

S-plus functions for robust estimators of the parameters of the Gaussian and the Lognormal distributions

A. Marazzi, A. Randriamiharisoa

March 1997

Revised: August 1999

Abstract

We describe a set of S-plus functions for several types of M-estimators for the parameters of the Gaussian and the Lognormal distributions. Beside the usual types of estimators based on Huber's Proposal 2 and the median absolute deviation estimators for the scale parameter, this set of S-plus functions includes the bias robust estimator for partitioned parameters described in Hampel et al. (1986), and the high breakdown point and high efficiency estimator with test for bias proposed by Yohai, Stahel, and Zamar (1991).

1 Introductory example

In order to use the functions described in this report, you have to load the libraries `robeth` and `robnrm`. Under S-plus for Windows, type:

```
library(robeth,T);library(robnrm,T)
```

Moreover, the example requires you to enter the functions listed in Appendix 2 as well as the following data.

```
oy <- c( 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,16,17,19,21,  
        22,26,28,29,32,33,34,35,36,37,40,43,44,49,60,68,81,96,134)  
fy <- c(51,59,34,32,32, 9, 6,12, 9,11,11, 8, 4, 3, 4, 6, 2, 1,  
        1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1)  
oz <- c( 1, 2, 3, 4, 5, 6, 7, 8, 9,16,115,198,374)  
fz <- c( 2, 6, 5, 5, 4, 2, 2, 1, 1, 1, 1, 1, 1)
```

The vector `oy` represents the observed lengths of stay (LOS) in days of 315 patients hospitalized in Belgium (BE) during 1988 for certain "disorders of the nervous system"; the vector `fy` contains their observed frequencies. Similarly, `oz` and `fz` represent the frequency distribution of 32 LOS of patients hospitalized during the same year in Switzerland (CH) for the same kind of illness. LOS is an important indicator of hospital costs. The LOS means of medically homogeneous groups of patients (e.g., the groups considered here) within a hospital are used for hospital planning; the means of different hospitals are used to compare costs and explore possible reductions. To expand the frequency distributions in individual LOS values type:

```
y0 <- NULL; for (i in 1:length(oy)) y0 <- c(y0,rep(oy[i],fy[i]))  
z0 <- NULL; for (i in 1:length(oz)) z0 <- c(z0,rep(oz[i],fz[i]))
```

In order to draw the corresponding histograms (truncated at LOS=50), open a graphic windows (e.g., by means of `win.graph()` under Windows) and type:

```
rng <- c(0,50); pr <- hists(y0,z0)
```

The arithmetic means can be computed with the help of the function `mean()`; they are 7.87 (BE) and 25.47 (CH). We will use the Lognormal model – one of the simplest models for asymmetric distribution – to approximate the histograms. We are not bothered by the use of a continuous model for discrete data, because LOS is essentially a rounded continuous variable with a large number of values, and because we interpret the model as a simple mathematical approximation of reality. The classical maximum likelihood procedure for fitting Lognormal distributions is applied as follows:

```
y <- y0; z <- z0; m <- length(y); n <- length(z)
gy <- log(y); gz <- log(z)
ly <- mean(gy); sy <- sqrt(var(gy)); my <- exp(ly+sy^2/2)
lz <- mean(gz); sz <- sqrt(var(gz)); mz <- exp(lz+sz^2/2)
```

Here, `ly` and `lz` are the location estimates of the transformed data, `sy` and `sz` the scale estimates; `my` (7.16) and `mz` (13.10) are the mean estimates based on the Lognormal model. The graphs of the two adjusted densities are obtained with

```
grd <- seq(rng[1],rng[2],length=100); dens(ly,sy,lz,sz,col=4)
```

Note that the fit for the Swiss data is very unsatisfactory.

We now test the hypothesis that the two population means (according to the model) are equal. We use a simple test procedure, based on the asymptotic normality of the maximum likelihood estimates. Their asymptotic variances can be estimated by:

$$\begin{aligned} V.ly &<- sy^2; V.sy <- sy^2/2; V.my <- my^2*V.ly+(my*sy)^2*V.sy & (F) \\ V.lz &<- sz^2; V.sz <- sz^2/2; V.mz <- mz^2*V.lz+(mz*sz)^2*V.sz \end{aligned}$$

Therefore, a simple test statistic (`tg.0`) is given by:

```
sd <- sqrt(V.my/my^2/m+V.mz/mz^2/n) # st.dev. of log(mz/my)
tg.0 <- log(mz/my)/sd
```

The null distribution of this test statistic is approximately standard normal; thus, an approximation of the attained significance level (ASL) is:

```
ASL.a <- pnorm(tg.0) # = 0.964
```

Another approximation can be computed by means of a parametric bootstrap test procedure, using the function `boot.test0()`:

```
mu <- (my+mz)/2
.Random.seed <- c(31,51,42,3,21,2,13,63,50,22,6,3)
tg.star <- boot.test0(y,z,sy,sz,mu,nboot=1000,tg.0)$lambdastar
boot.dist(tg.star,tg.0)
ASL.b <- sum(tg.star <= tg.0)/nboot # = 0.984 (L)
```

On the ground of these approximations, we should conclude that the Swiss mean is significantly larger than the Belgian mean.

We now remove the two most extreme outliers (198 and 374 days) from the Swiss sample and repeat the same analysis.

```

y <- y0;          m <- length(y)
z <- z0[!z0 > 197]; n <- length(z)
pr <- hists(y0,z0); dens(ly,sy,lz,sz,col=4) # for comparison
gy <- log(y);    gz <- log(z)
ly <- mean(gy);  sy <- sqrt(var(gy)); my <- exp(ly+sy^2/2)
lz <- mean(gz);  sz <- sqrt(var(gz)); mz <- exp(lz+sz^2/2)
dens(ly,sy,lz,sz)

```

Note that the fit for the remaining Swiss data is now more satisfactory and that $\text{mean}(y) = 7.87$, $\text{mean}(z) = 8.10$, $my = 7.16$, $mz = 6.05$. Repeating the test procedures, (i.e., retyping lines (F) to (L)) produces an attained significance level `ASL.a` of 0.20 or `ASL.b` of 0.22, i.e., the Swiss and the Belgian means are not significantly different. We conclude that this analysis is largely determined by the outliers.

Usually, around 500 groups of patients are routinely analyzed in the management of a single hospital. This clearly requires an automatic procedure for conveniently treating the outliers and producing reliable statistics. One of the most simple automatic procedures that can be applied to this problem is based on simultaneous M-estimates of location and scale (with Huber's Proposal 2 for scale). These estimates can be computed, using the complete data set, as follows:

```

y <- y0; z <- z0
by <- 1.46; bz <- 1.26 # See Appendix 1
ey <- M.E.lnorm(y,b1=by,b2=by,cov=T)
ez <- M.E.lnorm(z,b1=bz,b2=bz,cov=T)

```

According to this procedure, the Lognormal distribution is automatically adjusted to the "majority" of the data. Note that the density for the Swiss data is already satisfactory:

```

pr <- hists(y0,z0)
dens(ey$lambda,ey$sigma,ez$lambda,ez$sigma,col=4)

```

The mean estimates and the test statistic `tg.0` are obtained as follows:

```

my <- ey$mu; mz <- ez$mu
sd <- sqrt(ey$V.mu/my^2/m+ez$V.mu/mz^2/n)
tg.0 <- log(mz/my)/sd; pnorm(tg.0) # = 0.060

```

The bootstrap test is computed by means of the function `boot.test1`:

```

mu <- (my+mz)/2
.Random.seed <- c(31,51,42,3,21,2,13,63,50,22,6,3)
tg.star <- boot.test1(y,z,ey$sigma,ez$sigma,mu,by=by,bz=bz,opt=1,
                    Taby=NULL,Tabz=NULL,nboot=1000,tg.0)$lambdastar
boot.dist(tg.star,tg.0); ASL <- sum(tg.star <= tg.0)/nboot # = 0.74

```

The results of this automatic procedure are similar to those based on the maximum likelihood procedure when outliers are removed.

Section 2 briefly defines four types of estimators for which S-plus functions are made available. Section 3 contains some notes on computations. Section 4 completes the introductory example. The S-plus functions are described in Appendix 1.

2 Methods

The purpose of this section is to give precise definitions of the procedures implemented in the `robnorm` library. The description is not intended as an introduction to the methods and the reader is referred to several sources for complete informations.

2.1 The models

A random variable Y is distributed according to the *Gaussian distribution*, with location parameter λ and scale parameter σ , if its density function is

$$f_{\theta}(y) = (\sqrt{2\pi}\sigma)^{-1} \exp(-(y - \lambda)^2 / (2\sigma^2)), \quad -\infty < y < \infty, \quad -\infty < \lambda < \infty, \quad \sigma > 0, \quad (1)$$

where $\theta = (\lambda, \sigma)^T$ (a column vector). We denote the cumulative distribution function by $F_{\theta}(y)$, and use the abbreviation $\varphi(y)$ for the standard Gaussian density $f_{0,1}(y)$ and $\Phi(y)$ for the corresponding cumulative distribution function. A random variable X has a Lognormal distribution with parameter $\theta = (\lambda, \sigma)^T$, if $Y = \ln(X)$ has a Gaussian distribution with density f_{θ} . We define $\mu = E(X) = \exp(\lambda + \sigma^2/2)$.

2.2 The estimators

All estimators considered in this report are based on a sample y_1, \dots, y_n of Y . If a sample x_1, \dots, x_n of X is available, the transformed sample $y_i = \ln(x_i)$, $i = 1, \dots, n$ and the estimates $\hat{\lambda}$ and $\hat{\sigma}$ of λ and σ are first computed on the ground of the Gaussian model. The mean μ is estimated by $\hat{\mu} = \exp(\hat{\lambda} + \hat{\sigma}^2/2)$. All estimators can be viewed as members of the class of M-estimators.

M-estimators. In general, an *M-estimator* is defined as a solution $\hat{\theta}$ of a system of equations

$$\sum_{i=1}^n \psi(y_i, \theta) = 0, \quad (2)$$

where ψ denotes a given vector of (bounded) functions of (y, θ) with values in \mathbb{R}^2 . General results concerning the asymptotic properties of *M-estimators* are available in the literature (see, e.g., Rieder, 1994). In particular, let F denote the distribution of Y and $\hat{\theta}(F)$ the solution of

$$\int \psi(y, \theta) dF(y) = 0 \quad (3)$$

and note that $\hat{\theta}$ is the solution of (3) for $F = F_n$, (the empirical distribution of y_1, \dots, y_n) i.e., $\hat{\theta} = \hat{\theta}(F_n)$. Under certain regularity conditions (including observation independence) $\hat{\theta}$ is asymptotically normally distributed with mean $\hat{\theta}(F)$ and *asymptotic covariance matrix*

$$V(\hat{\theta}, F) = M(\psi, F)^{-1} Q(\psi, F) M(\psi, F)^{-T}, \quad (4)$$

where

$$Q(\psi, F) = \int \psi(y, \hat{\theta}(F))\psi(y, \hat{\theta}(F))^T dF(y), \quad (5)$$

$$M(\psi, F) = - \int \left[\frac{\partial}{\partial \theta} \psi(y, \theta) \right]_{\theta=\hat{\theta}(F)} dF(y). \quad (6)$$

The estimators of λ and σ considered in this report are asymptotically independent. Their asymptotic variances at the model distribution F_θ may be written in the form

$$V(\hat{\lambda}, F_\theta) = \sigma^2 Q_1 / M_1^2 \quad \text{and} \quad V(\hat{\sigma}, F_\theta) = \sigma^2 Q_2 / M_2^2, \quad (7)$$

where Q_1 , Q_2 , M_1 , and M_2 are scalars. The asymptotic variance of $\hat{\mu}$ at the model F_θ is

$$V(\hat{\mu}, F_\theta) = (\mu\sigma)^2 V(\hat{\sigma}, F_\theta) + \mu^2 V(\hat{\lambda}, F_\theta). \quad (8)$$

In applications, $V(\hat{\theta}, F_{\hat{\theta}})/n$ is used as an approximation of the covariance matrix of $\hat{\theta}$; thus, $V(\hat{\lambda}, F_{\hat{\theta}})/n$, $V(\hat{\sigma}, F_{\hat{\theta}})/n$ and $V(\hat{\mu}, F_{\hat{\theta}})/n$ are approximations of the variances of $\hat{\lambda}$, $\hat{\sigma}$, and $\hat{\mu}$.

In the following, we restrict our attention to the four specific types of estimators implemented in `robnorm`. These estimators, also discussed in Marazzi (1997), are:

- the simultaneous M-estimator of λ and σ described in Huber (1981, p. 136) and Hampel et al. (1986, p. 234, “Huber’s proposal 2”);
- the M-estimator of λ combined with the median absolute deviation estimator of σ described in Hampel et al. (1986, p. 236);
- the standardized bias robust B_s^p -estimator for partitioned parameters λ and σ introduced by Hampel et al. (1986, p. 254);
- the high breakdown point and high efficiency estimator of λ with a test for bias – introduced by Yohai, Stahel, and Zamar (1991, pp. 365-374) for the linear model – combined with the Q_n -estimate of σ defined by Rousseeuw and Croux (1993).

The M-estimator of location with Huber’s Proposal-2 for scale. Let $\underline{b} = (b_1, b_2)$ be a pair of user defined tuning constants. The estimator $(\hat{\lambda}, \hat{\sigma})$ is defined as the solution of the system of two equations

$$\sum_{i=1}^n \psi_{b_1}((y_i - \lambda)/\sigma) = 0, \quad (9)$$

$$\sum_{i=1}^n \chi_{b_2}((y_i - \lambda)/\sigma) = (n - 1)\beta, \quad (10)$$

where $\psi_b(x) = \max[-b, \min[b, x]]$ is Huber’s function with parameter b , $\chi_b(s) = \psi_b(s)^2$, and $\beta = \int \psi_{b_2}^2(s) d\Phi(s)$. The asymptotic variances of $\hat{\lambda}$ and $\hat{\sigma}$ at the model F_θ are given by (7) with:

$$Q_1 = \int \psi_{b_1}^2(z) d\Phi(z), \quad Q_2 = \int (\psi_{b_2}^2(z) - \beta)^2 d\Phi(z), \quad (11)$$

$$M_1 = \int \psi_{b_1}(z) z d\Phi(z), \quad M_2 = \int (\psi_{b_2}^2(z) - \beta) (z^2 - 1) d\Phi(z) \quad (12)$$

In applications, the scale parameter σ in (7) is estimated by $\hat{\sigma}$.

Remark. The ROBETH library uses $\chi_{b_2}(x) = \psi_{b_2}^2(x)/2$ in place of $\chi_{b_2}(x) = \psi_{b_2}(x)^2$ and makes available the subroutine LIBETH to compute the constant $\text{BETA} = \int \psi_{b_2}^2(s) d\Phi(s)$. Thus $\beta = 2 \cdot \text{BETA}$.

The M-estimator of location with the median absolute deviation estimator of scale. Let b_1 be a user defined tuning constant. The estimator $(\hat{\lambda}, \hat{\sigma})$ is the solution of the system of two equations

$$\sum_{k=1}^n \psi_{b_1}(y_i - \lambda)/\sigma = 0, \quad (13)$$

$$\sigma = \text{med}_i(|y_i - \lambda|)/\beta_0, \quad (14)$$

where $\psi_b(x) = \max[-b, \min[b, x]]$ is Huber's function with parameter b and $\beta_0 = \Phi^{-1}(3/4)$. The asymptotic variances of $\hat{\lambda}$ and $\hat{\sigma}$ at the model F_θ are given by (7) with:

$$Q_1 = \int \psi_{b_1}^2(z) d\Phi(z), \quad Q_2 = \int (\text{sign}(|z| - \beta_0))^2 d\Phi(z) = 1, \quad (15)$$

$$M_1 = \int \psi_{b_1}(z) z d\Phi(z), \quad M_2 = \int (\text{sign}(|z| - \beta_0)) (z^2 - 1) d\Phi(z) \quad (16)$$

In applications, the scale parameter σ in (7) is estimated by $\hat{\sigma}$.

The standardized bias robust estimator for partitioned parameters. The general definition of the standardized bias robust B_s^p -estimator for partitioned parameters is given in Hampel et al. (1986, p. 236) and a B_s^p -estimator of the parameters of the Gamma distribution is discussed Marazzi and Ruffieux (1996), with the name of "shrinking component estimator", and Marazzi (1997).

Step 1. For a given pair $\underline{b} = (b_1, b_2)$ of tuning constants and a given σ , solve the following equations for the elements a_{11}, a_{21}, a_{22} of a lower triangular matrix $A^{\underline{b}}(\sigma)$ and for the components c_1 and c_2 of a vector $c^{\underline{b}}(\sigma)$:

$$\int \psi_{b_1}^2 \left[\frac{a_{11}}{\sigma} (z - c_1 \sigma) \right] d\Phi(z) = 1, \quad (17)$$

$$\int \psi_{b_1} \left[\frac{a_{11}}{\sigma} (z - c_1 \sigma) \right] \cdot \psi_{b_2} \left[\frac{a_{21}}{\sigma} (z - c_1 \sigma) + \frac{a_{22}}{\sigma} (z^2 - 1 - c_2 \sigma) \right] d\Phi(z) = 0, \quad (18)$$

$$\int \psi_{b_2}^2 \left[\frac{a_{21}}{\sigma} (z - c_1 \sigma) + \frac{a_{22}}{\sigma} (z^2 - 1 - c_2 \sigma) \right] d\Phi(z) = 1, \quad (19)$$

$$\int \psi_{b_1} \left[\frac{a_{11}}{\sigma} (z - c_1 \sigma) \right] d\Phi(z) = 0, \quad (20)$$

$$\int \psi_{b_2} \left[\frac{a_{21}}{\sigma} (z - c_1 \sigma) + \frac{a_{22}}{\sigma} (z^2 - 1 - c_2 \sigma) \right] d\Phi(z) = 0, \quad (21)$$

where $\psi_b(x) = \max[-b, \min[b, x]]$ denotes Huber's function with parameter b .

Step 2. The estimate $(\hat{\lambda}, \hat{\sigma})$ is the solution of the system of two equations

$$\frac{1}{n} \sum \psi_{b_1}(a_1(\sigma)s_1(y; \lambda, \sigma)) = 0, \quad (22)$$

$$\frac{1}{n} \sum \psi_{b_2}(a_2(\sigma)[s_2(y; \lambda, \sigma) - c_2(\sigma)]) = 0, \quad (23)$$

where

$$s_1(y, \lambda) = \partial \ln(f_\theta(y))/\partial \lambda = z/\sigma, \quad s_2(y, \sigma) = \partial \ln(f_\theta(y))/\partial \sigma = (z^2 - 1)/\sigma, \quad (24)$$

are the score functions of the Gaussian model, with $z = (y - \lambda)/\sigma$.

It is easy to see (Marazzi, 1997) that the solution of (17)-(21) has the form

$$A^b(\sigma) = \sigma \begin{pmatrix} a_{11} & 0 \\ 0 & a_{22} \end{pmatrix}, \quad c^b(\sigma) = \frac{1}{\sigma} \begin{pmatrix} 0 \\ c_2 \end{pmatrix}, \quad (25)$$

where a_{11} , a_{22} , and c_2 are determined for $\sigma = 1$. The asymptotic variances of $\hat{\lambda}$ and $\hat{\sigma}$ at the model F_θ are given by (7) with:

$$Q_1 = \int \psi_{b_1}^2(a_{11}z) d\Phi(z), \quad Q_2 = \int \psi_{b_2}^2(a_{22}(z^2 - q)) d\Phi(z), \quad (26)$$

$$M_1 = \int \psi_{b_1}(a_{11}z)z d\Phi(z), \quad M_2 = \int \psi_{b_2}^2[a_{22}(z^2 - q)](z^2 - 1) d\Phi(z) \quad (27)$$

where $q = 1 + c_2$. In applications, the scale parameter σ in (7) is estimated by $\hat{\sigma}$.

High breakdownpoint and high efficiency estimate of location (and scale) with test for bias. For constant k define the bisquare function

$$\chi_k(s) := \begin{cases} 3(s/k)^2 - 3(s/k)^4 + (s/k)^6 & \text{if } |s| \leq k, \\ 1 & \text{if } |s| > k, \end{cases}$$

and perform the following steps.

1. Compute the initial location and scale estimates with high breakdown point. To this end, calculate the S-estimate $\lambda^{(0)}$ defined by

$$\lambda^{(0)} = \operatorname{argmin}_\lambda S_{k_0}(\lambda), \quad (28)$$

where $S_{k_0}(\lambda)$ is the solution S of

$$\sum_{i=1}^n \chi_{k_0}((y_i - \lambda)/S) = (n - 1)\beta. \quad (29)$$

Here $k_0 = 1.548$, and $\beta = \int \chi_{k_0}(s) d\Phi(s) = 0.5$. Let $\sigma^{(0)} = S(\lambda^{(0)})$ be the estimate of σ corresponding to $\lambda^{(0)}$, and let $r_i^{(0)} = y_i - \lambda^{(0)}$ for $i = 1, \dots, n$.

2. Compute the final coefficient estimate with high efficiency. This is the MM-estimate $\lambda^{(1)}$ defined as the local minimum of

$$Q_{\text{MM}}(\lambda) = \sum_{i=1}^n \rho\left(\frac{y_i - \lambda}{\sigma^{(0)}}\right), \quad (30)$$

such that

$$Q_{\text{MM}}(\lambda^{(1)}) \leq Q_{\text{MM}}(\lambda^{(0)}), \quad (31)$$

where $\rho(s) = \chi_k(s)$ and $k = k_1 = 4.687$.

3. Compute the scale estimate $\sigma^{(1)}$, as the solution of

$$\frac{1}{n-1} \sum_{i=1}^n \chi_{k_0}\left(\frac{r_i^{(1)}}{\sigma}\right) = 0.5. \quad (32)$$

4. Compute the test statistic for the test for bias. Let $\psi_{k_0}(\cdot) = \chi'_{k_0}(\cdot)$ and $\psi_{k_1}(\cdot) = \chi'_{k_1}(\cdot)$, where $'$ denotes the derivative with respect to the argument, and calculate

$$\begin{aligned} \tilde{r}_i &:= r_i^{(0)} / \sigma^{(0)} && \text{for } i = 1, \dots, n, \\ v_0 &:= \frac{(1/n) \sum \psi'_{k_0}(\tilde{r}_i)}{(\sigma^{(0)}/n) \sum \psi_{k_0}(\tilde{r}_i) \tilde{r}_i}, \\ \psi_i^{(0)} &:= \frac{\psi_{k_0}(\tilde{r}_i)}{(1/n) \sum \psi'_{k_0}(\tilde{r}_i)} && \text{for } i = 1, \dots, n, \\ \psi_i^{(1)} &:= \frac{\psi_{k_1}(\tilde{r}_i)}{(1/n) \sum \psi'_{k_1}(\tilde{r}_i)} && \text{for } i = 1, \dots, n, \\ d^2 &:= \frac{1}{n} \sum (\psi_i^{(1)} - \psi_i^{(0)})^2, \\ T &:= \frac{2n(\sigma^{(1)} - \sigma^{(0)})}{v_0 d^2 \sigma^{(0)^2}.} \end{aligned} \quad (33)$$

If T is larger than the α quantile χ_α^2 (a typical α is 0.95) of the χ^2 -distribution with 1 degrees of freedom, then the procedure gives a warning that the bias due to outliers may be unacceptably high so that inference based on the final MM-estimates is dangerous. In this case, the initial estimates $\lambda^{(0)}$ and $\sigma^{(0)}$ can be used as exploratory estimates. Otherwise, the bias is not high, and $\lambda^{(1)}$ is returned with $\sigma^{(0)}$ as a scale estimate. In this case, inference is possible.

5. Optionally, return the scale estimate Q_n defined by Rousseeuw and Croux (1993) in place of $\sigma^{(0)}$. The Q_n -estimate is defined as

$$Q_n = 2.219 \cdot \{|y_i - y_j|, i < j\}_{(k)}, \quad (34)$$

where $k = \binom{h}{2}$ for $h = \lceil n/2 \rceil + 1$ (the notation $\lceil s \rceil$ indicates the largest integer smaller than s). In other words, Q_n is the first quartile of the set of absolute differences $|y_i - y_j|$, $i < j$.

The asymptotic variance of $\lambda^{(1)}$ at the model F_θ is $\sigma^2 Q_1 / M_1^2$, with

$$Q_1 = \int \psi_{k_1}^2(z) d\Phi(z), \quad M_1 = \int \psi_{k_1}(z) z d\Phi(z). \quad (35)$$

The asymptotic variance of $\sigma^{(0)}$ at the model F_θ is $\sigma^2 Q_2 / M_2^2$, with

$$Q_2 = \int (\chi_{k_0}(z) - \beta)^2 d\Phi(z), \quad M_2 = \int (\chi_{k_0}(z) - \beta) (z^2 - 1) d\Phi(z) \quad (36)$$

The asymptotic variance of Q_n at the model F_θ is

$$V(Q_n, F_\theta) = 0.6089 \cdot \sigma^2. \quad (37)$$

In applications, the scale σ is estimated by $\sigma^{(0)}$ or by Q_n .

3 Notes on computation

Computation of integrals. In order to evaluate the asymptotic variances, several integrals must be computed numerically. The following formulae are used

(a) If Υ_d^2 is the cumulative probability of the χ^2 -distribution with d degrees of freedom, then

$$\int \psi_b(z) z d\Phi(z) = 2\Phi(b) - 1 = \Upsilon_1^2(b^2) \quad (38)$$

and

$$\int_{|z| \leq b} |z|^{2k} d\Phi(z) = 1 \cdot 3 \cdots [2k - 1] \cdot \Upsilon_{1+2k}^2(b^2). \quad (39)$$

(b) If $v = \varphi(z)$, then $dv = -z\varphi(z)dz = -zd\Phi(z)$ and

$$\begin{aligned} \int g(z)(z^2 - 1)d\Phi &= \int \{-g(z)z\}\{-z\varphi(z)\}dz - \int g(z)d\Phi \\ &= \{-g(z)z\}\varphi(z) \Big|_{-\infty}^{\infty} - \int \{-g(z)z\}'\varphi(z)dz - \int g(z)d\Phi = \int g'(z)z d\Phi. \end{aligned} \quad (40)$$

Computation of the B_s^p -estimator. For $\sigma = 1$, equations (17–(21) reduce to:

$$a_{11}^2 \int \psi_{b_1/a_{11}}^2(z) d\Phi(z) - 1 = 0, \quad (41)$$

$$a_{22}^2 \int \psi_{b_2/a_{22}}^2(z^2 - q) d\Phi(z) - 1 = 0, \quad (42)$$

$$\int \psi_{b_2/a_{22}}(z^2 - q) d\Phi(z) = 0. \quad (43)$$

A bisection method is used to solve (41) for a_{11} . Starting with $q = 1$, two bisection algorithms are used alternately for solving (42) and (43) for a_{22} and q , respectively, until convergence.

The system (22)-(23) is solved using a simple modification of the algorithm described in Marazzi and Ruffieux (1996), Section 4.3. Let

$$S_{p1}(\lambda, \sigma) = \frac{1}{n} \sum \psi_{b_1}(a_1(\sigma)s_1(y; \lambda, \sigma)), \quad (44)$$

$$S_{p2}(\lambda, \sigma) = \frac{1}{n} \sum \psi_{b_2}(a_2(\sigma)[s_2(y; \lambda, \sigma) - c_2(\sigma)]). \quad (45)$$

The function S_{p1} is inverted with respect to λ , obtaining the function $\bar{\lambda}(\sigma)$. Then, a bisection method is used to solve $S_{p2}(\bar{\lambda}(\sigma), \sigma) = 0$ with respect to σ .

Computation of the MM-estimator. The S-estimate $\lambda^{(0)}$ is obtained as follows:

- Let h be a given integer (e.g., $h = 100$) and $\delta = (\max(y_i) - \min(y_i))/h$.
For $j = 0, \dots, h$, let $\dot{\lambda}(j) = \min(y_i) + j\delta$, $\dot{r}_i(j) := y_i - \dot{\lambda}(j)$ ($i = 1, \dots, n$), and let $\dot{\sigma}(j)$ be the solution of $\sum_{i=1}^n \chi_{k_0}(\dot{r}_i(j)/\sigma) = (n-1)\beta$ with respect to σ .
Finally, Let $j^* = \operatorname{argmin}_j(\dot{\sigma}(j))$ and $\lambda^* = \dot{\lambda}(j^*)$.
- Once the approximation λ^* has been obtained, the S-estimate is computed by refining λ^* . The optimization algorithm described in Marazzi (1993) for the subroutine RYWALG is used to obtain $\lambda^{(0)}$ as a local minimum of S such that $S(\lambda^{(0)}) \leq S(\lambda^*)$.

The estimate $\sigma^{(1)}$ is computed by means of the subroutine RYSIGM of ROBETH and the MM-estimate $\lambda^{(1)}$ is obtained by means of the the ROBETH subroutine RYWALG (Marazzi, 1993, p. 59, Remark 1, and p. 83).

4 Advanced examples

Tuning constants. The simultaneous M-estimator with Huber's Proposal 2 for scale, used in the introductory example, depends on the tuning constants $\underline{b} = (b_1, b_2)$ that must be chosen by the user. For simplicity, we use M-estimators with a single tuning constant $b = b_1 = b_2$. The most common rule for determining the tuning constants is to require that the asymptotic relative efficiency (ARE) of the M-estimator with respect to the maximum likelihood estimator, at the model, equals a given value, e.g, 90%; the higher the value of ARE, the more the estimate is sensitive to outliers. Unfortunately, the ARE of the lognormal mean estimate depends on the estimated value of the scale parameter. Thus, the choice of b must be made in two steps: first, a preliminary scale estimate is obtained, using a low value of the tuning constant in order to ensure a reasonably high degree of robustness; second, the tuning constant is refined on the ground of the preliminary estimate.

In the Introductory example, using $b_1 = b_2 = 1.3$, one obtains $\mathbf{sy} = 0.71$ and $\mathbf{sz} = 1.077$. To compute refined tuning constants such that the ARE of the lognormal mean estimate equals 85%, we need two more functions:

```

ARE.mu    <- function(b1=1.5,b2=1.5,l=0,s=1){
v.mu.1    <- AV.ME.lnorm(b1=b1, b2=b2, lambda=1,sigma=s,opt=1)$V.mu
v.mu.0    <- AV.ME.lnorm(b1=Inf,b2=Inf,lambda=1,sigma=s,opt=1)$V.mu
v.mu.0/v.mu.1}

```

and

```

ARE.mu.level <- function(b,lambda=lambda,sigma=sigma,level=level){
ARE <- ARE.mu(b1=b,b2=b,l=lambda,s=sigma)
ARE-level}

```

The equation $ARE=0.85$, is solved (for the two estimates) as follows:

```

0 <- uniroot(ARE.mu.level,c(0.5,3),lambda=0,sigma=0.710,level=0.85)
0$root                # = 1.257
0 <- uniroot(ARE.mu.level,c(0.5,3),lambda=0,sigma=1.077,level=0.85)
0$root                # = 1.461

```

Bias curves. We consider a sequence of data sets that depend on an index $i = 1, \dots, 100$ and on the number $\varepsilon_i = i/(100 + i)$. Each data set consists of 100 percentile points of the Lognormal distribution with $\lambda = 0$, $\sigma = 1$, $\mu = \exp(0.5)$ and of i identical outliers with value 500. Thus, ε_i is the percentage of outliers in the data set. For each data set, we use the functions `M.E.lnorm(...,opt=1,...)` and `MM.E.lnorm()` to estimate μ . Suppose that $\hat{\mu}_{1,i}$ and $\hat{\mu}_{4,i}$ denote the estimates; we compute the relative bias $(\hat{\mu}_{1,i} - \mu)/\mu$ and $(\hat{\mu}_{4,i} - \mu)/\mu$ and plot them against ε_i , for $i = 1, \dots, 100$.

```

Mu      <- exp(0.5)
b       <- 1.43
ctrl    <- MM.E.control(k1=3.56)
eps <- mu1 <- mu4 <- NULL

for (i in 0:50) {
ep <- i/(100+i); eps <- c(eps,ep)
ly <- qnorm(ppoints(100)); y <- c(exp(ly),rep(500,i))
e1 <- M.E.lnorm(y,b1=b,b2=b,opt=1)
e4 <- MM.E.lnorm(y,control=ctrl,isigma=2)
mu1 <- e1$mu;      mu4 <- e4$mu
Mu1 <- c(Mu1,mu1); Mu4 <- c(Mu4,mu4)
cat(ep,mu1,mu4," \n")}

par(mfrow=c(1,1))
plot(eps,(Mu4-Mu)/Mu,type="n",xlab="eps",ylab="rel.bias",ylim=c(0,1.5))
lines(eps,(Mu4-Mu)/Mu,lty=1,lwd=1); text(0.28,(3-Mu)/Mu,"MM",cex=1.5)
lines(eps,(Mu1-Mu)/Mu,lty=2,lwd=1); text(0.22,(4-Mu)/Mu,"Hp2",cex=1.5)

```

The diagram shows that the MM-estimator is more resistant to outliers than the M-estimator. It can be shown that the breakdownpoint of the MM-estimator is 50%, whereas the breakdownpoint of the M-estimator is only 0.27. Moreover, the ARE of the MM-estimator computed above, with $k_1 = 3.56$, is 0.90 whereas the ARE of the M-estimator with $b = 1.43$ is 0.85.

The B_{ξ}^p -estimate

```
> library(robeth,T); library(robnrm,T); win.graph()
> lambda <- 5; sigma <- 3
> n <- 50; eps <- 0.06
> n1 <- n*(1-eps); n2 <- n-n1
> y <- c(rnorm(n1,lambda,sigma), runif(n2,min=0,max=50))
> Table <- Tab.gauss(b1=1.5,b2=1.7,tol=0.0001)
> ey <- M.E.gauss(y,b1=1.5,b2=1.7,opt=3,Table,cov=T)
> Sp2 <- Sp2.gauss(seq(1.5,6.5,length=50),y,Table)
> Crv.gauss(Sp2, Title="ey"); abline(h=0, lty=2)
> ey$lambda; ey$sigma; ey$V.lambda; ey$V.sigma
```

References

- Gradshteyn I.S., Ryzhik I.M. (1980). *Table of Integrals, Series, and Products*. Academic Press, New York.
- Hampel F.R., Ronchetti E.M., Rousseeuw P.J., Stahel W.A. (1986). *Robust statistics: The approach based on influence functions*, Wiley, New York.
- Huber P.J. (1980). *Robust statistics*. Wiley, New York.
- Marazzi A. (1993). Algorithms, Routines, and S-functions for Robust Statistics. Wadsworth & Brooks Cole, Belmont, California. Reprinted by Chapman and Hall, New York.
- Marazzi A., Ruffieux C. (1996). Implementing M-estimators of the Gamma distribution. In: H. Rieder (Ed.), *Robust statistics, data analysis, and computer intensive methods, In honor of Peter Huber's 60th birthday*, Lecture Notes in Statistics, 109, Springer Verlag, Heidelberg.
- Marazzi A., (1997). Estimating and testing the means of asymmetric distributions using M-estimators. Technical report. Institut Universitaire de médecine sociale et préventive, Bugnon 17, CH-1005 Lausanne.
- Piessens R., de Doncker-Kapenga E., Überhuber C.W., Kahaner D.K. (1980). *QUADPACK: A subroutine library for automatic integration*, Springer Verlag, Berlin Heidelberg.
- Rieder, H. (1994). *Robust Asymptotic Statistics*. Springer-Verlag, New York.
- Rousseeuw P.J., Croux C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88, pp. 1273-1283.
- Yohai V.J., Stahel W.A., Zamar R.H. (1991). A procedure for robust estimation and inference in linear regression, in Stahel W.A. and Weisberg S.W., Eds., *Directions in robust statistics and diagnostics, Part II*, Springer-Verlag, New York.

Appendix 1: S-Plus functions

<code>M.E.gauss</code>	M-estimates of the parameters of the Gaussian distribution.
<code>M.E.lnorm</code>	M-estimates of the parameters of the Lognormal distribution.
<code>MM.E.gauss</code>	MM-estimates of the parameters of the Gaussian distribution.
<code>MM.E.lnorm</code>	MM-estimates of the parameters of the Lognormal distribution.
<code>AV.ME.gauss</code>	Asymptotic variances of the estimates returned by <code>M.E.gauss</code> .
<code>AV.ME.lnorm</code>	Asymptotic variances of the estimates returned by <code>M.E.lnorm</code> .
<code>AV.MM.gauss</code>	Asymptotic variances of the estimates returned by <code>MM.E.gauss</code> .
<code>AV.MM.lnorm</code>	Asymptotic variances of the estimates returned by <code>MM.E.lnorm</code> .
<code>MM.E.control</code>	Control parameters for <code>MM.E.gauss</code> , and <code>MM.E.lnorm</code> .
<code>Tab.gauss</code>	Computation and storage of the matrix A_b and the vector c_b for B_s^p -estimates in the Gaussian and the Lognormal cases.
<code>Crv.gauss</code>	Visual check. This function draws the graph of $S_{p2}(\bar{\lambda}(\sigma), \sigma)$ as a function of σ .
<code>Sp2.gauss</code>	Preliminary computations for <code>Crv.gauss</code> .

These functions have been developed using parts of the subroutine library ROBETH described in Marazzi (1993). Several choices concerning the specification of the estimators (e.g., the choice of the function ψ) and the specification of the computational algorithms, have been made; the user cannot change many of these choices. A more flexible tool for programming and computing M-estimators is provided by ROBETH. The functions must be loaded into S-plus. For the Windows version of S-plus, a library is made available (use the command `library(robnrm,T)`).

function `M.E.gauss`

M-estimates of the parameters of the Gaussian distribution.

Specification

```
function M.E.gauss(y, b1=1.5, b2=1.5, opt=2, Table=NULL, cov=F,
                  maxit=300, tol=0.001)
```

Purpose

If `opt = 1`, `M.E.gauss` computes the solution of equations (9)-(10). If `opt = 2`, `M.E.gauss` computes the solution of equations (13)-(14). If `opt = 3`, `M.E.gauss` computes the solution of equations (22)-(23). In this case, the functions $A_b(\sigma)$ and $c_b(\sigma)$ are assumed to be given as input arguments. They must be computed before calling `M.E.gauss()` by means of the function `Tab.gauss()`. The asymptotic variances of the parameter estimates can be obtained by setting `cov=T` on input.

Method

The constants β and β_0 are computed by means of the ROBETH subroutines LIBETH and LIBET0. If `opt = 1` or `opt = 2`, the initial values are obtained by means of LILARS and the solution by means of LYHALG. For `OPT = 3`, the solution of $S_{pj}(\theta, \sigma) = 0$, $j = 1, 2$, is computed according to the algorithm described in Marazzi and Ruffieux (1997), Section 4.3.

Arguments

<code>y</code>	Observation vector.
<code>b1</code>	Tuning constant b_1 . If <code>opt = 3</code> , <code>b1</code> is extracted from <code>Table</code> .
<code>b2</code>	Tuning constant b_2 . If <code>opt = 3</code> , <code>b2</code> is extracted from <code>Table</code> .
<code>opt</code>	See Purpose.
<code>Table</code>	<code>Table</code> is the object (list) returned by the function <code>Tab.gauss()</code> . It must be created before calling <code>M.E.gauss()</code> with <code>opt = 3</code> . The main components of <code>Table</code> are the matrix $A_b(1)$ and the vector $c_b(1)$.
<code>cov</code>	If <code>cov=T</code> on input, the asymptotic variances of $\hat{\lambda}$ and $\hat{\sigma}$ are returned.
<code>maxit</code>	Maximum number of iterations for the iterative algorithms.
<code>tol</code>	Required relative precision of the solution.

Value

List with the following components:

<code>lambda</code>	Estimate $\hat{\lambda}$.
<code>sigma</code>	Estimate $\hat{\sigma}$.
<code>ok</code>	<code>ok = 1</code> indicates that a solution has been reached within the desired accuracy.
<code>V.lambda</code>	Asymptotic variance of $\hat{\lambda}$.
<code>V.sigma</code>	Asymptotic variance of $\hat{\sigma}$.
<code>call</code>	The calling sequence.

function `M.E.lnorm`

M-estimates of the parameters of the Lognormal distribution.

Specification

```
function M.E.lnorm(x, b1=1.5, b2=1.5, opt=2, Table=NULL, cov=F,  
                  maxit=300, tol=0.001)
```

Purpose

`M.E.lnorm()` transforms x_i to y_i as follows:

$$y_i = \begin{cases} \ln(x_i) & \text{if } x_i > 0, \\ 0.5 & \text{if } x_i \leq 0. \end{cases}$$

Then, it applies `M.E.gauss()` to $y = (y_1, \dots, y_n)$ in order to obtain M-estimates $\hat{\lambda}$ and $\hat{\sigma}$ of λ and σ . Finally, it computes $\hat{\mu} = \exp(\hat{\lambda} + \hat{\sigma}^2/2)$ and asymptotic variances of $\hat{\lambda}$, $\hat{\sigma}$, and $\hat{\mu}$.

Method

See `M.E.gauss()`.

Arguments

`x` Observation vector.
See `M.E.gauss()` for the other arguments.

Value

List with the following components:

<code>lambda</code>	Estimate $\hat{\lambda}$.
<code>sigma</code>	Estimate $\hat{\sigma}$.
<code>ok</code>	If <code>ok = 1</code> on output, a solution has been reached within the desired accuracy.
<code>mu</code>	The estimated mean $\hat{\mu} = \exp(\hat{\lambda} + \hat{\sigma}^2/2)$ of the Lognormal distribution.
<code>V.lambda</code>	Asymptotic variance of $\hat{\lambda}$.
<code>V.sigma</code>	Asymptotic variance of $\hat{\sigma}$.
<code>V.mu</code>	Asymptotic variance of $\hat{\mu}$.
<code>call</code>	The calling sequence.

function `MM.E.gauss`

High breakdown point and high efficiency estimates of the parameters of the Gaussian distribution with test for bias.

Specification

```
function MM.E.gauss(y, cov=F, isigma=1, onlyS=F, test=F, level=0.9,  
                    control, ...)
```

Purpose

`MM.E.gauss()` computes the estimates described in Section 2.4. The default values of the control parameters are collected in the auxiliary function `MM.E.control()`.

Method

See Section 2.4 and Section 3.

Arguments

<code>y</code>	Observation vector.
<code>cov</code>	If <code>cov=T</code> on input, the asymptotic variances of the parameter estimates are returned.
<code>isigma</code>	If <code>isigma = 1</code> , the scale estimate $\sigma^{(0)}$ is returned. If <code>isigma = 2</code> , the Q_n scale estimate is returned.
<code>onlyS</code>	If <code>onlyS=T</code> , the function just returns the S-estimates $\lambda^{(0)}$ and $\sigma^{(0)}$.
<code>test</code>	If <code>test=T</code> , the test for bias is performed. In this case, the returned location estimate is $\lambda^{(0)}$ or $\lambda^{(1)}$ depending on the value of the test statistic T and the level <code>level</code> .
<code>level</code>	Level α for the test for bias.
<code>control</code>	A list returned by the auxiliary function <code>MM.E.control()</code> .
<code>...</code>	List of control parameters to be modified.

Value

List with the following components

<code>lambda</code>	Location estimate $\lambda^{(1)}$ or $\lambda^{(0)}$, according to the test for bias.
<code>sigma</code>	Scale estimate $\sigma^{(0)}$ or Q_n .
<code>tbias</code>	The value of the test statistic T .
<code>pchi</code>	The probability that a χ^2 random variable with one degree of freedom is greater than <code>tbias</code> .
<code>V.lambda</code>	Asymptotic variance of the location estimate returned in <code>lambda</code> .
<code>V.sigma</code>	Asymptotic variance of the scale estimate returned in <code>sigma</code> .
<code>T.lambda</code>	The location estimate not returned in <code>lambda</code> .
<code>T.sigma</code>	Auxiliary scale estimate $\sigma^{(1)}$.
<code>call</code>	The calling sequence.

function `MM.E.lnorm`

High breakdown point and high efficiency estimates of the parameters of the Lognormal distribution with test for bias.

Specification

```
function MM.E.lnorm(x, cov=F, isigma=1, onlyS=T, test=F, level=0.9,  
                    control, ...)
```

Purpose

`M.E.lnorm()` transform x_i to y_i as follows:

$$y_i = \begin{cases} \ln(x_i) & \text{if } x_i > 0, \\ 0.5 & \text{if } x_i \leq 0. \end{cases}$$

Then, it applies `MM.E.gauss()` to $y = (y_1, \dots, y_n)$ in order to obtain estimates of λ and σ . Finally, it computes the corresponding estimate of μ and asymptotic variances of the returned estimates.

Method

See `MM.E.gauss()`.

Arguments

`x` Observation vector.

See `M.E.gauss` for the other arguments.

Value

List with the following components

<code>lambda</code>	Location estimate $\lambda^{(1)}$ or $\lambda^{(0)}$, according to test for bias.
<code>sigma</code>	Scale estimate $\sigma^{(0)}$ or Q_n .
<code>mu</code>	Estimate of μ based on the returned estimates <code>lambda</code> and <code>sigma</code> .
<code>tbias</code>	The value of the test statistic T .
<code>pchi</code>	The probability that a χ^2 random variable with one degree of freedom is greater than <code>tbias</code> .
<code>V.lambda</code>	Asymptotic variance of the location estimate returned in <code>lambda</code> .
<code>V.sigma</code>	Asymptotic variance of the scale estimate returned in <code>sigma</code> .
<code>V.mu</code>	Asymptotic variance of the mean estimate returned in <code>mu</code> .
<code>T.lambda</code>	The location estimate not returned in <code>lambda</code> .
<code>T.sigma</code>	Auxiliary scale estimate $\sigma^{(1)}$.
<code>call</code>	The calling sequence.

function `AV.ME.gauss`

Asymptotic variances of the estimates returned by `M.E.gauss()`.

Specification

```
function AV.ME.gauss(b1=1.5, b2=1.5, sigma=1, opt=2, Table=NULL)
```

Purpose

This function returns the asymptotic variances of the estimates $\hat{\lambda}$ and $\hat{\sigma}$ computed by means of `M.E.gauss()`. If `opt=3`, the function $A_b(\sigma)$ and $c_b(\sigma)$ are assumed to be given as input arguments. They must be computed before calling `AV.ME.gauss()` by means of the function `Tab.gauss()`.

Method

Formulae (11)-(12) (`opt=1`), (15)-(16) (`opt=2`), and (26)-(27) (`opt=3`).

Arguments

<code>b1</code>	Tuning constant b_1 . If <code>opt=3</code> , <code>b1</code> is extracted from <code>Table</code> .
<code>b2</code>	Tuning constant b_2 . If <code>opt=3</code> , <code>b2</code> is extracted from <code>Table</code> .
<code>sigma</code>	Estimate $\hat{\sigma}$.
<code>opt</code>	See Method.
<code>Table</code>	<code>Table</code> is the object (list) returned by the function <code>Tab.gauss()</code> and must be created before calling <code>M.E.gauss()</code> with <code>opt = 3</code> . The main components of <code>Table</code> usually are the matrix $A_b(1)$ and the vector $c_b(1)$.

Value

List with the following components

<code>V.lambda</code>	Asymptotic variance of $\hat{\lambda}$.
<code>V.sigma</code>	Asymptotic variance of $\hat{\sigma}$.

function AV.ME.lnorm

Asymptotic variances of the estimates returned by M.E.lnorm().

Specification

```
function AV.ME.lnorm(b1=1.5, b2=1.5, lambda=0, sigma=1, opt=2,  
                    Table=NULL)
```

Purpose

This function returns the asymptotic variances of the estimates $\hat{\lambda}$, $\hat{\sigma}$ computed by means of M.E.lnorm() and the corresponding estimate of the asymptotic variance of $\hat{\mu} = \exp(\hat{\lambda} + \hat{\sigma}^2/2)$. If opt=3, the function $A_b(\sigma)$ and $c_b(\sigma)$ are assumed to be given as input arguments. They must be computed before calling AV.ME.gauss() by means of the function Tab.gauss().

Method

AV.ME.lnorm() calls AV.ME.gauss() to compute the asymptotic variances of $\hat{\lambda}$ and $\hat{\sigma}$. The variance of $\hat{\mu}$ is given by $V(\hat{\mu}, F_{\hat{\theta}}) = (\hat{\mu}\hat{\sigma})^2V(\hat{\sigma}, F_{\hat{\theta}}) + \hat{\mu}^2V(\hat{\lambda}, F_{\hat{\theta}})$.

Arguments

See AV.ME.gauss().

Value

List with the following components

V.lambda	Asymptotic variance of $\hat{\lambda}$.
V.sigma	Asymptotic variance of $\hat{\sigma}$.
V.mu	Asymptotic variance of $\hat{\mu}$.

function AV.MM.gauss

Asymptotic variances of the estimates $\lambda^{(1)}$ and $\sigma^{(0)}$ returned by MM.E.gauss().

Specification

```
function AV.MM.gauss(k0=1.5477, k1=4.6873, sigma=1)
```

Purpose

Returns the asymptotic variances of the estimates returned by MM.E.gauss().

Method

Formulae (35)-(36).

Arguments

<code>k0</code>	Tuning constant k_0 .
<code>k1</code>	Tuning constant k_1 .
<code>sigma</code>	Estimate $\hat{\sigma}$.

Value

List with the following components

<code>V.lambda</code>	Asymptotic variance of $\lambda^{(1)}$.
<code>V.sigma</code>	Asymptotic variance of $\sigma^{(0)}$.

function `AV.MM.lnorm`

Asymptotic variances of the estimates $\lambda^{(1)}$ $\sigma^{(0)}$, and the corresponding estimate of the lognormal mean returned by `MM.E.lnorm()`.

Specification

```
function AV.MM.lnorm(k0=1.5477, k1=4.6873, lambda=0, sigma=1)
```

Purpose

Returns the asymptotic variances of the estimates returned by `MM.E.lnorm()`.

Method

`AV.MM.lnorm()` calls `AV.MM.gauss()` to compute the asymptotic variances of the location and scale estimates $\lambda^{(1)}$ and $\sigma^{(0)}$ returned by `MM.E.lnorm()`. The variance of the lognormal mean is computed according to (8).

Arguments

See `AV.MM.lnorm()`.

Value

List with the following components

<code>V.lambda</code>	Asymptotic variance of $\lambda^{(1)}$.
<code>V.sigma</code>	Asymptotic variance of $\sigma^{(0)}$.
<code>V.mu</code>	Asymptotic variance of the lognormal mean estimate.

function Tab.gauss

Computation and storage of the matrix A_b and the vector c_b (for $\sigma = 1$).

Specification

```
function Tab.gauss(b1=1.5, b2=1.7, a.interval=c(0.1,10),  
                  c.interval=c(-2,2), maxit=30, tol=0.0001)
```

Purpose

For a given value of the tuning constant $\underline{b} = (b_1, b_2)$, this function computes the matrix $A^{\underline{b}}$ and the vector $c^{\underline{b}} = (0, c_2)$, as solutions of equations (17)-(21) for $\sigma = 1$.

Method

See Section 3.

Arguments

b1	Tuning constant b_1 .
b2	Tuning constant b_2 .
a.interval	The range in which a_{11} and a_{22} are allowed to vary in the searching algorithm.
c.interval	The range in which c_2 is allowed to vary in the searching algorithm.
maxit	Maximum number of cycles of the iterative algorithm for a_{22} and c_2 .
tol	Desired relative precision of a_{11} , a_{22} , and c_2 .

Value

List with the following components

Table	Table is a matrix with 1 row. The row contains $(c_1 = 0, c_2, a_{11}, a_{21} = 0, a_{22})$.
call	The calling sequence.

function Sp2.gauss

Numerical computations for drawing a graph of $S_{p2}(\bar{\lambda}(\sigma), \sigma)$.

Specification

```
function Sp2.gauss(grid, y, Table, tol=0.001, maxit=300)
```

Purpose

`Sp2.gauss()` computes the function $S_{p2}(\bar{\lambda}(\sigma), \sigma)$ defined in (44)-(45) (where $\bar{\lambda}(\sigma)$ is the solution of $S_{p1}(\lambda, \sigma) = 0$ for given σ) for all values of σ contained in the vector argument `grid`. The graph of $S_{p2}(\bar{\lambda}(\sigma), \sigma)$, that can be drawn by means of the function `Crv.gauss()`, should cross the horizontal axis for $\sigma = \hat{\sigma}$. A neat intersection indicates a precise determination of $\hat{\sigma}$.

Method

`Sp2.gauss()` uses a regula-falsi method to invert $S_{p1}(\lambda, \sigma)$ with respect to λ , for all values of σ given in `grid`, and obtain an accurate solution $\bar{\lambda}(\sigma)$ of $S_{p1}(\lambda, \sigma) = 0$. It returns $S_{p2}(\bar{\lambda}(\sigma), \sigma)$ for $\sigma \in \text{grid}$. The values of $c_b(\sigma)$ and $A_b(\sigma)$ contained in `Table`, are used to compute S_{p1} and S_{p2} .

Arguments

<code>grid</code>	Vector containing the values of σ for which computations are desired. For example, <code>grid=seq(sigma1,sigma2,length=100)</code> , where <code>sigma1</code> and <code>sigma2</code> are the limits of the interval for the graph.
<code>y</code>	Observation vector.
<code>Table</code>	Matrix containing values of $A_b(\sigma)$ and $c_b(\sigma)$ for $\sigma = 1$. This table must be created by means of the function <code>Tab.gauss()</code> before calling <code>Sp2.gauss()</code> .
<code>tol</code>	Required relative precision of $\bar{\lambda}(\sigma)$ in the regula-falsi algorithm.
<code>maxit</code>	Maximum number of iterations in the regula-falsi algorithm.

Value

`Sp2` Matrix with `length(grid)` rows. The i th row of `Sp2` contains $(\sigma, S_{p2}(\bar{\lambda}(\sigma), \sigma), Iterm)$, where σ is the i th element of `grid`. *Iterm* is equal to 1, if the regula-falsi algorithm converged, 0 otherwise.

function `Crv.gauss`

Draws the graph of the function computed by `Sp2.gauss`

Specification

```
function Crv.gauss(Sp2, x.lim=c(0,0), y.lim=c(0,0),
                  Title="", Xlab="sigma", Ylab="")
```

Purpose

`Crv.gauss()` draws the graph of the function $(S_{p2}(\bar{\lambda}(\sigma), \sigma))$ prepared by means of `Sp2.gauss()`.

Method

`Crv.gauss()` calls `plot()` in order to draw the function defined by the first two columns of the matrix object produced by `Sp2.gauss()`.

Arguments

<code>Sp2</code>	Object returned by <code>Sp2.gauss</code> .
<code>x.lim</code>	Argument of the S-plus function <code>par()</code> : limits for the horizontal axis. By default (<code>x.lim=c(0,0)</code>), the extreme values of σ found in <code>Sp2</code> are used.

<code>y.lim</code>	Argument of the S-plus function <code>par()</code> : limits for the vertical axis. By default (<code>y.lim=c(0,0)</code>), the extreme values found in the second column of <code>Sp2</code> are used.
<code>Title</code>	Plot title.
<code>Xlab</code>	Horizontal axis label.
<code>Ylab</code>	Vertical axis label.

Value

None.

function `MM.E.control`

Control parameters for the estimation procedure of the MM-estimate of location and scale.

Specification

```
function MM.E.control(tlo = 0.0001, mxf = 50, mxs = 50, ntm = 50,
                     tls = 1e-06, k0 = 1.5477, k1 = 4.6873, h = 100)
```

Purpose

This function returns the list formed by all the input parameters and the corresponding value of $\text{Beta0} = \int \chi_{k_0}(s) d\Phi(s)$.

Arguments

<code>tlo</code>	Relative tolerance in the iterative algorithms.
<code>mxf</code>	Maximum number of iterations in computing the location estimate.
<code>mxs</code>	Maximum number of iterations in computing the scale estimate.
<code>ntm</code>	Parameter used in iteration monitoring. When the number of iterations is a multiple of <code>ntm</code> , the current parameter values are printed.
<code>tls</code>	Tolerance for denominators. If a scale estimate is less than <code>tls</code> , the scale estimate is set equal to <code>tls</code> .
<code>k0</code>	The tuning constant k_0 .
<code>k1</code>	The tuning constant k_1 .
<code>h</code>	The number of subdivisions of the interval $(\min(y_i), \max(y_i))$ used in the computation of the estimate $\lambda^{(0)}$. See Section 3.

Value

List containing the desired values for each of the control parameters, plus the value `Beta0` of β .

Appendix 2

```
boot.test0 <- function(y,z,sigmay,sigmaz,mu,nboot,t0){
  call <- match.call();
  m <- length(y); n <- length(z); lambdastar <- NULL
  lambday <- log(mu)-sigmay^2/2
  lambdaz <- log(mu)-sigmaz^2/2
  for (i in 1:nboot) {
    y <- rlnorm(m,meanlog=lambday,sdlog=sigmay)
    z <- rlnorm(n,meanlog=lambdaz,sdlog=sigmaz)
    gy <- log(y); gz <- log(z)
    ly <- mean(gy); sy <- sqrt(var(gy)); my <- exp(ly+sy^2/2)
    lz <- mean(gz); sz <- sqrt(var(gz)); mz <- exp(lz+sz^2/2)
    V.ly <- sy^2; V.sy <- sy^2/2; V.my <- my^2*V.ly+(my*sy)^2*V.sy
    V.lz <- sz^2; V.sz <- sz^2/2; V.mz <- mz^2*V.lz+(mz*sz)^2*V.sz
    sd <- sqrt(V.my/my^2/m+V.mz/mz^2/n)
    tg <- log(mz/my)/sd
    lambdastar <- c(lambdastar,tg)
    asl <- sum(lambdastar <= t0)/i
    cat(i,format(round(c(my,mz,sd,tg,asl),3)),"\n")}
  list(lambdastar = lambdastar, call = call)}
```

```
boot.test1 <- function(y,z,sigmay,sigmaz,mu,by,bz,opt=1,
                      Taby=NULL,Tabz=NULL,nboot,t0){
  call <- match.call();
  m <- length(y); n <- length(z); lambdastar <- NULL
  lambday <- log(mu)-sigmay^2/2
  lambdaz <- log(mu)-sigmaz^2/2
  for (i in 1:nboot) {
    y <- rlnorm(m,meanlog=lambday,sdlog=sigmay)
    z <- rlnorm(n,meanlog=lambdaz,sdlog=sigmaz)
    ey <- M.E.lnorm(y,b1=by,b2=by,opt=opt,Table=Taby,cov=T)
    ez <- M.E.lnorm(z,b1=bz,b2=bz,opt=opt,Table=Tabz,cov=T)
    my <- ey$mu; mz <- ez$mu
    sd <- sqrt(ey$V.mu/my^2/m+ez$ V.mu/mz^2/n)
    tg <- log(mz/my)/sd
    lambdastar <- c(lambdastar,tg)
    asl <- sum(lambdastar <= t0)/i
    cat(i,format(round(c(my,mz,sd,tg,asl),3)),"\n")}
  list(lambdastar = lambdastar, call = call)}
```

```

hists <- function(y,z,range=rng){ par(mfrow=c(2,1))
hsy <- hist(y, probability=T, plot=F, nclass=50, xlim=range)
barplot(hsy$counts, hsy$breaks, histo=T, nclass=50, xlim=range)
pr1 <- par(); title("y / BE",cex=1.2)
hsz <- hist(z, probability=T, plot=F, nclass=200, xlim=range)
barplot(hsz$counts, hsz$breaks, histo=T, nclass=200, xlim=range)
pr2 <- par(); title("z / CH",cex=1.2)
list(pr1=pr1,pr2=pr2)}

```

```

dens <- function(ly,sy,lz,sz,grid=grd,prs=pr,col=1){
par(pr$pr1); den <- dlnorm(grid,meanlog=ly,sdlog=sy)
lines(grid,den,col=col,lty=1,lwd=2)
par(pr$pr2); den <- dlnorm(grid,meanlog=lz,sdlog=sz)
lines(grid,den,col=col,lty=1,lwd=2)}

```

```

boot.dist <- function(z,tg.0,nc=30){
par(mfrow=c(1,1))
hs <- hist(z,probability=T,plot=F,nclass=nc,xlim=c(-3,3))
barplot(hs$counts,hs$breaks,histo=T,xlim=c(-3,3))
grid <- seq(-3,3,length=100)
dens <- dnorm(grid,0,1); lines(grid,dens,lwd=2); abline(v=tg.0,lwd=2)
title("bootstrap null distribution",cex=1.2)}

```