Package ‘lognorm’

October 13, 2022

Title Functions for the Lognormal Distribution

Version 0.1.10

Author Thomas Wutzler

Maintainer Thomas Wutzler <twutz@bgc-jena.mpg.de>

Description The lognormal distribution
(Limpert et al. (2001) <doi:10.1641/0006-3568(2001)051%5B0341:lndats%5D2.0.co;2>)
can characterize uncertainty that is bounded by zero.
This package provides estimation of distribution parameters, computation of
moments and other basic statistics, and an approximation of the distribution
of the sum of several correlated lognormally distributed variables
(Lo 2013 <doi:10.12988/ams.2013.39511>) and the approximation of the
difference of two correlated lognormally distributed variables
(Lo 2012 <doi:10.1155/2012/838397>).

Imports Matrix

Suggests markdown, rmarkdown, testthat, knitr, dplyr, ggplot2,
mvtnorm, purrr, tidyr

VignetteBuilder knitr

License GPL-2

RoxygenNote 7.1.1

URL `https://github.com/bgctw/lognorm`

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2021-11-21 17:50:10 UTC

R topics documented:

computeEffectiveAutoCorr 2
computeEffectiveNumObs 3
estimateDiffLognormal 4
estimateParmsLognormFromSample 5

estimateSumLognormalSample 6
getCorrMatFromAcf 8
getLognormMoments 9
getParmsLognormForMedianAndUpper 10
scaleLogToOrig 12
seCor 13
setMatrixOffDiagonals 14
varCor 15

Index **16**

`computeEffectiveAutoCorr` *Estimate vector of effective components of the autocorrelation*

Description

Estimate vector of effective components of the autocorrelation

Usage

```
computeEffectiveAutoCorr(res, type = "correlation")
```

Arguments

`res`	numeric of autocorrelated numbers, usually observation - model residuals
`type`	type of residuals (see `acf`)

Details

Returns all components before first negative autocorrelation

Value

numeric vector: strongest components of the autocorrelation function

References

Zieba 2011 Standard Deviation of the Mean of Autocorrelated Observations Estimated with the Use of the Autocorrelation Function Estimated From the Data

Examples

```
# generate autocorrelated time series
res <- stats::filter(rnorm(1000), filter = rep(1,5), circular = TRUE)
res[100:120] <- NA
(effAcf <- computeEffectiveAutoCorr(res))
```

computeEffectiveNumObs

Compute the effective number of observations taking into account autocorrelation

Description

Compute the effective number of observations taking into account autocorrelation

Usage

```
computeEffectiveNumObs(
  res,
  effAcf = computeEffectiveAutoCorr(res),
  na.rm = FALSE,
  exact.na = TRUE
)
```

Arguments

`res`	numeric of autocorrelated numbers, usually observation - model residuals
`effAcf`	autocorrelation coefficients. The first entry is fixed at 1 for zero distance.
`na.rm`	if not set to TRUE will return NA in there are missings in the series
`exact.na`	if set to FALSE then do not count and correct for missing in the sum of autocorrelation terms. This is faster, but results are increasingly biased high with increasing number of missings.

Details

Assumes records of all times present. DO NOT REMOVE OR FILTER NA records before. The length of the time series is used.

Handling of NA values: The formula from Zieba 2011 is extended to subtract the number of missing pairs in the count of correlation terms. If ‘exact.na=false‘ the original formula is used (after trimming edge-NAs).

Value

integer scalar: effective number of observations

References

Zieba & Ramza (2011) Standard Deviation of the Mean of Autocorrelated Observations Estimated with the Use of the Autocorrelation Function Estimated From the Data. Metrology and Measurement Systems,Walter de Gruyter GmbH, 18 10.2478/v10178-011-0052-x

Bayley & Hammersley (1946) The "effective" number of independent observations in an autocorrelated time series. Supplement to the Journal of the Royal Statistical Society, JSTOR,8,184-197

Examples

```
# generate autocorrelated time series
res <- stats::filter(rnorm(1000), filter = rep(1,5), circular = TRUE)
res[100:120] <- NA
# plot the series of autocorrelated random variables
plot(res)
# plot their empirical autocorrelation function
acf(res, na.action = na.pass)
#effAcf <- computeEffectiveAutoCorr(res)
# the effective number of parameters is less than number of 1000 samples
(nEff <- computeEffectiveNumObs(res, na.rm = TRUE))
```

---

estimateDiffLognormal *Inference on the difference of two lognormals*

Description

The distribution of y = a - b + s, where a and b are two lognormal random variables and s is a constant to be estimated, can be approximated by a lognormal distribution.

Usage

```
estimateDiffLognormal(mu_a, mu_b, sigma_a, sigma_b, corr = 0)

pDiffLognormalSample(
  mu_a,
  mu_b,
  sigma_a,
  sigma_b,
  corr = 0,
  q = 0,
  nSample = 1e+05
)
```

Arguments

mu_a	center parameter of the first term
mu_b	center parameter of the second term
sigma_a	scale parameter of the first term
sigma_b	scale parameter of the second term
corr	correlation between the two random variables
q	vector of quantiles
nSample	number of samples

Value

estimateDiffLognormal: numeric vector with components mu, sigma, and shift, the components of the shifted lognormal distribution.

pDiffLognormalSample: vector of probabilities

Functions

- `estimateDiffLognormal`: Estimate the shifted-lognormal approximation to difference of two lognormals
- `pDiffLognormalSample`: Distribution function for the difference of two lognormals based on sampling. Default provides the probability that the difference is significantly larger than zero.

`estimateParmsLognormFromSample`
Estimate lognormal distribution parameters from a sample

Description

Estimate lognormal distribution parameters from a sample

Usage

```
estimateParmsLognormFromSample(x, na.rm = FALSE)

estimateStdErrParms(x, na.rm = FALSE)
```

Arguments

`x`	numeric vector of sampled values
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

The expected value of a can be determined with higher accuracy the larger the sample. Here, the uncorrelated assumption is applied at the log scale and distribution parameters are returned with the same expected value as the sample, but with uncertainty (sigma) decreased by sqrt(nfin - 1).

Since with low relative error, the lognormal becomes very close to the normal distribution, the distribution of the mean can be well approximated by a normal with sd(mean(x)) ~ sd(x)/sqrt(n-1).

Value

numeric vector with components `mu` and `sigma`, i.e., the center parameter (mean at log scale, log(median)) and the scale parameter (standard deviation at log scale)

Functions

- `estimateParmsLognormFromSample`: Estimate lognormal distribution parameters from a sample
- `estimateStdErrParms`: Estimate parameters of the lognormal distribution of the mean from an uncorrelated sample

Examples

```
.mu <- log(1)
.sigma <- log(2)
n = 200
x <- exp(rnorm(n, mean = .mu, sd = .sigma))
exp(pL <- estimateParmsLognormFromSample(x)) # median and multiplicative stddev
c(mean(x), meanx <- getLognormMoments(pL["mu"],pL["sigma"])[,"mean"])
c(sd(x), sdx <- sqrt(getLognormMoments(pL["mu"],pL["sigma"])[,"var"]))

# stddev decreases (each sample about 0.9) to about 0.07
# for the mean with n replicated samples
se <- estimateStdErrParms(x)
sqrt(getLognormMoments(se["mu"],se["sigma"])[,"var"])
sd(x)/sqrt(n-1) # well approximated by normal
# expected value stays the same
c(meanx, getLognormMoments(se["mu"],se["sigma"])[,"mean"])
```

---

estimateSumLognormalSample

Estimate the parameters of the lognormal approximation to the sum

---

Description

Estimate the parameters of the lognormal approximation to the sum

Estimate the parameters of the lognormal approximation to the sum

Usage

```
estimateSumLognormalSample(
  mu,
  sigma,
  resLog,
  effAcf = computeEffectiveAutoCorr(resLog),
  isGapFilled = logical(0),
  na.rm = TRUE
)

estimateSumLognormalSampleExpScale(mean, sigmaOrig, ...)

estimateSumLognormal(
```

```
  mu,
  sigma,
  effAcf = c(),
  corr = Diagonal(length(mu)),
  corrLength = if (inherits(corr, "ddiMatrix")) 0 else nTerm,
  sigmaSum = numeric(0),
  isStopOnNoTerm = FALSE,
  na.rm = isStopOnNoTerm
)
```

Arguments

`mu`	numeric vector of center parameters of terms at log scale
`sigma`	numeric vector of scale parameter of terms at log scale
`resLog`	time series of model-residuals at log scale to estimate correlation
`effAcf`	numeric vector of effective autocorrelation This overrides arguments `corr` and `corrLength`
`isGapFilled`	logical vector whether entry is gap-filled rather than an original measurement, see details
`na.rm`	neglect terms with NA values in mu or sigma
`mean`	numeric vector of expected values
`sigmaOrig`	numeric vector of standard deviation at original scale
`...`	further arguments passed to `estimateSumLognormalSample`
`corr`	numeric matrix of correlations between the random variables
`corrLength`	integer scalar: set correlation length to smaller values to speed up computation by neglecting correlations among terms further apart. Set to zero to omit correlations.
`sigmaSum`	numeric scalar: possibility to specify a precomputed scale parameter instead of computing it.
`isStopOnNoTerm`	if no finite estimate is provided then by default NA is returned for the sum. Set this to TRUE to issue an error instead.

Details

If there are no gap-filled values, i.e. `all(!isGapFilled)` or `!length(isGapFilled)` (the default), distribution parameters are estimated using all the samples. Otherwise, the scale parameter (uncertainty) is first estimated using only the non-gapfilled records.

Also use isGapFilled == TRUE for records, where sigma cannot be trusted. When setting sigma to missing, this is also affecting the expected value.

If there are only gap-filled records, assume uncertainty to be (before v0.1.5: the largest uncertainty of given gap-filled records.) the mean of the given multiplicative standard deviation

Value

numeric vector with components `mu`, `sigma`, and `nEff`, i.e. the parameters of the lognormal distribution at log scale and the number of effective observations.

Functions

- `estimateSumLognormalSample`: In addition to `estimateSumLognormal` take care of missing values and estimate correlation terms.
- `estimateSumLognormalSampleExpScale`: Before calling `estimateSumLognormalSample` estimate lognormal parameters from value and its uncertainty given on original scale.
- `estimateSumLognormal`: Estimate the parameters of the lognormal approximation to the sum

References

Lo C (2013) WKB approximation for the sum of two correlated lognormal random variables. Applied Mathematical Sciences, Hikari, Ltd., 7 , 6355-6367 10.12988/ams.2013.39511

Examples

```
# distribution of the sum of two lognormally distributed random variables
mu1 = log(110)
mu2 = log(100)
sigma1 = log(1.2)
sigma2 = log(1.6)
(coefSum <- estimateSumLognormal(
c(mu1,mu2), c(sigma1,sigma2) ))
# repeat with correlation
(coefSumCor <- estimateSumLognormal(
c(mu1,mu2), c(sigma1,sigma2), effAcf = c(1,0.9) ))
# expected value is equal, but variance with correlated variables is larger
getLognormMoments(coefSum["mu"],coefSum["sigma"])
getLognormMoments(coefSumCor["mu"],coefSumCor["sigma"])
```

---

getCorrMatFromAcf — *Construct the full correlation matrix from autocorrelation components.*

Description

Construct the full correlation matrix from autocorrelation components.

Usage

```
getCorrMatFromAcf(nRow, effAcf)
```

Arguments

`nRow`	number of rows in correlation matrix
`effAcf`	numeric vector of effective autocorrelation components . The first entry, which is defined as 1, is not used.

`getLognormMoments` *Compute summary statistics of a log-normal distribution*

Description

Compute summary statistics of a log-normal distribution

Usage

```
getLognormMoments(mu, sigma, m = exp(mu + sigma2/2) - shift, shift = 0)

getLognormMedian(mu, sigma, shift = 0)

getLognormMode(mu, sigma, shift = 0)
```

Arguments

`mu`	numeric vector: location parameter
`sigma`	numeric vector: scale parameter
`m`	mean at original scale, may override default based on mu
`shift`	shift for the shifted lognormal distribution

Value

for `getLognormMoments` a numeric matrix with columns `mean` (expected value at original scale) , `var` (variance at original scale) , and `cv` (coefficient of variation: sqrt(var)/mean). For the other functions a numeric vector of the required summary.

Functions

- `getLognormMoments`: get the expected value, variance, and coefficient of variation
- `getLognormMedian`: get the median
- `getLognormMode`: get the mode

References

Limpert E, Stahel W & Abbt M (2001) Log-normal Distributions across the Sciences: Keys and Clues. Oxford University Press (OUP) 51, 341,10.1641/0006-3568(2001)051[0341:lndats]2.0.co;2

See Also

scaleLogToOrig

Examples

```
# start by estimating lognormal parameters from moments
.mean <- 1
.var <- c(1.3,2)^2
parms <- getParmsLognormForMoments(.mean, .var)
#
# computed moments must equal previous ones
(ans <- getLognormMoments(parms[,"mu"], parms[,"sigma"]))
cbind(.var, ans[,"var"])
#
getLognormMedian(mu = log(1), sigma = log(2))
getLognormMode(mu = log(1), sigma = c(log(1.2),log(2)))
```

---

getParmsLognormForMedianAndUpper

Calculate mu and sigma of lognormal from summary statistics.

Description

Calculate mu and sigma of lognormal from summary statistics.

Usage

```
getParmsLognormForMedianAndUpper(median, upper, sigmaFac = qnorm(0.99))

getParmsLognormForMeanAndUpper(mean, upper, sigmaFac = qnorm(0.99))

getParmsLognormForLowerAndUpper(lower, upper, sigmaFac = qnorm(0.99))

getParmsLognormForLowerAndUpperLog(lowerLog, upperLog, sigmaFac = qnorm(0.99))

getParmsLognormForModeAndUpper(mle, upper, sigmaFac = qnorm(0.99))

getParmsLognormForMoments(mean, var, sigmaOrig = sqrt(var))

getParmsLognormForExpval(mean, sigmaStar)
```

Arguments

`median`	geometric mu (median at the original exponential scale)
`upper`	numeric vector: value at the upper quantile, i.e. practical maximum
`sigmaFac`	sigmaFac=2 is 95% sigmaFac=2.6 is 99% interval.
`mean`	expected value at original scale
`lower`	value at the lower quantile, i.e. practical minimum
`lowerLog`	value at the lower quantile, i.e. practical minimum at log scale

upperLog	value at the upper quantile, i.e. practical maximum at log scale
mle	numeric vector: mode at the original scale
var	variance at original scale
sigmaOrig	standard deviation at original scale
sigmaStar	multiplicative standard deviation

Details

For getParmsLognormForMeanAndUpper there are two valid solutions, and the one with lower sigma , i.e. the not so strongly skewed solution is returned.

Value

numeric matrix with columns ‘mu‘ and ‘sigma‘, the parameter of the lognormal distribution. Rows correspond to rows of inputs.

Functions

- getParmsLognormForMedianAndUpper: Calculates mu and sigma of lognormal from median and upper quantile.
- getParmsLognormForMeanAndUpper: Calculates mu and sigma of lognormal from mean and upper quantile.
- getParmsLognormForLowerAndUpper: Calculates mu and sigma of lognormal from lower and upper quantile.
- getParmsLognormForLowerAndUpperLog: Calculates mu and sigma of lognormal from lower and upper quantile at log scale.
- getParmsLognormForModeAndUpper: Calculates mu and sigma of lognormal from mode and upper quantile.
- getParmsLognormForMoments: Calculate mu and sigma from moments (mean anc variance)
- getParmsLognormForExpval: Calculate mu and sigma from expected value and geometric standard deviation

References

Limpert E, Stahel W & Abbt M (2001) Log-normal Distributions across the Sciences: Keys and Clues. Oxford University Press (OUP) 51, 341,10.1641/0006-3568(2001)051[0341:lndats]2.0.co;2

Examples

```
# example 1: a distribution with mode 1 and upper bound 5
(thetaEst <- getParmsLognormForModeAndUpper(1,5))
mle <- exp(thetaEst[1] - thetaEst[2]^2)
all.equal(mle, 1, check.attributes = FALSE)

# plot the distributions
xGrid = seq(0,8, length.out = 81)[-1]
dxEst <- dlnorm(xGrid, meanlog = thetaEst[1], sdlog = thetaEst[2])
```

```
plot( dxEst~xGrid, type = "l",xlab = "x",ylab = "density")
abline(v = c(1,5),col = "gray")

# example 2: true parameters, which should be rediscovered
theta0 <- c(mu = 1, sigma = 0.4)
mle <- exp(theta0[1] - theta0[2]^2)
perc <- 0.975 # some upper percentile, proxy for an upper bound
upper <- qlnorm(perc, meanlog = theta0[1], sdlog = theta0[2])
(thetaEst <- getParmsLognormForModeAndUpper(
  mle,upper = upper,sigmaFac = qnorm(perc)) )

#plot the true and the rediscovered distributions
xGrid = seq(0,10, length.out = 81)[-1]
dx <- dlnorm(xGrid, meanlog = theta0[1], sdlog = theta0[2])
dxEst <- dlnorm(xGrid, meanlog = thetaEst[1], sdlog = thetaEst[2])
plot( dx~xGrid, type = "l")
#plot( dx~xGrid, type = "n")
#overplots the original, coincide
lines( dxEst ~ xGrid, col = "red", lty = "dashed")

# example 3: explore varying the uncertainty (the upper quantile)
x <- seq(0.01,1.2,by = 0.01)
mle = 0.2
dx <- sapply(mle*2:8,function(q99){
  theta = getParmsLognormForModeAndUpper(mle,q99,qnorm(0.99))
  #dx <- dDistr(x,theta[,"mu"],theta[,"sigma"],trans = "lognorm")
  dx <- dlnorm(x,theta[,"mu"],theta[,"sigma"])
})
  matplot(x,dx,type = "l")
# Calculate mu and sigma from expected value and geometric standard deviation
.mean <- 1
.sigmaStar <- c(1.3,2)
(parms <- getParmsLognormForExpval(.mean, .sigmaStar))
# multiplicative standard deviation must equal the specified value
cbind(exp(parms[,"sigma"]), .sigmaStar)
```

`scaleLogToOrig` *Scale standard deviation between log and original scale.*

Description

When comparing values at log scale that have different sd at original scale, better compare log(mean) instead of mu.

Usage

```
scaleLogToOrig(logmean, sigma)

scaleOrigToLog(mean, sd)
```

Arguments

`logmean`	log of the expected value
`sigma`	standard deviation at log scale
`mean`	expected value at original scale
`sd`	standard deviation at original scale

Value

numeric matrix with columns `mean`, and `sd` at original scale

Functions

- `scaleLogToOrig`: get logmean and sigma at log scale
- `scaleOrigToLog`: get mean and sd at original scale

Examples

```
xLog <- data.frame(logmean = c(0.8, 0.8), sigma = c(0.2, 0.3))
xOrig <- as.data.frame(scaleLogToOrig(xLog$logmean, xLog$sigma))
xLog2 <- as.data.frame(scaleOrigToLog(xOrig$mean, xOrig$sd))
all.equal(xLog, xLog2)
xLog3 <- as.data.frame(getParmsLognormForMoments(xOrig$mean, xOrig$sd^2))
all.equal(xLog$sigma, xLog3$sigma) # but mu  < logmean
```

`seCor` *Compute the standard error accounting for empirical autocorrelations*

Description

Compute the standard error accounting for empirical autocorrelations

Usage

```
seCor(
  x,
 effCor = if (missing(effCov)) computeEffectiveAutoCorr(x) else effCov/var(x, na.rm =
    TRUE),
  na.rm = FALSE,
  effCov,
  nEff = computeEffectiveNumObs(x, effCor, na.rm = na.rm)
)
```

Arguments

`x`	numeric vector
`effCor`	numeric vector of effective correlation components first entry at zero lag equals one. See `computeEffectiveAutoCorr`
`na.rm`	logical. Should missing values be removed?
`effCov`	alternative to specifying effCor: numeric vector of effective covariance components first entry is the variance. See `computeEffectiveAutoCorr`
`nEff`	possibility to specify precomputed number of effective observations for speedup.

Details

The default uses empirical autocorrelation estimates from the supplied data up to first negative component. For short series of `x` it is strongly recommended to to provide `effCov` that was estimated on a longer time series.

Value

numeric scalar of standard error of the mean of x

`setMatrixOffDiagonals` *set off-diagonal values of a matrix*

Description

set off-diagonal values of a matrix

Usage

```
setMatrixOffDiagonals(x, diag = 1:length(value), value, isSymmetric = FALSE)
```

Arguments

`x`	numeric square matrix
`diag`	integer vector specifying the diagonals 0 is the center +1 the first row to upper and -2 the second row to lower
`value`	numeric vector of values to fill in
`isSymmetric`	set to TRUE to to only specify the upper diagonal element but also set the lower in the mirrored diagonal

Value

matrix with modified diagonal elements

varCor — *Compute the unbiased variance accounting for empirical autocorrelations*

Description

Compute the unbiased variance accounting for empirical autocorrelations

Usage

```
varCor(
  x,
  effCor = computeEffectiveAutoCorr(x),
  na.rm = FALSE,
  nEff = computeEffectiveNumObs(x, effAcf = effCor)
)
```

Arguments

`x`	numeric vector
`effCor`	numeric vector of effective correlation components first entry at zero lag equals one. See `computeEffectiveAutoCorr` The effective correlation is passed to `computeEffectiveNumObs`.
`na.rm`	logical. Should missing values be removed?
`nEff`	possibility to specify precomputed number of effective observations for speedup.

Details

The default uses empirical autocorrelation estimates from the supplied data up to first negative component. For short series of `x` it is strongly recommended to to provide `effCov` that was estimated on a longer time series.

Value

numeric scalar of unbiased variation of x

Index

acf, *2*

computeEffectiveAutoCorr, 2, *14, 15*
computeEffectiveNumObs, 3, *15*

estimateDiffLognormal, 4
estimateParmsLognormFromSample, 5
estimateStdErrParms (estimateParmsLognormFromSample), 5
estimateSumLognormal (estimateSumLognormalSample), 6
estimateSumLognormalSample, 6
estimateSumLognormalSampleExpScale (estimateSumLognormalSample), 6

getCorrMatFromAcf, 8
getLognormMedian (getLognormMoments), 9
getLognormMode (getLognormMoments), 9
getLognormMoments, 9
getParmsLognormForExpval (getParmsLognormForMedianAndUpper), 10
getParmsLognormForLowerAndUpper (getParmsLognormForMedianAndUpper), 10
getParmsLognormForLowerAndUpperLog (getParmsLognormForMedianAndUpper), 10
getParmsLognormForMeanAndUpper (getParmsLognormForMedianAndUpper), 10
getParmsLognormForMedianAndUpper, 10
getParmsLognormForModeAndUpper (getParmsLognormForMedianAndUpper), 10
getParmsLognormForMoments (getParmsLognormForMedianAndUpper), 10

pDiffLognormalSample (estimateDiffLognormal), 4

scaleLogToOrig, 12
scaleOrigToLog (scaleLogToOrig), 12
seCor, 13
setMatrixOffDiagonals, 14

varCor, 15