must follow a common HPP.

If *resample* is "no" (default) no resampling is performed. If *resample* is "permutation" the function uses a permutation method to compute p-values for the tests. For the Poisson tests H_0 is then extended to RP. If *resample* is "bootstrap" the function uses a bootstrap method to compute p-values for the tests. For the Poisson tests H_0 is again extended to RP. Resampling tests may be slow, especially for larger datasets.

The argument *resnum* is the number of resampling replications executed to construct the distribution of the test statistic under H_0.

****

2.5.2 Tests for Non-Exponential Distribution

These tests are just special cases of the TTT tests for trend in a Poisson process of Section 2.5.1. For exponentially distributed i.i.d data, the arrival times of the standard TTT transformation (Section 2.4.5) will be distributed according to a uniform distribution. Therefore the standard tests for trend in Poisson processes may be used to test for deviations from exponential distribution of possibly censored i.i.d data.

The main intension of theses tests is to test the assumption of exponentially distributed interarrival times of the residual processes of Section 2.3.4.

rs.test.exp(data, type="all", combine=F)

Required Arguments

x: either counting process data on the form specified by the function *rs.read.frame* or a data vector which may have a names attribute indicating a censored observation (see *rs.tbf*). Data from an *rsframe* will be converted into times between failures.

type: vector indicating which tests should be used. If "laplace" is an element of *type* the function returns the p-value for the Laplace test, likewise for "military-handbook" and "anderson-darling". If type="all" (default) p-values for all the test are returned.

combine: if FALSE (default) interfailure times from all processes are used to compute the TTT-transformation (see details). If TRUE a ttt-transform and test statistic are computed for each individual system separately. The test statistics are then combined to give a combined test statistic for the whole dataset. *combine* can only be used for data on the form specified by the function rs.read.frame'

Output

A data frame with test statistics, p-values against non-exponential distribution and p-values against increasing failure rate (where possible) for the specified tests with the input dataset.

Purpose

To test a given dataset for deviations from exponential distribution. The dataset may be either a vector of observations (with a names attribute to indicate censoring) or an *rsframe*. In the latter case the interarrival times are tested.

Details

These are originally tests for trend, here used as tests for deviations from exponential distribution. See the function *rs.test.trend* for details on the test statistics. The null hypothesis is an exponential distribution. The Laplace and military handbook tests are tests against monotonically increasing or decreasing failure rate, the Anderson-Darling test will detect more general deviations from an exponential distribution.

The testing procedure consists of computing the (standard) ttt-transformation for the dataset. If the null hypothesis is true this can be regarded as a realization from a homogeneous Poisson process, and tests for trend in Poisson processes are therefore used to test for deviations from an exponential distribution.

If *combine* is TRUE and x is a multi system *rsframe*, TTT transformation and testing are done for each system separately. The test statistics are then combined into an overall test statistic for the whole dataset. The null hypothesis is then that the interfailure times from different systems are distributed according to exponential distributions with possibly differing parameters (heterogeneity between systems). In the case of no heterogeneity between systems this is a weaker test than the test above.

****

References

Aalen, O. (1978). Nonparametric inference for a family of counting processes, *Annals of Statistics* **6**: 721–728.

Abramowitz, M. & Stegun, I. A. (1972). *Handbook of Mathematical Functions*, Dover Publications, New York.

Andersen, P. K., Borgan, Ø., Gill, R. D. & Keiding, N. (1993). *Statistical Models Based on Counting Processes*, Springer-Verlag, New York.

Ascher, H. & Feingold, H. (1984). *Repairable Systems Reliability. Modeling, Inference, Misconceptions and their Causes*, Marcel Dekker, New York.

Bain, M. J., Engelhardt, F. & Wright, F. T. (1985). Tests for increasing trend in the intensity of a poisson process: A power study, *Journal of The American Statistical Association* **80**: 419–422.

Barlow, R. E. & Davis, B. (1977). Analysis of time between failures of repairable components, *in* J. B. Fussel & G. R. Burdick (eds), *Nuclear Systems Reliability Engineering and Risk Asessment*, SIAM, Philadelphia, pp. 543–561.

Brandt, H., Korsaksel, M. & Rosland, G. (1995). Statistisk modellering og inferens for reparerbare systemer, Project work, Norwegian Institute of Technology. In Norwegian.

Cox, D. R. & Lewis, P. A. (1966). *The Statistical Analysis of Series of Events*, Methuen, London.

Elvebakk, G. (1998). Robustification of trend tests by resampling techniques, *in* S. Lydersen, G. A. Hansen & H. A. Sandtorv (eds), *Safety and Reliability, Proceedings of the European Conference of Safety and Reliability - ESREL'98*, Vol. 2, The European Safety and Reliability Association, Balkema, Rotterdam, Netherlands, pp. 1365–1372.

Elvebakk, G. (1999). Extending the use of some traditional trend tests for repairable systems by resampling techniques, Paper II of this thesis.

Elvebakk, G., Lindqvist, B. H. & Heggland, K. (1999). The trend renewal process for statistical analysis of repairable systems, Paper I of this thesis.

Follmann, D. A. & Goldberg, M. S. (1988). Distinguishing heterogeneity from decreasing hazard rates, *Technometrics* **30**: 389–396.

Heggland, K. (1994). Analyse av reparerbare systemer: Modellering av heterogenitet, spesielt innen trend-fornyelsesprosesser, Project work, Norwegian Institute of Technology. In Norwegian.

Kaplan, E. L. & Meier, P. (1958). Nonparametric estimation from incomplete observations, *Journal of the American Statistical Association* pp. 457–481.

Kvaløy, J. T. & Lindqvist, B. H. (1998). TTT-based tests for trend in repairable systems data, *Reliability Engeneering and System Safety* **60**: 13–28.

Lewis, P. A. & Robinson, D. W. (1974). Testing for monotone trend in a modulated renewal process, *in* F. Proschan & R. J.Serfling (eds), *Reliability and Biometry*, SIAM, Philadelphia, pp. 163–182.

Lindqvist, B. H., Kjønstad, G. A. & Meland, N. (1994). Testing for trend in repairable systems data, *Proceedings of ESREL'94*, The European Safety and Reliability Association, La Baule, France.

Lundal, O. K. (1994). Grafisk analyse av reparerbare systemer ved hjelp av splus, Project work, Norwegian Institute of Technology. In Norwegian.

Mann, H. B. (1945). Nonparametric tests against trend, *Econometrica* **13**: 245–259.

Nelson, W. (1969). Hazard plotting for incomplete failure data, *Journal of Quality Technology* **1**(1): 27–52.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*, 2 edn, Cambridge University Press, Cambridge, UK.

Tveit, I. M. (1993). Heterogenitet i pålitelighetsanalyse, Project work, Norwegian Institute of Technology. In Norwegian.

MIL-HDBK-189, Reliability Growth Managment (1981). Headquaters. U.S. Army Communications Research and Development Command, ATTN: DRDCO-PT, Fort Monmouth, NJ 07702.