Package: GB2 (via r-universe)

August 31, 2024

Version 2.1.1

Date 2015-05-01

Title Generalized Beta Distribution of the Second Kind: Properties, Likelihood, Estimation

Author Monique Graf <`monique.p.n.graf@bluewin.ch`>, Desislava Nedyalkova <`desislava.nedyalkova@gmail.com`>.

Maintainer Desislava Nedyalkova <`desislava.nedyalkova@gmail.com`>

Depends R (>= 3.1.0)

Imports cubature, hypergeo, laeken, numDeriv, stats, survey

Suggests simFrame

Description Package GB2 explores the Generalized Beta distribution of the second kind. Density, cumulative distribution function, quantiles and moments of the distributions are given. Functions for the full log-likelihood, the profile log-likelihood and the scores are provided. Formulas for various indicators of inequality and poverty under the GB2 are implemented. The GB2 is fitted by the methods of maximum pseudo-likelihood estimation using the full and profile log-likelihood, and non-linear least squares estimation of the model parameters. Various plots for the visualization and analysis of the results are provided. Variance estimation of the parameters is provided for the method of maximum pseudo-likelihood estimation. A mixture distribution based on the compounding property of the GB2 is presented (denoted as ``compound" in the documentation). This mixture distribution is based on the discretization of the distribution of the underlying random scale parameter. The discretization can be left or right tail. Density, cumulative distribution function, moments and quantiles for the mixture distribution are provided. The compound mixture distribution is fitted using the method of maximum pseudo-likelihood estimation. The fit can also incorporate the use of auxiliary information. In this new version of the package, the mixture case is complemented with new functions for variance estimation by linearization and comparative density plots.

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-22 05:53:16 UTC

Contents

Compound 2
CompoundAuxDensPlot 5
CompoundAuxFit 6
CompoundAuxVarest 9
CompoundDensPlot 12
CompoundFit 13
CompoundIndicators 15
CompoundMoments 17
CompoundQuantiles 18
CompoundVarest 20
Contindic 22
Contprof 24
Fisk 25
gb2 26
Gini 27
Indicators 28
LogDensity 30
LogLikelihood 31
MLfitGB2 32
MLfullGB2 34
MLprofGB2 36
Moments 37
NonlinearFit 39
PlotsML 41
ProfLogLikelihood 42
RobustWeights 43
Thomae 44
Varest 46

Index **51**

`Compound` *Compound Distribution based on the Generalized Beta Distribution of the Second Kind*

Description

Mixture distribution based on the compounding property of the GB2, in short "compound GB2". Decomposition of the GB2 distribution with respect to the left and right tail of the distribution. Calculation of the component densities and cumulative distribution functions. Calculation of the compound density function and the compound cumulative distribution function.

Usage

```
fg.cgb2(x, shape1, scale, shape2, shape3, pl0, decomp="r")
dl.cgb2(x, shape1, scale, shape2, shape3, pl0, decomp="r")
pl.cgb2(y, shape1, scale, shape2, shape3, pl0, decomp="r", tol=1e-05)
dcgb2(x, shape1, scale, shape2, shape3, pl0, pl, decomp="r")
pcgb2(y, shape1, scale, shape2, shape3, pl0, pl, decomp="r")
prcgb2(y1, y2, shape1, scale, shape2, shape3, pl0, pl, decomp="r", tol=1e-08,
debug=FALSE)
```

Arguments

`x`	numeric; can be a vector. The value(s) at which the compound density and the component densities are calculated, `x` is positive.
`y`	numeric; can be a vector. The value(s) at which the compound distribution function and the component distribution functions are calculated.
`y1`, `y2`	numeric values.
`shape1`, `scale`, `shape2`, `shape3`	numeric; positive parameters of the GB2 distribution.
`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`pl`	numeric; a vector of fitted proportions. Sums to one. If `pl` is equal to `pl0`, we obtain the GB2 distribution.
`decomp`	string; specifying if the decomposition of the GB2 is done with respect to the right tail ("r") or the left tail ("l") of the distribution. By default, `decomp` = "r" - right tail decomposition.
`debug`	logical; By default, `debug` = `FALSE`.
`tol`	numeric; tolerance with default 0, meaning to iterate until additional terms do not change the partial sum.

Details

The number of components L is given by the length of the vector `pl0`. In our examples $L = 3$. Let N denote the length of the vector `x`. Function `fg.cgb2` calculates the L gamma factors which multiply the GB2 density in order to obtain the component density f_ℓ. These component densities are calculated using the function `dl.cgb2`. Function `pl.cgb2` calculates the corresponding L cumulative component distribution functions. Function `dcgb2` calculates the resulting compound density function. Function `pcgb2` calculates the compound cumulative distribution function for a vector of values `y` and function `prcgb2`, given 2 arguments `y1` and `y2`, calculates the probability $P(min(y1, y2) < Y < max(y1, y2))$, where the random variable Y follows a compound GB2 distribution.

Value

`fg.cgb2` returns a matrix of size $N \times L$ of the Gamma factors, `dl.cgb2` returns a matrix of size $N \times L$ of component densities, `pl.cgb2` returns a matrix containing the L component cdfs, `dcgb2` returns a matrix of size $N \times 1$ of the GB2 compound density function, `pcgb2` returns a matrix of size $N \times 1$ of the GB2 compound distribution function and `prcgb2` returns a probability between 0 and 1.

Author(s)

Monique Graf and Desislava Nedyalkova

References

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

Examples

```
#\dontrun{

#\library{cubature}

# GB2 parameters
af <- 5
bf <- 20000
pf <- 0.45
qf <- 0.75

p0 <- rep(1/3,3)
p1 <- c(0.37,0.43,0.2)

# a vector of values
x <- rep(20000*seq(1,2,length.out=9))

#Gamma components
fg.cgb2(20000,af,bf,pf,qf,p0)
fg.cgb2(Inf,af,bf,pf,qf,p0,"l")

#Component densities
dl.cgb2(x,af,bf,pf,qf,p0)
dl.cgb2(20000,af,bf,pf,qf,p0,"l")

#Component cdf
pl.cgb2(25000,af,bf,pf,qf,p0)

#Compound cdf
pcgb2(x,af,bf,pf,qf,p0,p1)
prcgb2(37000,38000,af,bf,pf,qf,p0,p1,"l")
#}
```

`CompoundAuxDensPlot` *Comparison of the compound GB2 and kernel densities by group*

Description

Function `dplot.cavgb2` produces a plot in which the compound and kernel (Epanechnikov) densities are plotted by group.

Usage

```
dplot.cavgb2(group, x, shape1, scale, shape2, shape3, pl0, pl, w=rep(1,length(x)),
xmax = max(x)*(2/3), ymax=2e-05, decomp="r", choicecol=1:length(levels(group)),
xlab="")
```

Arguments

`group`	numeric; a factor variable giving the group membership of each sampled unit.
`x`	numeric; can be a vector. The value(s) at which the density is calculated, used for the kernel estimate only. `x` is positive.
`shape1`, `scale`, `shape2`, `shape3`	numeric; positive parameters of the GB2 distribution. On the plot they are denotes as `a`, `b`, `p`, `q` and `pl0` respectively.
`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`pl`	numeric; a vector of fitted proportions (output of `pkl.cavgb2`). Sums to one. If `pl` is equal to `pl0`, we obtain the GB2 distribution.
`w`	numeric; weights.
`xmax`	numeric; scale on the horizontal axis. By default is equal to $max(x) * (2/3)$.
`ymax`	numeric; scale on the vertical axis. By default is equal to 2e-05.
`decomp`	string; specifying if the decomposition of the GB2 is done with respect to the right tail ("r") or the left tail ("l") of the distribution. By default, `decomp` = "r" - right tail decomposition.
`choicecol`	numeric vector of length the number of groups; defines the color with which the density curves will be plotted for each group.
`xlab`	string; label for x. The default is " ".

Details

The legend is placed interactively.

Value

`dplot.cavgb2` plots a graph with two curves - the GB2 density, the compound GB2 per group and the corresponding kernel estimate.

Author(s)

Monique Graf and Desislava Nedyalkova

`CompoundAuxFit` *Fitting the Compound Distribution based on the GB2 by the Method of Pseudo Maximum Likelihood Estimation using Auxiliary Information*

Description

Calculates the log-likelihood, the score functions of the log-likelihood and fits the compound distribution based on the GB2 and using auxiliary information.

Usage

```
pkl.cavgb2(z, lambda)
lambda0.cavgb2(pl0, z, w=rep(1, dim(z)[1]))
logl.cavgb2(fac, z, lambda, w=rep(1, dim(fac)[1]))
scores.cavgb2(fac, z, lambda, w=rep(1, dim(fac)[1]))
ml.cavgb2(fac, z, lambda0, w = rep(1, dim(fac)[1]), maxiter = 100, fnscale=length(w))
```

Arguments

`z`	numeric; a matrix of auxiliary variables.
`lambda`	numeric; a matrix of parameters.
`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`w`	numeric; vector of weights of length the number of rows of the matrix `fac`. By default `w` is a vector of 1.
`fac`	numeric; a matrix of Gamma factors.
`lambda0`	numeric; a matrix of initial parameters.
`maxiter`	numeric; maximum number of iterations to perform. By default `maxiter` = 100.
`fnscale`	numeric; parameter of the `optim` function. By default `fnscale` is equal to the lenth of the vector of weights (value of `fnscale` in the preceding version of the package). Permits to solve some convergence problems (see `optim`).

Details

We model the probabilities p_ℓ with auxiliary variables. Let z_k denote the vector of auxiliary information for unit k. This auxiliary information modifies the probabilities p_ℓ at the unit level. Denote by $p_{k,\ell}$ the weight of the density f_ℓ for unit k. For $\ell = 1, ..., L-1$, we pose a linear model for the log-ratio $v_{k,\ell}$:

$$\log(p_{k,\ell}/p_{k,L}) = v_{k,\ell} = \sum_{i=1}^{I} \lambda_{\ell i} z_{ki} = \mathbf{z}_k^T \boldsymbol{\lambda}_{\boldsymbol{\ell}}.$$

Function `pkl.cavgb2` calculates the $p_{k,\ell}$. Function `lambda0.cavgb2` calculates the initial values $\lambda_{\ell i}$, $i = 1, ..., I$, $\ell = 1, ..., L-1$. Let

$$\bar{z}_i = \sum_k w_k z_{ki} / \sum_k w_k$$

be the mean value of the i-th explanatory variable. Writing

$$\log(\hat{p}_\ell^{(0)} / \hat{p}_L^{(0)}) = v_\ell^{(0)} = \sum_{i=1}^{I} \lambda_{\ell i}^{(0)} \bar{z}_i,$$

we can choose $\lambda_{\ell i}^{(0)} = v_\ell^{(0)} / (I \bar{z}_i)$. Analogically to the ordinary fit of the compound distribution based on the GB2 `CompoundFit`, we express the log-likelihood as a weighted mean of $log(f) = log(\sum(p_{k,\ell} f_\ell(x_k))$, evaluated at the data points, where f is the GB2 compound density. The scores are obtained as the weighted sums of the first derivatives of the log-likelihood, with respect to the parameters $\lambda_\ell,\ \ell = 1, ..., L-1$, evaluated at the data points. Function `ml.cavgb2` performs maximum likelihood estimation through the general-purpose optimization function `optim` from package `stats`. The considered method of optimization is "BFGS" (`optim`). Once we have the fitted parameters $\hat{\lambda}$ we can deduce the fitted parameters $v\hat{k}\ell$ and $\hat{p_{k\ell}}$ in function of $\bar{z}$ and $\hat{\lambda}_\ell$.

Value

`pkl.cavgb2` returns a matrix of probabilities. `lambda0.cavgb2` returns a matrix of size $I \times L - 1$. `logl.cavgb2` returns the value of the pseudo log-likelihood. `scores.cavgb2` returns the weighted sum of the scores of the log-likelihood. `ml.cavgb2` returns a list containing two objects - the vector of fitted coefficients $\hat{\lambda_\ell}$ and the output of the "BFGS" fit.

Author(s)

Monique Graf and Desislava Nedyalkova

See Also

`optim`

Examples

```
## Not run:

library(simFrame)
data(eusilcP)

# Stratified cluster sampling
set.seed(12345)
srss <- SampleControl(design = "region", grouping = "hid", size = c(200*3, 1095*3, 1390*3,
 425*3, 820*3, 990*3, 400*3, 450*3, 230*3), k = 1)

# Draw a sample
s1 <- draw(eusilcP,srss)
#names(s1)
```

```
# Creation of auxiliary variables
ind <- order(s1[["hid"]])
ss1 <- data.frame(hid=s1[["hid"]], region=s1[["region"]], hsize=s1[["hsize"]],
peqInc=s1[["eqIncome"]], age=s1[["age"]], pw=s1[[".weight"]])[ind,]
ss1[["child"]] <- as.numeric((ss1[["age"]]<=14))
ss1[["adult"]] <- as.numeric((ss1[["age"]]>=20))
sa <- aggregate(ss1[,c("child","adult")],list(ss1[["hid"]]),sum)
names(sa)[1] <- "hid"
sa[["children"]] <- as.numeric((sa[["child"]]>0))
sa[["single_a"]] <- as.numeric((sa[["adult"]]==1))
sa[["sa.ch"]] <- sa[["single_a"]]*sa[["children"]]
sa[["ma.ch"]] <- (1-sa[["single_a"]])*sa[["children"]]
sa[["nochild"]] <- 1-sa[["children"]]

# New data set
ns <- merge(ss1[,c("hid","region","hsize","peqInc","pw")],
sa[,c("hid","nochild","sa.ch","ma.ch")], by="hid")

# Ordering the data set
ns <- ns[!is.na(ns$peqInc),]
index <- order(ns$peqInc)
ns <- ns[index,]

#  Truncate at 0
ns <- ns[ns$peqInc>0,]
# income
peqInc <- ns$peqInc
# weights
pw <- ns$pw

# Adding the weight adjustment
c1 <- 0.1
pwa <- robwts(peqInc,pw,c1,0.001)[[1]]
corr <- mean(pw)/mean(pwa)
pwa <- pwa*corr

ns <- data.frame(ns, aw=pwa)

# Empirical indicators with original weights
emp.ind <- c(main.emp(peqInc, pw),
              main.emp(peqInc[ns[["nochild"]]==1], pw[ns[["nochild"]]==1]),
              main.emp(peqInc[ns[["sa.ch"]]==1], pw[ns[["sa.ch"]]==1]),
              main.emp(peqInc[ns[["ma.ch"]]==1], pw[ns[["ma.ch"]]==1]))

# Matrix of auxiliary variables
z <- ns[,c("nochild","sa.ch","ma.ch")]
#unique(z)
z <- as.matrix(z)

# global GB2 fit, ML profile log-likelihood
gl.fit <- profml.gb2(peqInc,pwa)$opt1
agl.fit <- gl.fit$par[1]
bgl.fit <- gl.fit$par[2]
```

```
pgl.fit <- prof.gb2(peqInc,agl.fit,bgl.fit,pwa)[3]
qgl.fit <- prof.gb2(peqInc,agl.fit,bgl.fit,pwa)[4]

# Likelihood and convergence
proflikgl <- -gl.fit$value
convgl <- gl.fit$convergence

# Fitted GB2 parameters and indicators
profgb2.par <- c(agl.fit, bgl.fit, pgl.fit, qgl.fit)
profgb2.ind <- main.gb2(0.6, agl.fit, bgl.fit, pgl.fit, qgl.fit)

# Initial lambda and pl
pl0 <- c(0.2,0.6,0.2)
lambda0 <- lambda0.cavgb2(pl0, z, pwa)

# left decomposition
decomp <- "l"
facgl <- fg.cgb2(peqInc, agl.fit, bgl.fit, pgl.fit, qgl.fit, pl0 ,decomp)
fitcml <- ml.cavgb2(facgl, z, lambda0, pwa, maxiter=500)
fitcml
convcl <- fitcml[[2]]$convergence
convcl
lambdafitl <- fitcml[[1]]
pglfitl <-  pkl.cavgb2(diag(rep(1,3),lambdafitl)
row.names(pglfitl) <- colnames(z)

## End(Not run)
```

`CompoundAuxVarest` *Variance Estimation under the Compound GB2 Distribution Using Auxiliary Information*

Description

Calculation of variance estimates of the parameters of the compound GB2 distribution and of the estimated compound GB2 indicators under a complex survey design (see package `survey`).

Usage

```
scoreU.cavgb2(fac, z, lambda)
scorez.cavgb2(U,z)
varscore.cavgb2(SC, w=rep(1,dim(SC)[1]))
desvar.cavgb2(data=data, SC=SC, ids=NULL, probs=NULL, strata = NULL, variables = NULL,
fpc=NULL, nest = FALSE, check.strata = !nest, weights=NULL, pps=FALSE,
variance=c("HT","YG"))
hess.cavgb2(U, P, z, w=rep(1, dim(z)[1]))
vepar.cavgb2(ml, Vsc, hess)
veind.cavgb2(group, vepar, shape1, scale, shape2, shape3, pl0, P, decomp="r")
```

Arguments

`fac`	numeric; a matrix of Gamma factors.
`z`	numeric; a matrix of auxiliary variables.
`lambda`	numeric; a matrix of parameters.
`U`	numeric; a matrix of scores $U_{k,\ell}$ (output of the `scoreU.cavgb2` function).
`SC`	numeric; scores, output of `scorez.cavgb2`.
`w`	numeric; vector of extrapolation weights. By default `w` is a vector of 1.
`data`	dataset containing the design information per unit
`ids, probs, strata, variables, fpc, nest, check.strata, weights, pps, variance`	parameters of `svydesign`.
`P`	numeric; matrix of mixture probabilities (output of `pkl.cavgb2`).
`ml`	numeric; estimated values of the vector of v's. Output of the `ml.cavgb2` function (the second element in the list).
`Vsc`	numeric; 4 by 4 matrix. Variance of the scores `SC`, computed in `varscore.cavgb2` or with the design information in `desvar.cavgb2`.
`hess`	numeric; Hessian (bread) for the sandwich variance estimate (output of `hess.cavgb2`).
`group`	numeric; a factor variable of the same length as the sample size giving the group membership in the special case when the auxiliary information defines group membership.
`vepar`	numeric; output of `vepar.cavgb2`.
`shape1, scale, shape2, shape3`	numeric; positive parameters of the GB2 distribution.
`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`decomp`	string; specifying if the decomposition of the GB2 is done with respect to the right tail ("r") or the left tail ("l") of the distribution.

Details

The $N \times L$ matrix of fitted mixture probabilities `P`$= (p_{k,\ell})$ depends on the $N \times I$ matrix `z` of auxiliary variables. `P` has as many distinct rows as there are distinct rows in `z`. The $N \times L$ matrix of gamma factors `fac`$= F$, output of `fg.cgb2` depends on the vector of initial probabilities $p_{0,\ell}$ only. The $N \times (L-1)$ matrix of scores `U` is defined as

$$U(k,\ell) = p_{k,\ell}\left(\frac{F(k,\ell)}{\sum_{j=1}^{L} p_{k,j}\, F(k,j)} - 1\right).$$

The linearized scores are the columns of a $N \times I(L-1)$ matrix

$$SC(k,\ I(\ell-1)+i) = U(k,\ell)\, z(k,i).$$

Function `varscore.cavgb2` calculates the middle term of the sandwich variance estimator, that is the $(I(L-1) \times I(L-1))$ estimated variance-covariance matrix of the $I(L-1)$ weighted sums of the columns of SC, without design information. `desvar.cavgb2` calculates the design-based variance-covariance matrix of the $I(L-1)$ weighted sums of the columns of SC, invoking `svydesign` and `svytotal` of package `survey`. `hess.cavgb2` calculates the Hessian ($I(L-1) \times I(L-1)$ matrix of second derivatives of the pseudo-log-likelihood with respect to the parameters). It should be negative definite. If not, the maximum likelihood estimates are spurious. `vepar.cavgb2` calculates the sandwich variance estimate of the vectorized matrix of parameters `lambda`. `veind.cavgb2` calculates estimates, std error, covariance and correlation matrices of the indicators under the compound GB2 with auxiliary variables in the particular case where the unique combinations of the auxiliary variables define a small number of groups. Group membership is specified by the vector `group` of length N.

Value

`scoreU.cavgb2` returns a $N \times (L-1)$ matrix of scores `U`. `scorez.cavgb2` returns a $N \times I(L-1)$ matrix whose columns are the linearized scores `SC`. `varscore.cavgb2` returns the variance-covariance estimate of the weighted sums of scores `SC`, given by weighted cross products. `desvar.cavgb2` returns a list of two elements. The first is the output of svytotal and the second is the design-based variance-covariance matrix of the weighted sums of the scores SC. `hess.cavgb2` returns the matrix of second derivatives of the likelihood with respect to the parameters (bread for the sandwich variance estimate). `vepar.cavgb2` returns a list of five elements - [["type"]] with value "parameter", [["estimate"]] estimated parameters, [["stderr"]] corresponding standard errors, [["Vcov"]] variance -covariance matrix and [["Vcor"]] - correlation matrix. `veind.cavgb2` returns a list of five elements: [["type"]] with value "indicator", followed by a list with as many arguments as `length(levels(group))`. Each argument is itself a list with 5 arguments: [["group"]] group name, [["estimate"]] estimated indicators under the compound GB2, [["stderr"]] corresponding standard errors, [["Vcov"]] variance -covariance matrix and [["Vcor"]] - correlation matrix.

Author(s)

Monique Graf and Desislava Nedyalkova

References

Davison, A. (2003), *Statistical Models*. Cambridge University Press.

Freedman, D. A. (2006), On The So-Called "Huber Sandwich Estimator" and "Robust Standard Errors". *The American Statistician*, **60**, 299–302.

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

Pfeffermann, D. and Sverchkov, M. Yu. (2003), Fitting Generalized Linear Models under Informative Sampling. In, Skinner, C.J. and Chambers, R.L. (eds.). *Analysis of Survey Data*, chapter 12, 175–195. Wiley, New York.

Examples

```
## Not run:
# Example (following of example in CompoundAuxFit)

# Scores U
```

```
U <- scoreU.cavgb2(facgl, z, lambdafitl)

# Scores multiplied by z
SC <- scorez.cavgb2(U,z)

# Naive variance estimate of sum of scores
(Vsc <- varscore.cavgb2(SC,w=pwa))

# Design based variance of sum of scores
(desv <- desvar.cavgb2(data=ns,SC=SC,id=~hid,strata=~region,weights=~pwa))

# Hessian
hess <- hess.cavgb2(U,pglfitl,z,w=pwa)

# 1. Sandwich variance-covariance matrix estimate of parameters using Vsc:
Param1 <- vepar.cavgb2(fitcml,Vsc, hess)
Param1

# 2. Sandwich variance-covariance matrix estimate of parameters using
# the design variance:
Param2 <- vepar.cavgb2(fitcml,desv$Vtheta, hess)
Param2

# 3. Indicators and conditional variances : takes a long time!
(Indic <- veind.cavgb2(group,Param2 ,agl.fit,bgl.fit,pgl.fit,qgl.fit,
                       pl0, pglfitl, decomp="l") )

## End(Not run)
```

`CompoundDensPlot` *Comparison of the GB2, compound GB2 and kernel densities*

Description

Function `dplot.cgb2` produces a plot in which the three densities are plotted.

Usage

```
dplot.cgb2(x,shape1, scale, shape2, shape3, pl0, pl, w=rep(1,length(x)), decomp="r",
xmax = max(x)*(2/3), choicecol=1:3, kernel="epanechnikov", adjust=1, title=NULL,
ylim=NULL)
```

Arguments

`x`	numeric; can be a vector. The value(s) at which the density is calculated, used for the kernel estimate only. `x` is positive.
`shape1, scale, shape2, shape3`	numeric; positive parameters of the GB2 distribution. On the plot they are denotes as `a`, codeb, `p`, `q` and `pl0` respectively.

`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`pl`	numeric; a vector of mixture probabilities (output of `ml.cgb2`). Sums to one. If `pl` is equal to `pl0`, we obtain the GB2 distribution.
`w`	numeric; weights.
`decomp`	string; specifying if the decomposition of the GB2 is done with respect to the right tail ("r") or the left tail ("l") of the distribution. By default, `decomp` = "r" - right tail decomposition.
`xmax`	numeric; maximum `x` value to be plotted.
`choicecol`	numeric vector of length 3; defines the color with which the density curves will be plotted.
`adjust`	numeric; graphical parameter of the generic function `density`.
`title`	string; title of the plot. By default is equall to NULL (no title).
`ylim`	string; scaling of parameters. By default is equall to NULL (automatic scaling).
`kernel`	string; the kernel used for the kernel density estimate. The default value is "Epanechnikov" (see `plot.density`).

Details

The legend is placed interactively.

Value

`dplot.cgb2` plots a graph with three curves - the GB2 density, the compound GB2 density and the corresponding kernel estimate

Author(s)

Monique Graf and Desislava Nedyalkova

`CompoundFit` *Fitting the Compound Distribution based on the GB2 by the Method of Maximum Likelihood Estimation*

Description

Calculates the log-likelihood, the score functions of the log-likelihood, the weighted mean of scores, and fits the parameters of the Compound Distribution based on the GB2.

Usage

```
vofp.cgb2(pl)
pofv.cgb2(vl)
logl.cgb2(fac, pl, w=rep(1, dim(fac)[1]))
scores.cgb2(fac, pl, w=rep(1, dim(fac)[1]))
ml.cgb2(fac, pl0, w=rep(1, dim(fac)[1]), maxiter=100, fnscale=length(w))
```

Arguments

`pl0`	numeric; vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`pl`	numeric; vector of fitted proportions. Sums to one. If `pl` is equal to `pl0`, we obtain the GB2 distribution.
`fac`	numeric; matrix of Gamma factors (output of `fac.cgb2`.
`vl`	numeric; vector of parameters. Its length is equal to the length of `pl` - 1.
`w`	numeric; vector of weights of length the number of rows of the matrix `fac`. By default `w` is a vector of 1.
`maxiter`	numeric; maximum number of iterations to perform. By default `maxiter` = 100.
`fnscale`	numeric; an overall scaling parameter used in the function `optim`. By default it is equal to the length of the vector of weights `w`.

Details

There are only $L - 1$ parameters to estimate, because the probabilities p_ℓ sum to 1 (L is the dimension of the vector of probailities p_ℓ). Knowing this, we change the parameters p_ℓ to $v_\ell = log(p_\ell/p_L)$, $\ell = 1, ..., L-1$. This calculation is done through the function `vofp.cgb2`. `pofv.cgb2` calculates the p_ℓ in function of the given v_ℓ. We express the log-likelihood as a weighted mean of $log(f) = log(\sum(p_\ell f_\ell)$, evaluated at the data points, where f is the GB2 compound density. If the weights are not available, then we suppose that `w` $= 1$. Analogically, the scores are obtained as weighted sums of the first derivatives of the log-likelihood, with respect to the parameters v_ℓ, $\ell = 1, ..., L-1$, evaluated at the data points. Function `ml.cgb2` performs maximum likelihood estimation through the general-purpose optimization function `optim` from package `stats`. The considered method of optimization is BFGS.

Value

`vofp.cgb2` returns a vector of length $L - 1$, where L is the length of the vector p_ℓ. `pofv.cgb2` returns a vector of length ℓ. `logl.cgb2` returns the value of the pseudo log-likelihood. `scores.cgb2` returns a vector of the weighted mean of the scores of length $L-1$. `ml.cgb2` returns a list containing two objects - the vector of fitted proportions $\hat{p_\ell}$ and the output of the BFGS fit.

Author(s)

Monique Graf and Desislava Nedyalkova

See Also

`optim`

Examples

```
## Not run:
# GB2 parameters:
a <- 4
b <- 1950
p <- 0.8
```

```
q <- 0.6

# Proportions defining the component densities:
pl0 <- rep(1/3,3)

# Mixture probabilities
pl <- c(0.1,0.8,0.1)

# Random generation:
n <- 10000
set.seed(12345)
x <- rcgb2(n,a,b,p,q,pl0,pl,decomp="l")

# Factors in component densities
fac <- fg.cgb2(x,a,b,p,q, pl0,decomp="l")

# Estimate the mixture probabilities:
estim <- ml.cgb2(fac,pl0)

# estimated mixture probabilities:
estim[[1]]
#[1] 0.09724319 0.78415797 0.11859883

## End(Not run)
```

`CompoundIndicators` *Indicators of Poverty and Social Exclusion under the Compound Distribution based on the GB2*

Description

Functions to calculate four primary social welfare indicators under the compound GB2 distribution, i.e. the at-risk-of-poverty threshold, the at-risk-of-poverty rate, the relative median at-risk-of-poverty gap, and the income quintile share ratio.

Usage

```
arpt.cgb2(prop, shape1, scale, shape2, shape3, pl0, pl, decomp="r")
arpr.cgb2(prop, shape1, shape2, shape3, pl0, pl, decomp="r")
rmpg.cgb2(arpr, shape1, shape2, shape3, pl0, pl, decomp="r")
qsr.cgb2(shape1, shape2, shape3, pl0, pl, decomp="r")
main.cgb2(prop, shape1, scale, shape2, shape3, pl0, pl, decomp="r")
```

Arguments

`prop`	numeric; proportion (in general is set to 0.6).
`arpr`	numeric; the value of the at-risk-of-poverty rate.
`shape1, scale, shape2, shape3`	numeric; positive parameters of the GB2 distribution.

`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`pl`	numeric; a vector of mixture probabilities. Sums to one. If $pl = pl0$ we obtain the GB2 distribution.
`decomp`	string; specifying if the decomposition of the GB2 is done with respect to the right tail ("r") or the left tail ("l") of the distribution. By default, `decomp` = "r" - right tail decomposition.

Details

The four indicators are described in details in the case of the GB2. The difference here is that we need to give an initial vector of proportions, fitted proportions and define for which decomposition (left or right) the indicators should be calculated.

Value

`arpt.cgb2` gives the ARPT, `arpr.cgb2` the ARPR, `rmpg.cgb2` the RMPG, `qsr.cgb2` gives the QSR and `main.cgb2` calculates the median, the mean, the ARPR, the RMPG and the QSR under the compound GB2.

Author(s)

Monique Graf

References

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

See Also

`arpr.gb2` for details on the welfare indicators under the GB2.

Examples

```
# GB2 parameters
a <- 3.9
b <- 18873
p <- 0.97
q <- 1.03

# Proportions defining the component densities
p0 <- rep(1/3,3)

# Mixture probabilities
pl <- c(0.39,0.26,0.35)

# for the right discretization
arpt <- arpt.cgb2(0.6, a, b, p, q, p0, pl)
```

```
arpr <- arpr.cgb2(0.6, a, p, q, p0, pl)
rmpg <- rmpg.cgb2(arpr, a, p, q, p0, pl)
qsr <- qsr.cgb2(a, p, q, p0, pl)

# for the left discretization
arptleft <- arpt.cgb2(0.6, a, b, p, q, p0, pl, "l")
```

`CompoundMoments` *Moments of the Compound Distribution based on the GB2*

Description

These functions calculate the moment of order `k` and incomplete moment of order `k` of a GB2 compound random variable X as well as the moment of order `k` for each component density.

Usage

```
mkl.cgb2(k, shape1, scale, shape2, shape3, pl0, decomp="r")
moment.cgb2(k, shape1, scale, shape2 ,shape3, pl0, pl, decomp="r")
incompl.cgb2(x, k, shape1, scale, shape2, shape3, pl0, pl, decomp="r")
```

Arguments

`x`	numeric; vector of quantiles.
`k`	numeric; order of the moment.
`shape1`, `scale`, `shape2`, `shape3`	numeric; positive parameters of the GB2 distribution.
`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`pl`	numeric; a vector of mixture probabilities. Sums to one. If $pl = pl0$ we obtain the GB2 distribution.
`decomp`	string; specifying if the decomposition of the GB2 is done with respect to the right tail ("r") or the left tail ("l") of the distribution. By default, `decomp` = "r" - right tail decomposition.

Value

`mkl.cgb2` returns a vector of the moments of the component densities, `moment.cgb2` returns the moment of order `k` and `incompl.cgb2` - the incomplete moment of order `k`.

Author(s)

Monique Graf

References

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

Examples

```
#\dontrun{

#\library{cubature}

# GB2 parameters
af <- 5
bf <- 20000
pf <- 0.45
qf <- 0.75

p0 <- rep(1/3,3)
p1 <- c(0.37,0.43,0.2)

# moments for the component densities
mkl.cgb2(1,af,bf,pf,qf,p0)
mkl.cgb2(-1,af,bf,pf,qf,p0,"l")

#Moment of order k
moment.cgb2(0.5,af,bf,pf,qf,p0,p1)
moment.cgb2(0.5,af,bf,pf,qf,p0,p1,"l")

#Incomplete moment of order k
incompl.cgb2(20000,1,af,bf,pf,qf,p0,p1)
incompl.cgb2(20000,1,af,bf,pf,qf,p0,p1,"l")
#}
```

`CompoundQuantiles` *Quantiles and random generation of the Compound Distribution based on the GB2*

Description

Calculation of the quantiles of a compound GB2 random variable. Random generation of compound GB2 variables.

Usage

```
qcgb2(prob, shape1, scale, shape2, shape3, pl0, pl, decomp="r", tol=1e-08, ff=1.5,
debug=FALSE, maxiter=50)
rcgb2(n, shape1, scale, shape2, shape3, pl0, pl, decomp="r", tol=1e-02, maxiter=100,
debug = FALSE)
```

Arguments

`prob`	numeric; vector of probabilities between 0 and 1.
`shape1`, `scale`, `shape2`, `shape3`	numeric; positive parameters of the GB2 distribution.
`n`	numeric; number of observations. If `length(n) > 1`, the length is taken to be the number required.
`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`pl`	numeric; a vector of mixture probabilities. Sums to one. If $pl = pl0$ we obtain the GB2 distribution.
`decomp`	string; specifying if the decomposition of the GB2 is done with respect to the right tail ("r") or the left tail ("l") of the distribution. By default, `decomp` = "r" - right tail decomposition.
`ff`	numeric; a tuning parameter.
`debug`	logical; By default, `debug = FALSE`.
`maxiter`	numeric; maximum number of iterations to perform.
`tol`	numeric; tolerance with default 0, meaning to iterate until additional terms do not change the partial sum.

Value

`qcgb2` returns a vector of quantiles and `rcgb2` return a vector of size `n` of GB2 compound random deviates.

Author(s)

Monique Graf and Desislava Nedyalkova

References

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

Examples

```
#\dontrun{

#\library{cubature}

# GB2 parameters
af <- 5
bf <- 20000
pf <- 0.45
qf <- 0.75

p0 <- rep(1/3,3)
```

```
p1 <- c(0.37,0.43,0.2)

#Quantiles

qcgb2(0.5,af,bf,pf,qf,p0,p1)
qcgb2(1,af,bf,pf,qf,p0,p1)
qcgb2(c(0.5,0.8),af,bf,pf,qf,p0,p1)

#Random generation
rcgb2(10,af,bf,pf,qf,p0,p1)

#}
```

`CompoundVarest` *Variance Estimation of the Compound GB2 Distribution*

Description

Calculation of variance estimates of the parameters of the compound GB2 distribution and of the estimated compound GB2 indicators under cluster sampling.

Usage

```
scoreU.cgb2(fac, pl)
varscore.cgb2(U, w=rep(1,dim(U)[1]))
desvar.cgb2(data=data, U=U, ids=NULL, probs=NULL, strata = NULL, variables = NULL,
fpc=NULL, nest = FALSE, check.strata = !nest, weights=NULL, pps=FALSE,
variance=c("HT","YG"))
hess.cgb2(U, pl, w=rep(1,dim(U)[1]))
vepar.cgb2(ml, Vsc, hess)
derivind.cgb2(shape1, scale, shape2, shape3, pl0, pl, prop=0.6, decomp="r")
veind.cgb2(Vpar, shape1, scale, shape2, shape3, pl0, pl, decomp="r")
```

Arguments

`fac`	numeric; matrix of Gamma factors (output of `fac.cgb2`.
`pl`	numeric; a vector of fitted mixture probabilities. Sums to one. If `pl` is equal to `pl0`, we obtain the GB2 distribution.
`U`	numeric; vector of scores. Output of the `scoreU.cgb2` function.
`w`	numeric; vector of some extrapolation weights. By default `w` is a vector of 1.
`data`	dataset containing the design information per unit.
`ids, probs, strata, variables, fpc, nest, check.strata, weights, pps, variance`	parameters of `svydesign`.
`ml`	numeric; output of the `ml.cgb2` function. A list with two components. First component: estimated mixture probabilities. Second component: list containing the output of `optim`.

`Vsc`	numeric; 4 by 4 matrix.
`hess`	numeric; Hessian (bread) for the sandwich variance estimate.
`shape1`, `scale`, `shape2`, `shape3`	numeric; positive parameters of the GB2 distribution.
`pl0`	numeric; a vector of initial proportions defining the number of components and the weight of each component density in the decomposition. Sums to one.
`prop`	numeric; proportion (in general is set to 0.6).
`decomp`	string; specifying if the decomposition of the GB2 is done with respect to the right tail ("r") or the left tail ("l") of the distribution.
`Vpar`	numeric; 4 by 4 matrix. Output of the function `vepar.cgb2`.

Details

Function `scoreU.cgb2` calculates the $N \times (L-1)$ matrix of scores `U` is defined as

$$U(k,\ell) = p_\ell \left(\frac{F(k,\ell)}{\sum_{j=1}^{L} p_j \, F(k,j)} - 1 \right),$$

where $p_\ell, \ell = 1, .., L$ is the vector of fitted mixture probabilities and F is the $N \times L$ matrix of gamma factors, output of `fg.cgb2`. The linearized scores are the columns of `U`. They serve to compute the linearization approximation of the covariance matrix of the parameters $v_\ell = \log(p_\ell / p_L), \ell = 1, ..., L-1$. Function `varscore.cgb2` calculates the middle term of the sandwich variance estimator, that is the $((L-1) \times (L-1))$ estimated variance-covariance matrix of the $(L-1)$ weighted sums of the columns of `U`, without design information. `desvar.cgb2` calculates the design-based variance-covariance matrix of the $(L-1)$ weighted sums of the columns of `U`, invoking `svydesign` and `svytotal` of package `survey`. `hess.cgb2` calculates the Hessian $((L-1) \times (L-1))$ matrix of second derivatives of the pseudo-log-likelihood with respect to the parameters v_ℓ). It should be negative definite. If not, the maximum likelihood estimates are spurious. `vepar.cgb2` calculates the sandwich covariance matrix estimate of the vector of parameters `v`. `veind.cgb2` calculates estimates, standard error, covariance and correlation matrices of the indicators under the compound GB2.

Value

`scoreU.cgb2` returns a $N \times (L-1)$ matrix of scores <codeU. `varscore.cgb2` returns the variance-covariance estimate of the weighted sums of scores `U`, given by weighted cross products. `desvar.cgb2` returns a list of two elements. The first is the output of svytotal and the second is the design-based variance-covariance matrix of the weighted sums of the scores. `hess.cgb2` returns the matrix of second derivatives of the likelihood with respect to the parameters (bread for the sandwich variance estimate). `vepar.cgb2` returns a list of five elements - [["type"]] with value "parameter", [["estimate"]] estimated parameters, [["stderr"]] corresponding standard errors, [["Vcov"]] variance -covariance matrix and [["Vcor"]] - correlation matrix. `veind.cgb2` returns a list of five elements: [["type"]] with value "indicator", [["estimate"]] estimated indicators under the compound GB2, [["stderr"]] corresponding standard errors, [["Vcov"]] variance -covariance matrix and [["Vcor"]] - correlation matrix.

Author(s)

Monique Graf and Desislava Nedyalkova

References

Davison, A. (2003), *Statistical Models*. Cambridge University Press.

Freedman, D. A. (2006), On The So-Called "Huber Sandwich Estimator" and "Robust Standard Errors". *The American Statistician*, **60**, 299–302.

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

Pfeffermann, D. and Sverchkov, M. Yu. (2003), Fitting Generalized Linear Models under Informative Sampling. In, Skinner, C.J. and Chambers, R.L. (eds.). *Analysis of Survey Data*, chapter 12, 175–195. Wiley, New York.

Examples

```
## Not run:
# Example (following of example in CompoundFit)

# Estimated mixture probabilities:
(pl.hat <- estim[[1]])

# scores per unit
U <- scoreU.cgb2(fac, pl.hat)

# Conditional variances given a,b,p,q:

# 1. Variance of sum of scores:
(Vsc <- t(U)
(Vsc <- varscore.cgb2(U))

# 2. sandwich variance-covariance matrix estimate of (v_1,v_2):
(hess <-  hess.cgb2(U,pl.hat))
(Parameters <- vepar.cgb2(estim, Vsc, hess))

# 3. Theoretical indicators  (with mixture prob pl)
decomp <- "r"
(theoretical <- main.cgb2( 0.6,a,b,p,q,pl0, pl,decomp=decomp))

# Estimated indicators and conditional variances : takes a long time!
(Indic <- veind.cgb2(Parameters,a,b,p,q, pl0, pl.hat, decomp="r") )


## End(Not run)
```

`Contindic` *Sensitivity Analysis of Laeken Indicators on GB2 Parameters*

Description

Produces a contour plot of an indicator for a given `shape1`.

Usage

```
contindic.gb2(resol, shape1, shape21, shape22, shape31, shape32, fn, title, table=FALSE)
```

Arguments

`resol`	numeric; number of grid points horizontally and vertically.
`shape1`	numeric; positive parameter, first shape parameter of the GB2 distribution.
`shape21`, `shape22`, `shape31`, `shape32`	numeric; limits on the positive parameters of the Beta distribution.
`fn`	string; the name of the function to be used for the calculation of the values to be plotted.
`title`	string; title of the plot.
`table`	boolean; if `TRUE`, a table containing the values of the function `fn` at the different grid points is printed.

Details

An indicator is defined as a function of three parameters. The shape parameter, `shape1`, is held fixed. The shape parameters `shape2` and `shape3` vary between `shape21` and `shape22`, and `shape31` and `shape32`, respectively.

Value

A contour plot of a given indicator for a fixed value of the shape parameter `shape1`.

Author(s)

Monique Graf

See Also

`contour` (package `graphics`) for more details on contour plots.

Examples

```
par(mfrow=c(2,2))
shape21 <- 0.3
shape31 <- 0.36
shape22 <- 1.5
shape32 <- 1.5
shape11 <- 2.7
shape12 <- 9.2
resol <- 11
rangea <- round(seq(shape11, shape12 ,length.out=4),digits=1)
arpr <- function(shape1, shape2, shape3) 100*arpr.gb2(0.6, shape1, shape2, shape3)
fonc <- "arpr"
for (shape1 in rangea){
contindic.gb2(resol, shape1, shape21, shape22, shape31, shape32, arpr, "At-risk-of-poverty rate",
table=TRUE)
}
```

`Contprof` *Contour Plot of the Profile Log-likelihood of the GB2 Distribution*

Description

Produces a contour plot of the profile log-likelihood, which is a function of two parameters only.

Usage

```
contprof.gb2(z, w=rep(1,length(z)), resol, low=0.1, high=20)
```

Arguments

`z`	numeric; vector of data values.
`w`	numeric; vector of weights. Must have the same length as `z`. By default `w` is a vector of 1.
`resol`	numeric; number of grid points horizontally and vertically. For better graph quality, we recommend a value of 100.
`low`, `high`	numeric; lower and upper factors for scale.

Details

The matrix containing the values to be plotted (NAs are allowed) is of size `resol` × `resol`. The locations of the grid lines at which the values of the profile log-likelihood are measured are equally-spaced values between `low` and `high` multiplied by the initial parameters.

Value

A contour plot of the profile log-likelihood. The initial Fisk estimate is added as point "F".

Author(s)

Monique Graf

See Also

`fisk` for the Fisk estimate, `ProfLogLikelihood` for the profile log-likelihood and `contour` (package `graphics`) for more details on contour plots.

`Fisk` *Parameters of the Fisk Distribution*

Description

Calculation of the parameters a and b of the Fisk distribution, which is a GB2 distribution with $p = q = 1$. If m and v denote, respectively, the mean and variance of $log(z)$, then $\hat{a} = \pi/\sqrt{3 * v}$ and $\hat{b} = \exp(m)$.

Usage

```
fisk(z, w=rep(1, length(z)))
fiskh(z, w=rep(1, length(z)), hs=rep(1, length(z)))
```

Arguments

`z`	numeric; vector of data values.
`w`	numeric; vector of weights. Must have the same length as `z`. By default `w` is a vector of 1.
`hs`	numeric; vector of household sizes. Must have the same length as `z`. By default `hs` is a vector of 1.

Details

Function `fisk` first calculates the mean and variance of $log(z)$ and next the values of a and b under the Fisk distribution. Function `fiskh` first calculates the mean and variance of $log(z)$, assuming a sample of households, and next the values of a and b under the Fisk distribution.

Value

`fisk` and `fiskh` return vectors of length 4 containing the estimated parameters a, eqnb, as well as $p = 1$ and $q = 1$.

Author(s)

Monique Graf

References

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

See Also

`optim` for the general-purpose optimization

Examples

```
library(laeken)
data(eusilc)

# Income
inc <- as.vector(eusilc$eqIncome)

# Weights
w <- eusilc$rb050

#Fisk parameters
fpar <- fisk(inc, w)
```

---

gb2 *The Generalized Beta Distribution of the Second Kind*

Description

Density, distribution function, quantile function and random generation for the Generalized beta distribution of the second kind with parameters `a`, `b`, `p` and `q`.

Usage

```
dgb2(x, shape1, scale, shape2, shape3)
pgb2(x, shape1, scale, shape2, shape3)
qgb2(prob, shape1, scale, shape2, shape3)
rgb2(n, shape1, scale, shape2, shape3)
```

Arguments

`x`	numeric; vector of quantiles.
`shape1`	numeric; positive parameter.
`scale`	numeric; positive parameter.
`shape2, shape3`	numeric; positive parameters of the Beta distribution.
`prob`	numeric; vector of probabilities.
`n`	numeric; number of observations. If `length(n) > 1`, the length is taken to be the number required.

Details

The Generalized Beta distribution of the second kind with parameters `shape1` $= a$, `scale` $= b$, `shape2` $= p$ and `shape3` $= q$ has density

$$f(x) = \frac{a(x/b)^{ap-1}}{bB(p,q)(1+(x/b)^a)^{p+q}}$$

for $a > 0$, $b > 0$, $p > 0$ and $q > 0$, where $B(p, q)$ is the Beta function (`beta`). If `Z` follows a Beta distribution with parameters p and q and

$$y = \frac{z}{1 - z},$$

then

$$x = b * y^{1/a}$$

follows the GB2 distribution.

Value

`dgb2` gives the density, `pgb2` the distribution function, `qgb2` the quantile function, and `rgb2` generates random deviates.

Author(s)

Monique Graf

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, chapter 6. Wiley, Ney York.

McDonald, J. B. (1984) Some generalized functions for the size distribution of income. *Econometrica*, **52**, 647–663.

See Also

`beta` for the Beta function and `dbeta` for the Beta distribution.

Examples

```
a <- 3.9
b <- 18873
p <- 0.97
q <- 1.03
x <- qgb2(0.6, a, b, p, q)
y <- dgb2(x, a, b, p, q)
```

`Gini` *Computation of the Gini Coefficient for the GB2 Distribution and its Particular Cases.*

Description

Computes the Gini coefficient for the GB2 distribution using the function `gb2.gini`. Computes the Gini coefficient for the Beta Distribution of the Second Kind, Dagum and Singh-Maddala distributions.

Usage

```
gini.gb2(shape1, shape2, shape3)
gini.b2(shape2, shape3)
gini.dag(shape1, shape2)
gini.sm(shape1, shape3)
```

Arguments

`shape1`	numeric; positive parameter.
`shape2, shape3`	numeric; positive parameters of the Beta distribution.

Value

The Gini coefficient.

Author(s)

Monique Graf

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, chapter 6. Wiley, Ney York.

McDonald, J. B. (1984) Some generalized functions for the size distribution of income. *Econometrica*, **52**, 647–663.

See Also

`gb2.gini`

`Indicators` *Monetary Laeken Indicators under the GB2*

Description

Functions to calculate four primary social welfare indicators under the GB2, i.e. the at-risk-of-poverty threshold, the at-risk-of-poverty rate, the relative median at-risk-of-poverty gap, and the income quintile share ratio.

Usage

```
arpt.gb2(prop, shape1, scale, shape2, shape3)
arpr.gb2(prop, shape1, shape2, shape3)
rmpg.gb2(arpr, shape1, shape2, shape3)
qsr.gb2(shape1, shape2, shape3)
main.gb2(prop, shape1, scale, shape2, shape3)
main2.gb2(prop, shape1, scale, shape12, shape13)
```

Arguments

`prop`	numeric; proportion (in general is set to 0.6).
`arpr`	numeric; the value of the at-risk-of-poverty rate.
`shape1`	numeric; positive parameter.
`scale`	numeric; positive parameter.
`shape2, shape3`	numeric; positive parameters of the Beta distribution.
`shape12`	numeric; the product of the two parameters `shape1` and `shape2`.
`shape13`	numeric; the product of the two parameters `shape1` and `shape3`.

Details

In June 2006, the Social Protection Committee, which is a group of officials of the European Commisiion, adopts a set of common indicators for the social protection and social inclusion process. It consists of a portfolio of 14 overarching indicators (+11 context indicators) meant to reflect the overarching objectives (a) "social cohesion" and (b) "interaction with the Lisbon strategy for growth and jobs (launched in 2000) objectives"; and of three strand portfolios for social inclusion, pensions, and health and long-term care.

The at-risk-of-poverty threshold (or ARPT) is defined as 60% of the median national equivalized income.

The at-risk-of-poverty rate (or ARPR) is defined as the share of persons with an equivalised disposable income below the ARPT.

The relative median at-risk-of-poverty gap (or RMPG) is defined as the difference between the median equivalised income of persons below the ARPT and the ARPT itself, expressed as a percentage of the ARPT.

The income quintile share ratio (or QSR) is defined as the ratio of total income received by the 20% of the country's population with the highest income (top quintile) to that received by the 20% of the country's population with the lowest income (lowest quintile).

Let $x_{0.5}$ be the median of a GB2 with parameters `shape1` $= a$, `scale` $= b$, `shape2` $= p$ and `shape3` $= q$. Then,

$$ARPT(a, b, p, q) = 0.6x_{0.5}$$

The ARPR being scale-free, b can be chosen arbitrarily and can be fixed to 1.

The QSR is calculated with the help of the incomplete moments of order 1.

`main.gb2` and `main2.gb2` return a vector containing the following set of GB2 indicators: the median, the mean, the ARPR, the RMPG, the QSR and the Gini coefficient. The only difference is in the input parameters.

Value

`arpt.gb2` gives the ARPT, `arpr.gb2` the ARPR, `rmpg.gb2` the RMPG, and `qsr.gb2` calculates the QSR. `main.gb2` returns a vector containing the median of the distribution, the mean of the distribution, the ARPR, the RMPG, the QSR and the Gini coefficient. `main2.gb2` produces the same output as `main.gb2`.

Author(s)

Monique Graf

References

https://ec.europa.eu/social/main.jsp?langId=en&catId=750

See Also

`qgb2`, `incompl.gb2`

Examples

```
a <- 3.9
b <- 18873
p <- 0.97
q <- 1.03
ap <- a*p
aq <- a*q

arpt <- arpt.gb2(0.6, a, b, p, q)
arpr <- arpr.gb2(0.6, a, p, q)
rmpg <- rmpg.gb2(arpr, a, p, q)
qsr <- qsr.gb2(a, p, q)

ind1 <- main.gb2(0.6, a, b, p, q)
ind2 <- main2.gb2(0.6, a, b, ap, aq)
```

`LogDensity` *Log Density of the GB2 Distribution*

Description

Calculates the log density of the GB2 distribution for a single value or a vector of values. Calculates the first- and second-order partial derivatives of the log density evaluated at a single value.

Usage

```
logf.gb2(x, shape1, scale, shape2, shape3)
dlogf.gb2(xi, shape1, scale, shape2, shape3)
d2logf.gb2(xi, shape1, scale, shape2, shape3)
```

Arguments

`xi`	numeric; a data value.
`x`	numeric; a vector of data values.
`shape1`	numeric; positive parameter.
`scale`	numeric; positive parameter.
`shape2, shape3`	numeric; positive parameters of the Beta distribution.

Details

We calculate $log(f(x, \theta))$, where f is the GB2 density with parameters `shape1` $= a$, `scale` $= b$, `shape2` $= p$ and `shape3` $= q$, θ is the parameter vector. We calculate the first- and second-order partial derivatives of $log(f(x, \theta))$ with respect to the parameter vector θ.

Value

Depending on the input `logf.gb2` gives the log density for a single value or a vector of values. `dlogf.gb2` gives the vector of the four first-order partial derivatives of the log density and `d2logf.gb2` gives the 4×4 matrix of second-order partial derivatives of the log density.

Author(s)

Desislava Nedyalkova

References

Brazauskas, V. (2002) Fisher information matrix for the Feller-Pareto distribution. *Statistics & Probability Letters*, **59**, 159–167.

`LogLikelihood` *Full Log-likelihood of the GB2 Distribution*

Description

Calculates the log-likelihood, the score functions of the log-likelihood and the Fisher information matrix based on all four parameters of the GB2 distribution.

Usage

```
loglp.gb2(x, shape1, scale, shape2, shape3, w=rep(1, length(x)))
loglh.gb2(x, shape1, scale, shape2, shape3, w=rep(1, length(x)), hs=rep(1, length(x)))
scoresp.gb2(x, shape1, scale, shape2, shape3, w=rep(1, length(x)))
scoresh.gb2(x, shape1, scale, shape2, shape3, w=rep(1, length(x)), hs=rep(1, length(x)))
info.gb2(shape1, scale, shape2, shape3)
```

Arguments

`x`	numeric; vector of data values.
`shape1`	numeric; positive parameter.
`scale`	numeric; positive parameter.
`shape2, shape3`	numeric; positive parameters of the Beta distribution.
`w`	numeric; vector of weights. Must have the same length as `x`. By default `w` is a vector of 1.
`hs`	numeric; vector of household sizes. Must have the same length as `x`. By default `hs` is a vector of 1.

Details

We express the log-likelihood as a weighted mean of $log(f)$, evaluated at the data points, where f is the GB2 density with parameters `shape1` $= a$, `scale` $= b$, `shape2` $= p$ and `shape3` $= q$. If the weights are not available, then we suppose that `w` $= 1$. `loglp.gb2` calculates the log-likelihood in the case where the data is a sample of persons and `loglh.gb2` is adapted for a sample of households. Idem for the scores, which are obtained as weighted sums of the first derivatives of $log(f)$ with respect to the GB2 parameters, evaluated at the data points. The Fisher information matrix of the GB2 was obtained by Brazauskas (2002) and is expressed in terms of the second derivatives of the log-likelihood with respect to the GB2 parameters.

Author(s)

Monique Graf

References

Brazauskas, V. (2002) Fisher information matrix for the Feller-Pareto distribution. *Statistics & Probability Letters*, **59**, 159–167.

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, chapter 6. Wiley, Ney York.

`MLfitGB2` *Fitting the GB2 by the Method of Maximum Likelihood Estimation and Comparison of the Fitted Indicators with the Empirical Indicators*

Description

The function `mlfit.gb2` makes a call to `ml.gb2` and `profml.gb2`. Estimates of the GB2 parameters are calculated using maximum likelihood estimation based on the full and profile log-likelihoods. Empirical estimates of the set of primary indicators of poverty and social inclusion are calculated using the function `main.emp` (see package `laeken`) and these estimates are compared with the indicators calculated with the GB2 fitted parameters using the function `main.gb2`.

Usage

```
main.emp(z, w=rep(1, length(z)))
mlfit.gb2(z, w=rep(1, length(z)))
```

Arguments

`z`	numeric; vector of data values.
`w`	numeric; vector of weights. Must have the same length as `z`. By default `w` is a vector of 1.

Value

A list containing three different objects. The first is a data frame with the values of the fitted parameters for the full log-likelihood and the profile log-likelihood, the values of the two likelihoods, the values of the GB2 estimates of the indicators and the values of the empirical estimates of the indicators. The second and third objects are, respectively, the fit using the full log-likelihood and the fit using the profile log-likelihood.

Author(s)

Monique Graf and Desislava Nedyalkova

See Also

`optim` , `ml.gb2`, `profml.gb2`

Examples

```
# An example of using the function mlfit.gb2
# See also the examples of ml.gb2 and mlprof.gb2

## Not run:
library(laeken)
data(eusilc)

# Income
inc <- as.vector(eusilc$eqIncome)

# Weights
w <- eusilc$rb050

# Data set
d <- data.frame(inc, w)
d <- d[!is.na(d$inc),]

# Truncate at 0
inc <- d$inc[d$inc > 0]
w   <- d$w[d$inc > 0]

# ML fit
m1 <- mlfit.gb2(inc,w)

# GB2 fitted parameters and indicators through maximum likelihood estimation
m1[[1]]
# The fit using the full log-likelihood
m1[[2]]
# The fit using the profile log-likelihood
m1[[3]]

# ML fit, when no weights are avalable
m2 <- mlfit.gb2(inc)
# Results
```

```
m2[[1]]

## End(Not run)
```

MLfullGB2 *Maximum Likelihood Estimation of the GB2 Based on the Full Log-likelihood*

Description

Performs maximum pseudo-likelihood estimation through the general-purpose optimisation function `optim` from package `stats`. Two methods of optimization are considered: BFGS and L-BFGS-B (see `optim` documentation for more details). Initial values of the parameters to be optimized over (a, b, p and q) are given from the Fisk distribution and $p = q = 1$. The function to be maximized by `optim` is the negative of the full log-likelihood and the gradient is equal to the negative of the scores, respectively for the case of a sample of persons and a sample of households.

Usage

```
ml.gb2(z, w=rep(1, length(z)), method=1, hess=FALSE)
mlh.gb2(z, w=rep(1, length(z)), hs=rep(1, length(z)), method=1, hess = FALSE)
```

Arguments

`z`	numeric; vector of data values.
`w`	numeric; vector of weights. Must have the same length as `z`. By default `w` is a vector of 1.
`hs`	numeric; vector of household sizes. Must have the same length as `z`. By default `hs` is a vector of 1.
`method`	numeric; the method to be used by `optim`. By default, codemethod = 1 and the used method is BFGS. If `method` = 2, method L-BFGS-B is used.
`hess`	logical; By default, `hess = FALSE`, the hessian matrix is not calculated.

Details

Function `ml.gb2` performs maximum likelihood estimation through the general-purpose optimization function `optim` from package `stats`, based on the full log-likelihood calculated in a sample of persons. Function `mlh.gb2` performs maximum likelihood estimation through the general-purpose optimization function `optim` from package `stats`, based on the full log-likelihood calculated in a sample of households.

Value

`ml.gb2` and `mlh.gb2` return a list with 1 argument: `opt1` for the output of the BFGS fit or `opt2` for the output of the L-BFGS fit. Further values are given by the values of `optim`.

Author(s)

Monique Graf

References

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

See Also

`optim` for the general-purpose optimization and `fisk` for the Fisk distribution.

Examples

```
## Not run:
library(laeken)
data(eusilc)

# Income
inc <- as.vector(eusilc$eqIncome)

# Weights
w <- eusilc$rb050

# Data set
d <- data.frame(inc, w)
d <- d[!is.na(d$inc),]

# Truncate at 0
inc <- d$inc[d$inc > 0]
w   <- d$w[d$inc > 0]

# Fit using the full log-likelihood
fitf <- ml.gb2(inc, w)

# Fitted GB2 parameters
af <- fitf$par[1]
bf <- fitf$par[2]
pf <- fitf$par[3]
qf <- fitf$par[4]

# Likelihood
flik <- fitf$value

# If we want to compare the indicators

# GB2 indicators
indicf <- round(main.gb2(0.6,af,bf,pf,qf), digits=3)
# Empirical indicators
indice <- round(main.emp(inc,w), digits=3)
```

```
# Plots
plotsML.gb2(inc,af,bf,pf,qf,w)

## End(Not run)
```

`MLprofGB2` *Maximum Likelihood Estimation of the GB2 Based on the Profile Log-likelihood*

Description

`profml.gb2` performs maximum likelihood estimation based on the profile log-likelihood through the general-purpose optimisation function `optim` from package `stats`.

Usage

```
profml.gb2(z, w=rep(1, length(z)), method=1, hess = FALSE)
```

Arguments

`z`	numeric; vector of data values.
`w`	numeric; vector of weights. Must have the same length as `z`. By default `w` is a vector of 1.
`method`	numeric; the method to be used by `optim`. By default, codemethod = 1 and the used method is BFGS. If `method` = 2, method L-BFGS-B is used.
`hess`	logical; By default, `hess` = `FALSE`, the hessian matrix is not calculated.

Details

Two methods are considered: BFGS and L-BFGS-B (see `optim` documentation for more details). Initial values of the parameters to be optimized over (a and b) are given from the Fisk distribution. The function to be maximized by `optim` is the negative of the profile log-likelihood (`proflogl.gb2`) and the gradient is equal to the negative of the scores (`profscores.gb2`).

Value

A list with 1 argument: `opt1` for the output of the BFGS fit or `opt2` for the output of the L-BFGS fit. Further values are given by the values of `optim`.

Author(s)

Monique Graf

References

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

See Also

`optim` for the general-purpose optimization, `link{ml.gb2}` for the fit using the full log-likelihood and `fisk` for the Fisk distribution.

Examples

```
library(laeken)
data(eusilc)

# Income
inc <- as.vector(eusilc$eqIncome)

# Weights
w <- eusilc$rb050

# Data set
d <- data.frame(inc,w)
d <- d[!is.na(d$inc),]

# Truncate at 0
inc <- d$inc[d$inc > 0]
w   <- d$w[d$inc > 0]

# Fit using the profile log-likelihood
fitp <- profml.gb2(inc, w)$opt1

# Fitted GB2 parameters
ap <- fitp$par[1]
bp <- fitp$par[2]
pp <- prof.gb2(inc, ap, bp, w)[3]
qp <- prof.gb2(inc, ap, bp, w)[4]

# Profile log-likelihood
proflik <- fitp$value

# If we want to compare the indicators
## Not run:
# GB2 indicators
indicp <- round(main.gb2(0.6,ap,bp,pp,qp), digits=3)
# Empirical indicators
indice <- round(main.emp(inc,w), digits=3)

## End(Not run)

# Plots
## Not run: plotsML.gb2(inc,ap,bp,pp,qp,w)
```

`Moments` *Moments and Other Properties of a GB2 Random Variable*

Description

These functions calculate the moments of order `k` and incomplete moments of order `k` of a GB2 random variable X as well as the expectation, the variance, the kurtosis and the skewness of $log(X)$.

Usage

```
moment.gb2(k, shape1, scale, shape2, shape3)
incompl.gb2(x, k, shape1, scale, shape2, shape3)
el.gb2(shape1, scale, shape2, shape3)
vl.gb2(shape1, shape2, shape3)
sl.gb2(shape2, shape3)
kl.gb2(shape2, shape3)
```

Arguments

`x`	numeric; vector of quantiles.
`k`	numeric; order of the moment.
`shape1`	numeric; positive parameter.
`scale`	numeric; positive parameter.
`shape2, shape3`	numeric; positive parameters of the Beta distribution.

Details

Let X be a random variable following a GB2 distribution with parameters `shape1` $= a$, `scale` $= b$, `shape2` $= p$ and `shape3` $= q$. Moments and incomplete moments of X exist only for $-ap \leq k \leq aq$. Moments are given by

$$E(X^k) = b^k \frac{\Gamma(p + k/a)\Gamma(q - k/a)}{\Gamma(p)\Gamma(q)}$$

This expression, when considered a function of `k`, can be viewed as the moment-generating function of $Y = log(X)$. Thus, it is useful to compute the moments of $log(X)$, which are needed for deriving, for instance, the Fisher information matrix of the GB2 distribution. Moments of $log(X)$ exist for all `k`.

Value

`moment.gb2` gives the moment of order `k`, `incompl.gb2` gives the incomplete moment of order `k`, `El.gb2` gives the expectation of $log(X)$, `vl.gb2` gives the variance of $log(X)$, `sl.gb2` gives the skewness of $log(X)$, `kl.gb2` gives the kurtosis of $log(X)$.

Author(s)

Monique Graf

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, chapter 6. Wiley, Ney York.

See Also

`gamma` for the Gamma function and related functions (`digamma`, `trigamma` and `psigamma`).

Examples

```
a <- 3.9
b <- 18873
p <- 0.97
q <- 1.03
k <- 2
x <- qgb2(0.6, a, b, p, q)
moment.gb2(k, a, b, p, q)
incompl.gb2(x, k, a, b, p, q)
vl.gb2(a, p, q)
kl.gb2(p, q)
```

`NonlinearFit` *Fitting the GB2 by Minimizing the Distance Between a Set of Empirical Indicators and Their GB2 Expressions*

Description

Fitting the parameters of the GB2 distribution by optimizing the squared weighted distance between a set of empirical indicators, i.e. the median, the ARPR, the RMPG, the QSR and the Gini coefficient, and the GB2 indicators using nonlinear least squares (function `nls` from package `stats`).

Usage

```
nlsfit.gb2(med, ei4, par0=c(1/ei4[4],med,1,1), cva=1, bound1=par0[1]*max(0.2,1-2*cva),
bound2=par0[1]*min(2,1+2*cva), ei4w=1/ei4)
```

Arguments

`med`	numeric; the empirical median.
`ei4`	numeric; the values of the empirical indicators.
`par0`	numeric; vector of initial values for the GB2 parameters a, b, p and q. The default is to take a equal to the inverse of the empirical Gini coefficient, b equal to the empirical median and $p = q = 1$.
`cva`	numeric; the coefficient of variation of the ML estimate of the parameter a. The default value is 1.
`bound1, bound2`	numeric; the lower and upper bounds for the parameter a in the algorithm. The default values are $0.2 * a_0$ and $2 * a_0$, where a_0 is the initial value of the parameter a.
`ei4w`	numeric; vector of weights of to be passed to the `nls` function. The default values are the inverse of the empirical indicators.

Details

We consider the following set of indicators $A = (median, ARPR, RMPG, QSR, Gini)$ and their corresponding GB2 expressions A_{GB2}. We fit the parameters of the GB2 in two consecutive steps. In the first step, we use the set of indicators, excluding the median, and their corresponding expressions in function of a, ap and aq. The bounds for a are defined in function of the coefficient of variation of the fitted parameter $\hat(a)$. The nonlinear model that is passed to `nls` is given by:

$$\sum_{i=1}^{4} c_i (A_{empir,i} - A_{GB2,i}(a, ap, aq))^2,$$

where the weights c_i take the differing scales into account and are given by the vector `ei4w`. ap and aq are bounded so that the constraints for the existence of the moments of the GB2 distribution and the excess for calculating the Gini coefficient are fulfilled, i.e. $ap \geq 1$ and $aq \geq 2$. In the second step, only the the parameter b is estimated, optimizing the weighted square difference between the empirical median and the GB2 median in function of the already obtained NLS parameters a, p and q.

Value

`nlsfit.gb2` returns a list of three values: the fitted GB2 parameters, the first fitted object and the second fitted object.

Author(s)

Monique Graf and Desislava Nedyalkova

See Also

`nls`, `Thomae`, `moment.gb2`

Examples

```
# Takes long time to run, as it makes a call to the function ml.gb2

## Not run:
library(laeken)
data(eusilc)

# Personal income
inc <- as.vector(eusilc$eqIncome)

# Sampling weights
w <- eusilc$rb050

# Data set
d <- data.frame(inc, w)
d <- d[!is.na(d$inc),]

# Truncate at 0
d <- d[d$inc > 0,]
```

```
inc <- d$inc
w   <- d$w

# ML fit, full log-likelihood
fitf <- ml.gb2(inc, w)$opt1

# Estimated parameters
af <- fitf$par[1]
bf <- fitf$par[2]
pf <- fitf$par[3]
qf <- fitf$par[4]

gb2.par <- c(af, bf, pf, qf)

# Empirical indicators
indicEMP  <- main.emp(inc, w)
indicEMP <- c(indicEMP[1],indicEMP[3:6])
indicE <- round(indicEMP, digits=3)

# Nonlinear fit
nn <- nlsfit.gb2(indicEMP[1,3:6],indicEMP[3:6])
an <- nn[[1]][1]
bn <- nn[[1]][2]
pn <- nn[[1]][3]
qn <- nn[[1]][4]

# GB2 indicators
indicNLS <- c(main.gb2(0.6, an, bn, pn, qn)[1], main.gb2(0.6, an, bn, pn, qn)[3:6])
indicML <- c(main.gb2(0.6, af, bf, pf, qf)[1], main.gb2(0.6, af, bf, pf, qf)[3:6])
indicN <- round(indicNLS, digits=3)
indicM <- round(indicML, digits=3)

# Likelihoods
nlik <- loglp.gb2(inc, an, bn, pn, qn, w)
mlik <- loglp.gb2(inc, af, bf, pf, qf, w)

# Results
type=c("Emp. est", "NLS", "ML full")
results <- data.frame(type=type,
        median=c(indicE[1], indicN[1], indicM[1]),
        ARPR=c(indicE[2], indicN[2], indicM[2]),
        RMPG=c(indicE[3], indicN[3], indicM[3]),
        QSR =c(indicE[4], indicN[4], indicM[4]),
        GINI=c(indicE[5], indicN[5], indicM[5]),
        likelihood=c(NA, nlik, mlik),
        a=c(NA, an, af), b=c(NA, bn, bf) ,p=c(NA, pn, pf), q=c(NA, qn, qf))


## End(Not run)
```

PlotsML *Cumulative Distribution Plot and Kernel Density Plot for the Fitted GB2*

Description

Function `plotsML.gb2` produces two plots. The first is a plot of the empirical cumulative distribution function versus the fitted cumulative distibution function. The second is a plot of the kernel density versus the fitted GB2 density. Function `saveplot` saves locally the produced plot.

Usage

```
plotsML.gb2(z, shape1, scale, shape2, shape3, w=rep(1,length(z)))
saveplot(name, pathout)
```

Arguments

`z`	numeric; vector of data values.
`w`	numeric; vector of weights. Must have the same length as `z`. By default `w` is a vector of 1.
`shape1`	numeric; positive parameter.
`scale`	numeric; positive parameter.
`shape2, shape3`	numeric; positive parameters of the Beta distribution.
`name`	string; the name of the plot.
`pathout`	string; the path of the folder or device where the plot will be saved.

Details

The used kernel is "Epanechnikov" (see `plot`).

Author(s)

Monique Graf and Desislava Nedyalkova

ProfLogLikelihood *Profile Log-likelihood of the GB2 Distribution*

Description

Expression of the parameters `shape2` $= p$ and `shape3` $= q$ of the GB2 distribution as functions of `shape1` $= a$ and `scale` $= b$, profile log-likelihood of the GB2 distribution, scores of the profile log-likelihood.

Usage

```
prof.gb2(x, shape1, scale, w=rep(1, length(x)))
proflogl.gb2(x, shape1, scale, w=rep(1, length(x)))
profscores.gb2(x, shape1, scale, w=rep(1, length(x)))
```

Arguments

`x`	numeric; vector of data values.
`shape1`	numeric; positive parameter.
`scale`	numeric; positive parameter.
`w`	numeric; vector of weights. Must have the same length as `x`. By default `w` is a vector of 1.

Details

Using the full log-likelihood equations for the GB2 distribution, the parameters p and q can be estimated as functions of a and b. These functions are plugged into the log-likelihood expression, which becomes a function of a and b only. This is obtained by reparametrizing the GB2, i.e. we set $r = \frac{p}{p+q}$ and $s = p + q$. More details can be found in Graf (2009).

Value

`prof` returns a vector containing the values of r, s, p, q as well as two other parameters used in the calculation of the profile log-likelihood and its first derivatives. `proflogl.gb2` returns the value of the profile log-likelihood and `profscores.gb2` returns the vector of the first derivatives of the profile log-likelihhod with respect to a and b.

Author(s)

Monique Graf and Desislava Nedyalkova

References

Graf, M. (2009) The Log-Likelihood of the Generalized Beta Distribution of the Second Kind. *working paper*, SFSO.

`RobustWeights` *Robustification of the sampling weights*

Description

Calculation of a Huber-type correction factor by which the vector of weights is multiplied.

Usage

```
robwts(x, w=rep(1,length(x)), c=0.01, alpha=0.001)
```

Arguments

`x`	numeric; vector of data values.
`w`	numeric; vector of weights. Must have the same length as `x`. By default `w` is a vector of 1.
`c`	numeric; a constant which can take different values, e.g. 0.01,0.02. By default $c = 0.1$.
`alpha`	numeric; a probability in the interval $(0, 1)$. By default $alpha = 0.001$.

Details

If x denotes the observed value and x_α the α-th qiantile of the Fisk distribution, then we define our scale as:

$$d = \frac{x_{1-\alpha}}{b} - \frac{x_\alpha}{b}$$

. Next, the correction factor is calculated as follows:

$$corr = \max\left\{c, \min\left(1, \frac{d}{|b/x - 1|}, \frac{d}{|x/b - 1|}\right)\right\}$$

Value

`robwts` returns a list of two elements: the vector of correction factors by which the weights are multiplied and the vector of corrected (robustified) weights.

Author(s)

Monique Graf

References

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

`Thomae` *Maximum Excess Representation of a Generalized Hypergeometric Function Using Thomae's Theorem*

Description

Defines Thomae's arguments from the upper (`U`) and lower (`L`) parameters of a ${}_3F_2(U, L; 1)$. Computes the optimal combination leading to the maximum excess. Computes the optimal combination of Thomae's arguments and calculates the optimal representation of the ${}_3F_2(U, L; 1)$ using the `genhypergeo_series` function from package `hypergeo`. Computes the Gini coefficient for the GB2 distribution, using Thomae's theorem.

Usage

```
ULg(U, L)
combiopt(g)
Thomae(U, L, lB, tol, maxiter, debug)
gb2.gini(shape1, shape2, shape3, tol=1e-08, maxiter=10000, debug=FALSE)
```

Arguments

`U`	numeric; vector of length 3 giving the upper arguments of the generalized hypergeometric function ${}_3F_2$.
`L`	numeric; vector of length 2 giving the lower arguments of the generalized hypergeometric function ${}_3F_2$.
`g`	numeric; vector of Thomae's permuting arguments.
`lB`	numeric; ratio of beta functions (a common factor in the expression of the Gini coefficient under the GB2).
`shape1`	numeric; positive parameter.
`shape2`, `shape3`	numeric; positive parameters of the Beta distribution.
`tol`	numeric; tolerance with default 0, meaning to iterate until additional terms do not change the partial sum.
`maxiter`	numeric; maximum number of iterations to perform.
`debug`	boolean; if `TRUE`, returns the list of changes to the partial sum.

Details

Internal use only. More details can be found in Graf (2009).

Value

`ULg` returns a list containing Thomae's arguments and the excess, `combiopt` gives the optimal combination of Thomae's arguments, `Thomae` returns the optimal representation of the ${}_3F_2(U, L; 1)$, `gb2.gini` returns the value of the Gini coefficient under the GB2.

Author(s)

Monique Graf

References

Graf, M. (2009) An Efficient Algorithm for the Computation of the Gini coefficient of the Generalized Beta Distribution of the Second Kind. *ASA Proceedings of the Joint Statistical Meetings*, 4835–4843. American Statistical Association (Alexandria, VA).

McDonald, J. B. (1984) Some generalized functions for the size distribution of income. *Econometrica*, **52**, 647–663.

See Also

`genhypergeo_series`, `gini.gb2`

`Varest` *Variance Estimation of the Parameters of the GB2 Distribution*

Description

Calculation of variance estimates of the estimated GB2 parameters and the estimated GB2 indicators under cluster sampling.

Usage

```
varscore.gb2(x, shape1, scale, shape2, shape3, w=rep(1,length(x)), hs=rep(1,length(x)))
vepar.gb2(x, Vsc, shape1, scale, shape2, shape3, w=rep(1,length(x)), hs=rep(1,length(x)))
derivind.gb2(shape1, scale, shape2, shape3)
veind.gb2(Vpar, shape1, scale, shape2, shape3)
```

Arguments

`x`	numeric; vector of data values.
`Vsc`	numeric; 4 by 4 matrix.
`shape1`	numeric; positive parameter.
`scale`	numeric; positive parameter.
`shape2, shape3`	numeric; positive parameters of the Beta distribution.
`w`	numeric; vector of weights. Must have the same length as `x`. By default `w` is a vector of 1.
`hs`	numeric; vector of household sizes. Must have the same length as `x`. By default `w` is a vector of 1.
`Vpar`	numeric; 4 by 4 matrix.

Details

Knowing the first and second derivatives of $log(f)$, and using the sandwich variance estimator (see Freedman (2006)), the calculation of the variance estimates of the GB2 parameters is straightforward. `Vsc` is a square matrix of size the number of parameters, e.g. the estimated design variance-covariance matrix of the estimated parameters. We know that the GB2 estimates of the Laeken indicators are functions of the GB2 parameters. In this case, the variance estimates of the fitted indicators are obtained using the delta method. The function `veind.gb2` uses `Vpar`, the sandwich variance estimator of the vector of parameters, in order to obtain the sandwich variance estimator of the indicators. More details can be found in Graf and Nedyalkova (2011).

Value

`varscore.gb2` calculates the middle term of the sandwich variance estimator under simple random cluster sampling. `vepar.gb2` returns a list of two elements: the estimated variance-covariance matrix of the estimated GB2 parameters and the second-order partial derivative of the pseudo log-likelihood function. The function `veind.gb2` returns the estimated variance-covariance matrix of the estimated GB2 indicators. `derivind.gb2` calculates the numerical derivatives of the GB2 indicators and is for internal use only.

Author(s)

Monique Graf and Desislava Nedyalkova

References

Davison, A. (2003), *Statistical Models*. Cambridge University Press.

Freedman, D. A. (2006), On The So-Called "Huber Sandwich Estimator" and "Robust Standard Errors". *The American Statistician*, **60**, 299–302.

Graf, M., Nedyalkova, D., Muennich, R., Seger, J. and Zins, S. (2011) AMELI Deliverable 2.1: Parametric Estimation of Income Distributions and Indicators of Poverty and Social Exclusion. *Technical report*, AMELI-Project.

Pfeffermann, D. and Sverchkov, M. Yu. (2003), Fitting Generalized Linear Models under Informative Sampling. In, Skinner, C.J. and Chambers, R.L. (eds.). *Analysis of Survey Data*, chapter 12, 175–195. Wiley, New York.

Examples

```
# An example of variance estimation of the GB2 parameters,
# using the dataset "eusilcP" from the R package simFrame.
# Takes long time to run

## Not run:
library(survey)
library(simFrame)
data(eusilcP)

# Draw a sample from eusilcP
# 1-stage simple random cluster sampling of size 6000 (cluster = household)
# directly,
#s <- draw(eusilcP[, c("hid", "hsize", "eqIncome")], grouping = "hid", size = 6000)

# or setting up 250 samples, and drawing the first one.
# This sample setup can be used for running a simulation.
set.seed(12345)
scs <- setup(eusilcP, grouping = "hid", size = 6000, k = 250)
s <- draw(eusilcP[, c("region", "hid", "hsize", "eqIncome")], scs, i=1)

# The number of observations (persons) in eusilcP (58654 persons)
\dontrun{N <- dim(eusilcP)[1]}
# The number of households in eusilcP (25000 households)
Nh <- length(unique(eusilcP$hid))

# Survey design corresponding to the drawn sample
sdo = svydesign(id=~hid, fpc=rep(Nh,nrow(s)), data=s)
\dontrun{summary(sdo)}

# Truncated sample (truncate at 0)
s <- s[!is.na(s$eqIncome),]
str <- s[s$eqIncome > 0, ]
eqInc <- str$eqIncome
```

```
w <- str$.weight

# Designs for the truncated sample
sdotr <- subset(sdo, eqIncome >0)
sddtr = svydesign(id=~hid, strata=~region, fpc=NULL, weights=~.weight, data=str)
\dontrun{summary(sdotr)}
\dontrun{summary(sddtr)}

# Fit by maximum likelihood
fit <- ml.gb2(eqInc,w)$opt1
af <- fit$par[1]
bf <- fit$par[2]
pf <- fit$par[3]
qf <- fit$par[4]
mlik <- -fit$value

# Estimated parameters and indicators, empirical indicators
gb2.par <- round(c(af, bf, pf, qf), digits=3)
emp.ind <- main.emp(eqInc, w)
gb2.ind <- main.gb2(0.6, af, bf, pf, qf)

# Scores
scores <- matrix(nrow=length(eqInc), ncol=4)
for (i in 1:length(eqInc)){
scores[i,] <- dlogf.gb2(eqInc[i], af, bf, pf, qf)
}

# Data on households only
sh <- unique(str)
heqInc <- sh$eqIncome
hw <- sh$.weight
hhs <- sh$hsize
hs <- as.numeric(as.vector(hhs))

# Variance of the scores
VSC <- varscore.gb2(heqInc, af, bf, pf, qf, hw, hs)

# Variance of the scores using the explicit designs, and package survey
DV1 <- vcov(svytotal(~scores[,1]+scores[,2]+scores[,3]+scores[,4], design=sdotr))
DV2 <- vcov(svytotal(~scores[,1]+scores[,2]+scores[,3]+scores[,4], design=sddtr))

# Estimated variance-covariance matrix of the parameters af, bf, pf and qf
VCMP <- vepar.gb2(heqInc, VSC, af, bf, pf, qf, hw, hs)[[1]]
DVCMP1 <- vepar.gb2(heqInc, DV1, af, bf, pf, qf, hw, hs)[[1]]
DVCMP2 <- vepar.gb2(heqInc, DV2, af, bf, pf, qf, hw, hs)[[1]]

\dontrun{diag(DVCMP1)/diag(VCMP)}
\dontrun{diag(DVCMP2)/diag(VCMP)}
\dontrun{diag(DV1)/diag(VSC)}
\dontrun{diag(DV2)/diag(VSC)}

# Standard errors of af, bf, pf and qf
se.par <- sqrt(diag(VCMP))
```

```
sed1.par <- sqrt(diag(DVCMP1))
sed2.par <- sqrt(diag(DVCMP2))

# Estimated variance-covariance matrix of the indicators (VCMI)
VCMI <- veind.gb2(VCMP, af, bf, pf, qf)
DVCMI1 <- veind.gb2(DVCMP1, af, bf, pf, qf)
DVCMI2 <- veind.gb2(DVCMP2, af, bf, pf, qf)

# Standard errors and confidence intervals
varest.ind <- diag(VCMI)
se.ind <- sqrt(varest.ind)
lci.ind <- gb2.ind - 1.96*se.ind
uci.ind <- gb2.ind + 1.96*se.ind
inCI <- as.numeric(lci.ind <= emp.ind & emp.ind <= uci.ind)

# under the sampling design sdotr

varestd1.ind <- diag(DVCMI1)
sed1.ind <- sqrt(varestd1.ind)
lcid1.ind <- gb2.ind - 1.96*sed1.ind
ucid1.ind <- gb2.ind + 1.96*sed1.ind
inCId1 <- as.numeric(lcid1.ind <= emp.ind & emp.ind <= ucid1.ind)

#under the sampling design sddtr

varestd2.ind <- diag(DVCMI2)
sed2.ind <- sqrt(varestd2.ind)
lcid2.ind <- gb2.ind - 1.96*sed2.ind
ucid2.ind <- gb2.ind + 1.96*sed2.ind
inCId2 <- as.numeric(lcid2.ind <= emp.ind & emp.ind <= ucid2.ind)

#coefficients of variation .par (parameters), .ind (indicators)
cv.par <- se.par/gb2.par
names(cv.par) <- c("am","bm","pm","qm")
cvd1.par <- sed1.par/gb2.par
names(cvd1.par) <- c("am","bm","pm","qm")
cvd2.par <- sed2.par/gb2.par
names(cvd2.par) <- c("am","bm","pm","qm")

cv.ind <- se.ind/gb2.ind
cvd1.ind <- sed1.ind/gb2.ind
cvd2.ind <- sed2.ind/gb2.ind


#results
res <- data.frame(am = af, bm = bf, pm = pf, qm = qf, lik = mlik,
  median = gb2.ind[[1]], mean = gb2.ind[[2]], ARPR = gb2.ind[[3]],
    RMPG = gb2.ind[[4]], QSR = gb2.ind[[5]], Gini = gb2.ind[[6]],
  emedian = emp.ind[[1]], emean = emp.ind[[2]], eARPR = emp.ind[[3]],
    eRMPG = emp.ind[[4]], eQSR = emp.ind[[5]], eGini = emp.ind[[6]],
  cva = cv.par[1], cvb = cv.par[2], cvp= cv.par[3], cvq = cv.par[4],
  cvd1a = cvd1.par[1], cvd1b = cvd1.par[2], cvd1p= cvd1.par[3], cvd1q = cvd1.par[4],
  cvd2a = cvd2.par[1], cvd2b = cvd2.par[2], cvd2p= cvd2.par[3], cvd2q = cvd2.par[4],
```

```
  cvmed = cv.ind[[1]], cvmean = cv.ind[[2]], cvARPR = cv.ind[[3]],
  cvRMPG = cv.ind[[4]], cvQSR = cv.ind[[5]], cvGini = cv.ind[[6]],
  cvd1med = cvd1.ind[[1]], cvd1mean = cvd1.ind[[2]], cvd1ARPR = cvd1.ind[[3]],
  cvd1RMPG = cvd1.ind[[4]], cvd1QSR = cvd1.ind[[5]], cvd1Gini = cvd1.ind[[6]],
  cvd2med = cvd2.ind[[1]], cvd2mean = cvd2.ind[[2]], cvd2ARPR = cvd2.ind[[3]],
  cvd2RMPG = cvd2.ind[[4]], cvd2QSR = cvd2.ind[[5]], cvd2Gini = cvd2.ind[[6]])

  res <- list(parameters = data.frame(am = af, bm = bf, pm = pf, qm = qf, lik = mlik),
              cv.parameters.naive = cv.par,
              cv.parameters.design1 = cvd1.par,
              cv.parameters.design2 = cvd2.par,
    GB2.indicators = gb2.ind,
              emp.indicators = emp.ind,
              cv.indicators.naive = cv.ind,
              cv.indicators.design1 = cvd1.ind,
              cv.indicators.design2 = cvd2.ind)
res
\dontrun{inCI}

## End(Not run)
```

Index

∗ **distribution**
    Compound, 2
    CompoundAuxFit, 6
    CompoundAuxVarest, 9
    CompoundFit, 13
    CompoundIndicators, 15
    CompoundMoments, 17
    CompoundQuantiles, 18
    CompoundVarest, 20
    Contprof, 24
    Fisk, 25
    gb2, 26
    Gini, 27
    Indicators, 28
    LogDensity, 30
    LogLikelihood, 31
    MLfitGB2, 32
    MLfullGB2, 34
    MLprofGB2, 36
    Moments, 37
    NonlinearFit, 39
    PlotsML, 42
    ProfLogLikelihood, 42
    Thomae, 44
    Varest, 46
∗
    Compound, 2
    NonlinearFit, 39

arpr.cgb2 *(*CompoundIndicators*)*, 15
arpr.gb2 *(*Indicators*)*, 28
arpt.cgb2 *(*CompoundIndicators*)*, 15
arpt.gb2 *(*Indicators*)*, 28

beta, *27*

combiopt *(*Thomae*)*, 44
Compound, 2
CompoundAuxDensPlot, 5
CompoundAuxFit, 6
CompoundAuxVarest, 9
CompoundDensPlot, 12
CompoundFit, *7*, 13
CompoundIndicators, 15
CompoundMoments, 17
CompoundQuantiles, 18
CompoundVarest, 20
Contindic, 22
contindic.gb2 *(*Contindic*)*, 22
contour, *23, 24*
Contprof, 24
contprof.gb2 *(*Contprof*)*, 24

d2logf.gb2 *(*LogDensity*)*, 30
dbeta, *27*
dcgb2 *(*Compound*)*, 2
density, *13*
derivind.cgb2 *(*CompoundVarest*)*, 20
derivind.gb2 *(*Varest*)*, 46
desvar.cavgb2, *10*
desvar.cavgb2 *(*CompoundAuxVarest*)*, 9
desvar.cgb2 *(*CompoundVarest*)*, 20
dgb2 *(*gb2*)*, 26
dl.cgb2 *(*Compound*)*, 2
dlogf.gb2 *(*LogDensity*)*, 30
dplot.cavgb2 *(*CompoundAuxDensPlot*)*, 5
dplot.cgb2 *(*CompoundDensPlot*)*, 12

el.gb2 *(*Moments*)*, 37

fg.cgb2 *(*Compound*)*, 2
Fisk, 25
fisk, *24*, *35*, *37*
fisk *(*Fisk*)*, 25
fiskh *(*Fisk*)*, 25

gamma, *39*
gb2, 26
gb2.gini, *27, 28*
gb2.gini *(*Thomae*)*, 44

genhypergeo_series, *45*
Gini, 27
gini.b2 *(*Gini*)*, 27
gini.dag *(*Gini*)*, 27
gini.gb2, *45*
gini.gb2 *(*Gini*)*, 27
gini.sm *(*Gini*)*, 27

hess.cavgb2 *(*CompoundAuxVarest*)*, 9
hess.cgb2 *(*CompoundVarest*)*, 20

incompl.cgb2 *(*CompoundMoments*)*, 17
incompl.gb2, *30*
incompl.gb2 *(*Moments*)*, 37
Indicators, 28
info.gb2 *(*LogLikelihood*)*, 31

kl.gb2 *(*Moments*)*, 37

lambda0.cavgb2 *(*CompoundAuxFit*)*, 6
LogDensity, 30
logf.gb2 *(*LogDensity*)*, 30
logl.cavgb2 *(*CompoundAuxFit*)*, 6
logl.cgb2 *(*CompoundFit*)*, 13
loglh.gb2 *(*LogLikelihood*)*, 31
LogLikelihood, 31
loglp.gb2 *(*LogLikelihood*)*, 31

main.cgb2 *(*CompoundIndicators*)*, 15
main.emp *(*MLfitGB2*)*, 32
main.gb2, *32*
main.gb2 *(*Indicators*)*, 28
main2.gb2 *(*Indicators*)*, 28
mkl.cgb2 *(*CompoundMoments*)*, 17
ml.cavgb2 *(*CompoundAuxFit*)*, 6
ml.cgb2, *13*
ml.cgb2 *(*CompoundFit*)*, 13
ml.gb2, *33*
ml.gb2 *(*MLfullGB2*)*, 34
mlfit.gb2 *(*MLfitGB2*)*, 32
MLfitGB2, 32
MLfullGB2, 34
mlh.gb2 *(*MLfullGB2*)*, 34
MLprofGB2, 36
moment.cgb2 *(*CompoundMoments*)*, 17
moment.gb2, *40*
moment.gb2 *(*Moments*)*, 37
Moments, 37

nls, *40*
nlsfit.gb2 *(*NonlinearFit*)*, 39
NonlinearFit, 39

optim, *6*, *7*, *14*, *25*, *33*, *35*, *37*

pcgb2 *(*Compound*)*, 2
pgb2 *(*gb2*)*, 26
pkl.cavgb2, *5*
pkl.cavgb2 *(*CompoundAuxFit*)*, 6
pl.cgb2 *(*Compound*)*, 2
plot, *42*
plot.density, *13*
PlotsML, 41
plotsML.gb2 *(*PlotsML*)*, 42
pofv.cgb2 *(*CompoundFit*)*, 13
prcgb2 *(*Compound*)*, 2
prof.gb2 *(*ProfLogLikelihood*)*, 42
proflogl.gb2, *36*
proflogl.gb2 *(*ProfLogLikelihood*)*, 42
ProfLogLikelihood, *24*, 42
profml.gb2, *33*
profml.gb2 *(*MLprofGB2*)*, 36
profscores.gb2, *36*
profscores.gb2 *(*ProfLogLikelihood*)*, 42

qcgb2 *(*CompoundQuantiles*)*, 18
qgb2, *30*
qgb2 *(*gb2*)*, 26
qsr.cgb2 *(*CompoundIndicators*)*, 15
qsr.gb2 *(*Indicators*)*, 28

rcgb2 *(*CompoundQuantiles*)*, 18
rgb2 *(*gb2*)*, 26
rmpg.cgb2 *(*CompoundIndicators*)*, 15
rmpg.gb2 *(*Indicators*)*, 28
RobustWeights, 43
robwts *(*RobustWeights*)*, 43

saveplot *(*PlotsML*)*, 42
scores.cavgb2 *(*CompoundAuxFit*)*, 6
scores.cgb2 *(*CompoundFit*)*, 13
scoresh.gb2 *(*LogLikelihood*)*, 31
scoresp.gb2 *(*LogLikelihood*)*, 31
scoreU.cavgb2 *(*CompoundAuxVarest*)*, 9
scoreU.cgb2 *(*CompoundVarest*)*, 20
scorez.cavgb2 *(*CompoundAuxVarest*)*, 9
sl.gb2 *(*Moments*)*, 37
survey, *9*, *11*, *21*
svydesign, *10*, *20*

Thomae, *40*, 44

ULg *(Thomae)*, 44

Varest, 46
varscore.cavgb2 *(CompoundAuxVarest)*, 9
varscore.cgb2 *(CompoundVarest)*, 20
varscore.gb2 *(Varest)*, 46
veind.cavgb2 *(CompoundAuxVarest)*, 9
veind.cgb2 *(CompoundVarest)*, 20
veind.gb2 *(Varest)*, 46
vepar.cavgb2 *(CompoundAuxVarest)*, 9
vepar.cgb2 *(CompoundVarest)*, 20
vepar.gb2 *(Varest)*, 46
vl.gb2 *(Moments)*, 37
vofp.cgb2 *(CompoundFit)*, 13