Package 'ExtDist'

October 17, 2014

Version 0.3.7

License GPL (>= 2)

Title Extended Probability Distribution Functions

Description The package provides a consistent, unified and extensible framework for parameter estimation of probability distributions; it extends parameter estimation procedures to allow for weighted samples; moreover, it extends the gallery of available distributions.

Author Haizhen Wu <h.wu2@massey.ac.nz>,A. Jonathan R. Godfrey <A.J.Godfrey@massey.ac.nz>,Kondaswamy Govindaraju <k.govindaraju@massey.ac.nz>

Maintainer Haizhen Wu <h.wu2@massey.ac.nz>

Imports numDeriv, optimx, VGAM, SuppDists, truncdist

Suggests knitr, ggplot2, xtable, PerformanceAnalytics

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2014-10-17 13:05:48

R topics documented:

`ExtDist-package` *Extended Probability Distribution Functions*

Description

The package provides a consistent, unified and extensible framework for parameter estimation of probability distributions; it extends parameter estimation procedures to allow for weighted samples; moreover, it extends the gallery of available distributions.

Details

Package:	ExtDist
Version:	0.3.7
License:	GPL (>= 2)
Imports:	numDeriv, optimx, VGAM, SuppDists, truncdist
Suggests:	knitr, ggplot2, xtable, PerformanceAnalytics
Roxygen:	list(wrap = FALSE)
VignetteBuilder:	knitr
Built:	R 3.1.1; ; 2014-10-17 03:52:27 UTC; unix

Index:

```
Beta                    The standard Beta Distribution.
Beta_ab                 The four-Parameter beta Distribution.
Burr                    The Burr's Distribution.
DistSelCriteriaValues   Distribution Selection Criteria Values.
ExtDist-package         Extended Probability Distribution Functions
```

Gamma	The Gamma Distribution.
Gumbel	The Gumbel Distribution.
JohnsonSB	The Johnson SB Distribution.
JohnsonSU	The Johnson SU Distribution.
Laplace	The Laplace Distribution.
Logistic	The Logistic Distribution.
Normal	The Normal Distribution.
Normal_sym_trunc_ab	The symmetric truncated normal distribution.
Normal_trunc_ab	The truncated normal distribution.
SRTB_ab	The Symmetric-Reflected Truncated Beta (SRTB) Distribution.
SSRTB	The standard Symmetric-Reflected Truncated Beta (SRTB) Distribution.
Triangular	The Triangular Distribution.
Uniform	The Uniform Distribution.
Weibull	The Weibull Distribution.
bestDist	Best distribution for (weighted) sample.
compareDist	Compare sample and fitted distributions
eDist	S3 methods from manuplating eDist objects.
eval.estimation	Parameter Estimation Evaluation.
wmle	Weighted Maximum Likelihood Estimation.

Further information is available in the following vignettes:

Distributions-Beta	Distributions-Beta (source)
Distributions-Normal	Distributions-Normal (source)
Distributions	Distributions-Index (source)
ParaEst-and-DistSel-by-ExtDist	Parameter-Estimation-and-Distribution-Selection-by-ExtDist (source)

Author(s)

Haizhen Wu <h.wu2@massey.ac.nz>, A. Jonathan R. Godfrey <A.J.Godfrey@massey.ac.nz>, Kondaswamy Govindaraju <k.govindaraju@massey.ac.nz>

Maintainer: Haizhen Wu <h.wu2@massey.ac.nz>

`bestDist` *Best distribution for (weighted) sample.*

Description

A function to choose the best fitted distibution based on specified criteria.

Usage

```
bestDist(X, w = rep(1, length(X))/length(X), candDist = c("Beta_ab",
  "Laplace", "Normal"), criterion = "AICc")
```

Arguments

X	obersevations.
w	weights of sample.
candDist	a vector of names of candidate distributions.
criterion	the criterion based on which the best fitted distribution is choosen.

Details

Details

Value

the name of best fitted distribution and the parameter estimates.

Examples

```
X <- rBeta_ab(30, a = 0, b = 1, shape1 = 2, shape2 = 10 )
bestDist(X, candDist = c("Beta_ab","Laplace","Normal"), criterion = "logLik")
bestDist(X, candDist = c("Beta_ab","Laplace","Normal"), criterion = "AIC")
bestDist(X, candDist = c("Beta_ab","Laplace","Normal"), criterion = "AICc")
bestDist(X, candDist = c("Beta_ab","Laplace","Normal"), criterion = "BIC")
bestDist(X, candDist = c("Beta_ab","Laplace","Normal"), criterion = "MDL")

w <- c(0.32, 1.77, 1.22, 0.64, 0.38, 0.93, 1.63, 1.34, 0.57, 1.73, 1.67, 0.67,
0.09, 1, 1.55, 0.53, 0.76, 1.06, 1.13, 1.31, 1.18, 1.64, 0.07, 1.41, 1.18,
0.69, 0.28, 1.27, 0.9, 1.08)
bestDist(X, w, candDist = c("Beta_ab","Laplace","Normal"), criterion = "logLik")
bestDist(X, w, candDist = c("Beta_ab","Laplace","Normal"), criterion = "AIC")
bestDist(X, w, candDist = c("Beta_ab","Laplace","Normal"), criterion = "AICc")
bestDist(X, w, candDist = c("Beta_ab","Laplace","Normal"), criterion = "BIC")
bestDist(X, w, candDist = c("Beta_ab","Laplace","Normal"), criterion = "MDL")

# parameter for best distribution
best_dist <- bestDist(X, candDist = c("Beta_ab","Laplace","Normal"), criterion = "logLik")
attributes(best_dist)$best.dist.par
```

Beta *The standard Beta Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Beta distribution

Usage

```
dBeta(x, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

pBeta(q, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

qBeta(p, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

rBeta(n, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

eBeta(X, w, method = "numerical.MLE")

lBeta(X, w, shape1 = 2, shape2 = 3, params = list(shape1, shape2),
  logL = TRUE)

sBeta(X, w, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

iBeta(X, w, shape1 = 2, shape2 = 3, params = list(shape1, shape2))
```

Arguments

x,q	vector of quantiles.
shape1,shape2	shape parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lBeta gives log likelihood.
...	other parameters

Details

standard Beta Distribution

See Distributions-Beta

Value

dBeta gives the density; pBeta gives the distribution function; qBeta gives the quantile function; rBeta generates random variables; eBeta estimate the parameters; sBeta gives observed scorn function

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
shape1 <- 1
shape2 <- 2
X <- rBeta(n, shape1, shape2)
(est.par <- eBeta(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dBeta(den.x,shape1=est.par$shape1,shape2=est.par$shape2)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qBeta((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot", xlab="Theoretical Quantiles",
ylab="Sample Quantiles", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

plot((1:n-0.5)/n, pBeta(sort(X), params=est.par), main="P-P Plot", xlab="Theoretical Percentile",
ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(shape1=1, shape2=2)
X <- rBeta(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eBeta(X,w) # estimated parameters of weighted sample
eBeta(X) # estimated parameters of unweighted sample

# Alternative parameter estimation methods
(est.par <- eBeta(X, method = "numerical.MLE"))

# Extracting shape parameters
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rBeta,edist=eBeta,n = 1000, rep.num = 1e3,
params = list(shape1=2, shape2=5), method ="numerical.MLE")
eval.estimation(rdist=rBeta,edist=eBeta,n = 1000, rep.num = 1e3,
params = list(shape1=2, shape2=5), method ="MOM")

# evaluate the precision of estimation by Hessian matrix
X <- rBeta(1000, shape1, shape2)
(est.par <- eBeta(X))
H <- attributes(eBeta(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
```

```
lBeta(X,param = est.par)
lBeta(X,param = est.par, logL=FALSE)
sBeta(X,param = est.par)
iBeta(X,param = est.par)
```

Beta_ab *The four-Parameter beta Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the 4 Parameter beta distribution

Usage

```
dBeta_ab(x, shape1 = 2, shape2 = 3, a = 0, b = 1,
  params = list(shape1, shape2, a, b))

pBeta_ab(q, shape1 = 2, shape2 = 3, a = 0, b = 1, params = list(shape1
  = 2, shape2 = 5, a = 0, b = 1))

qBeta_ab(p, shape1 = 2, shape2 = 3, a = 0, b = 1, params = list(shape1
  = 2, shape2 = 5, a = 0, b = 1))

rBeta_ab(n, shape1 = 2, shape2 = 3, a = 0, b = 1,
  params = list(shape1, shape2, a, b))

eBeta_ab(X, w, method = "numerical.MLE")

lBeta_ab(X, w, shape1 = 2, shape2 = 3, a = 0, b = 1,
  params = list(shape1, shape2, a, b), logL = TRUE)

sBeta_ab(X, w, shape1 = 2, shape2 = 3, a = 0, b = 1,
  params = list(shape1, shape2, a, b))
```

Arguments

x,q	vector of quantiles.
shape1,shape2	shape parameters.
a,b	boundary parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.

w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lBeta_ab gives log likelihood.
...	other parameters

Details

four-Parameter beta Distribution

See Distributions-Four-Parameter-Beta

Value

dBeta_ab gives the density; pBeta_ab gives the distribution function; qBeta_ab gives the quantile function; rBeta_ab generates random variables; eBeta_ab estimate the parameters; sBeta_ab gives observed scorn function

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
a <- 1
b <- 2
shape1 <- 2
shape2 <- 5
X <- rBeta_ab(n, shape1, shape2, a, b)
(est.par <- eBeta_ab(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dBeta_ab(den.x,params = est.par)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qBeta_ab((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(a,b), ylim = c(a,b))
abline(0,1)

plot((1:n-0.5)/n, pBeta_ab(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(shape1=2, shape2=5, a= 1, b=2)
X <- rBeta_ab(n, params=par)
```

```
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eBeta_ab(X,w) # estimated parameters of weighted sample
eBeta_ab(X) # estimated parameters of unweighted sample

# Extracting boundary and shape parameters
est.par[attributes(est.par)$par.type=="boundary"]
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rBeta_ab,edist=eBeta_ab,n = 1000, rep.num = 1e3,
params = list(shape1=2, shape2=5, a=0, b=1), method ="numerical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rBeta_ab(1000, shape1, shape2, a, b)
(est.par <- eBeta_ab(X))
H <- attributes(eBeta_ab(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lBeta_ab(X,param = est.par)
lBeta_ab(X,param = est.par, logL=FALSE)
sBeta_ab(X,param = est.par)
```

Burr *The Burr's Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Burr distribution

Usage

```
dBurr(x, b = 1, g = 2, s = 2, params = list(b = 1, g = 2, s = 2))

pBurr(q, b = 1, g = 2, s = 2, params = list(b = 1, g = 2, s = 2))

qBurr(p, b = 1, g = 2, s = 2, params = list(b = 1, g = 2, s = 2))

rBurr(n, b = 1, g = 2, s = 2, params = list(b = 1, g = 2, s = 2))

eBurr(X, w, method = "numerical.MLE")

lBurr(X, w, b = 1, g = 2, s = 2, params = list(b = 1, g = 2, s = 2),
  logL = TRUE)
```

Arguments

x,q	vector of quantiles.
b	scale parameters.
g,s	shape parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lBurr gives log likelihood.
...	other parameters

Details

Burr's Distribution

See Distributions-Burr

Value

dBurr gives the density; pBurr gives the distribution function; qBurr gives the quantile function; rBurr generates random variables; eBurr estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
b = 1; g = 2; s = 2
X <- rBurr(n, b = 1, g = 2, s = 2)
(est.par <- eBurr(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dBurr(den.x, b=est.par$b, g=est.par$g, s=est.par$s)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qBurr((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(0,5), ylim = c(0,5))
abline(0,1)
```

```
plot((1:n-0.5)/n, pBurr(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(b=1, g=2, s =2)
X <- rBurr(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eBurr(X,w) # estimated parameters of weighted sample
eBurr(X) # estimated parameters of unweighted sample

# Extracting shape or scale parameters
est.par[attributes(est.par)$par.type=="scale"]
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rBurr,edist=eBurr,n = 1000, rep.num = 1e3, params = list(b=1, g=2, s =2))

# evaluate the precision of estimation by Hessian matrix
X <- rBurr(1000, b = 1, g = 2, s = 2)
(est.par <- eBurr(X))
H <- attributes(eBurr(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lBurr(X,param = est.par)
lBurr(X,param = est.par, logL=FALSE)
```

`compareDist` *Compare sample and fitted distributions*

Description

A method to Compare sample and fitted distributions

Usage

```
compareDist(X, Dist1, Dist2 = NULL, Dist3 = NULL)
```

Arguments

`X`	- the (unweighted) sample
`Dist1,Dist2,Dist3`	- the eDist objects or names of the distribution to be fitted.

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
n <- 100
par <- list(shape1=1, shape2=2)
X <- rBeta(n, params=par)
compareDist(X, "Beta", "Normal")
compareDist(X, "Beta", eNormal(X))
```

DistSelCriteriaValues *Distribution Selection Criteria Values.*

Description

A function to calculate the distribution selection criteria values for a list of candidate distributions

Usage

```
DistSelCriteriaValues(X, w = rep(1, length(X))/length(X),
  candDist = c("Beta_ab", "Laplace", "Normal"), criteria = c("logLik",
  "AIC", "AICc", "BIC", "MDL"))
```

Arguments

X	obersevations.
w	weights of sample.
candDist	a vector of names of candidate distributions.
criteria	a vector of criteria to be calculated

Details

Details

Value

a table contains mle parameter estimates and corresponding distribution selection criteria values.

Examples

```
X <- rBeta_ab(30, a = 0, b = 1, shape1 = 2, shape2 = 10 )
DistSelCriteriaValues(X, candDist = c("Beta_ab","Laplace","Normal"),
                         criteria = c("logLik","AIC","AICc","BIC","MDL"))

w <- c(0.32, 1.77, 1.22, 0.64, 0.38, 0.93, 1.63, 1.34, 0.57, 1.73, 1.67, 0.67,
0.09, 1, 1.55, 0.53, 0.76, 1.06, 1.13, 1.31, 1.18, 1.64, 0.07, 1.41, 1.18,
0.69, 0.28, 1.27, 0.9, 1.08)
DistSelCriteriaValues(X, w, candDist = c("Beta_ab","Laplace","Normal"),
                            criteria = c("logLik","AIC","AICc","BIC","MDL"))
```

---

eDist *S3 methods from manuplating eDist objects.*

Description

S3 methods from manuplating eDist objects

Usage

```
## S3 method for class 'eDist'
logLik(object, ...)

## S3 method for class 'eDist'
AIC(object, ..., k = 2)

AICc(object)

## S3 method for class 'eDist'
AICc(object, ...)

## S3 method for class 'eDist'
BIC(object, ...)

## S3 method for class 'eDist'
vcov(object, ..., corr = FALSE)

MDL(object)

## S3 method for class 'eDist'
MDL(object, ...)

## S3 method for class 'eDist'
print(x, ...)
```

Arguments

object,x	a eDist object, which is the output of the parameter estimation functions.
...	other parameters
k	numeric, the penalty per parameter to be used; the default k = 2 is the classical AIC.
corr	logic, if the vcov return correlation matrix (instead of variance-covariance matrix) or not.

Details

S3 methods from manuplating eDist objects.

Author(s)

A. Jonathan R. Godfrey and Haizhen Wu

References

Myung, I. (2000). The Importance of Complexity in Model Selection. Journal of mathematical psychology, 44(1), 190-204.

Examples

```
X <- rnorm(20)
est.par <- eNormal(X, method ="numerical.MLE")
logLik(est.par)
AIC(est.par)
AICc(est.par)
BIC(est.par)
MDL(est.par)
vcov(est.par)
vcov(est.par,corr=TRUE)
print(est.par)
```

---

eval.estimation *Parameter Estimation Evaluation.*

Description

A function to evaluate the parameter estimation function.

Usage

```
eval.estimation(rdist, edist, n = 20, rep.num = 1000, params,
  method = "numerical.MLE")
```

Arguments

rdist	random variable generating function.
edist	parameter estimation function.
n	sample size
rep.num	number of replacates
params	true parameters of the distribution.
method	esimation method

Details

Parameter Estimation Evaluation

Details

Value

a list containing the mean, sd of the estimated parameters. na.cont is the number of "na"s appeared in the parameter estimation.

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
eval.estimation(rdist=rNormal,edist=eNormal,n = 10, rep.num = 1e3,
params = list(mean = 1,sd = 5))
eval.estimation(rdist=rNormal,edist=eNormal,n = 1000, rep.num = 1e3,
params = list(mean = 1,sd = 5))
```

Gamma *The Gamma Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Gamma distribution

Usage

```
dGamma(x, shape = 2, scale = 2, params = list(shape = 2, scale = 2))

pGamma(q, shape = 2, scale = 2, params = list(shape = 2, scale = 2))

qGamma(p, shape = 2, scale = 2, params = list(shape = 2, scale = 2))

rGamma(n, shape = 2, scale = 2, params = list(shape = 2, scale = 2))

eGamma(X, w, method = "numerical.MLE")

lGamma(X, w, shape = 2, scale = 2, params = list(shape = 2, scale = 2),
  logL = TRUE)
```

Arguments

x,q	vector of quantiles.
shape	shape parameter.
scale	scale parameter.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lGamma gives log likelihood.
...	other parameters

Details

Gamma Distribution

See Distributions-Gamma

Value

dGamma gives the density; pGamma gives the distribution function; qGamma gives the quantile function; rGamma generates random variables; eGamma estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
shape <- 1.5
scale <- 0.5
X <- rGamma(n, shape, scale)
(est.par <- eGamma(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dGamma(den.x,shape=est.par$shape,scale=est.par$scale)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qGamma((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(0,5), ylim = c(0,5))
abline(0,1)

plot((1:n-0.5)/n, pGamma(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(shape=1, scale=2)
X <- rGamma(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eGamma(X,w) # estimated parameters of weighted sample
eGamma(X) # estimated parameters of unweighted sample

# Extracting shape or scale parameters
est.par[attributes(est.par)$par.type=="shape"]
est.par[attributes(est.par)$par.type=="scale"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rGamma,edist=eGamma,n = 1000, rep.num = 1e3, params = list(shape=1, scale=2))

# evaluate the precision of estimation by Hessian matrix
X <- rGamma(1000, shape, scale)
(est.par <- eGamma(X))
H <- attributes(eGamma(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lGamma(X,param = est.par)
lGamma(X,param = est.par, logL=FALSE)
```

Gumbel *The Gumbel Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Gumbel distribution

Usage

```
dGumbel(x, location = 0, scale = 1, params = list(location = 0, scale =
  1))

pGumbel(q, location = 0, scale = 1, params = list(location = 0, scale =
  1))

qGumbel(p, location = 0, scale = 1, params = list(location = 0, scale =
  1))

rGumbel(n, location = 0, scale = 1, params = list(location = 0, scale =
  1))

eGumbel(X, w, method = "numerical.MLE")

lGumbel(X, w, location = 0, scale = 1, params = list(location = 0, scale =
  1), logL = TRUE)
```

Arguments

x,q	vector of quantiles.
location	location parameter.
scale	scale parameter.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lGumbel gives log likelihood.
...	other parameters

Details

Gumbel Distribution

See Distributions-Gumbel

Value

dGumbel gives the density; pGumbel gives the distribution function; qGumbel gives the quantile function; rGumbel generates random variables; eGumbel estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
location <- 1.5
scale <- 0.5
X <- rGumbel(n, location, scale)
(est.par <- eGumbel(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dGumbel(den.x,location=est.par$location,scale=est.par$scale)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qGumbel((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(0,5), ylim = c(0,5))
abline(0,1)

plot((1:n-0.5)/n, pGumbel(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(location=1, scale=2)
X <- rGumbel(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eGumbel(X,w) # estimated parameters of weighted sample
eGumbel(X) # estimated parameters of unweighted sample

# Extracting location or scale parameters
est.par[attributes(est.par)$par.type=="location"]
est.par[attributes(est.par)$par.type=="scale"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rGumbel,edist=eGumbel,n = 1000, rep.num = 1e3,
   params = list(location=1, scale=2))

# evaluate the precision of estimation by Hessian matrix
X <- rGumbel(1000, location, scale)
(est.par <- eGumbel(X))
```

```
H <- attributes(eGumbel(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lGumbel(X,param = est.par)
lGumbel(X,param = est.par, logL=FALSE)
```

JohnsonSB *The Johnson SB Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the 4 Parameter beta distribution

Usage

```
dJohnsonSB(x, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2))

pJohnsonSB(q, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2))

qJohnsonSB(p, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2))

rJohnsonSB(n, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2))

eJohnsonSB(X, w, method = "numerical.MLE")

lJohnsonSB(X, w, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2),
  logL = TRUE)
```

Arguments

x,q	vector of quantiles.
gamma,delta	shape parameters.
xi,lambda	location-scale parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.

w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lJohnsonSB gives log likelihood.
...	other parameters

Details

Johnson SB Distribution

Johnson SB is bounded in [xi,xi+lambda]. See Distributions-Johnson-SB

Value

dJohnsonSB gives the density; pJohnsonSB gives the distribution function; qJohnsonSB gives the quantile function; rJohnsonSB generates random variables; eJohnsonSB estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
gamma = -0.5; delta = 2; xi = -0.5; lambda = 2
X <- rJohnsonSB(n, gamma, delta, xi, lambda)
(est.par <- eJohnsonSB(X))
# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dJohnsonSB(den.x,params = est.par)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qJohnsonSB((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(a,b), ylim = c(a,b))
abline(0,1)

plot((1:n-0.5)/n, pJohnsonSB(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2)
X <- rJohnsonSB(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eJohnsonSB(X,w) # estimated parameters of weighted sample
eJohnsonSB(X) # estimated parameters of unweighted sample

# Extracting location, scale and shape parameters
```

```
est.par[attributes(est.par)$par.type=="location"]
est.par[attributes(est.par)$par.type=="scale"]
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rJohnsonSB,edist=eJohnsonSB,n = 1000, rep.num = 1e3,
params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2), method ="numerical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rJohnsonSB(1000, gamma, delta, xi, lambda)
(est.par <- eJohnsonSB(X))
H <- attributes(eJohnsonSB(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lJohnsonSB(X,param = est.par)
lJohnsonSB(X,param = est.par, logL=FALSE)
```

JohnsonSU *The Johnson SU Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the 4 Parameter beta distribution

Usage

```
dJohnsonSU(x, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2))

pJohnsonSU(q, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2))

qJohnsonSU(p, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2))

rJohnsonSU(n, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2))

eJohnsonSU(X, w, method = "numerical.MLE")

lJohnsonSU(X, w, gamma = -0.5, delta = 2, xi = -0.5, lambda = 2,
  params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2),
  logL = TRUE)
```

Arguments

x,q	vector of quantiles.
gamma,delta	shape parameters.
xi,lambda	location-scale parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lJohnsonSU gives log likelihood.
...	other parameters

Details

Johnson SU Distribution

See Distributions-Johnson-SU

Value

dJohnsonSU gives the density; pJohnsonSU gives the distribution function; qJohnsonSU gives the quantile function; rJohnsonSU generates random variables; eJohnsonSU estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
gamma = -0.5; delta = 2; xi = -0.5; lambda = 2
X <- rJohnsonSU(n, gamma, delta, xi, lambda)
(est.par <- eJohnsonSU(X))
# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dJohnsonSU(den.x,params = est.par)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qJohnsonSU((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(-5,5), ylim = c(-5,5))
abline(0,1)

plot((1:n-0.5)/n, pJohnsonSU(sort(X), params=est.par), main="P-P Plot",
```

```
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2)
X <- rJohnsonSU(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eJohnsonSU(X,w) # estimated parameters of weighted sample
eJohnsonSU(X) # estimated parameters of unweighted sample

# Extracting shape parameters
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rJohnsonSU,edist=eJohnsonSU,n = 1000, rep.num = 1e3,
params = list(gamma = -0.5, delta = 2, xi = -0.5, lambda = 2), method ="numerical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rJohnsonSU(1000, gamma, delta, xi, lambda)
(est.par <- eJohnsonSU(X))
H <- attributes(eJohnsonSU(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lJohnsonSU(X,param = est.par)
lJohnsonSU(X,param = est.par, logL=FALSE)
```

Laplace *The Laplace Distribution.*

Description

Density (d), distribution (p), and quantile (q), random number generation (r), and parameter estimation (e) functions for the Laplace distribution. Parameter estimation can only be based on an unweighted i.i.d. sample.

Usage

```
dLaplace(x, mu = 0, b = 1, params = list(mu, b))

pLaplace(q, mu = 0, b = 1, params = list(mu, b))

qLaplace(p, mu = 0, b = 1, params = list(mu, b))

rLaplace(n, mu = 0, b = 1, params = list(mu, b))
```

```
eLaplace(X, w, method = "numerical.MLE")

lLaplace(x, w = 1, mu = 0, b = 1, params = list(mu, b), logL = TRUE)
```

Arguments

x,q	vector of quantiles.
mu	location parameter.
b	scale parameter.
params	a list that includes all named parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical, it is assumed that the log likelihood is desired. Set to FALSE if the likelihood is wanted.
...	other parameters

Details

Laplace Distribution

no details yet.

Value

dLaplace gives the density; pLaplace gives the distribution function, qLaplace gives the quantile function, rLaplace generates random variables and eLaplace estimates the parameters. lLaplace will provide the log-likelihood.

Note

The estimation of the population mean is done using the median of the sample. Unweighted samples are not yet catered for in the eLaplace() function.

Author(s)

A. Jonathan R. Godfrey and Haizhen Wu

Examples

```
# Parameter estimation
n <- 500
mu <- 1
b <- 2
X <- rLaplace(n, mu, b)
```

```
(est.par <- eLaplace(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dLaplace(den.x,mu=est.par$mu,b=est.par$b)
hist(X, breaks=10, col="red", probability=TRUE, ylim=c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qLaplace((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim=c(-5,5), ylim=c(-5,5))
abline(0,1)

plot((1:n-0.5)/n, pLaplace(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim=c(0,1), ylim=c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(mu=1, b=2)
X <- rLaplace(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eLaplace(X,w) # estimated parameters of weighted sample
eLaplace(X) # estimated parameters of unweighted sample

# Alternative parameter estimation methods
eLaplace(X, method="numerical.MLE")

# Extracting location or scale parameters
est.par[attributes(est.par)$par.type=="location"]
est.par[attributes(est.par)$par.type=="scale"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rLaplace,edist=eLaplace,n=1000, rep.num=1e3, params=list(mu=1, b=2))
eval.estimation(rdist=rLaplace,edist=eLaplace,n=1000, rep.num=1e3, params=list(mu=1, b=2),
method ="analytical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rLaplace(1000, mu, b)
(est.par <- eLaplace(X))
H <- attributes(eLaplace(X, method="numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lLaplace(X,param=est.par)
lLaplace(X,param=est.par, logL=FALSE)
```

`Logistic` *The Logistic Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Logistic distribution

Usage

```
dLogistic(x, location = 0, scale = 1, params = list(location = 0, scale =
  1))

pLogistic(q, location = 0, scale = 1, params = list(location = 0, scale =
  1))

qLogistic(p, location = 0, scale = 1, params = list(location = 0, scale =
  1))

rLogistic(n, location = 0, scale = 1, params = list(location = 0, scale =
  1))

eLogistic(X, w, method = "numerical.MLE")

lLogistic(X, w, location = 0, scale = 1, params = list(location = 0, scale
  = 1), logL = TRUE)
```

Arguments

`x,q`	vector of quantiles.
`location`	location parameter.
`scale`	scale parameter.
`params`	a list includes all parameters
`p`	vector of probabilities.
`n`	number of observations.
`X`	sample observations.
`w`	weights of sample.
`method`	parameter estimation method.
`logL`	logical; if TRUE, lLogistic gives log likelihood.
`...`	other parameters

Details

Logistic Distribution

See Distributions-Logistic

Value

dLogistic gives the density; pLogistic gives the distribution function; qLogistic gives the quantile function; rLogistic generates random variables; eLogistic estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
location <- 1.5
scale <- 0.5
X <- rLogistic(n, location, scale)
(est.par <- eLogistic(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dLogistic(den.x,location=est.par$location,scale=est.par$scale)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qLogistic((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(-5,5), ylim = c(-5,5))
abline(0,1)

plot((1:n-0.5)/n, pLogistic(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(location=1, scale=2)
X <- rLogistic(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eLogistic(X,w) # estimated parameters of weighted sample
eLogistic(X) # estimated parameters of unweighted sample

# Extracting location or scale parameters
est.par[attributes(est.par)$par.type=="location"]
est.par[attributes(est.par)$par.type=="scale"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rLogistic,edist=eLogistic,n = 1000, rep.num = 1e3,
   params = list(location=1, scale=2))

# evaluate the precision of estimation by Hessian matrix
X <- rLogistic(1000, location, scale)
(est.par <- eLogistic(X))
H <- attributes(eLogistic(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
```

```
lLogistic(X,param = est.par)
lLogistic(X,param = est.par, logL=FALSE)
```

Normal *The Normal Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Normal distribution

Usage

```
dNormal(x, mean = 0, sd = 1, params = list(mean, sd))

pNormal(q, mean = 0, sd = 1, params = list(mean, sd))

qNormal(p, mean = 0, sd = 1, params = list(mean, sd))

rNormal(n, mean = 0, sd = 1, params = list(mean, sd))

eNormal(X, w, method = "numerical.MLE")

lNormal(X, w, mean = 0, sd = 1, params = list(mean, sd), logL = TRUE)

sNormal(X, w, mean = 0, sd = 1, params = list(mean, sd))

iNormal(X, w, mean = 0, sd = 1, params = list(mean, sd))
```

Arguments

x,q	vector of quantiles.
mean	location parameter.
sd	scale parameter.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lNormal gives log likelihood.
...	other parameters

Details

Normal Distribution

See Distributions-Normal

Value

dNormal gives the density; pNormal gives the distribution function; qNormal gives the quantile function; rNormal generates random variables; eNormal estimate the parameters; sNormal gives observed scorn function

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
mean <- 1
sd <- 2
X <- rNormal(n, mean, sd)
(est.par <- eNormal(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dNormal(den.x,mean=est.par$mean,sd=est.par$sd)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qNormal((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(-5,5), ylim = c(-5,5))
abline(0,1)

plot((1:n-0.5)/n, pNormal(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(mean=1, sd=2)
X <- rNormal(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eNormal(X,w) # estimated parameters of weighted sample
eNormal(X) # estimated parameters of unweighted sample

# Alternative parameter estimation methods
eNormal(X, method = "numerical.MLE")
eNormal(X, method = "bias.adjusted.MLE")
mean(X); sd(X); sd(X)*sqrt((n-1)/n)
```

```
# Extracting location or scale parameters
est.par[attributes(est.par)$par.type=="location"]
est.par[attributes(est.par)$par.type=="scale"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rNormal,edist=eNormal,n = 1000, rep.num = 1e3, params = list(mean=1, sd=2))
eval.estimation(rdist=rNormal,edist=eNormal,n = 1000, rep.num = 1e3, params = list(mean=1, sd=2),
method ="analytical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rNormal(1000, mean, sd)
(est.par <- eNormal(X))
H <- attributes(eNormal(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lNormal(X,param = est.par)
lNormal(X,param = est.par, logL=FALSE)
sNormal(X,param = est.par)
iNormal(X,param = est.par)
```

Normal_sym_trunc_ab *The symmetric truncated normal distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the symmetric truncated normal distribution

Usage

```
dNormal_sym_trunc_ab(x, sigma = 0.3, a = 0, b = 1, params = list(sigma,
  a, b))

pNormal_sym_trunc_ab(q, sigma = 0.3, a = 0, b = 1, params = list(mu = 2,
  sigma = 5, a = 0, b = 1))

qNormal_sym_trunc_ab(p, sigma = 0.3, a = 0, b = 1, params = list(mu = 2,
  sigma = 5, a = 0, b = 1))

rNormal_sym_trunc_ab(n, mu = 2, sigma = 3, a = 0, b = 1,
  params = list(sigma, a, b))

eNormal_sym_trunc_ab(X, w, method = "numerical.MLE")
```

```
lNormal_sym_trunc_ab(X, w, mu = 2, sigma = 3, a = 0, b = 1,
  params = list(sigma, a, b), logL = TRUE)
```

Arguments

x,q	vector of quantiles.
a,b	boundary parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
mu,sigma	shape parameters.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lNormal_sym_trunc_ab gives log likelihood.
...	other parameters

Details

symmetric truncated normal Distribution

See Distributions-Truncted-Normal

Value

dNormal_sym_trunc_ab gives the density; pNormal_sym_trunc_ab gives the distribution function; qNormal_sym_trunc_ab gives the quantile function; rNormal_sym_trunc_ab generates random variables; eNormal_sym_trunc_ab estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
a <- 0; b <- 1; sigma <- 0.3
X <- rNormal_sym_trunc_ab(n, sigma, a, b)
(est.par <- eNormal_sym_trunc_ab(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dNormal_sym_trunc_ab(den.x,params = est.par)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)
```

```
# Q-Q plot and P-P plot
plot(qNormal_sym_trunc_ab((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(a,b), ylim = c(a,b))
abline(0,1)

plot((1:n-0.5)/n, pNormal_sym_trunc_ab(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(sigma=0.3, a= 0, b=1)
X <- rNormal_sym_trunc_ab(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eNormal_sym_trunc_ab(X,w) # estimated parameters of weighted sample
eNormal_sym_trunc_ab(X) # estimated parameters of unweighted sample

# Extracting boundary and shape parameters
est.par[attributes(est.par)$par.type=="boundary"]
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rNormal_sym_trunc_ab,edist=eNormal_sym_trunc_ab,n = 1000, rep.num = 1e3,
params = list(mu=2, sigma=5, a=0, b=1), method ="numerical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rNormal_sym_trunc_ab(1000, sigma, a, b)
(est.par <- eNormal_sym_trunc_ab(X))
H <- attributes(eNormal_sym_trunc_ab(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lNormal_sym_trunc_ab(X,param = est.par)
lNormal_sym_trunc_ab(X,param = est.par, logL=FALSE)
```

Normal_trunc_ab *The truncated normal distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the truncated normal distribution

Usage

```
dNormal_trunc_ab(x, mu = 2, sigma = 3, a = 0, b = 1, params = list(mu,
  sigma, a, b))
```

```
pNormal_trunc_ab(q, mu = 2, sigma = 3, a = 0, b = 1, params = list(mu
  = 2, sigma = 5, a = 0, b = 1))

qNormal_trunc_ab(p, mu = 2, sigma = 3, a = 0, b = 1, params = list(mu
  = 2, sigma = 5, a = 0, b = 1))

rNormal_trunc_ab(n, mu = 2, sigma = 3, a = 0, b = 1, params = list(mu,
  sigma, a, b))

eNormal_trunc_ab(X, w, method = "numerical.MLE")

lNormal_trunc_ab(X, w, mu = 2, sigma = 3, a = 0, b = 1,
  params = list(mu, sigma, a, b), logL = TRUE)
```

Arguments

x,q	vector of quantiles.
mu,sigma	shape parameters.
a,b	boundary parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lNormal_trunc_ab gives log likelihood.
...	other parameters

Details

truncated normal Distribution

See Distributions-Truncted-Normal

Value

dNormal_trunc_ab gives the density; pNormal_trunc_ab gives the distribution function; qNormal_trunc_ab gives the quantile function; rNormal_trunc_ab generates random variables; eNormal_trunc_ab estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
a <- 1
b <- 2
mu <- 2
sigma <- 5
X <- rNormal_trunc_ab(n, mu, sigma, a, b)
(est.par <- eNormal_trunc_ab(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dNormal_trunc_ab(den.x,params = est.par)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qNormal_trunc_ab((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(a,b), ylim = c(a,b))
abline(0,1)

plot((1:n-0.5)/n, pNormal_trunc_ab(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(mu=2, sigma=5, a= 1, b=2)
X <- rNormal_trunc_ab(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eNormal_trunc_ab(X,w) # estimated parameters of weighted sample
eNormal_trunc_ab(X) # estimated parameters of unweighted sample

# Extracting boundary and shape parameters
est.par[attributes(est.par)$par.type=="boundary"]
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rNormal_trunc_ab,edist=eNormal_trunc_ab,n = 1000, rep.num = 1e3,
params = list(mu=2, sigma=5, a=0, b=1), method ="numerical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rNormal_trunc_ab(1000, mu, sigma, a, b)
(est.par <- eNormal_trunc_ab(X))
H <- attributes(eNormal_trunc_ab(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lNormal_trunc_ab(X,param = est.par)
lNormal_trunc_ab(X,param = est.par, logL=FALSE)
```

SRTB_ab *The Symmetric-Reflected Truncated Beta (SRTB) Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Symmetric-Reflected Truncated Beta (SRTB).

Usage

```
dSRTB_ab(x, shape1 = 2, shape2 = 3, a = 0, b = 1,
  params = list(shape1, shape2, a, b))

pSRTB_ab(q, shape1 = 2, shape2 = 3, a = 0, b = 1, params = list(shape1
  = 2, shape2 = 5, a = 0, b = 1))

qSRTB_ab(p, shape1 = 2, shape2 = 3, a = 0, b = 1, params = list(shape1
  = 2, shape2 = 5, a = 0, b = 1))

rSRTB_ab(n, shape1 = 2, shape2 = 3, a = 0, b = 1,
  params = list(shape1, shape2, a, b))

eSRTB_ab(X, w, method = "numerical.MLE")

lSRTB_ab(X, w, shape1 = 2, shape2 = 3, a = 0, b = 1,
  params = list(shape1, shape2, a, b), logL = TRUE)
```

Arguments

x,q	vector of quantiles.
shape1,shape2	shape parameters.
a,b	boundary parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lSRTB_ab gives log likelihood.
...	other parameters

Details

See Distributions-Four-Parameter-Beta

Value

dSRTB_ab gives the density; pSRTB_ab gives the distribution function; qSRTB_ab gives the quantile function; rSRTB_ab generates random variables; eSRTB_ab estimate the parameters; sSRTB_ab gives observed scorn function

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
a <- 1
b <- 2
shape1 <- 2
shape2 <- 10
X <- rSRTB_ab(n, shape1, shape2, a, b)
(est.par <- eSRTB_ab(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dSRTB_ab(den.x,params = est.par)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qSRTB_ab((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(a,b), ylim = c(a,b))
abline(0,1)

plot((1:n-0.5)/n, pSRTB_ab(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(shape1=2, shape2=10, a= 1, b=2)
X <- rSRTB_ab(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eSRTB_ab(X,w) # estimated parameters of weighted sample
eSRTB_ab(X) # estimated parameters of unweighted sample

# Extracting boundary and shape parameters
est.par[attributes(est.par)$par.type=="boundary"]
est.par[attributes(est.par)$par.type=="shape"]
```

```
# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rSRTB_ab,edist=eSRTB_ab,n = 1000, rep.num = 1e3,
params = list(shape1=2, shape2=10, a=0, b=1), method ="numerical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rSRTB_ab(1000, shape1, shape2, a, b)
(est.par <- eSRTB_ab(X))
H <- attributes(eSRTB_ab(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lSRTB_ab(X,param = est.par)
lSRTB_ab(X,param = est.par, logL=FALSE)
```

SSRTB *The standard Symmetric-Reflected Truncated Beta (SRTB) Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the SSRTB distribution

Usage

```
dSSRTB(x, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

pSSRTB(q, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

qSSRTB(p, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

rSSRTB(n, shape1 = 2, shape2 = 3, params = list(shape1, shape2))

eSSRTB(X, w, method = "numerical.MLE")

lSSRTB(X, w, shape1 = 2, shape2 = 3, params = list(shape1, shape2),
  logL = TRUE)
```

Arguments

x,q	vector of quantiles.
shape1,shape2	shape parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.

X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lSSRTB gives log likelihood.
...	other parameters

Details

Standard Symmetric-Reflected Truncated Beta (SRTB) Distribution

See Distributions-SSRTB

Value

dSSRTB gives the density; pSSRTB gives the distribution function; qSSRTB gives the quantile function; rSSRTB generates random variables; eSSRTB estimate the parameters; sSSRTB gives observed scorn function

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
shape1 <- 2
shape2 <- 10
X <- rSSRTB(n, shape1, shape2)
(est.par <- eSSRTB(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dSSRTB(den.x,shape1=est.par$shape1,shape2=est.par$shape2)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qSSRTB((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot", xlab="Theoretical Quantiles",
ylab="Sample Quantiles", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

plot((1:n-0.5)/n, pSSRTB(sort(X), params=est.par), main="P-P Plot", xlab="Theoretical Percentile",
ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(shape1=1, shape2=2)
X <- rSSRTB(n, params=par)
```

```
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eSSRTB(X,w) # estimated parameters of weighted sample
eSSRTB(X) # estimated parameters of unweighted sample

# Alternative parameter estimation methods
(est.par <- eSSRTB(X, method = "numerical.MLE"))

# Extracting shape parameters
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rSSRTB,edist=eSSRTB,n = 1000, rep.num = 1e3,
params = list(shape1=2, shape2=5), method ="numerical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rSSRTB(1000, shape1, shape2)
(est.par <- eSSRTB(X))
H <- attributes(eSSRTB(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lSSRTB(X,param = est.par)
lSSRTB(X,param = est.par, logL=FALSE)
```

Triangular *The Triangular Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Triangular distribution

Usage

```
dTriangular(x, a = 0, b = 1, theta = 0.5, params = list(a, b, theta))

pTriangular(q, a = 0, b = 1, theta = 0.5, params = list(a, b, theta))

qTriangular(p, a = 0, b = 1, theta = 0.5, params = list(a, b, theta))

rTriangular(n, a = 0, b = 1, theta = 0.5, params = list(a, b, theta))

eTriangular(X, w, method = "numerical.MLE")

lTriangular(X, w, a = 0, b = 1, theta = 0.5, params = list(a, b, theta),
  logL = TRUE)
```

Arguments

x,q	vector of quantiles.
a,b	boundary parameters.
theta	shape parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lTriangular gives log likelihood.
...	other parameters

Details

Triangular Distribution

See Distributions-Four-Parameter-Beta

Value

dTriangular gives the density; pTriangular gives the distribution function; qTriangular gives the quantile function; rTriangular generates random variables; eTriangular estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
a <- 1
b <- 2
theta <- 0.5
X <- rTriangular(n, a, b, theta)
(est.par <- eTriangular(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dTriangular(den.x,params = est.par)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qTriangular((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(a,b), ylim = c(a,b))
```

```
abline(0,1)

plot((1:n-0.5)/n, pTriangular(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(a=0, b=1, theta=0.5)
X <- rTriangular(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eTriangular(X,w) # estimated parameters of weighted sample
eTriangular(X) # estimated parameters of unweighted sample

# Extracting boundary and shape parameters
est.par[attributes(est.par)$par.type=="boundary"]
est.par[attributes(est.par)$par.type=="shape"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rTriangular,edist=eTriangular,n = 1000, rep.num = 1e3,
params = list(a=0, b=1, theta=0.5), method ="numerical.MLE")

# evaluate the precision of estimation by Hessian matrix
X <- rTriangular(1000, a, b, theta)
(est.par <- eTriangular(X))
H <- attributes(eTriangular(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lTriangular(X,param = est.par)
lTriangular(X,param = est.par, logL=FALSE)
sTriangular(X,param = est.par)
```

`Uniform` *The Uniform Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Uniform distribution

Usage

```
dUniform(x, a = 2, b = 3, params = list(a, b))

pUniform(q, a = 2, b = 3, params = list(a, b))

qUniform(p, a = 2, b = 3, params = list(a, b))
```

```
rUniform(n, a = 2, b = 3, params = list(a, b))

eUniform(X, w, method = "numerical.MLE")

lUniform(X, w, a = 2, b = 3, params = list(a, b), logL = TRUE)
```

Arguments

x,q	vector of quantiles.
a,b	boundary parameters.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lUniform gives log likelihood.
...	other parameters

Details

Uniform Distribution ————————————————————

See Distributions-Uniform

Value

dUniform gives the density; pUniform gives the distribution function; qUniform gives the quantile function; rUniform generates random variables; eUniform estimate the parameters; sUniform gives observed scorn function

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
a <- 1
b <- 2
X <- rUniform(n, a, b)
(est.par <- eUniform(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
```

```
den.y <- dUniform(den.x,a=est.par$a,b=est.par$b)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qUniform((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot", xlab="Theoretical Quantiles",
ylab="Sample Quantiles", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

plot((1:n-0.5)/n, pUniform(sort(X), params=est.par), main="P-P Plot", xlab="Theoretical Percentile",
ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(a=1, b=2)
X <- rUniform(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eUniform(X,w) # estimated parameters of weighted sample
eUniform(X) # estimated parameters of unweighted sample

# Alternative parameter estimation methods
(est.par <- eUniform(X, method = "numerical.MLE"))

# Extracting boundary parameters
est.par[attributes(est.par)$par.type=="boundary"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rUniform,edist=eUniform,n = 1000, rep.num = 1e3,
params = list(a=2, b=5), method ="numerical.MLE")
eval.estimation(rdist=rUniform,edist=eUniform,n = 1000, rep.num = 1e3,
params = list(a=2, b=5), method ="MOM")

# evaluate the precision of estimation by Hessian matrix
X <- rUniform(1000, a, b)
(est.par <- eUniform(X))
H <- attributes(eUniform(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lUniform(X,param = est.par)
lUniform(X,param = est.par, logL=FALSE)
sUniform(X,param = est.par)
iUniform(X,param = est.par)
```

`Weibull` *The Weibull Distribution.*

Description

Density, distribution function, quantile function, random generation function and parameter estimation function (based on weighted or unweighted i.i.d. sample) for the Weibull distribution

Usage

```
dWeibull(x, shape = 2, scale = 2, params = list(shape = 2, scale = 2))

pWeibull(q, shape = 2, scale = 2, params = list(shape = 2, scale = 2))

qWeibull(p, shape = 2, scale = 2, params = list(shape = 2, scale = 2))

rWeibull(n, shape = 2, scale = 2, params = list(shape = 2, scale = 2))

eWeibull(X, w, method = "numerical.MLE")

lWeibull(X, w, shape = 2, scale = 2, params = list(shape = 2, scale = 2),
  logL = TRUE)
```

Arguments

x,q	vector of quantiles.
shape	shape parameter.
scale	scale parameter.
params	a list includes all parameters
p	vector of probabilities.
n	number of observations.
X	sample observations.
w	weights of sample.
method	parameter estimation method.
logL	logical; if TRUE, lWeibull gives log likelihood.
...	other parameters

Details

Weibull Distribution

See Distributions-Weibull

Value

dWeibull gives the density; pWeibull gives the distribution function; qWeibull gives the quantile function; rWeibull generates random variables; eWeibull estimate the parameters

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
# Parameter estimation
n <- 500
shape <- 1.5
scale <- 0.5
X <- rWeibull(n, shape, scale)
(est.par <- eWeibull(X))

# Histogram and fitted density
den.x <- seq(min(X),max(X),length=100)
den.y <- dWeibull(den.x,shape=est.par$shape,scale=est.par$scale)
hist(X, breaks=10, col="red", probability=TRUE, ylim = c(0,1.1*max(den.y)))
lines(den.x, den.y, col="blue", lwd=2)

# Q-Q plot and P-P plot
plot(qWeibull((1:n-0.5)/n, params=est.par), sort(X), main="Q-Q Plot",
xlab="Theoretical Quantiles", ylab="Sample Quantiles", xlim = c(0,5), ylim = c(0,5))
abline(0,1)

plot((1:n-0.5)/n, pWeibull(sort(X), params=est.par), main="P-P Plot",
xlab="Theoretical Percentile", ylab="Sample Percentile", xlim = c(0,1), ylim = c(0,1))
abline(0,1)

# A weighted parameter estimation example
n <- 10
par <- list(shape=1, scale=2)
X <- rWeibull(n, params=par)
w <- c(0.13, 0.06, 0.16, 0.07, 0.2, 0.01, 0.06, 0.09, 0.1, 0.12)
eWeibull(X,w) # estimated parameters of weighted sample
eWeibull(X) # estimated parameters of unweighted sample

# Extracting shape or scale parameters
est.par[attributes(est.par)$par.type=="shape"]
est.par[attributes(est.par)$par.type=="scale"]

# evaluate the performance of the parameter estimation function by simulation
eval.estimation(rdist=rWeibull,edist=eWeibull,n = 1000, rep.num = 1e3,
   params = list(shape=1, scale=2))

# evaluate the precision of estimation by Hessian matrix
X <- rWeibull(1000, shape, scale)
(est.par <- eWeibull(X))
H <- attributes(eWeibull(X, method = "numerical.MLE"))$nll.hessian
fisher_info <- solve(H)
sqrt(diag(fisher_info))

# log-likelihood, score vector and observed information matrix
lWeibull(X,param = est.par)
lWeibull(X,param = est.par, logL=FALSE)
```

wmle *Weighted Maximum Likelihood Estimation.*

Description

A general weighted maximum likelihood estimation function.

Usage

```
wmle(X, w, distname, initial, lower, upper, loglik.fn, score.fn, obs.info.fn)
```

Arguments

X	observation.
w	frequency (or weights) of observation.
distname	name of distribution to be estimated.
initial	initial value of the parameters.
lower	the lower bound of the parameters.
upper	the upper bound of the parameters.
loglik.fn	function to compute (weighted) log likelihood
score.fn	function to compute (weighted) score
obs.info.fn	function to compute observed information matrix

Details

Weighted Maximum Likelihood Estimation

Value

weighted mle estimates.

Author(s)

Haizhen Wu and A. Jonathan R. Godfrey

Examples

```
#if only density function available
n <- 200
X <- rnorm(n)
dFoo <- function(x, params = list(mean=0, sd=1)){
  mean <- params$mean
  sd <- params$sd
  return(dnorm(x, mean = mean, sd = sd))
```

```
}
wmle(X, w=rep(1,n), distname = "Foo",
     initial=list(mean = 0, sd = 1),
     lower=list(mean = -Inf, sd = 0),
     upper=list(mean = Inf, sd = Inf))

#if log likehood function available
lFoo2 <- function(X,w, params = list(mean=0, sd=1)){
  mean <- params$mean
  sd <- params$sd
  return(sum(w*log(dnorm(X, mean = mean, sd = sd))))
}
wmle(X, w=rep(1,n), loglik.fn = lFoo2,
     initial=list(mean = 0, sd = 1),
     lower=list(mean = -Inf, sd = 0),
     upper=list(mean = Inf, sd = Inf))

#if score function available
sFoo <- function(X,w, params = list(mean=0, sd=1)){
  mean <- params$mean
  sd <- params$sd
  score1 <- sum(w*(X-mean)/sd^2)
  score2 <- sum(w*((X-mean)^2-sd^2)/sd^3)
  score <- c(score1,score2)
  return(score)
}
lFoo <- lFoo2
wmle(X, w=rep(1,n), loglik.fn = lFoo, score.fn= sFoo,
     initial=list(mean = 0, sd = 1),
     lower=list(mean = -Inf, sd = 0),
     upper=list(mean = Inf, sd = Inf))

# if score function & observed information matrix available
iFoo <- function(X,w, params = list(mean=0, sd=1)){
  mean <- params$mean
  sd <- params$sd

  n <- length(X)
  if(missing(w)){
    w <- rep(1,n)
  } else {
    w <- n*w/sum(w)
  }

  info11 <- sum(w*rep(1/sd^2,n))
  info12 <- sum(w*2*(X-mean)/sd^3)
  info21 <- info12
  info22 <- sum(w*(3*(X-mean)^2-sd^2)/sd^4)
  info <- matrix(c(info11,info12,info21,info22), nrow=2,ncol=2)
  rownames(info) <- colnames(info) <- c("mean","sd")
  return(info)
}
wmle(X, w=rep(1,n), loglik.fn = lFoo, score.fn= sFoo, obs.info.fn=iFoo,
```

```
     initial=list(mean = 0, sd = 1),
     lower=list(mean = -Inf, sd = 0),
     upper=list(mean = Inf, sd = Inf))

# Speed comparison
N <- 1000
if(exists("sFoo2")) rm(sFoo2)
if(exists("iFoo2")) rm(iFoo2)
pt <- proc.time()
for(i in 1:N){
  wmle(X, w=rep(1,n), loglik.fn = lFoo2,
       initial=list(mean = 0, sd = 1),
       lower=list(mean = -Inf, sd = 0),
       upper=list(mean = Inf, sd = Inf))
}
proc.time() - pt

sFoo2 <- sFoo
if(exists("iFoo2")) rm(iFoo2)
pt <- proc.time()
for(i in 1:N){
  wmle(X, w=rep(1,n), loglik.fn = lFoo2, score.fn= sFoo2,
       initial=list(mean = 0, sd = 1),
       lower=list(mean = -Inf, sd = 0),
       upper=list(mean = Inf, sd = Inf))
}
proc.time() - pt

sFoo2 <- sFoo; iFoo2 <- iFoo
pt <- proc.time()
for(i in 1:N){
  wmle(X, w=rep(1,n), loglik.fn = lFoo2, score.fn= sFoo2, obs.info.fn=iFoo2,
       initial=list(mean = 0, sd = 1),
       lower=list(mean = -Inf, sd = 0),
       upper=list(mean = Inf, sd = Inf))
}
proc.time() - pt
```

Index