Package ‘acid’

February 1, 2016

Type Package

Title Analysing Conditional Income Distributions

Version 1.1

Date 2016-02-01

Author Alexander Sohn

Maintainer Alexander Sohn <asohn@uni-goettingen.de>

Description Functions for the analysis of income distributions for subgroups of the population as defined by a set of variables like age, gender, region, etc. This entails a Kolmogorov-Smirnov test for a mixture distribution as well as functions for moments, inequality measures, entropy measures and polarisation measures of income distributions. This package thus aides the analysis of income inequality by offering tools for the exploratory analysis of income distributions at the disaggregated level.

Depends gamlss,gamlss.dist,Hmisc,stats,graphics,splines

Imports grDevices,utils,datasets,methods

Suggests ineq

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2016-02-01 13:34:52

R topics documented:

`acid-package` *Analysing Conditional Income Distributions*

Description

Functions for the analysis of income distributions for subgroups of the population as defined by a set of variables like age, gender, region, etc. This entails a Kolmogorov-Smirnov test for a mixture distribution as well as functions for moments, inequality measures, entropy measures and polarisation measures of income distributions. This package thus aides the analysis of income inequality by offering tools for the exploratory analysis of income distributions at the disaggregated level.

Details

Package:	acid
Type:	Package
Version:	1.1
Date:	2015-01-06
License:	GPL-3

sadr.test, polarisation.ER, gini.den

Author(s)

Alexander Sohn <asohn@uni-goettingen.de>

References

Klein, N. and Kneib, T., Lang, S. and Sohn, A. (2015): Bayesian Structured Additive Distributional Regression with an Application to Regional Income Inequality in Germany, in: Annals of Applied Statistics, Vol. 9(2), pp. 1024-1052.

Sohn, A., Klein, N. and Kneib. T. (2014): A New Semiparametric Approach to Analysing Conditional Income Distributions, in: SOEPpapers, No. 676.

See Also

gamlss, ineq

---

arithmean.GB2 — *Mean of the Generalised Beta Distribution of Second Kind*

Description

This function calculates the expectation of the Generalised Beta Distribution of Second Kind.

Usage

```
arithmean.GB2(b, a, p, q)
```

Arguments

b	the parameter b of the Dagum distribution as defined by Kleiber and Kotz (2003).
a	the parameter a of the Dagum distribution as defined by Kleiber and Kotz (2003).
p	the parameter p of the Dagum distribution as defined by Kleiber and Kotz (2003).
q	the parameter q of the Dagum distribution as defined by Kleiber and Kotz (2003).

Value

returns the expectation.

Author(s)

Alexander Sohn

References

Kleiber, C. and Kotz, S. (2003): Statistical Size Distributions in Economics and Actuarial Sciences, Wiley, Hoboken.

Examples

```
a.test<- 4
b.test<- 20000
p.test<- 0.7
q.test<- 1
alpha.test<-1
GB2sample<-rGB2(10000,b.test,a.test,p.test,q.test)
arithmean.GB2(b.test,a.test,p.test,q.test)
mean(GB2sample)
```

---

atkinson *Atkinson Inequality Index*

Description

This function computes the Atkinson inequality index for a vector of observations.

Usage

```
atkinson(x, epsilon = 1)
```

Arguments

x	a vector of observations.
epsilon	inequality aversion parameter as denoted by Atkinson (1970). The default is epsilon=1.

Value

returns the selected Atkinson inequality index.

Author(s)

Alexander Sohn

References

Atkinson, A.B. (1970): On the Measurment of Inequality, in: Journal of Economic Theory, Vol. 2(3), pp. 244-263.

See Also

ineq

Examples

```
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
atkinson(x)
```

---

atkinson.den *Atkinson Index for an Income Distribution*

Description

This function approximates the Atkinson index for a distribution specified by a vector of densities and a corresponding income vector. A point mass at zero is allowed.

Usage

```
atkinson.den(incs, dens, epsilon = 1, pm0 = NA,
lower = NULL, upper = NULL, zero.approx = NULL)
```

Arguments

incs	a vector with income values.
dens	a vector with the corresponding densities.
epsilon	inequality aversion parameter as denoted by Atkinson (1970). The default is epsilon=1.
pm0	the point mass for zero incomes. If not specified no point mass is assumed.
lower	the lower bound of the income range considered.
upper	the upper bound of the income range considered.
zero.approx	a scalar which replaces zero-incomes, such that the Atkinson index involving a logarithm return finite values.

Value

AIM	the approximation of the selected Atkinson inequality measure.
epsilon	the inequality aversion parameter used.
mean	the approximated expected value of the distribution.
pm0	the point mass for zero incomes used.
lower	the lower bound of the income range considered used.
upper	the upper bound of the income range considered used.
zero.approx	the zero approximation used.

Author(s)

Alexander Sohn

References

Atkinson, A.B. (1970): On the Measurment of Inequality, in: Journal of Economic Theory, Vol. 2(3), pp. 244-263.

See Also

atkinson , atkinson.md

Examples

```
## without point mass at zero
incs<-seq(0,500,by=0.01)
dens<-dLOGNO(incs,2,1)
plot(incs,dens,type="l",xlim=c(0,100))
atkinson.den(incs=incs,dens=dens,epsilon=1)$AIM
atkinson(rLOGNO(50000,2,1),epsilon=1)
atkinson.den(incs=incs,dens=dens,epsilon=0.5)$AIM
atkinson(rLOGNO(50000,2,1),epsilon=0.5)

## with point mass at zero
incs<-c(seq(0,100,by=0.1),seq(100.1,1000,by=1),seq(1001,10000,by=10))
dens<-dLOGNO(incs,2,1)/2
dens[1]<-0.5
plot(incs,dens,type="l",ylim=c(0,max(dens[-1])),xlim=c(0,100))
#without zero approx zeros
atkinson.den(incs=incs,dens=dens,epsilon=1,pm0=0.5)$AIM
atkinson(c(rep(0,25000),rLOGNO(25000,2,1)),epsilon=1)
atkinson.den(incs=incs,dens=dens,epsilon=0.5,pm0=0.5)$AIM
atkinson(c(rep(0,25000),rLOGNO(25000,2,1)),epsilon=0.5)
#with zero approximation
atkinson.den(incs=incs,dens=dens,epsilon=0.5,pm0=0.5,zero.approx=1)$AIM
atkinson(c(rep(1,25000),rLOGNO(25000,2,1)),epsilon=0.5)
atkinson.den(incs=incs,dens=dens,epsilon=1,pm0=0.5,zero.approx=0.01)$AIM
atkinson(c(rep(0.01,250000),rLOGNO(250000,2,1)),epsilon=1)
```

---

atkinson.GB2 *Atkinson Index for a Generalised Beta Distribution of Second Kind*

Description

This function computes the Atkinson index (I(epsilon)) for Generalised Beta Distribution of Second Kind. The function is exact for the values epsilon=0, epsilon=1 and epsilon=2. For other values of epsilon, the function provides a numerical approximation.

Usage

```
atkinson.GB2(b, a, p, q, epsilon = NULL, ylim = c(0, 1e+06), zeroapprox = 0.01)
```

Arguments

b	the parameter b of the Dagum distribution as defined by Kleiber and Kotz (2003).
a	the parameter a of the Dagum distribution as defined by Kleiber and Kotz (2003).
p	the parameter p of the Dagum distribution as defined by Kleiber and Kotz (2003).
q	the parameter q of the Dagum distribution as defined by Kleiber and Kotz (2003).
epsilon	inequality aversion parameter as denoted by Atkinson (1970). The default is epsilon=1.
ylim	limits of the interval of y considered needed for the approximation of the entropy measure. The default is [0,1e+06].
zeroapprox	an approximation for zero needed for the approximation of the entropy measure. The default is 0.01.

Value

returns the selected Atkinson inequality index.

Author(s)

Alexander Sohn

References

Atkinson, A.B. (1970): On the Measurment of Inequality, in: Journal of Economic Theory, Vol. 2(3), pp. 244-263.

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 87-166, Elsevier, Amsterdam.

See Also

ineq

Examples

```
a.test<- 4
b.test<- 20000
p.test<- 0.7
q.test<- 1
epsilon.test<-1
GB2sample<-rGB2(1000,b.test,a.test,p.test,q.test)
atkinson.GB2(b.test,a.test,p.test,q.test,epsilon=epsilon.test,ylim=c(0,1e+07))
atkinson(GB2sample, epsilon.test)
```

atkinson.md *Atkinson Index for a Mixture of Income Distributions*

Description

This function uses Monte Carlo methods to estimate the Atkinson index for a mixture of two continuous income distributions and a point mass for zero-incomes.

Usage

```
atkinson.md(n, epsilon = 1, dist1, dist2, theta, p0, p1, p2,
dist.para.table, zero.approx)
```

Arguments

`n`	sample size used to estimate the Atkinson index.
`epsilon`	inequality aversion parameter as denoted by Atkinson (1970). The default is epsilon=1.
`dist1`	character string with the name of the first continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
`dist2`	character string with the name of the second continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
`theta`	vector with the parameters of dist1 and dist2. Order must be the same as in the functions for the distributions.
`p0`	scalar with probability mass for the point mass.
`p1`	scalar with probability mass for dist1.
`p2`	scalar with probability mass for dist2.
`dist.para.table`	a table of the same form as `dist.para.t` with distribution name, function name and number of parameters.
`zero.approx`	a scalar which replaces zero-incomes, such that the Atkinson index involving a logarithm return finite values.

Value

`AIM`	the selected Atkinson inequality measure.
`epsilon`	the inequality aversion parameter used.
`y`	a vector with the simulated incomes to estimate the entropy measure.
`y2`	a vector with the zero-replaced simulated incomes to estimate the entropy measure.
`zero.replace`	a logical vector indicating whether a zero has been replaced.
`stat`	a vector with the simulated group the observation was chosen from. 0 is the point mass, 1 dist1 and 2 dist2.

Author(s)

Alexander Sohn

References

Atkinson, A.B. (1970): On the Measurment of Inequality, in: Journal of Economic Theory, Vol. 2(3), pp. 244-263.

See Also

ineq , atkinson , atkinson.den

Examples

```
theta<-c(2,1,5,2)
x<- c(rgamma(50000,2,1),rgamma(50000,5,2))
para<-1

data(dist.para.t)
atkinson.md(10000,para,"gamma","gamma",theta,0,0.5,0.5,dist.para.t,zero.approx=1)$AIM
atkinson(x,1)
```

---

cdf.mix.dag — *Cumulative Density Function of Dagum Mixture Distribution*

Description

This function yields the cdf of a mixture distribution consisting of a point mass (at the lower end), a uniform distribution (above the point mass and below the Dagum distribution) and a Dagum distribution.

Usage

```
cdf.mix.dag(q, pi0, thres0 = 0, pi1, thres1, mu, sigma, nu, tau)
```

Arguments

q	a vector of quantiles.
pi0	the probability mass at thres0.
thres0	the location of the probability mass at the lower end of the distribution.
pi1	the probability mass of the uniform distribution.
thres1	the upper bound of the uniform distribution.
mu	the parameter mu of the Dagum distribution as defined by the function GB2.
sigma	the parameter sigma of the Dagum distribution as defined by the function GB2.
nu	the parameter nu of the Dagum distribution as defined by the function GB2.
tau	the parameter tau of the Dagum distribution as defined by the function GB2.

Value

returns the cumulative density for the given quantiles.

Author(s)

Alexander Sohn

References

Sohn, A., Klein, N. and Kneib. T. (2014): A New Semiparametric Approach to Analysing Conditional Income Distributions, in: SOEPpapers, No. 676.

See Also

gamlss.dist , gamlss.family

Examples

```
pi0.s<-0.2
pi1.s<-0.1
thres0.s<-0
thres1.s<-25000
mu.s<-20000
sigma.s<-5
nu.s<-0.5
tau.s<-1

cdf.mix.dag(50000,pi0.s,thres0.s,pi1.s,thres1.s,mu.s,sigma.s,nu.s,tau.s)
```

---

cdf.mix.LN — *Cumulative Density Function of Log-Normal Mixture Distribution*

Description

This function yields the cdf of a mixture distribution consisting of a point mass (at the lower end), a uniform distribution (above the point mass and below the log-normal distribution) and a log-normal distribution.

Usage

```
cdf.mix.LN(q, pi0, thres0 = 0, pi1, thres1, mu, sigma)
```

Arguments

`q`	a vector of quantiles.
`pi0`	the probability mass at thres0.
`thres0`	the location of the probability mass at the lower end of the distribution.
`pi1`	the probability mass of the uniform distribution.
`thres1`	the upper bound of the uniform distribution.
`mu`	the parameter mu of the Dagum distribution as defined by the function GB2.
`sigma`	the parameter sigma of the Dagum distribution as defined by the function GB2.

Value

returns the cumulative density for the given quantiles.

Author(s)

Alexander Sohn

References

Sohn, A., Klein, N., Kneib. T. (2014): A New Semiparametric Approach to Analysing Conditional Income Distributions, in: SOEPpapers, No. 676.

See Also

`gamlss.dist`, `gamlss.family`

Examples

```
pi0.s<-0.2
pi1.s<-0.1
thres0.s<-0
thres1.s<-25000
mu.s<-10
sigma.s<-2

cdf.mix.LN(50000,pi0.s,thres0.s,pi1.s,thres1.s,mu.s,sigma.s)
```

`coeffvar` *Coefficient of Variation*

Description

This function computes the Coefficient of Variation for a vector of observations.

Usage

```
coeffvar(x)
```

Arguments

`x`	a vector of observations.

Value

`cv`	returns the coefficient of variation without bias correction.
`bccv`	returns the coefficient of variation with bias correction.

Warning

Weighting is not properly accounted for in the sample adjustment of bccv!

Author(s)

Alexander Sohn

References

Atkinson, A.B. and Bourguignon, F. (2000): Income Distribution and Economics, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

See Also

`ineq`

Examples

```
# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
coeffvar(x)
```

`confband.kneib` *Simultaneous Confidence Bands*

Description

This function computes simultaneous confidence bands for samples of the presumed distribution of the parameter estimator.

Usage

```
confband.kneib(samples, level = 0.95)
```

Arguments

`samples`	matrix containing samples of the presumed distribution of the parameter estimator.
`level`	the desired confidence level.

Value

`lower`	a vector containing the lower bound of the confidence band.
`upper`	a vector containing the lower bound of the confidence band.

Note

This function is taken from the work of T. Krivobokova, T. Kneib and G. Claeskens.

Author(s)

Alexander Sohn

References

T. Krivobokova, T. Kneib, G. Claeskens (2010): Simultaneous Confidence Bands for Penalized Spline Estimators, in: Journal of the American Statistical Association, Vol. 105(490), pp.852-863.

Examples

```
mu<-1:20
n<-1000
mcmc<-matrix(NA,n,20)
for(i in 1:20){
  mcmc[,i]<- rnorm(n,mu[i],sqrt(i))
}

plot(mu,type="l",ylim=c(-10,30),lwd=3)
lines(confband.pw(mcmc)$lower,lty=2)
lines(confband.pw(mcmc)$upper,lty=2)
lines(confband.kneib(mcmc)$lower,lty=3)
lines(confband.kneib(mcmc)$upper,lty=3)
```

`confband.pw` *Pointwise Confidence Bands*

Description

This function computes pointwise confidence bands for samples of the presumed distribution of the parameter estimator.

Usage

```
confband.pw(samples, level = 0.95)
```

Arguments

`samples`	matrix containing samples of the presumed distribution of the parameter estimator.
`level`	the desired confidence level.

Value

`lower`	a vector containing the lower bound of the confidence band.
`upper`	a vector containing the lower bound of the confidence band.

Note

This function is mainly derived from the work of T. Krivobokova, T. Kneib and G. Claeskens.

Author(s)

Alexander Sohn

References

T. Krivobokova, T. Kneib, G. Claeskens (2010): Simultaneous Confidence Bands for Penalized Spline Estimators, in: Journal of the American Statistical Association, Vol. 105(490), pp.852-863.

Examples

```
mu<-1:20
n<-1000
mcmc<-matrix(NA,n,20)
for(i in 1:20){
  mcmc[,i]<- rnorm(n,mu[i],sqrt(i))
}

plot(mu,type="l",ylim=c(-10,30),lwd=3)
lines(confband.pw(mcmc)$lower,lty=2)
lines(confband.pw(mcmc)$upper,lty=2)
lines(confband.kneib(mcmc)$lower,lty=3)
lines(confband.kneib(mcmc)$upper,lty=3)
```

`dat` *ACID Simulated Data*

Description

This is some simulated income data from a mixture model as used in Sohn et al (2014).

Usage

```
data(dat)
```

Format

The format is: List of 4 $ dag.para:'data.frame': 8 obs. of 1 variable: ..$ parameters: num [1:8] 0.2 0.1 0 25000 20000 5 0.5 1 $ dag.s :'data.frame': 100 obs. of 3 variables: ..$ cat: int [1:100] 3 1 3 1 2 3 3 1 3 3 ... ..$ y : num [1:100] 36410 0 58165 0 15034 ... ..$ w : int [1:100] 1 1 1 2 1 3 2 1 1 1 ... $ LN.para :'data.frame': 6 obs. of 1 variable: ..$ parameters: num [1:6] 0.2 0.1 0 25000 10 2 $ LN.s :'data.frame': 100 obs. of 3 variables: ..$ cat: int [1:100] 3 3 1 3 3 3 3 3 3 3 ... ..$ y : num [1:100] 29614 29549 0 33068 463941 ... ..$ w : int [1:100] 1 2 1 1 1 1 1 2 1 1 ...

Details

The data contains information on whether the person is unemployed (cat=1), precariously employed (cat=2) or in standard employment(cat=3), the corresponding parameters used to generate the truncated distribution - both for Log-normal and Dagum.

References

Sohn, A., Klein, N., Kneib. T. (2014): A New Semiparametric Approach to Analysing Conditional Income Distributions, in: SOEPpapers, No. 676.

Examples

```
data(dat)
str(dat)
```

---

den.md — *Density for a Mixture of Income Distributions*

Description

This function computes the p-value for a mixture of two continuous income distributions and a point mass for zero-incomes.

Usage

```
den.md(y, dist1, dist2, theta, p0, p1, p2, dist.para.table)
```

Arguments

`y`	a vector with incomes. If a zero income is included, it must be the first element.
`dist1`	character string with the name of the first continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
`dist2`	character string with the name of the second continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
`theta`	vector with the parameters of dist1 and dist2. Order must be the same as in the functions for the distributions.

p0	scalar with probability mass for the point mass.
p1	scalar with probability mass for dist1.
p2	scalar with probability mass for dist2.
dist.para.table	a table of the same form as dist.para.t with distribution name, function name and number of parameters.

Value

returns the density for given values of y.

Author(s)

Alexander Sohn

See Also

ysample.md, pval.md

Examples

```
data(dist.para.t)
ygrid<-seq(0,20,by=0.1)#c(seq(0,1e5,by=100),seq(1.1e5,1e6,by=100000))
theta<-c(5,1,10,1.5)
p0<-0.2
p1<-0.3
p2<-0.5
n <-100000
y.sim <- ysample.md(n, "norm", "norm", theta, p0, p1, p2, dist.para.t)
den<-den.md(ygrid,"norm", "norm", theta,
              p0, p1, p2, dist.para.table=dist.para.t)
hist(y.sim,freq=FALSE)
#hist(y.sim,breaks=c(seq(0,1e5,by=100),seq(1.1e5,1e6,by=100000)),xlim=c(0,2e4),ylim=c(0,0.001))
lines(ygrid,den,col=2)
```

---

dist.para.t *Distributions and their Parameters*

Description

A data frame providing information on the number of parameters of distributions used for analysing conditional income distributions.

Usage

```
data(dist.para.t)
```

Format

A data frame with the following 3 variables.

`Distribution` name of the distribution.

`dist` function of the distribution.

`Parameters` the number of parameters for the distribution.

Examples

```
data(dist.para.t)
dist.para.t
```

`entropy` *Measures of the Generalised Entropy Family*

Description

This function computes the Measures of the Generalised Entropy Family for a vector of observations.

Usage

```
entropy(x, alpha = 1)
```

Arguments

`x`	a vector of observations.
`alpha`	the parameter for the generalised entropy family of measures, denoted by alpha by Cowell (2000). Note that this parameter notation differs from the notation used in the ineq package.

Value

returns the entropy measure.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

See Also

`ineq`

Examples

```
# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
entropy(x)
```

---

entropy.GB2 — *Entropy Measures for a Generalised Beta Distribution of Second Kind*

Description

This function computes four standard entropy measures from the generalised entropy class of inequality indices (I(alpha)) for Generalised Beta Distribution of Second Kind, namely the mean logarithmic deviation (I(0)), the Theil index (I(1)) as well as a bottom-sensitive index (I(-1)) and a top-sensitive index (I(2)). For other values of alpha, the function provides a numerical approximation.

Usage

```
entropy.GB2(b, a, p, q, alpha = NULL, ylim = c(0, 1e+06), zeroapprox = 0.01)
```

Arguments

`b`	the parameter b of the Dagum distribution as defined by Kleiber and Kotz (2003).
`a`	the parameter a of the Dagum distribution as defined by Kleiber and Kotz (2003).
`p`	the parameter p of the Dagum distribution as defined by Kleiber and Kotz (2003).
`q`	the parameter q of the Dagum distribution as defined by Kleiber and Kotz (2003).
`alpha`	measure for the entropy measure as denoted by Cowell (2000). The default is alpha=1, i.e. the Theil Index.
`ylim`	limits of the interval of y considered needed for the approximation of the entropy measure. The default is [0,1e+06].
`zeroapprox`	an approximation for zero needed for the approximation of the entropy measure. The default is 0.01.

Value

returns the selected entropy measure.

Author(s)

Alexander Sohn

References

Kleiber, C. and Kotz, S. (2003): Statistical Size Distributions in Economics and Actuarial Sciences, Wiley, Hoboken.

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 87-166, Elsevier, Amsterdam.

Jenkins, S.P. (2009): Distributionally-Sensitive Inequality Indices and the GB2 Income Distribution, in: Review of Income and Wealth, Vol. 55(2), pp.392-398.

Examples

```
a.test<- 4
b.test<- 20000
p.test<- 0.7
q.test<- 1
alpha.test<-1
GB2sample<-rGB2(1000,b.test,a.test,p.test,q.test)
entropy.GB2(b.test,a.test,p.test,q.test,alpha=alpha.test,ylim=c(0,1e+07))
entropy(GB2sample, alpha.test)
```

---

entropy.md — *Generalised Entropy Measure for a Mixture of Income Distributions*

Description

This function uses Monte Carlo methods to estimate an entropy measure for a mixture of two continuous income distributions and a point mass for zero-incomes.

Usage

```
entropy.md(n, alpha = 1, dist1, dist2, theta,
p0, p1, p2, dist.para.table, zero.approx)
```

Arguments

n	sample size used to estimate the entropy measure.
alpha	the parameter for the generalised entropy family of measures, denoted by alpha by Cowell (2000). Note that this parameter notation differs from the notation used in the ineq package.
dist1	character string with the name of the first continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
dist2	character string with the name of the second continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
theta	vector with the parameters of dist1 and dist2. Order must be the same as in the functions for the distributions.

`p0`	scalar with probability mass for the point mass.
`p1`	scalar with probability mass for dist1.
`p2`	scalar with probability mass for dist2.
`dist.para.table`	a table of the same form as `dist.para.t` with distribution name, function name and number of parameters.
`zero.approx`	a scalar which replaces zero-incomes (and negative incomes), such that entropy measures involving a logarithm return finite values.

Value

`entropy`	the estimated entropy measure.
`alpha`	the entropy parameter used.
`y`	a vector with the simulated incomes to estimate the entropy measure.
`y2`	a vector with the zero-replaced simulated incomes to estimate the entropy measure.
`zero.replace`	a logical vector indicating whether a zero has been replaced.
`stat`	a vector with the simulated group the observation was chosen from. 0 is the point mass, 1 dist1 and 2 dist2.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 87-166, Elsevier, Amsterdam.

See Also

`dist.para.t`, `entropy`

Examples

```
theta<-c(2,1,5,2)
x<- c(rgamma(500,2,1),rgamma(500,5,2))
para<-1
entropy(x,para)
data(dist.para.t)
entropy.md(100,para,"gamma","gamma",theta,0,0.5,0.5,dist.para.t,zero.approx=1)$entropy
```

frac.ranks — *Fractional Ranks*

Description

This function computes fractional ranks which are required for the S-Gini coefficient.

Usage

```
frac.ranks(x, w = NULL)
```

Arguments

`x`	a vector with sorted income values.
`w`	a vector of weights.

Value

returns the fractional ranks.

Author(s)

Alexander Sohn

References

van Kerm, P. (2009): sgini - Generalized Gini and Concentration coefficients (with factor decomposition) in Stata', CEPS/INSTEAD, Differdange, Luxembourg.

See Also

`sgini`, `sgini.den`

gini — *Gini Coefficient*

Description

This function computes the Gini coefficient for a vector of observations.

Usage

```
gini(x)
```

Arguments

`x`	a vector of observations.

Value

Gini	the Gini coefficient for the sample.
bcGini	the bias-corrected Gini coefficient for the sample.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 87-166, Elsevier, Amsterdam.

See Also

ineq

Examples

```
# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
gini(x)
```

---

gini.Dag *Gini Coefficient for the Dagum Distribution*

Description

This function computes the Gini coefficient for the Dagum Distribution.

Usage

```
gini.Dag(a, p)
```

Arguments

a	the parameter a of the Dagum distribution as defined by Kleiber and Kotz (2003).
p	the parameter p of the Dagum distribution as defined by Kleiber and Kotz (2003).

Value

returns the Gini coefficient.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

See Also

gini

Examples

```
a.test<- 4
b.test<- 20000
p.test<- 0.7
alpha.test<-1
GB2sample<-rGB2(10000,b.test,a.test,p.test,1)
gini.Dag(a.test,p.test)
gini(GB2sample)
```

---

gini.den *Gini Coefficient for an Income Distribution*

Description

This function approximates the Gini coefficient for a distribution specified by a vector of densities and a corresponding income vector. A point mass at zero is allowed.

Usage

```
gini.den(incs, dens, pm0 = NA,
lower = NULL, upper = NULL)
```

Arguments

incs	a vector with sorted income values.
dens	a vector with the corresponding densities.
pm0	the point mass for zero incomes. If not specified no point mass is assumed.
lower	the lower bound of the income range considered.
upper	the upper bound of the income range considered.

Value

Gini	the approximation of the Gini coefficient.
pm0	the point mass for zero incomes used.
lower	the lower bound of the income range considered used.
upper	the upper bound of the income range considered used.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

See Also

weighted.gini

Examples

```
mu<-2
sigma<-1
incs<-c(seq(0,500,by=0.01),seq(501,50000,by=1))
dens<-dLOGNO(incs,mu,sigma)
plot(incs,dens,type="l",xlim=c(0,100))
gini.den(incs=incs,dens=dens)$Gini
gini(rLOGNO(5000000,mu,sigma))$Gini
2*pnorm(sigma/sqrt(2))-1 #theoretical Gini
```

---

gini.gamma *Gini Coefficient for the Gamma Distribution*

Description

This function computes the Gini coefficient for the gamma distribution.

Usage

```
gini.gamma(p)
```

Arguments

p	the shape parameter p of the gamma distribution as defined by Kleiber and Kotz (2003).

Value

returns the Gini coefficient.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

Kleiber, C. and Kotz, S. (2003): Statistical Size Distributions in Economics and Actuarial Sciences, Wiley, Hoboken.

See Also

gini

Examples

```
shape.test <- 5
scale.test  <- 50000
y <- rgamma(10000,shape=shape.test,scale=scale.test)
gini(y)
gini.gamma(shape.test)
```

---

gini.md *Gini Coefficient for a Mixture of Income Distributions*

Description

This function uses Monte Carlo methods to estimate the Gini coefficient for a mixture of two continuous income distributions and a point mass for zero-incomes.

Usage

```
gini.md(n, dist1, dist2, theta,
p0, p1, p2, dist.para.table)
```

Arguments

n	sample size used to estimate the gini coefficient.
dist1	character string with the name of the first continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
dist2	character string with the name of the second continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
theta	vector with the parameters of dist1 and dist2. Order must be the same as in the functions for the distributions.
p0	scalar with probability mass for the point mass.
p1	scalar with probability mass for dist1.

p2 scalar with probability mass for dist2.

dist.para.table
: a table of the same form as dist.para.t with distribution name, function name and number of parameters.

Value

gini the estimated Gini coefficient.

y a vector with the simulated incomes to estimate the Gini coefficient.

stat a vector with the simulated group the observation was chosen from. 0 is the point mass, 1 dist1 and 2 dist2.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 87-166, Elsevier, Amsterdam.

See Also

dist.para.t, gini

Examples

```
theta<-c(2,1,5,2)
x<- c(rnorm(500,2,1),rnorm(500,5,2))
gini(x)$Gini
data(dist.para.t)
gini.md(1000,"norm","norm",theta,0,0.5,0.5,dist.para.t)$gini
```

---

ineq.md *Three Inequality Measures for a Mixture of Income Distributions*

Description

This function uses Monte Carlo methods to estimate an the mean logarithmic deviation, the Theil Index and the Gini Coefficient for a mixture of two continuous income distributions and a point mass for zero-incomes.

Usage

```
ineq.md(n, dist1, dist2, theta,
p0, p1, p2, dist.para.table, zero.approx)
```

Arguments

`n`	sample size used to estimate the gini coefficient.
`dist1`	character string with the name of the first continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
`dist2`	character string with the name of the second continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
`theta`	vector with the parameters of dist1 and dist2. Order must be the same as in the functions for the distributions.
`p0`	scalar with probability mass for the point mass.
`p1`	scalar with probability mass for dist1.
`p2`	scalar with probability mass for dist2.
`dist.para.table`	a table of the same form as `dist.para.t` with distribution name, function name and number of parameters.
`zero.approx`	a scalar which replaces zero-incomes (and negative incomes), such that entropy measures involving a logarithm return finite values.

Value

`MLD`	the estimated mean logarithmic deviation.
`Theil`	the estimated Theil index.
`Gini`	the estimated Gini coefficient.
`y`	a vector with the simulated incomes to estimate the entropy measure.
`y2`	a vector with the zero-replaced simulated incomes to estimate the entropy measure.
`zero.replace`	a logical vector indicating whether a zero has been replaced.
`stat`	a vector with the simulated group the observation was chosen from. 0 is the point mass, 1 dist1 and 2 dist2.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 87-166, Elsevier, Amsterdam.

See Also

`dist.para.t`, `gini`, `entropy`

Examples

```
theta<-c(0,1,5,2)
x<- c(rgamma(500,2,1),rgamma(500,5,2))
entropy(x,0)
entropy(x,1)
gini(x)$Gini
data(dist.para.t)
im<-ineq.md(100,"gamma","gamma",theta,0,0.5,0.5,dist.para.t,zero.approx=1)
im$MLD
im$Theil
im$Gini
```

---

km.GB2 *k-th Moment of the Generalised Beta Distribution of Second Kind*

Description

Calculates the k-th moment of the Generalised Beta Distribution of Second Kind.

Usage

```
km.GB2(b, a, p, q, k)
```

Arguments

b	the parameter b of the Dagum distribution as defined by Kleiber and Kotz (2003).
a	the parameter a of the Dagum distribution as defined by Kleiber and Kotz (2003).
p	the parameter p of the Dagum distribution as defined by Kleiber and Kotz (2003).
q	the parameter q of the Dagum distribution as defined by Kleiber and Kotz (2003).
k	order of the moment desired.

Value

returns the k-th moment.

Author(s)

Alexander Sohn

References

Kleiber, C. and Kotz, S. (2003): Statistical Size Distributions in Economics and Actuarial Sciences, Wiley, Hoboken.

Examples

```
a.test<- 4
b.test<- 20000
p.test<- 0.7
q.test<- 1
alpha.test<-1
GB2sample<-rGB2(10000,b.test,a.test,p.test,q.test)
km.GB2(b.test,a.test,p.test,q.test,k=1)
mean(GB2sample)
```

---

midks.plot — *Plot Comparing Parametric and Empirical Cumulative Density Functions*

Description

This function plots a graph entailing the empirical cdf and the parametrically specified cdf composed of a mixture distribution either by cdf.mix.dag or cdf.mix.LN.

Usage

```
midks.plot(x.seq, y, dist, w.emp = NULL, ...)
```

Arguments

`x.seq`	the sequence on the x-axis for which the parametric distribution is plotted.
`y`	a vector of observed incomes.
`dist`	a function specifying the parametric cdf.
`w.emp`	the weights of the observations contained in y.
`...`	arguments to be passed to dist.

Author(s)

Alexander sohn

See Also

`midks.test`,`cdf.mix.dag` ,`cdf.mix.LN`

Examples

```
# parameter values
pi0.s<-0.2
pi1.s<-0.1
thres0.s<-0
thres1.s<-25000
mu.s<-20000
```

```
sigma.s<-5
nu.s<-0.5
tau.s<-1
x.seq<-seq(0,200000,by=1000)

# generate sample
n<-100
s<-as.data.frame(matrix(NA,n,3))
names(s)<-c("cat","y","w")
s[,1]<-sample(1:3,n,replace=TRUE,prob=c(pi0.s,pi1.s,1-pi0.s-pi1.s))
s[,3]<-rep(1,n)
for(i in 1:n){
  if(s$cat[i]==1){s$y[i]<-0
  }else if(s$cat[i]==2){s$y[i]<-runif(1,thres0.s,thres1.s)
  }else s$y[i]<-rGB2(1,mu=mu.s,sigma=sigma.s,nu=nu.s,tau=tau.s)+thres1.s
}
# display
midks.plot(x.seq,s$y,dist=cdf.mix.dag,pi0=pi0.s,thres0=thres0.s,pi1=pi1.s,
thres1=thres1.s,mu=mu.s,sigma=sigma.s,nu=nu.s,tau=tau.s)
```

`midks.test` *Kolmogorov-Smirnov Test assessing a Parametric Mixture for a Conditional Income Distribution*

Description

This function performs a Kolmogorov-Smirnov test for a parametrically specified cdf composed of a mixture distribution either by cdf.mix.dag or cdf.mix.LN.

Usage

```
midks.test(x, y, ..., w = NULL, pmt = NULL)
```

Arguments

`x`	a vector of observed incomes.
`y`	a function specifying the parametric cdf.
`...`	arguments to be passed to y.
`w`	the weights of the observations contained in y.
`pmt`	point mass threshold equivalent to thres0 in y.

Value

`statistic`	returns the test statistic.
`method`	returns the methodology - currently always One-sample KS-test.
`diffpm`	the difference of the probability for the point mass.
`diff1`	the upper difference between for the continuous part of the cdfs.
`diff2`	the lower difference between for the continuous part of the cdfs.

Author(s)

Alexander Sohn

References

Sohn, A., Klein, N. and Kneib. T. (2014): A New Semiparametric Approach to Analysing Conditional Income Distributions, in: SOEPpapers, No. 676.

Examples

```
# parameter values
pi0.s<-0.2
pi1.s<-0.1
thres0.s<-0
thres1.s<-25000
mu.s<-20000
sigma.s<-5
nu.s<-0.5
tau.s<-1

# generate sample
n<-100
s<-as.data.frame(matrix(NA,n,3))
names(s)<-c("cat","y","w")
s[,1]<-sample(1:3,n,replace=TRUE,prob=c(pi0.s,pi1.s,1-pi0.s-pi1.s))
s[,3]<-rep(1,n)
for(i in 1:n){
  if(s$cat[i]==1){s$y[i]<-0
  }else if(s$cat[i]==2){s$y[i]<-runif(1,thres0.s,thres1.s)
  }else s$y[i]<-rGB2(1,mu=mu.s,sigma=sigma.s,nu=nu.s,tau=tau.s)+thres1.s
}

# midks.test
midks.test(s$y,cdf.mix.dag,pi0=pi0.s,thres0=thres0.s,pi1=pi1.s,thres1=thres1.s,mu=mu.s,
sigma=sigma.s,nu=nu.s,tau=tau.s,w=s$w,pmt=thres0.s)$statistic
```

`params` *Parameter estimators obtained from Structured Additive Distributional Regression*

Description

A list containing parameter estimates as obtained from Structured Additive Distributional Regression

Usage

```
data(params)
```

Format

The format is: List of 16 $ mcmcsize : num 1000 $ ages : int [1:40] 21 22 23 24 25 26 27 28 29 30 ... $ unems : num [1:23] 0 1 2 3 4 5 6 7 8 9 ... $ educlvls : num [1:2] -1 1 $ bulas : chr [1:16] "SH" "HH" "NDS" "Bremen" ... $ aft.v : num [1:3447] 4.85 6.5 5.92 5.76 6.05 ... $ bft.v : num [1:3447] 78169 65520 47184 58763 46188 ... $ cft.v : num [1:3447] 1.177 0.299 0.818 0.522 0.836 ... $ mupt.v : num [1:3447] 10.21 9.46 9.66 9.77 9.68 ... $ sigmapt.v: num [1:3447] 1.07 1.25 1.85 1.21 1.74 ... $ muemp.v : num [1:3447] 3.25 2.68 2.08 3.53 2.43 ... $ muunemp.v: num [1:3447] -2.691 -0.813 -1.919 -1.542 -1.765 ... $ punemp.v : num [1:3447] 0.0658 0.3104 0.1314 0.18 0.1496 ... $ pemp.v : num [1:3447] 0.934 0.69 0.869 0.82 0.85 ... $ pft.v : num [1:3447] 0.898 0.644 0.77 0.796 0.78 ... $ ppt.v : num [1:3447] 0.0359 0.0452 0.0987 0.024 0.0706 ...

Examples

```
data(params)
str(params)
## maybe str(params) ; plot(params) ...
```

`pens.parade` *Pen's Parade*

Description

This function plots Pen's parade.

Usage

```
pens.parade(x, bodies = TRUE, feet = 0, ...)
```

Arguments

`x`	a vector of observed incomes.
`bodies`	a logical value indicating whether lines, i.e. the bodies, should be drawn.
`feet`	a numeric value indicating where the lines originate.
`...`	additional arguments passed to the plot function.

Author(s)

Alexander Sohn

References

Atkinson, A.B. (1975): The Economics of Inequality, Cleardon Press, Oxford.

Examples

```
a.test<- 4
b.test<- 20000
p.test<- 0.7
q.test<- 1
alpha.test<-1
GB2sample<-rGB2(100,b.test,a.test,p.test,q.test)
pens.parade( GB2sample)
```

`polarisation.EGR` *Polarisation Measure from Esteban, Gradin and Ray (2007)*

Description

This function computes the polarisation measure proposed in Esteban, Gradin and Ray (2007) which accounts for deviations from an n-spike representation of strata in society.

Usage

```
polarisation.EGR(alpha, beta, rho, y, f = NULL, dist = NULL,
weights = NULL, pm0 = NA, lower = NULL, upper = NULL, ...)
```

Arguments

`alpha`	a scalar containing the alpha parameter from Esteban and Ray (1994) on the sensitivity to polarisation.
`beta`	a scalar containing the beta parameter from Esteban, Gradin and Ray (2007) on the weight assigned to the error in the n-spike representation.
`rho`	a dataframe with the group means in the first column and their respective population shares in the second. The groups need to be exogenously defined. Note: the two columns should be named `means` and `shares` respectively. Otherwise a warning will appear.

`y`	a vector of incomes. If f is NULL and dist is NULL, this includes all incomes of all observations in the sample, i.e. all observations comprising the aggregate distribution. If either f or dist is not NULL, then this gives the incomes where the density is evaluated.
`f`	a vector of user-defined densities of the aggregate distribution for the given incomes in y.
`dist`	character string with the name of the distribution used. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
`weights`	an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If non-NULL, observations in y are weighted accordingly.
`pm0`	the point mass for zero incomes used in the gini.den function. If not specified no point mass is assumed.
`lower`	the lower bound of the income range considered used in the gini.den function.
`upper`	the upper bound of the income range considered used in the gini.den function.
`...`	arguments to be passed to the distribution function used, e.g. mean and sd for the normal distribution.

Value

`P`	the polarisation measure proposed by Esteban, Gradin and Ray (2007).
`PG`	the adjusted polarisation measure proposed by Gradin (2000).
`alpha`	the alpha parameter used.
`beta`	the beta parameter used.
`beta`	the distribution option used, i.e. whether only y, f or dist was used.

Author(s)

Alexander Sohn

References

Esteban, J. and Ray, D. (1994): On the Measurment of Polarization, in: Econometrica, Vol. 62(4), pp. 819-851.

Esteban, J., Gradin, C. and Ray, D. (2007): Extensions of a Measure of Polarization, with an Application to the Income Distribution of five OECD Countries.

Gradin, C. (2000): Polarization by Sub-populations in Spain, 1973-91, in Review of Income and Wealth, Vol. 46(4), pp.457-474.

See Also

`polarisation.ER`

Examples

```
## example 1
y<-rnorm(1000,5,0.5)
y<-sort(y)
m.y<-mean(y)
sd.y<-sd(y)
y1<-y[1:(length(y)/4)]
m.y1<-mean(y1)
sd.y1<-sd(y1)
y2<-y[(length(y)/4+1):length(y)]
m.y2<-mean(y2)
sd.y2<-sd(y2)
means<-c(m.y1,m.y2)
share1<- length(y1)/length(y)
share2<- length(y2)/length(y)
shares<- c(share1,share2)
rho<-data.frame(means=means,shares=shares)
alpha<-1
beta<-1
den<-density(y)
polarisation.ER(alpha,rho,comp=FALSE)
polarisation.EGR(alpha,beta,rho,y)$P
polarisation.EGR(alpha,beta,rho,y=den$x,f=den$y)$P
polarisation.EGR(alpha,beta,rho,y=seq(0,10,by=0.1),dist="norm",
mean=m.y,sd=sd.y)$P
polarisation.EGR(alpha,beta,rho,y=seq(0,10,by=0.1),dist="norm",
mean=m.y,sd=sd.y)$PG

## example 2
y1<-rnorm(100,5,1)
y2<-rnorm(100,1,0.1)
y <- c(y1,y2)
m.y1<-mean(y1)
sd.y1<-sd(y1)
m.y2<-mean(y2)
sd.y2<-sd(y2)
means<-c(m.y1,m.y2)
share1<- length(y1)/length(y)
share2<- length(y2)/length(y)
shares<- c(share1,share2)
rho<-data.frame(means=means,shares=shares)
alpha<-1
beta<-1
polarisation.EGR(alpha,beta,rho,y=seq(0,10,by=0.1),dist="norm",
                 mean=c(m.y1,m.y2),sd=c(sd.y1,sd.y2))$P
```

---

polarisation.ER *Polarisation Measure from Esteban and and Ray (1994)*

Description

This function computes the polarisation measure proposed in Esteban and and Ray (1994).

Usage

```
polarisation.ER(alpha, rho, comp = FALSE)
```

Arguments

alpha	a scalar containing the alpha parameter from Esteban and Ray (1994) on the sensitivity to polarisation.
rho	a dataframe with the group means in the first column and their respective population shares in the second. The groups need to be exogenously defined. Note: the two columns should be named means and "shares" respectively. Otherwise a warning will appear.
comp	logical; if TRUE, all components pf p_i^(a+alpha)*p_j*abs(y_i-y_j)

Value

P	the polarisation measure proposed by Esteban and Ray (1994).
means	the means stored in rho.
shares	the shares stored in rho..
ERcomp	if comp is TRUE, the components aggregated in P.

Author(s)

Alexander Sohn

References

Esteban, J. and Ray, D. (1994): On the Measurment of Polarization, in: Econometrica, Vol. 62(4), pp. 819-851.

See Also

polarisation.EGR

Examples

```
means<-rnorm(10)+5
shares<-  rep(1,length(means))
shares<-shares/sum(shares)
rho<-data.frame(means=means,shares=shares)
alpha<-1
polarisation.ER(alpha,rho,comp=FALSE)
```

pval.md *P-Value for a Mixture of Income Distributions*

Description

This function computes the p-value for a mixture of two continuous income distributions and a point mass for zero-incomes.

Usage

```
pval.md(y, dist1, dist2, theta, p0, p1, p2, dist.para.table)
```

Arguments

- `y` a vector with incomes. If a zero income is included, it must be the first element.
- `dist1` character string with the name of the first continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
- `dist2` character string with the name of the second continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
- `theta` vector with the parameters of dist1 and dist2. Order must be the same as in the functions for the distributions.
- `p0` scalar with probability mass for the point mass.
- `p1` scalar with probability mass for dist1.
- `p2` scalar with probability mass for dist2.
- `dist.para.table` a table of the same form as `dist.para.t` with distribution name, function name and number of parameters.

Value

returns the p-value.

Author(s)

Alexander Sohn

See Also

`ysample.md`, `den.md`

Examples

```
data(dist.para.t)
ygrid<-seq(0,1e5,by=1000)
theta<-c(5,1,10,3)
p0<-0.2
p1<-0.3
p2<-0.5
n <-10000
y.sim <- ysample.md(n, "LOGNO", "LOGNO", theta, p0, p1, p2, dist.para.t)
pval<-pval.md(ygrid,"LOGNO", "LOGNO", theta,
              p0, p1, p2, dist.para.table=dist.para.t)
mean(y.sim<=ygrid[10])
pval[10]
```

---

sadr.test *Misspecification Test assessing a Parametric Conditional Income Distribution*

Description

This function performs a misspecificaton test for a parametrically specified cdf estimated by (Bayesian) Structured Additive Distributional Regression.

Usage

```
sadr.test(data, y.pos = NULL, dist1, dist2, params.m, mcmc = TRUE, mcmc.params.a,
 ygrid, bsrep = 10, n.startvals = 300, dist.para.table)
```

Arguments

data	a dataframe including dependent variable and all explanatory variables.
y.pos	an integer indicating the position of the dependent variable in the dataframe.
dist1	character string with the name of the first continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
dist2	character string with the name of the second continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
params.m	a matrix with the estimated parameter values (in colums) for each individual (in rows). The order of the parameters must be as follows: parameters for the first distribution, parameters for the second distribution, probability of zero income, probability of dist1, probability of dist2 and probability of dist1 given employment/non-zero income.
mcmc	logical; if TRUE, uncertainty as provided by the MCMC samples is considered.

`mcmc.params.a` an array, with the mcmc samples for all the parameters specified by structured additive distributional regression. In the first dimension should be the MCMC realisations, in the second dimension the individuals and in the third the parameters. The order of the parameters must be as follows: parameters for the first distribution, parameters for the second distribution, probability of zero income, probability of dist1, probability of dist2 and probability of dist1 given employment/non-zero income.

`ygrid` vector yielding the grid on which the cdf is specified.

`bsrep` integer giving the number of bootstrap repitions in order to determine the distributions of the test statistics under the null.

`n.startvals` integer giving the maximum number of observations used to estimate the test statistic.

`dist.para.table`
a table of the same form as `dist.para.t` with distribution name, function name and number of parameters.

Value

`teststat.ks` Kolmogorov-Smirnov test statistic.

`pval.ks` p-value based on the Kolmogorov-Smirnov test statistic.

`teststat.cvm` Cramer-von-Mises test statistic.

`pval.cvm` p-value based on the Cramer-von-Mises test statistic.

`test` type cdf considered for the test.

`param.distributions`
parametric distributions assumed for dist1 and dist2.

`teststat.ks.bs` bootstrap results of Kolmogorov-Smirnov test statistic under null.

`teststat.cvm.bs`
bootstrap results of Cramer-von-Mises test statistic under null.

Author(s)

Alexander Sohn

References

Rothe, C. and Wied, D. (2013): Misspecification Testing in a Class of Conditional Distributional Models, in: Journal of the American Statistical Association, Vol. 108(501), pp.314-324.

Sohn, A. (forthcoming): Scars from the Past and Future Earning Distributions.

Examples

```
# ### functions not run - take considerable time!
#
# library(acid)
# data(dist.para.t)
# data(params)
# ### example one - two normals, no mcmc
```

```
# dist1<-"norm"
# dist2<-"norm"
# ## generating data
# set.seed(1234)
# n<-1000
# sigma<-0.1
# X.theta<-c(1,10,1,10)
# X.gen<-function(n,paras){
#   X<-matrix(c(round(runif(n,paras[1],paras[2])),round(runif(n,paras[3],
#             paras[4]))),ncol=2)
#   return(X)
# }
# X <- X.gen(n,X.theta)
# beta.mu1   <- 1
# beta.sigma1<- 0.1
# beta.mu2   <- 2
# beta.sigma2<- 0.1
# pi0        <- 0.3
# pi01       <- 0.8
# pi1        <- (1-pi0)*pi01
# pi2        <- 1-pi0-pi1
#
# params.m<-matrix(NA,n,8)
# params.m[,1]<-(0+beta.mu1)*X[,1]
# params.m[,2]<-(0+beta.sigma1)*X[,1]
# params.m[,3]<-(0+beta.mu2)*X[,2]
# params.m[,4]<-(0+beta.sigma2)*X[,2]
# params.m[,5]<-pi0
# params.m[,6]<-pi1
# params.m[,7]<-pi2
# params.m[,8]<-pi01
#
# params.mF<-matrix(NA,n,8)
# params.mF[,1]<-(10+beta.mu1)*X[,1]
# params.mF[,2]<-(0+beta.sigma1)*X[,1]
# params.mF[,3]<-(0+beta.mu2)*X[,2]
# params.mF[,4]<-(2+beta.sigma2)*X[,2]
# params.mF[,5]<-pi0
# params.mF[,6]<-pi1
# params.mF[,7]<-pi2
# params.mF[,8]<-pi01
# # starting repititions
# reps<-30
# tsreps1T<-rep(NA,reps)
# tsreps2T<-rep(NA,reps)
# tsreps1F<-rep(NA,reps)
# tsreps2F<-rep(NA,reps)
# sys.t<-Sys.time()
# for(r in 1:reps){
#   Y <- rep(NA,n)
#   for(i in 1:n){
#     Y[i] <- ysample.md(1,dist1,dist2,theta=params.m[i,1:4],params.m[i,5],
#     params.m[i,6],params.m[i,7],dist.para.t)
```

```
#   }
#   dat<-cbind(Y,X)
#   y.pos<-1
#   ygrid<-seq(min(Y),round(max(Y)*1.2,-1),by=1)
#   tsT<-sadr.test(data=dat,y.pos=NULL,dist1="norm",dist2="norm",
#   params.m=params.m,mcmc=FALSE,mcmc.params=NA,ygrid=ygrid, bsrep=100,
#   n.startvals=30000,dist.para.table=dist.para.t)
#   tsreps1T[r]<-tsT$pval.ks
#   tsreps2T[r]<-tsT$pval.cvm
#   tsF<-sadr.test(data=dat,y.pos=NULL,dist1="norm",dist2="norm",
#   params.m=params.mF,mcmc=FALSE,mcmc.params=NA,ygrid=ygrid, bsrep=100,
#   n.startvals=30000,dist.para.table=dist.para.t)
#   tsreps1F[r]<-tsF$pval.ks
#   tsreps2F[r]<-tsF$pval.cvm
# }
# time.taken<-Sys.time()-sys.t
# time.taken
# cbind(tsreps1T,tsreps2T,tsreps1F,tsreps2F)
#
# data(dist.para.t)
# data(params)
#
# ### example two - Dagum and log-normal - no mcmc
# ##putting list elements from params into matrix form for params.m
# params.m<-matrix(NA,length(params$aft.v),6+4)
# params.m[,1]<-params[[which(names(params)=="bft.v")]]
# params.m[,2]<-params[[which(names(params)=="aft.v")]]
# params.m[,3]<-params[[which(names(params)=="cft.v")]]
# params.m[,4]<-1
# params.m[,5]<-params[[which(names(params)=="mupt.v")]]
# params.m[,6]<-params[[which(names(params)=="sigmapt.v")]]
# params.m[,7]<-params[[which(names(params)=="punemp.v")]]
# params.m[,8]<-params[[which(names(params)=="pft.v")]]
# params.m[,9]<-params[[which(names(params)=="ppt.v")]]
# params.m[,10]<-params[[which(names(params)=="pemp.v")]]
#
# set.seed(123)
# reps<-30
# tsreps1T<-rep(NA,reps)
# tsreps2T<-rep(NA,reps)
# tsreps1F<-rep(NA,reps)
# tsreps2F<-rep(NA,reps)
# sys.t<-Sys.time()
# for(r in 1:reps){
#   ## creates variables under consideration and dimnames
#   n  <- dim(params.m)[1]
#   mcmcsize<-params$mcmcsize
#   ages <- params$ages
#   unems <- params$unems
#   educlvls <- params$educlvls
#   OW <- params$OW
#   ## simulate two samples
#   ages.s <- sample(ages,n,replace=TRUE)
```

```
#   unems.s<- sample(unems,n,replace=TRUE)
#   edu.s  <- sample(c(-1,1),n,replac=TRUE)
#   OW.s   <- sample(c(-1,1),n,replac=TRUE)
#   y.sim<-rep(NA,n)
#   p.sel<-sample(1:dim(params.m)[1],n)
#   for(i in 1:n){
#     p<-p.sel[i]
#     #p<-sample(1:n,1) #select a random individual
#     y.sim[i]<-ysample.md(1,"GB2","LOGNO",
#                        theta=c(params$bft.v[p],params$aft.v[p],
#                                params$cft.v[p],1,
#                                params$mupt.v[p],params$sigmapt.v[p]),
#                        params$punemp.v[p],params$pft.v[p],params$ppt.v[p],
#                        dist.para.t)
#   }
#   dat<-cbind(y.sim,ages.s,unems.s,edu.s,OW.s)
#   y.simF<- rnorm(n,mean(y.sim),sd(y.sim))
#   y.simF[y.simF<0]<-0
#   datF<-dat
#   datF[,1]<-y.simF
#   ygrid <- seq(0,1e6,by=1000) #quantile(y,taus)
#   ##executing test
#   tsT<-sadr.test(data=dat,y.pos=NULL,dist1="GB2",dist2="LOGNO",params.m=
#                  params.m[p.sel,],mcmc=FALSE,mcmc.params=NA,ygrid=ygrid,
#                  bsrep=100,n.startvals=30000,dist.para.table=dist.para.t)
#   tsreps1T[r]<-tsT$pval.ks
#   tsreps2T[r]<-tsT$pval.cvm
#   tsF<-sadr.test(data=datF,y.pos=NULL,dist1="GB2",dist2="LOGNO",
#                  params.m=params.m[p.sel,],mcmc=FALSE,mcmc.params=NA,
#                  ygrid=ygrid,
#                  bsrep=100,n.startvals=30000,dist.para.table=dist.para.t)
#   tsreps1F[r]<-tsF$pval.ks
#   tsreps2F[r]<-tsF$pval.cvm
# }
# time.taken<-Sys.time()-sys.t
# time.taken
# cbind(tsreps1T,tsreps2T,tsreps1F,tsreps2F)
#
#
#
#
#
# ### example three - two normals, with mcmc
# set.seed(1234)
# n<-1000 #no of observations
# m<-100 #no of mcmc samples
# sigma<-0.1
# X.theta<-c(1,10,1,10)
# #without weights
# X.gen<-function(n,paras){
#   X<-matrix(c(round(runif(n,paras[1],paras[2])),round(runif(n,paras[3],
#             paras[4]))),ncol=2)
#   return(X)
```

```
# }
# X <- X.gen(n,X.theta)
#
# beta.mu1   <- 1
# beta.sigma1<- 0.1
# beta.mu2   <- 2
# beta.sigma2<- 0.1
# pi0        <- 0.3
# pi01       <- 0.8
# pi1        <- (1-pi0)*pi01
# pi2        <- 1-pi0-pi1
#
# mcmc.params.a<-array(NA,dim=c(m,n,8))
# mcmc.params.a[,,1]<-(0+beta.mu1+rnorm(m,0,beta.mu1/10))%*%t(X[,1])
      #assume sd of mcmc as 10% of parameter value
# mcmc.params.a[,,2]<-(0+beta.sigma1+rnorm(m,0,beta.sigma1/10))%*%t(X[,1])
      #must not be negative!, may be for other seed!
# mcmc.params.a[,,3]<-(0+beta.mu2+rnorm(m,0,beta.mu2/10))%*%t(X[,2])
# mcmc.params.a[,,4]<-(0+beta.sigma2+rnorm(m,0,beta.sigma2/10))%*%t(X[,2])
      #must not be negative!, may be for other seed!
# mcmc.params.a[,,5]<-(pi0+rnorm(m,0,pi0/10))%*%t(rep(1,n))
# mcmc.params.a[,,8]<-(pi01+rnorm(m,0,pi01/10))%*%t(rep(1,n))
# mcmc.params.a[,,6]<-(1-mcmc.params.a[,,5])*mcmc.params.a[,,8]
# mcmc.params.a[,,7]<-1-mcmc.params.a[,,5]-mcmc.params.a[,,6]
#
# params.m<-apply(mcmc.params.a,MARGIN=c(2,3),FUN=quantile,probs=0.5)
#
# mcmc.params.aF<-array(NA,dim=c(m,n,8))
# mcmc.params.aF[,,1]<-(10+beta.mu1+rnorm(m,0,beta.mu1/10))%*%t(X[,1])
      #assume sd of mcmc as 10% of parameter value
# mcmc.params.aF[,,2]<-(0+beta.sigma1+rnorm(m,0,beta.sigma1/10))%*%t(X[,1])
      #must not be negative!, may be for other seed!
# mcmc.params.aF[,,3]<-(0+beta.mu2+rnorm(m,0,beta.mu2/10))%*%t(X[,2])
# mcmc.params.aF[,,4]<-(2+beta.sigma2+rnorm(m,0,beta.sigma2/10))%*%t(X[,2])
      #must not be negative!, may be for other seed!
# mcmc.params.aF[,,5]<-(pi0+rnorm(m,0,pi0/10))%*%t(rep(1,n))
# mcmc.params.aF[,,8]<-(pi01+rnorm(m,0,pi01/10))%*%t(rep(1,n))
# mcmc.params.aF[,,6]<-(1-mcmc.params.aF[,,5])*mcmc.params.aF[,,8]
# mcmc.params.aF[,,7]<-1-mcmc.params.aF[,,5]-mcmc.params.aF[,,6]
#
# params.mF<-apply(mcmc.params.aF,MARGIN=c(2,3),FUN=quantile,probs=0.5)
#
# reps<-30
# tsreps1T<-rep(NA,reps)
# tsreps2T<-rep(NA,reps)
# tsreps1F<-rep(NA,reps)
# tsreps2F<-rep(NA,reps)
# sys.t<-Sys.time()
# for(r in 1:reps){
#   Y <- rep(NA,n)
#   for(i in 1:n){
#     Y[i] <- ysample.md(1,dist1,dist2,theta=params.m[i,1:4],params.m[i,5],
#                         params.m[i,6],params.m[i,7],dist.para.t)
```

```
#   }
#   dat<-cbind(Y,X)
#   y.pos<-1
#   ygrid<-seq(min(Y),round(max(Y)*1.2,-1),by=1)
#   tsT<-sadr.test(data=dat,y.pos=NULL,dist1="norm",dist2="norm",params.m=
#                  params.m,mcmc=TRUE,mcmc.params=mcmc.params.a,ygrid=ygrid,
#                  bsrep=100,n.startvals=30000,dist.para.table=dist.para.t)
#   tsreps1T[r]<-tsT$pval.ks
#   tsreps2T[r]<-tsT$pval.cvm
#   tsF<-sadr.test(data=dat,y.pos=NULL,dist1="norm",dist2="norm",
#                  params.m=params.mF,mcmc=TRUE,mcmc.params=mcmc.params.aF,
#                  ygrid=ygrid, bsrep=100,n.startvals=30000,
#                  dist.para.table=dist.para.t)
#   tsreps1F[r]<-tsF$pval.ks
#   tsreps2F[r]<-tsF$pval.cvm
#   #c(ts$teststat.ks,ts$teststat.cvm)
#   #c(ts$pval.ks,ts$pval.cvm)
#
# }
# time.taken<-Sys.time()-sys.t
# time.taken
# cbind(tsreps1T,tsreps2T,tsreps1F,tsreps2F)
#
#
#
# ### example four - two normals, with mcmc and slight deviation from truth
#     in true params
# library(acid)
# data(dist.para.t)
# data(params)
# dist1<-"norm"
# dist2<-"norm"
#
# set.seed(1234)
# n<-1000 #no of observations
# m<-100 #no of mcmc samples
# sigma<-0.1
# X.theta<-c(1,10,1,10)
# #without weights
# X.gen<-function(n,paras){
#   X<-matrix(c(round(runif(n,paras[1],paras[2])),round(runif(n,paras[3],
#             paras[4]))),ncol=2)
#   return(X)
# }
# X <- X.gen(n,X.theta)
#
# beta.mu1   <- 1
# beta.sigma1<- 0.1
# beta.mu2   <- 2
# beta.sigma2<- 0.1
# pi0        <- 0.3
# pi01       <- 0.8
# pi1        <- (1-pi0)*pi01
```

```
# pi2        <- 1-pi0-pi1
#
# mcmc.params.a<-array(NA,dim=c(m,n,8))
# mcmc.params.a[,,1]<-(beta.mu1/10+beta.mu1+rnorm(m,0,beta.mu1/10))%*%t(X[,1])
       #assume sd of mcmc as 10% of parameter value
# mcmc.params.a[,,2]<-(0+beta.sigma1+rnorm(m,0,beta.sigma1/10))%*%t(X[,1])
       #must not be negative!, may be for other seed!
# mcmc.params.a[,,3]<-(0+beta.mu2+rnorm(m,0,beta.mu2/10))%*%t(X[,2])
# mcmc.params.a[,,4]<-(beta.sigma2/10+beta.sigma2+rnorm(m,0,
#                     beta.sigma2/10))%*%t(X[,2])
       #must not be negative!, may be for other seed!
# mcmc.params.a[,,5]<-(pi0+rnorm(m,0,pi0/10))%*%t(rep(1,n))
# mcmc.params.a[,,8]<-(pi01+rnorm(m,0,pi01/10))%*%t(rep(1,n))
# mcmc.params.a[,,6]<-(1-mcmc.params.a[,,5])*mcmc.params.a[,,8]
# mcmc.params.a[,,7]<-1-mcmc.params.a[,,5]-mcmc.params.a[,,6]
#
# params.m<-apply(mcmc.params.a,MARGIN=c(2,3),FUN=quantile,probs=0.5)
#
# mcmc.params.aF<-array(NA,dim=c(m,n,8))
# mcmc.params.aF[,,1]<-(10+beta.mu1+rnorm(m,0,beta.mu1/10))%*%t(X[,1])
       #assume sd of mcmc as 10% of parameter value
# mcmc.params.aF[,,2]<-(0+beta.sigma1+rnorm(m,0,beta.sigma1/10))%*%t(X[,1])
       #must not be negative!, may be for other seed!
# mcmc.params.aF[,,3]<-(0+beta.mu2+rnorm(m,0,beta.mu2/10))%*%t(X[,2])
# mcmc.params.aF[,,4]<-(2+beta.sigma2+rnorm(m,0,beta.sigma2/10))%*%t(X[,2])
       #must not be negative!, may be for other seed!
# mcmc.params.aF[,,5]<-(pi0+rnorm(m,0,pi0/10))%*%t(rep(1,n))
# mcmc.params.aF[,,8]<-(pi01+rnorm(m,0,pi01/10))%*%t(rep(1,n))
# mcmc.params.aF[,,6]<-(1-mcmc.params.aF[,,5])*mcmc.params.aF[,,8]
# mcmc.params.aF[,,7]<-1-mcmc.params.aF[,,5]-mcmc.params.aF[,,6]
#
# params.mF<-apply(mcmc.params.aF,MARGIN=c(2,3),FUN=quantile,probs=0.5)
#
# reps<-30
# tsreps1T<-rep(NA,reps)
# tsreps2T<-rep(NA,reps)
# tsreps1F<-rep(NA,reps)
# tsreps2F<-rep(NA,reps)
# sys.t<-Sys.time()
# for(r in 1:reps){
#   Y <- rep(NA,n)
#   for(i in 1:n){
#     Y[i] <- ysample.md(1,dist1,dist2,theta=params.m[i,1:4],params.m[i,5],
#                         params.m[i,6],params.m[i,7],dist.para.t)
#   }
#
#   dat<-cbind(Y,X)
#   y.pos<-1
#   ygrid<-seq(min(Y),round(max(Y)*1.2,-1),by=1)
#   tsT<-sadr.test(data=dat,y.pos=NULL,dist1="norm",dist2="norm",
#                  params.m=params.m,mcmc=TRUE,mcmc.params=mcmc.params.a,
#                  ygrid=ygrid, bsrep=100,n.startvals=30000,
#                  dist.para.table=dist.para.t)
```

```
#   tsreps1T[r]<-tsT$pval.ks
#   tsreps2T[r]<-tsT$pval.cvm
#   tsF<-sadr.test(data=dat,y.pos=NULL,dist1="norm",dist2="norm",
#                  params.m=params.mF,mcmc=TRUE,mcmc.params=mcmc.params.aF,
#                  ygrid=ygrid, bsrep=100,n.startvals=30000,
#                  dist.para.table=dist.para.t)
#   tsreps1F[r]<-tsF$pval.ks
#   tsreps2F[r]<-tsF$pval.cvm
#   #c(ts$teststat.ks,ts$teststat.cvm)
#   #c(ts$pval.ks,ts$pval.cvm)
#
# }
# time.taken<-Sys.time()-sys.t
# time.taken
# cbind(tsreps1T,tsreps2T,tsreps1F,tsreps2F)
```

sd.GB2 — *Standard Deviation of the Generalised Beta Distribution of Second Kind*

Description

This function calculates the standard deviation of the Generalised Beta Distribution of Second Kind.

Usage

```
sd.GB2(b, a, p, q)
```

Arguments

b	the parameter b of the Dagum distribution as defined by Kleiber and Kotz (2003).
a	the parameter a of the Dagum distribution as defined by Kleiber and Kotz (2003).
p	the parameter p of the Dagum distribution as defined by Kleiber and Kotz (2003).
q	the parameter q of the Dagum distribution as defined by Kleiber and Kotz (2003).

Value

returns the standard deviation.

Author(s)

Alexander Sohn

References

Kleiber, C. and Kotz, S. (2003): Statistical Size Distributions in Economics and Actuarial Sciences, Wiley, Hoboken.

Examples

```
a.test<- 4
b.test<- 20000
p.test<- 0.7
q.test<- 1
alpha.test<-1
GB2sample<-rGB2(10000,b.test,a.test,p.test,q.test)
sd.GB2(b.test,a.test,p.test,q.test)
sd(GB2sample)
```

`sgini` *Single-parameter Gini Coefficient*

Description

This function computes the Single-parameter Gini coefficient (a.k.a. generalised Gini coefficient or extended Gini coefficient) for a vector of observations.

Usage

```
sgini(x, nu = 2, w = NULL)
```

Arguments

`x`	a vector of observations.
`nu`	a scalar entailing the parameter that tunes the degree of the policy maker's aversion to inequality. See Yaari, 1988 for details.
`w`	a vector of weights.

Value

`Gini`	the Gini coefficient for the sample.
`bcGini`	the bias-corrected Gini coefficient for the sample.

Author(s)

Alexander Sohn

References

van Kerm, P. (2009): sgini - Generalized Gini and Concentration coefficients (with factor decomposition) in Stata', CEPS/INSTEAD, Differdange, Luxembourg.

Yaari, M.E. (1988): A Controversal Proposal Concerning Inequality Measurement, Journal of Economic Theory, Vol. 44, pp. 381-397.

Examples

```
set.seed(123)
x <- rnorm(100,10,1)
gini(x)$Gini
sgini(x,nu=2)$Gini
```

`sgini.den` *Single-parameter Gini Coefficient for an Income Distribution*

Description

This function approximates the Single-parameter Gini coefficient for a distribution specified by a vector of densities and a corresponding income vector. A point mass at zero is allowed.

Usage

```
sgini.den(incs, dens, nu = 2, pm0 = NA, lower = NULL, upper = NULL)
```

Arguments

`incs`	a vector with sorted income values.
`dens`	a vector with the corresponding densities.
`nu`	a scalar entailing the parameter that tunes the degree of the policy maker's aversion to inequality. See Yaari, 1988 for details.
`pm0`	the point mass for zero incomes. If not specified no point mass is assumed.
`lower`	the lower bound of the income range considered.
`upper`	the upper bound of the income range considered.

Value

`Gini`	the approximation of the Gini coefficient.
`pm0`	the point mass for zero incomes used.
`lower`	the lower bound of the income range considered used.
`upper`	the upper bound of the income range considered used.

Author(s)

Alexander Sohn

References

van Kerm, P. (2009): sgini - Generalized Gini and Concentration coefficients (with factor decomposition) in Stata', CEPS/INSTEAD, Differdange, Luxembourg.

Yaari, M.E. (1988): A Controversal Proposal Concerning Inequality Measurement, Journal of Economic Theory, Vol. 44, pp. 381-397.

See Also

gini

Examples

```
## without point mass
set.seed(123)
x <- rnorm(1000,10,1)
incs <- seq(1,20,length.out=1000)
dens <- dnorm(incs,10,1)
lower=NULL;upper=NULL;pm0<-NA
gini(x)$Gini
sgini(x,nu=2)$Gini
sgini.den(incs,dens)$Gini


## with point mass
set.seed(123)
x <- c(rep(0,1000),rnorm(1000,10,1))
incs <- c(0,seq(1,20,length.out=1000))
dens <- c(0.5,dnorm(incs[-1],10,1)/2)
gini(x)$Gini
sgini(x,nu=2)$Gini
sgini.den(incs,dens,pm = 0.5)$Gini
```

`skewness.GB2` *Skewness of the Generalised Beta Distribution of Second Kind*

Description

This function calculates the skewness of the Generalised Beta Distribution of Second Kind.

Usage

```
skewness.GB2(b, a, p, q)
```

Arguments

b	the parameter b of the Dagum distribution as defined by Kleiber and Kotz (2003).
a	the parameter a of the Dagum distribution as defined by Kleiber and Kotz (2003).
p	the parameter p of the Dagum distribution as defined by Kleiber and Kotz (2003).
q	the parameter q of the Dagum distribution as defined by Kleiber and Kotz (2003).

Value

returns the skewness.

Author(s)

Alexander Sohn

References

Kleiber, C. and Kotz, S. (2003): Statistical Size Distributions in Economics and Actuarial Sciences, Wiley, Hoboken.

Examples

```
a.test<- 4
b.test<- 20000
p.test<- 0.7
q.test<- 1
alpha.test<-1
GB2sample<-rGB2(10000,b.test,a.test,p.test,q.test)
skewness.GB2(b.test,a.test,p.test,q.test)
#require(e1071)
#skewness(GB2sample)#note that this estimation is highly unstable even for larger sample sizes.
```

---

theil.gamma *Theil Index for the Gamma Distribution*

Description

This function computes the Theil index for the gamma distribution.

Usage

```
theil.gamma(p)
```

Arguments

p	the shape parameter p of the gamma distribution as defined by Kleiber and Kotz (2003).

Value

returns the Theil index.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

Kleiber, C. and Kotz, S. (2003): Statistical Size Distributions in Economics and Actuarial Sciences, Wiley, Hoboken.

See Also

entropy

Examples

```
shape.test <- 5
scale.test  <- 50000
y <- rgamma(10000,shape=shape.test,scale=scale.test)
entropy(y,1)
theil.gamma(shape.test)
```

---

weighted.atkinson *Atkinson Inequality Index*

Description

This function computes the Atkinson inequality index for a vector of observations with corresponding weights.

Usage

```
weighted.atkinson(x, w = NULL, epsilon = 1, wscale = 1000)
```

Arguments

x	a vector of observations.
w	a vector of weights. If
epsilon	inequality aversion parameter as denoted by Atkinson (1970). The default is epsilon=1.
wscale	a scale by which the weights are adjusted such that can be rounded to natural numbers.

Value

returns the selected Atkinson inequality index.

Author(s)

Alexander Sohn

References

Atkinson, A.B. (1970): On the Measurment of Inequality, in: Journal of Economic Theory, Vol. 2(3), pp. 244-263.

See Also

ineq

Examples

```
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
w <- sample(1:2,length(x),replace=TRUE)
weighted.atkinson(x,w)
```

---

weighted.coeffvar *Coefficient of Variation*

Description

This function computes the Coefficient of Variation for a vector of observations and corresponding weights.

Usage

```
weighted.coeffvar(x, w)
```

Arguments

x	a vector of observations.
w	a vector of weights.

Value

cv	returns the coefficient of variation without bias correction.
bccv	returns the coefficient of variation with bias correction.

Warning

Weighting is not properly accounted for in the sample adjustment of bccv!

Author(s)

Alexander Sohn

References

Atkinson, A.B. and Bourguignon, F. (2000): Income Distribution and Economics, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

See Also

ineq

Examples

```
# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
w <- sample(1:10,length(x), replace=TRUE)
weighted.coeffvar(x,w)
```

---

weighted.entropy *Measures of the Generalised Entropy Family*

Description

This function computes the Measures of the Generalised Entropy Family for a vector of observations with corresponding weights.

Usage

```
weighted.entropy(x, w = NULL, alpha = 1)
```

Arguments

x	a vector of observations.
w	a vector of weights.
alpha	the parameter for the generalised entropy family of measures, denoted by alpha by Cowell (2000). Note that this parameter notation differs from the notation used in the ineq package.

Value

returns the entropy measure.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

See Also

ineq

Examples

```
# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
w <- sample(1:2,length(x),replace=TRUE)
weighted.entropy(x,w)
```

---

weighted.gini *Gini Coefficient*

Description

This function computes the Gini coefficient for a vector of observations with corresponding weights.

Usage

```
weighted.gini(x, w = NULL)
```

Arguments

x	a vector of observations.
w	a vector of weights.

Value

returns the Gini coefficient.

Author(s)

Alexander Sohn

References

Cowell, F.A. (2000): Measurement of Inequality, in: Atkinson and Bourguignon (eds.), Handbook of Income Distribution, pp. 1-86, Elsevier, Amsterdam.

See Also

ineq

Examples

```
# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
w <- sample(1:2,length(x),replace=TRUE)
weighted.gini(x,w)
```

---

weighted.moments *Moments of a Random Variable*

Description

This functions calculates the first three moments as well as mean, standard deviation and skewness for a vector of observations with corresponding weights.

Usage

```
weighted.moments(x, w8 = NULL)
```

Arguments

x	a vector of observations.
w8	a vector of weights.

Value

fm	returns the first moment.
weighted.mean	returns the mean.
sm	returns the second moment.
weighted.sd	returns the uncorrected (population) standard deviation.
wtd.sd	returns the sample-size corrected standard deviation estimate.
tm	returns the third moment.
w.skew.SAS	returns the skewness estimate as implemented in SAS.
w.skew.Stata	returns the skewness estimate as implemented in Stata.

Author(s)

Alexander Sohn

See Also

wtd.var

ysample.md *Sampling Incomes from a Mixture of Income Distributions*

Description

This function samples incomes from a mixture of two continuous income distributions and a point mass for zero-incomes.

Usage

```
ysample.md(n, dist1, dist2, theta, p0, p1, p2, dist.para.table)
```

Arguments

n	number of observations.
dist1	character string with the name of the first continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
dist2	character string with the name of the second continuous distribution used. Must be listed in dist.para.table. Must be equivalent to the respective function of that distribution, e.g. norm for the normal distribution.
theta	vector with the parameters of dist1 and dist2. Order must be the same as in the functions for the distributions.
p0	scalar with probability mass for the point mass.
p1	scalar with probability mass for dist1.
p2	scalar with probability mass for dist2.
dist.para.table	a table of the same form as dist.para.t with distribution name, function name and number of parameters.

Value

returns the sample of observations.

Author(s)

Alexander Sohn

See Also

pval.md

Examples

```
data(dist.para.t)
ygrid<-seq(0,1e5,by=1000)
theta<-c(5,1,10,3)
p0<-0.2
p1<-0.3
p2<-0.5
n <-10
ysample.md(n, "LOGNO", "LOGNO", theta, p0, p1, p2, dist.para.t)
```

Index