Package: sgt (via r-universe)

August 24, 2024

Version 2.0

Type Package

Title Skewed Generalized T Distribution Tree

Date 2015-08-30

Author Carter Davis

Maintainer Carter Davis <cdavis40@chicagobooth.edu>

Depends R (>= 3.0.0), optimx (>= 2013.8.7), numDeriv (>= 2014.2-1)

Description Density, distribution function, quantile function and random generation for the skewed generalized t distribution. This package also provides a function that can fit data to the skewed generalized t distribution using maximum likelihood estimation.

License GPL (>= 3)

Repository CRAN

NeedsCompilation no

Date/Publication 2015-09-04 01:00:18

Contents

sgt *The Skewed Generalized T Distribution*

Description

Density, distribution function, quantile function and random generation for the skewed generalized t distribution.

Usage

```
dsgt(x, mu = 0, sigma = 1, lambda = 0, p = 2, q = Inf,
mean.cent = TRUE, var.adj = TRUE, log = FALSE)
psgt(quant, mu = 0, sigma = 1, lambda = 0, p = 2, q = Inf,
mean.cent = TRUE, var.adj = TRUE, lower.tail = TRUE,
log.p = FALSE)
qsgt(prob, mu = 0, sigma = 1, lambda = 0, p = 2, q = Inf,
mean.cent = TRUE, var.adj = TRUE, lower.tail = TRUE,
log.p = FALSE)
rsgt(n, mu = 0, sigma = 1, lambda = 0, p = 2, q = Inf,
mean.cent = TRUE, var.adj = TRUE)
```

Arguments

x, quant	vector of quantiles.
prob	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
mu	vector of parameters. Note that if mean.cent == TRUE, mu is the mean of the distribution. Otherwise, mu is the mode of the distribution.
sigma	vector of variance parameters. The default is 1. The variance of the distribution increases as sigma increases. Must be strictly positive.
lambda	vector of skewness parameters. Note that -1 < lambda < 1. If lambda < 0, the distribution is skewed to the left. If lambda > 0, the distribution is skewed to the right. If lambda = 0, then the distribution is symmetric.
p, q	vector of parameters. Smaller values of p and q result in larger values for the kurtosis of the distribution. Allowed to be infinite. Note that p > 0, q > 0, otherwise NaNs will be produced.
mean.cent	logical; if TRUE, mu is the mean of the distribution, otherwise mu is the mode of the distribution. May only be used if p*q > 1, otherwise NaNs will be produced.
var.adj	logical or a positive scalar. If TRUE, then sigma is rescaled so that sigma is the variance. If FALSE, then sigma is not rescaled. If var.adj is a positive scalar, then sigma is rescaled by var.adj. May only be used if p*q > 2, otherwise NaNs will be produced.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \le x]$ otherwise, $P[X > x]$.

Details

If mu, sigma, lambda, p, or q are not specified they assume the default values of mu = 0, sigma = 1, lambda = 0, p = 2, and q = Inf. These default values yield a standard normal distribution.

See vignette('sgt') for the probability density function, moments, and various special cases of the skewed generalized t distribution.

Value

dsgt gives the density, psgt gives the distribution function, qsgt gives the quantile function, and rsgt generates random deviates.

The length of the result is determined by n for rsgt, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

sigma <= 0, lambda <= -1, lambda >= 1, p <= 0, and q <= 0 are errors and return NaN. Also, if mean.cent is TRUE but codep*q <= 1, the result is an error and NaNs are produced. Similarly, if var.adj is TRUE but codep*q <= 2, the result is an error and NaNs are produced.

Author(s)

Carter Davis, <carterdavis@byu.edu>

Source

For psgt, based on

a transformation of the cumulative probability density function that uses the incomplete beta function or incomplete gamma function.

For qsgt, based on

solving for the inverse of the psgt function that uses the inverse of the incomplete beta function or incomplete gamma function.

For rsgt, the algorithm simply uses the qsgt function with probabilities that are uniformly distributed.

References

Hansen, C., McDonald, J. B., and Newey, W. K. (2010) "Instrumental Variables Regression with Flexible Distributions" *Journal of Business and Economic Statistics*, volume 28, 13-25.

Kerman, S. C., and McDonald, J. B. (2012) "Skewness-Kurtosis Bounds for the Skewed Generalized T and Related Distributions" *Statistics and Probability Letters*, volume 83, 2129-2134.

Theodossiou, Panayiotis (1998) "Financial Data and the Skewed Generalized T Distribution" *Management Science*, volume 44, 1650-1661.

See Also

Distributions for other standard distributions which are special cases of the skewed generalized t distribution, including dt for the t distribution, dnorm for the normal distribution, and dunif for the uniform distribution. Other special cases of the skewed generalized t distribution include the generalized t distribution in the gamlss.dist package, the skewed t distribution in the skewt package, the exponential power distribution (also known as the generalized error distribution) in the normalp package, and the Laplace distribution in the rmutil package. Also see beta for the beta function.

Examples

```
require(graphics)

### This shows how to get a normal distribution
x = seq(-4,6,by=0.05)
plot(x, dnorm(x, mean=1, sd=1.5), type='l')
lines(x, dsgt(x, mu=1, sigma=1.5), col='blue')

### This shows how to get a cauchy distribution
plot(x, dcauchy(x, location=1, scale=1.3), type='l')
lines(x, dsgt(x, mu=1, sigma=1.3, q=1/2, mean.cent=FALSE, var.adj = sqrt(2)), col='blue')

### This shows how to get a Laplace distribution
plot(x, dsgt(x, mu=1.2, sigma=1.8, p=1, var.adj=FALSE), type='l', col='blue')

### This shows how to get a uniform distribution
plot(x, dunif(x, min=1.2, max=2.6), type='l')
lines(x, dsgt(x, mu=1.9, sigma=0.7, p=Inf, var.adj=FALSE), col='blue')
```

---

sgtmle — *Maximum Likelihood Estimation with the Skewed Generalized T Distribution*

Description

This function allows data to be fit to the skewed generalized t distribution using maximum likelihood estimation. This function uses the maxLik package to perform its estimations.

Usage

```
sgt.mle(X.f, mu.f = mu ~ mu, sigma.f = sigma ~ sigma,
lambda.f = lambda ~ lambda, p.f = p ~ p, q.f = q ~ q,
data = parent.frame(), start, subset,
method = c("Nelder-Mead", "BFGS"), itnmax = NULL,
hessian.method="Richardson",
gradient.method="Richardson",
mean.cent = TRUE, var.adj = TRUE, ...)
```

Arguments

X.f — A formula specifying the data, or the function of the data with parameters, that should be used in the maximisation procedure. X should be on the left-hand side and the right-hand side should be the data or function of the data that should be used.

mu.f, sigma.f, lambda.f, p.f, q.f — formulas including variables and parameters that specify the functional form of the parameters in the skewed generalized t log-likelihood function. mu, sigma, lambda, p, and q should be on the left-hand side of these formulas respectively.

data — an optional data frame in which to evaluate the variables in formula and weights. Can also be a list or an environment.

start — a named list or named numeric vector of starting estimates for every parameter.

subset — an optional vector specifying a subset of observations to be used in the fitting process.

method — A list of the optimization methods to be used, which is passed directly to the optimx function in the optimx package. See ?optimx for a list of methods that can be used. Note that the method that achieves the highest log-likelihood value is the method that is printed and reported. The default method is to use both "Nelder-Mead" and the "BFGS" methods.

itnmax — If provided as a vector of the same length as method, gives the maximum number of iterations or function values for the corresponding method. If a single number is provided, this will be used for all methods.

hessian.method — method used to calculate the hessian of the final estimates, either "Richardson" or "complex". This method is passed to the hessian function in the numDeriv package. See ?hessian for details.

gradient.method — method used to calculate the gradient of the final estimates, either "Richardson", "simple", or "complex". This method is passed to the grad function in the numDeriv package. See ?grad for details.

mean.cent, var.adj — arguments passed to the skewed generalized t distribution function (see ?dsgt).

... — further arguments that are passed to the control argument in the optimx function in the optimx package. See ?optimx for a list of arguments that can be used in the control argument.

Details

The parameter names are taken from start. If there is a name of a parameter or some data found on the right-hand side of one of the formulas but not found in data and not found in start, then an error is given.

This function simply uses the optimx function in the optimx package to maximize the skewed generalized t distribution log-likelihood function. It takes the method that returned the highest log-likelihood, and saves these results as the final estimates.

Value

sgt.mle returns a list of class "sgtest". A list of class "sgtest" has the following components:

- maximum: log-likelihood value of estimates (the last calculated value if not converged) of the method that achieved the greatest log-likelihood value.
- estimate: estimated parameter value with the method that achieved the greatest log-likelihood value.
- convcode: convcode returned from the optimx function in the optimx package of the method that achieved the greatest log-likelihood value. See ?optimx for the different convcode values.
- niter: The amount of iterations that the method which achieved the the greatest log-likelihood value used to reach its estimate.
- best.method.used: name of the method that achieved the greatest log-likelihood value.
- optimx: A data.frame of class "optimx" that contains the results of the optimx maximization for every method (*not* just the method that achieved the highest log-likelihood value). See ?optimx for details.
- gradient: vector, gradient value of the estimates with the method that achieved the greatest log-likelihood value.
- hessian: matrix, hessian of the estimates with the method that achieved the greatest log-likelihood value.
- varcov: variance/covariance matrix of the maximimum likelihood estimates
- std.error: standard errors of the estimates

Author(s)

Carter Davis, <carterdavis@byu.edu>

References

Davis, Carter, James McDonald, and Daniel Walton (2015). "A Generalized Regression Specification using the Skewed Generalized T Distribution" working paper.

See Also

The optimx package and its documentation. The sgt.mle simply uses its functions to maximize the skewed generalized t log-likelihood. Also, the sgt.mle function uses the numDeriv package to compute the final hessian and gradients of the estimates.

Examples

```
# SINGLE VARIABLE ESTIMATION:
### generate random variable
set.seed(7900)
n = 1000
x = rsgt(n, mu = 2, sigma = 2, lambda = -0.25, p = 1.7, q = 7)
```

```
### Get starting values and estimate the parameter values
start = list(mu = 0, sigma = 1, lambda = 0, p = 2, q = 10)
result = sgt.mle(X.f = ~ x, start = start, method = "nlminb")
print(result)
print(summary(result))

# REGRESSION MODEL ESTIMATION:
### Generate Random Data
set.seed(1253)
n = 1000
x1 = rnorm(n)
x2 = runif(n)
y = 1 + 2*x1 + 3*x2 + rnorm(n)
data = as.data.frame(cbind(y, x1, x2))

### Estimate Linear Regression Model
reg = lm(y ~ x1 + x2, data = data)
coef = as.numeric(reg$coefficients)
rmse = summary(reg)$sigma
start = c(b0 = coef[1], b1 = coef[2], b2 = coef[3],
g0 = log(rmse)+log(2)/2, g1 = 0, g2 = 0, d0 = 0,
d1 = 0, d2 = 0, p = 2, q = 10)

### Set up Model
X.f = X ~ y - (b0 + b1*x1 + b2*x2)
mu.f = mu ~ 0
sigma.f = sigma ~ exp(g0 + g1*x1 + g2*x2)
lambda.f = lambda ~ (exp(d0 + d1*x1 + d2*x2)-1)/(exp(d0 + d1*x1 + d2*x2)+1)

### Estimate Regression with a skewed generalized t error term
### This estimates the regression model from the Davis,
### McDonald, and Walton (2015) paper cited in the references section
### q is in reality infinite since the error term is normal
result = sgt.mle(X.f = X.f, mu.f = mu.f, sigma.f = sigma.f,
lambda.f = lambda.f, data = data, start = start,
var.adj = FALSE, method = "nlm")
print(result)
print(summary(result))
```

`summary.MLE` *Summary the Maximum-Likelihood Estimation with the Skewed Generalized T Distribution*

Description

Summary the maximum-likelihood estimation including standard errors and t-values.

Usage

```
## S3 method for class 'MLE'
```

```
summary(object, ...)
## S3 method for class 'mult.MLE'
summary(object, ...)
```

Arguments

object	object of class 'MLE' or of class 'mult.MLE', usually a result from maximum-likelihood estimation.
...	currently not used.

Value

summary.MLE returns an object of class 'summary.MLE' with the following components:

parameters	names of parameters used in the estimation procedure.
type	type of maximisation.
iterations	number of iterations.
code	code of success.
message	a short message describing the code.
loglik	the loglik value in the maximum.
estimate	numeric matrix, the first column contains the parameter estimates, the second the standard errors, third t-values and fourth corresponding probabilities.
fixed	logical vector, which parameters are treated as constants.
NActivePar	number of free parameters.
constraints	information about the constrained optimization. Passed directly further from maxim-object. NULL if unconstrained maximization.

summary.mult.MLE returns a list of class 'summary.mult.MLE' with components of class 'summary.MLE'.

Author(s)

Carter Davis, <carterdavis@byu.edu>

See Also

the maxLik CRAN package

Examples

```
### Showing how to fit a simple vector of data to the skewed
### generalized t distribution.
require(graphics)
require(stats)
set.seed(123456)
x = rt(100, df=10)
X.f = X ~ x
start = list(mu = 0, sigma = 2, lambda = 0, p = 2, q = 12)
result = sgt.mle(X.f = X.f, start = start, finalHessian = "BHHH")
```

```
sumResult = summary(result)
print(result)
coef(result)
print(sumResult)
### Note that the t distribution is a special case of the
### skewed generalized t distribution
```

---

summary.sgtest — *Summary the Maximum-Likelihood Estimation with the Skewed Generalized T Distribution*

Description

Summary the maximum-likelihood estimation.

Usage

```
## S3 method for class 'sgtest'
summary(object, ...)
```

Arguments

`object`	object of class `'sgtest'`, usually a result from maximum-likelihood estimation.
`...`	currently not used.

Value

`summary.sgtest` returns an object of class `'summary.sgtest'` with the following components:

`maximum`	log-likelihood value of estimates (the last calculated value if not converged) of the method that achieved the greatest log-likelihood value.
`estimate`	estimated parameter value with the method that achieved the greatest log-likelihood value.
`convcode`	`convcode` returned from the `optimx` function in the `optimx` package of the method that achieved the greatest log-likelihood value. See `?optimx` for the different `convcode` values.
`niter`	The amount of iterations that the method which achieved the the greatest log-likelihood value used to reach its estimate.
`best.method.used`	name of the method that achieved the greatest log-likelihood value.
`optimx`	A `data.frame` of class `"optimx"` that contains the results of the `optimx` maximization for every method (*not* just the method that achieved the highest log-likelihood value). See `?optimx` for details.
`gradient`	vector, gradient value of the estimates with the method that achieved the greatest log-likelihood value.

`hessian`	matrix, hessian of the estimates with the method that achieved the greatest log-likelihood value.
`varcov`	variance/covariance matrix of the maximimum likelihood estimates
`std.error`	standard errors of the estimates
`z.score`	the z score of the estimates
`p.value`	the p-values of the estimates
`summary.table`	a `data.frame` containing the estimates, standard errors, z scores, and p-values of the estimates.

Author(s)

Carter Davis, <cdavis40@chicagobooth.edu>

See Also

the `optimx` CRAN package

Examples

```
# SINGLE VARIABLE ESTIMATION:
### generate random variable
set.seed(7900)
n = 1000
x = rsgt(n, mu = 2, sigma = 2, lambda = -0.25, p = 1.7, q = 7)

### Get starting values and estimate the parameter values
start = list(mu = 0, sigma = 1, lambda = 0, p = 2, q = 10)
result = sgt.mle(X.f = ~ x, start = start, method = "nlminb")
print(result)
print(summary(result))
```

Index