Package 'SGB'

May 10, 2019

Type Package

Title Simplicial Generalized Beta Regression

Version 1.0

Date 2019-05-09

Author Monique Graf

Maintainer Monique Graf <monique.p.n.graf@bluewin.ch>

Description Main properties and regression procedures using a generalization of the Dirichlet distribution called Simplicial Generalized Beta distribution. It is a new distribution on the simplex (i.e. on the space of compositions or positive vectors with sum of components equal to 1). The Dirichlet distribution can be constructed from a random vector of independent Gamma variables divided by their sum. The SGB follows the same construction with generalized Gamma instead of Gamma variables. The Dirichlet exponents are supplemented by an overall shape parameter and a vector of scales. The scale vector is itself a composition and can be modeled with auxiliary variables through a log-ratio transformation. Graf, M. (2017, ISBN: 978-84-947240-0-8). See also the vignette enclosed in the package.

Depends Formula

Imports stats, MASS, numDeriv, alabama

Suggests knitr, goftest

License GPL(>=2)

Encoding latin1

LazyData true

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

`SGB-package` *Package SGB*

Description

Package SGB contains a generalization of the Dirichlet distribution, called the Simplicial Generalized Beta (SGB). It is a new distribution on the simplex (i.e. on the space of compositions or positive vectors with sum of components equal to 1). The Dirichlet distribution can be constructed from a random vector of independent Gamma variables divided by their sum. The SGB follows the same construction with generalized Gamma instead of Gamma variables. The Dirichlet exponents are supplemented by an overall shape parameter and a vector of scales. The scale vector is itself a composition and can be modeled with auxiliary variables through a log-ratio transformation.

Details

Index of help topics:

```
B2i                     Balances to isometric log-ratio
EZ.SGB                  Expectations of Z under the SGB distribution
EqualityConstr          Equality constraints for overall shape and/or
                        regression parameters and jacobian
GenGammaDistrib         Generalized Gamma distribution
GoodnessFit             Goodness of fit tests on the marginal
                        distributions of each part in a SGB model
Imputation              Imputation of missing parts in compositions
                        from a SGB model
InequalityConstr        Inequality constraints and jacobian
InitialParameters       Initial parameters estimates and comparison
```

MarginPlots	Histograms, quantile and probability plots for the z(u)-transforms of parts
SGB-package	Package SGB
SGBLik	SGB log-likelihood and gradient
SGBdistrib	Density and random generator for the SGB distribution
SGButil	Computation of scales and z-vectors
Tabulation	Tabulation of overall SGB regression results with AIC and matrix view of regression coefficients
arc	arc dataset
carseg	carseg dataset
covest.SGB	Classical and robust asymptotic covariance matrix
ocar	ocar data set
oilr	oilr data set
regSGB	Regression for compositions following a SGB distribution
stepSGB	Stepwise backward elimination for SGB regression
summaryA.SGB	Aitchison expectation and mode under the SGB distribution

Further information is available in the following vignettes:

vignette	SGB multivariate regression (source)

Author(s)

Monique Graf

Maintainer: Monique Graf <monique.p.n.graf@bluewin.ch>

References

Graf, M. (2017). A distribution on the simplex of the Generalized Beta type. *In J. A. Martin-Fernandez (Ed.), Proceedings CoDaWork 2017*, University of Girona (Spain), 71-90.

Examples

```
## Result of a regression object:
summary(oilr)
```

`arc` *arc dataset*

Description

39 (sand,silt,clay) compositions in an Arctic lake in function of depth.

Usage

```
data("arc")
```

Format

A data frame with 39 observations on the following 4 variables.

sand sand part

silt silt part

clay clay part

depth depth (m)

Source

Aitchison, J. (1986). *The Statistical Analysis of Compositional Data.*. Monographs on Statistics and Applied Probability. Chapman and Hall Ltd (reprinted 2003 with additional material by the Blackburn Press, London (UK).

References

Coakley, J.P. and Rust, B.R. (1968). Sedimentation in an Arctic lake. *J. Sed. Petrology*, **38**, 1290-1300.

Examples

```
data(arc)
str(arc)
```

B2i *Balances to isometric log-ratio*

Description

Coefficients of log of parts in a balance matrix, (+1) for numerator and (-1) for denominator, are transformed into the corresponding isometric log-ratio (ilr) coefficients

Usage

```
B2i(bal, balnames=FALSE)
```

Arguments

`bal`	a ($D-1$ x D) balance matrix with cells +1, 0 or -1.
`balnames`	logical, if TRUE, balance names are attributed to ilr transforms; if FALSE (default) ilr transforms are numbered $ilr1$ to $ilrD1$, where $D1 = D-1$ and D is the number of parts.

Details

Two scalars multiplying positive and negative cells respectively are defined for each row of the matrix bal in such a way that the resulting matrix defines the ilr transformation to apply to the log of a compositional vector. The output transformation matrix is transposed for application to a compositional dataset where the compositions are the rows.

Value

a $D \times (D-1)$ matrix giving the coefficients of the ilr transforms

References

Pawlowsky-Glahn, V., J. J. Egozcue, and R. Tolosana-Delgado (2007). Lecture Notes on Compositional Data Analysis.

Examples

```
bal <- matrix(c(1,-1,0,1,1,-1),nrow=2, byrow=TRUE)
colnames(bal) <- paste("l.P",1:3,sep="")
bal
B2i(bal)

rownames(bal) <- paste("B",1:2,sep="")
bal
B2i(bal,balnames=TRUE)
B2i(bal)
```

`carseg` *carseg dataset*

Description

Segment shares of car sales in five categories according to the size of the car chassis, with explanatory variables.

Usage

```
data("carseg")
```

Format

A data frame with 152 observations on the following 13 variables.

SA Segment share in category A

SB Segment share in category B

SC Segment share in category C

SD Segment share in category D

SE Segment share in category E

expend quarterly household expenditures

sent monthly confidence indicator made up of several branches

FBCF monthly households investment

PAC binary vector indicating the incentive period

PIB Gross domestic product

price gas oil price

rates monthly short term interest rates

month sequential month number (1 to 150)

Details

This dataset consists of simulated monthly segment market shares (SA to SE) corresponding to the 5 segments of a certain brand during 150 consecutive months (01/2003 to 08/2015). The set of explanatory variables was selected by Morais and Thomas-Agnan (2019) as being the most meaningful to explain the segment shares. Names have been simplified.

References

Morais, J. and Thomas-Agnan, C. (2019), Impact of economic context on automobile market segment shares: a compositional approach, *Case Studies in Business, Industry and Government Statistics*, in press.

Examples

```
data(carseg)
summary(carseg[,(6:12)])
```

`covest.SGB` *Classical and robust asymptotic covariance matrix*

Description

Computation of two covariance matrices of the estimators of parameters in a SGB regression. The first is based on the Hessian and the second is the sandwich estimator.

Usage

```
covest.SGB(x, d, u, V, weight=rep(1,dim(d)[1]), x0 = NULL, hessian = NULL, ind = NULL,
    shape1 = NULL)
```

Arguments

`x`	vector of parameters (shape1,coefi,shape2) where shape1 is the overall shape, coefi is the vector of regression coefficients (see `initpar.SGB`) and shape2 the vector of the D Dirichlet shape parameters; D: number of parts. shape1 and shape2 must be positive.
`d`	data matrix of explanatory variables (with constant vector if required in the model) $(n \times m)$; n: sample size, m: number of auxiliary variables.
`u`	data matrix of compositions (variables to be explained) $n \times D$.
`V`	full rank transformation of log(parts) into log-ratios, matrix $D \times (D-1)$.
`weight`	vector of length n; positive observation weights, default `rep(1,n)`. Should be scaled to sum to n.
`x0`	specification of the initial parameter vector of length $npar$ (optional), default: NULL, no specification.
`hessian`	Hessian matrix (optional), see `regSGB`, default: NULL, no specification. In this case the Hessian is computed numerically.
`ind`	vector of length equal to the number of fixed parameters; specifies the indices of the fixed components in the vector of parameters x (possible for `shape1` and `coefi` (regression coefficients) only).
`shape1`	fixed value of the overall shape parameter, if `heq = heqa.SGB` or `heq = heqab.SGB`. Default is 1.

Details

This function is internally called by regSGB. In this case the Hessian is the output of `auglag` and is numerically computed.
A design based covariance matrix of the parameters can be obtained by linearization as the covariance matrix of the `scores`.

Value

a list with

`summary`	Data frame with `Initial` = `x0` (if specified), `Estimate` = `x`, `StdErr1` = ordinary asymptotic standard error of parameters, `StdErr` = robust asymptotic standard error, `p.value` = asymptotic normal p-value based on `StdErr`. For `shape1`, H_0 is "shape1=shape1", or "shape1=1" if `shape1=NULL`. The other parameters are tested against 0. `signif` = significance code based on p.value.
`scores`	matrix $n \times npar$. Each row contains the (unweighted) derivatives of the log-density at a data point w.r.t the parameters.
`vcov1`	ordinary asymptotic covariance matrix, inverse of minus the Hessian.
`StdErr1`	vector of ordinary asymptotic standard error of parameters.
`varest2`	robust asymptotic covariance matrix.
`StdErr`	vector of robust asymptotic standard error of parameters.

References

Huber, P. J. (1967). The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1, pp. 221-233.

See Also

`regSGB` for creating `oilr`.

Examples

```
data(arc)
data(oilr)

## compositions
da <- as.matrix(log(arc[["depth"]]),ncol=1)
ua <- as.matrix(arc[,1:3])

## ilr transforms
 c1 <- 1/sqrt(2)
 c2 <- 1/sqrt(6)
 Vilr <- matrix(c(-c1,c1,0,-c2,-c2,2*c2),nrow=3)
 colnames(Vilr) <- c("ilr1","ilr2")
 Vilr

 covs <- covest.SGB(oilr[["par"]], da, ua, Vilr)

## Compare the ordinary and robust correlation matrices of parameters estimates.
```

```
## (Ordinary) covariance based on inverse Hessian
vcov1 <- covs[["vcov1"]]
StdErr1 <- covs[["StdErr1"]]
## Estimated correlation matrix
vcor1 <- diag(1/StdErr1) %*% vcov1 %*% diag(1/StdErr1)
round(vcor1,2)

## Robust (Huber's sandwich estimator):
StdErr <- covs[["StdErr"]]
vcov <- covs[["vcov"]]

## Estimated correlation matrix
round(diag(1/StdErr) %*% vcov %*% diag(1/StdErr),2)
```

`EqualityConstr` *Equality constraints for overall shape and/or regression parameters and jacobian*

Description

Setting of equality constraints on parameters.
`heqa.SGB` sets the overall shape parameter to `shape1`.
`heqb.SGB` sets specified regression parameters to 0.
`heqab.SGB` is a combination of both.
`heqa.SGB.jac`, `heqb.SGB.jac`, `heqab.SGB.jac` compute the corresponding Jacobians.

Usage

```
heqa.SGB(x, d, u, bound, shape1, ...)
heqa.SGB.jac(x, ...)
heqb.SGB(x, d, u, bound, shape1, index, ...)
heqb.SGB.jac(x, d, u, bound, shape1, index, ...)
heqab.SGB(x, d, u, bound, shape1, index, ...)
heqab.SGB.jac(x, d, u, bound, shape1, index, ...)
```

Arguments

x	current vector of parameters (`shape1`, `coefi`, `shape2`) where `shape1` is the overall shape, `coefi` is the vector of regression coefficients (see `initpar.SGB`) and `shape2` the vector of D Dirichlet shape parameters.
d	data matrix of explanatory variables (without constant vector) $(N \times m)$; N: sample size, m: number of explanatory variables.
u	data matrix of compositions (independent variables) $(N \times D)$; D: number of parts.
bound	not used.

shape1 — chosen fixed value of the overall shape parameter.
Default shape1 = 1 for heqa.SGB and heqab.SGB.
shape1 is not fixed in heqb.SGB.

index — vector of length equal to the number of fixed parameters; specifies the indices of the fixed components in the vector of parameters x, such that for
heqa.SGB,heqa.SGB.jac: index=1. The fixed value of the overall shape parameter is shape1 (by default 1).
heqb.SGB,heqb.SGB.jac: index= c(...) with ... the indices of regression parameters to be set to 0.
heqab.SGB,heqab.SGB.jac: index=c(1,...); shape1 is the fixed value of the overall shape parameter, and ... the indices of the regression parameters to be set to 0.

... — not used.

Details

These functions are invoked by regSGB through the specification of the function name, shape1 and/or index.

Value

heqa.SGB, heqb.SGB, heqab.SGB: vector of the same length as index specifying the current value of x[index] or x[1]-shape1, where x is the current vector of parameters. It should be near zero at convergence of the regression algorithm.
heqa.SGB.jac, heqb.SGB.jac, heqab.SGB.jac: the corresponding jacobian matrices of dimensions $length(index) \times length(x)$.

See Also

regSGB, summary.regSGB

Examples

```
## parameter vector for a 3 parts composition with one explanatory variable (+ intercept):
x <- c(1,3.2,0.04,0.05,6,7:9)

## shape1 fixed to 1.5:
heqa.SGB(x,d,u,bound,1.5)
heqa.SGB.jac(x)

## Parameters 3 (first slope) and 4 (second intercept) fixed to 0:
heqb.SGB(x,d,u,bound,shape1,c(3,4))
heqb.SGB.jac(x,d,u,bound,shape1,c(3,4))

## Parameters 1, 3, 4 fixed to 1.5, 0, 0 respectively:
heqab.SGB(x,d,u,bound,1.5,c(1,3,4))
heqab.SGB.jac(x,d,u,bound,1.5,c(1,3,4))
```

EZ.SGB *Expectations of Z under the SGB distribution*

Description

Expectations under Lebesgue and Aitchison measures for the transformed composition $Z = C((U/scale)^{shape1})$ and $C(Z^{1/shape1})$, where $C(.)$ is the closure operation.

Usage

```
EZ.SGB(D, x)
```

Arguments

x	vector of parameters (shape1,coefi,shape2) where shape1 is the overall shape, coefi is the vector of regression coefficients (see initpar.SGB) and shape2 the vector of D Dirichlet shape parameters
D	number of parts

Value

A matrix with 4 rows and D columns giving on each row the expectation of parts

EZ	$E(Z)$, expectation under the (ordinary) Lebesgue measure,
EAZ	$E_A(Z)$, expectation under the Aitchison measure,
EZa	$E(Z^{1/shape1})$, expectation under the (ordinary) Lebesgue measure,
EAZa	$E_A(Z^{1/shape1})$, expectation under the Aitchison measure.

See Also

zval

Examples

```
set.seed(1234)
x <- c(2,rnorm(4,0,1),1.8,3.1,4.0)
D <- 3
EZ.SGB(D,x)
```

GenGammaDistrib *Generalized Gamma distribution*

Description

Density and random generation of the generalized gamma distribution.

Usage

```
dggamma(x, shape1, scale, shape2)
rggamma(n, shape1, scale, shape2)
```

Arguments

`x`	vector of positive values
`n`	number of simulated vectors
`shape1`	overall shape parameter
`scale`	vector of scales. Should be of the same length as `x`
`shape2`	vector of Dirichlet parameters. Should be of the same length as `x`.

Details

log density at $x > 0$:
$log(shape1/scale) - lgamma(shape2) + (shape1 * shape2 - 1) * log(x/scale) - (x/scale)^{shape1}$

Value

dggamma: Generalized gamma density evaluated at x

rggamma: Generalized gamma random deviates

References

Stacy, E.W. (1962). "A Generalization of the Gamma Distribution." *Annals of Mathematical Statistics* **33**(3): 1187-1192.

Johnson, N.L.; Kotz, S; Balakrishnan, N. (1994) *Continuous Univariate Distributions*, Volume 1, 2nd Edition. Wiley. ISBN 0-471-58495-9 (Section 17.8.7)

Examples

```
set.seed(12345)
u1 <- rggamma(10,2,1,1.4) # 10 random deviates with scale 1
set.seed(12345)
u <- rggamma(10,2,1:10,1.4) # 10 random deviates with scale 1:10, repectively
u
u/u1
dggamma(u,2,1:10,1.4)
```

`GoodnessFit` *Goodness of fit tests on the marginal distributions of each part in a SGB model*

Description

Kolmogorov-Smirnov goodness of fit tests
Cramer-von-Mises goodness of fit tests.

Usage

```
ks.SGB(u,shape1,shape2,scale,alpha=0.05)
cvm.SGB(u,shape1,shape2,scale,alpha=0.05)

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

Arguments

`u`	data matrix of compositions (independent variables) $(N \times D)$; D: number of parts
`shape1`	positive number, overall shape parameter of the SGB distribution. See `SGBdistrib`.
`shape2`	vector of length D, Dirichlet shape parameters of the SGB distribution. See `SGBdistrib`.
`scale`	matrix of the same dimensions as `u`, containing the shape compositions, or positive number if the scales of all parts are identical. See `SGBdistrib`.
`alpha`	overall level of the test, default 0.05.
`x`	an object of class "testSGB".
`...`	further arguments passed to or from other methods.

Details

`ks.SGB` calls `ks.test` and `cvm.SGB` calls `cvm.test`.
Consider $z = C[(u/scale)^{shape1}]$, where $C[.]$ is the closure operation. The scale compositions $scale$ may be modelled with auxiliary variables. Under the SGB hypothesis, the components of z should be marginally beta-distributed. The functions provide D tests, one for each part.
Theoretically, the parameters should be known and not estimated on the data. Thus the test using estimated parameters is conservative.
The cutoff value is based on the false discovery rate for multiple comparisons (Benjamini and Hochberg, 1995), which is simply `alpha*i/D` for the i-th ordered p-value, i=1,...,D (number of tests). Reject the null hypothesis if at least one p-value is smaller than the cutoff. The overall level is then `alpha`. The proof of the result does not use an independence assumption between the tests.

Value

A list of class 'testSGB' with the following components:

- `method` either "One-sample Kolmogorov-Smirnov test" or "Cramer-von Mises test of goodness-of-fit"
- `Compositions` name of the dataset `u`
- `tests` data frame with D rows and 3 columns: test statistics for each part against the beta distribution and corresponding p-values and cutoff. Any p-value smaller than the cutoff means that the assumption of the beta distribution for all the margins is rejected.

A print method exists for the class "testSGB".

References

Benjamini, Y. and Y. Hochberg (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society.* Series B (Methodological) 57 (1), 289-300.

Birnbaum, Z. W. and Fred H. Tingey (1951), One-sided confidence contours for probability distribution functions. *The Annals of Mathematical Statistics*, **22**/4, 592-596.

Conover, William J. (1971), *Practical Nonparametric Statistics.* New York: John Wiley & Sons. Pages 295-301 (one-sample Kolmogorov test), 309-314 (two-sample Smirnov test).

Csorgo, S. and Faraway, J.J. (1996) The exact and asymptotic distributions of Cramer-von Mises statistics. *Journal of the Royal Statistical Society*, Series B **58**, 221-234.

Durbin, J. (1973), *Distribution theory for tests based on the sample distribution function.* SIAM.

Marsaglia, G., Wai Wan Tsang and Jingbo Wang (2003), Evaluating Kolmogorov's distribution. *Journal of Statistical Software*, **8**/18.

See Also

`SGBdistrib` for the theoretical distribution, `oilr` for the regression results.

Examples

```
## Generate 1000 random variates according to SGB(shape1,rep(1/3,3),shape2)
shape1 <- 0.6
shape2 <- c(10,20, 30)
rnum <- rSGB(1000,shape1,rep(1,3)/3,shape2)
ks.SGB(rnum,shape1=shape1, shape2=shape2,scale=1)
## same result as
ks.SGB(rnum,shape1=shape1,scale= matrix(rep(1/3,3000),ncol=3), shape2=shape2)
library(goftest)
cvm.SGB(rnum,shape1=shape1,scale= matrix(rep(1/3,3000),ncol=3), shape2=shape2)
```

```
## Arctic lake data

# oilr is a SGB regression object, see \code{\link{oilr}}.
data(oilr)      # regSGB object
data(arc)
ua <- arc[1:3]   # compositions

## Kolmogorov-Smirnov goodness of fit test
ks.SGB(ua,shape1=oilr[["par"]][1],shape2=oilr[["par"]][4:6],scale=oilr[["scale"]])
## Rounding shape1 affects the results less than rounding shape2.
ks.SGB(ua,shape1=round(oilr[["par"]][1],3),shape2=round(oilr[["par"]][4:6],1),
 scale=oilr[["scale"]])
ks.SGB(ua,shape1=round(oilr[["par"]][1],1),shape2=round(oilr[["par"]][4:6],3),
 scale=oilr[["scale"]])

## Cramer-von-Mises goodness of fit test
library(goftest)
cvm.SGB(ua,shape1=oilr[["par"]][1],shape2=oilr[["par"]][4:6],scale=oilr[["scale"]])
```

`Imputation` *Imputation of missing parts in compositions from a SGB model*

Description

Applied to a completely missing composition, the function returns the Aitchison expectation.
Applied to a partially missing composition, it returns the conditional Aitchison expectation, given the observed sub-composition.
Applied to a complete case, it returns the complete case.

Usage

```
impute.regSGB(obj, dsup, usup)
```

Arguments

`obj`	list, output of regSGB.
`dsup`	data frame with explanatory variables for the incomplete compositions. Missing values not allowed.
`usup`	compositions corresponding to `dsup`. On each row, the non-missing parts sum to 1.

Value

data frame with imputed compositions instead of missing or partially missing compositions. Complete cases are also returned.

Examples

```
## Arctic lake
data(arc)
arcmis <- arc
arc[11:13,]

## Introduce NA alues

arcmis[11,2] <- NA      # "core" observation
arcmis[12,3] <- NA      # outlying clay value
arcmis[13,1:3] <- NA    # totally missing observation
umis <- arcmis[,1:3]
umis <- umis/rowSums(umis,na.rm=TRUE)
umis[11:13,]

d <- data.frame(depth=arc[["depth"]])

## original compositions
arc[11:13,1:3]

## unconditional predicted value
MeanA.SGB(oilr[["par"]][1],oilr[["scale"]],oilr[["par"]][4:6] )[11:13,]

## predicted value given the sub-composition (sand,clay) for 11, (sand,silt) for 12
impute.regSGB(oilr,arcmis,umis)[11:13, ]

impute.regSGB(oilr,arcmis[11:13, ],umis[11:13, ])  # same result.
```

InequalityConstr *Inequality constraints and jacobian*

Description

Setting of inequality constraints on shape parameters.
`hin.SGB` sets inequality constraints on the shape parameters in a SGB regression.
`hin.SGB.jac` defines the corresponding Jacobian.

Usage

```
hin.SGB(x, d, u, bound, ...)
hin.SGB.jac(x, d, u, ...)
```

Arguments

x	vector of parameters (`shape1`, `coefi`, `shape2`) where `shape1` is the overall shape, `coefi` is the vector of regression coefficients (see `initpar.SGB`) and `shape2` the vector of D Dirichlet shape parameters.
d	data matrix of explanatory variables (without constant vector) $(N \times m)$; N: sample size, m: number of auxiliary variables.

u	data matrix of compositions (independent variables) $(N \times D)$; D: number of parts.
bound	the estimates of shapes are constrained by shape1*shape2[i] > bound, i=1,...,D. By default bound = 2.1.
...	not used.

Details

These functions are invoked internally by regSGB with bound specified by the user.
shape1 is constrained to be larger than 0.1, in order to avoid numerical problems and shape2 must be positive.
Moments of ratios of parts only exist up to bound. Thus bound = 2.1 guarantees the existence of variances of ratios of parts.

Value

hin.SGB : vector of length $D{+}1$ with the current value of c(shape1-0.1,shape1*shape2-bound).
It should be non-negative at convergence of the regression algorithm.
hin.SGB.jac : corresponding jacobian matrix of dimensions $(D+1)\times$ length(x).

Examples

```
## Parameter vector for a 3 parts composition with one explanatory variable (+ intercept):
x <- c(1,3.2,0.04,0.05,6,7:9)
bound <- 2.1
u <- t(c(0.1,0.5,0.4))  # only used to compute the number of parts.
hin.SGB(x, d, u, bound)
# = c(shape1-0.1, shape1*shape2-bound,shape2)
# all must be positive.
```

InitialParameters *Initial parameters estimates and comparison*

Description

initpar.SGB computes an initial vector of parameters.
condshape2 computes the shape2 parameters by the same method as initpar.SGB, but from an arbitrary set of parameters (shape1,coefi) (e.g. the result of a SGB regression fit). These approximations are compared with the shape2 estimates.
compushape2 is internally called by initpar.SGB and condshape2. It computes shape2 parameters in function of shape1 and given regression parameters coefi.

Usage

```
initpar.SGB(d, u, V, weight = rep(1, dim(u)[1]), shape1 = 1, Mean2 = TRUE)
condshape2(x,d,u,V)
compushape2(shape1, coefi, d, u, V)
```

Arguments

`d`	data matrix of explanatory variables (without constant vector) $(n \times m)$; n: sample size, m: number of auxiliary variables
`u`	data matrix of compositions (independent variables) $(n \times D)$; D: number of parts
`V`	full rank transformation of log(parts) into log-ratios, matrix $D \times (D-1)$
`weight`	vector of length n; positive observation weights, default `rep(1,n)`. Should be scaled to sum to n.
`shape1`	positive number, overall shape parameter
`Mean2`	logical, if TRUE (default), the computed `shape2` parameters are each replaced by their average.
`coefi`	vector of regression coefficients of length $(m+1)*(D-1)$, resp. $D-1$ constants, then $D-1$ coef. of the 1st expl. variable,..., $D-1$ coef. of the m-th expl. variable
`x`	fitted SGB regression parameters, see `regSGB`.

Details

The main function here is `initpar.SGB`. The initial value of `shape1` must be specified by the user; by default, it takes the value 1. In the initial regression model, each column of `log(u) %*% V` is regressed by OLS on the columns of `d`. `coefi` is the vector of regression parameters, first the $D-1$ terms associated with the first explanatory variable in `d`, and so on similarly for each explanatory variable. The initial scale compositions are computed by back-transforming the predicted values to the simplex and used to compute the vector $z = C[(u/scale)^{shape1}]$, where $C[.]$ is the closure operation. Wicker et al. (2008), see also Ng et al. (2011) p.74-75, describe a procedure to find initial values for the shape parameters in a Dirichlet distribution. Their method is used on the (approximate) Dirichlet vector z.

Value

`initpar.SGB`:
vector of length $(1+(D-1)*(m+1)+D)$ containing initial values for (`shape1`,`coefi`,`shape2`).
`condshape2`:
list with two components: 1. title and 2. data-frame with 2 columns: fitted `shape2` and Wicker's approximation.

References

Wicker, N., J. Muller, R. K. R. Kalathur, and O. Poch (2008). A maximum likelihood approximation method for Dirichlet's parameter estimation. *Computational Statistics & Data Analysis* **52** (3), 1315-1322.

Kai Wang Ng, Guo-Liang Tian, Man-Lai Tang (2011). *Dirichlet and Related Distributions: Theory, Methods and Applications*. Wiley Series in Probability and Statistics.

Examples

```
## Explanatory variable
da <- data.frame(l.depth=log(arc[["depth"]]))
damat <- as.matrix(da)
## Compositions
ua <- arc[,1:3]

## alr transforms
Va <- matrix(c(1,0,-1,0,1,-1),nrow=3)
colnames(Va) <- c("alr1","alr2")
Va

## Initial values
initpar.SGB(damat,ua,Va)
initpar.SGB(damat,ua,Va,Mean2=FALSE)

## Conditional shape2 values; same as parameters computed with initpar
condshape2(initpar.SGB(damat,ua,Va,Mean2=FALSE),damat,ua,Va)

## Comparison with fitted parameters
oa <- regSGB(damat, as.matrix(ua), Va)
condshape2(oa[["par"]],damat,ua,Va)
```

MarginPlots — *Histograms, quantile and probability plots for the z(u)-transforms of parts*

Description

These functions draw a plot for each part in the dataset.

Usage

```
hzbeta(u, obj, weight = rep(1,dim(u)[1]) )
qzbeta(u, obj, weight = rep(1,dim(u)[1]) )
pzbeta(u, obj, weight = rep(1,dim(u)[1]) )
```

Arguments

`u`	data matrix of compositions (independent variables) $(N \times D)$; D: number of parts
`obj`	list, result of regSGB. See `regSGB`.
`weight`	vector of length n; positive observation weights, default rep(1,n).

Details

Let U follow a $SGB(shape1, scale, shape2)$ distribution. Then the composition

$$Z = C[(U/scale)^{shape1}]$$

is called the $z(u)$-transform of U.
Z follows a $Dirichlet(shape2)$ distribution and each part $Z_i, i = 1, ..., D$ is Beta-distributed with parameters `(shape2[i],sum(shape2)-shape2[i])`.
Goodness of fit plots are produced for the parts of the $z(u)$-transforms against the Beta distribution.
Each function creates D plots, where D is the number of parts.
`hzbeta`: histograms and the corresponding Beta-densities,
`qzbeta`: marginal quantile plots,
`pzbeta`: marginal probability plots.
If `weight` is specified, weighted histgrams, quantile and probability plots are drawn.

Value

D plots are produced comparing the marginal distribution of the parts of the $z(u)$ compositions with the theoretical Beta distribution.

Examples

```
## Arctic lake data
data(arc)
# Compositions
ua <- arc[,1:3]

# SGB regression
data(oilr)

# plot
par(mfrow=c(3,3))
hzbeta(ua,oilr)
qzbeta(ua,oilr)
pzbeta(ua,oilr)
```

`ocar` *ocar data set*

Description

Car segment shares SGB regression with formula

Usage

```
data("ocar")
```

Format

List of 25 items, see `regSGB`.

Details

`ocar` is the same regression as `object3` in `regSGB`, Example 3.

Examples

```
data(ocar)
ocar
summary(ocar)   # regSGB summary
ocar[["kkt1"]]  # first KKT condition
ocar[["V"]]     # matrix of log-ratio transformation

#########################################################
## ocar has been created by the following commands:
## Car segment shares
 data(carseg)

## Extract the compositions
uc <- as.matrix(carseg[,(1:5)])

## Define the log-ratio transformation matrix
Vc <- matrix(c( 1, 0, 0, 0,
               -1, 1, 0, 0,
                0,-1, 1, 0,
                0, 0,-1, 1,
                0, 0, 0,-1),ncol=4,byrow=TRUE)
colnames(Vc) <- c("AB","BC","CD","DE")
rownames(Vc) <- colnames(uc)
Vc

## Formula
Form <- Formula(AB | BC | CD | DE ~  log(expend) + I(PAC*log(expend)) + log(sent) + log(FBCF) +
                  log(price) + rates)
ocar <- regSGB(Form, data = list(carseg, uc, Vc),shape10=4.4)
############################################################
```

`oilr` *oilr data set*

Description

Arctic lake SGB regression based on isometric log-ratio transforms

Usage

```
data("oilr")
```

Format

List of 25 items, see `regSGB`.

Examples

```
data(oilr)
oilr
summary(oilr)   # regSGB summary
oilr[["kkt1"]]  # first KKT condition
oilr[["V"]]     # matrix of log-ratio transformation

## oilr has been created by the following commands:
## Arctic lake data
 data(arc)
# Compositions
 ua <- arc[,1:3]

## ilr transforms
 c1 <- 1/sqrt(2)
 c2 <- 1/sqrt(6)
 Vilr <- matrix(c(-c1,c1,0,-c2,-c2,2*c2),nrow=3)
 colnames(Vilr) <- c("ilr1","ilr2")
 Vilr

## Formula
 F1 <- Formula(ilr1 | ilr2 ~ -1 + log(depth) )
# SGB regression object
 oilr <- regSGB(F1, data= list(arc, ua, Vilr), shape10=0.5, bound=2.1)
##########################################################
```

---

regSGB *Regression for compositions following a SGB distribution*

Description

Explanatory variables may influence the scale vector through a linear model applied to a log-ratio transform of the compositions. The shape parameters do not depend on explanatory variables. The overall shape parameter shape1 is common to all parts, whereas the Dirichlet shape parameters vector shape2 are specific to each part, i.e. shape2[j] is the Dirichlet parameter for u[i,j], i=1,...,n, (n=number of compositions in the dataset u).

Usage

```
regSGB(d, ...)

## Default S3 method:
regSGB(d, u, V, weight=rep(1,dim(d)[1]),
    shape10 = 1, bound = 2.1, ind = NULL, shape1 = NULL, Mean2 = TRUE,
    control.optim = list(trace=0,fnscale=-1),
    control.outer = list(itmax=1000,ilack.max=200,trace=TRUE, kkt2.check =TRUE,
    method = "BFGS"),...)
```

```
## S3 method for class 'formula'
regSGB(Formula, data= list(), weight=rep(1,dim(d)[1]),
    shape10 = 1, bound = 2.1, ind = NULL, shape1 = 1,  Mean2=TRUE,
    control.optim = list(trace=0,fnscale=-1),
    control.outer = list(itmax=1000,ilack.max=200,trace=TRUE,kkt2.check =TRUE,
    method = "BFGS"),...)

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

## S3 method for class 'regSGB'
summary(object, digits=3,...)
```

Arguments

`Formula`	formula of class Formula, see `Formula`.
`d`	data matrix of explanatory variables (without constant vector) $(n \times m)$; n: sample size, m: number of auxiliary variables.
`u`	data matrix of compositions (independent variables) $(n \times D)$; D: number of parts.
`V`	log-ratio transformation matrix $(D \times (D-1))$.
`data`	a list with 3 components `d`, `u` and `V`.
`weight`	vector of length n; positive observation weights, default rep(1,n). Should be scaled to sum to n.
`shape10`	positive number, initial value of the overall shape parameter, default 1.
`bound`	inequality constraints on the estimates of shapes: `shape1*shape2[i]` > `bound,i=1,...,D`. By default `bound` = `2.1`, see `InequalityConstr`.
`ind`	vector of length equal to the number of fixed parameters; see `index` in `EqualityConstr`. Default `ind` = NULL (no fixed parameters).
`shape1`	fixed value of the overall shape parameter if `min(ind)=1`. Default is 1.
`Mean2`	logical, if TRUE (default), the computed `shape2` parameters are each replaced by their average. See `initpar.SGB`.
`control.optim`	list of control parameters for optim, see `optim`. Default is from `auglag`, except `list(fnscale = -1)`. Always specify `fnscale = -1`.
`control.outer`	list of control parameters to be used by the outer loop in `constrOptim.nl`, see `auglag`. Default is from `auglag`, except `list(itmax = 1000,ilack.max = 200`.
`object`	an object of class "regSGB".
`digits`	number of decimal places for print, default 3.
`x`	an object of class "regSGB".
`...`	not used.

Details

It is advisable to use the formula to specify the model for easy comparison between models. Without formula, the `d` matrix of explanatory variables must contain exactly the variables used in the model, whereas with formula other variables can be included as well. Variable transformations can be utilized within the formula, see Example 4 below with the indicator `I` and the log.
Constraints on parameters can be introduced, see example 5 and `EqualityConstr` for more details.
Use `weight` for pseudo-likelihood estimation. `weight` is scaled to n, the sample size.
A design based covariance matrix of the parameters can be obtained by linearization as the covariance matrix of the `scores`.

Value

A list of class 'regSGB' with the following components:
The first 13 form the output from `auglag`.

`par`	Vector of length $npar$. Parameters that optimize the nonlinear objective function, satisfying constraints, if convergence is successful.
`value`	The value of the objective function at termination.
`counts`	A vector of length 2 denoting the number of times the objective and its gradient were evaluated, respectively.
`convergence`	An integer code indicating the type of convergence. 0 indicates successful convergence. Positive integer codes indicate failure to converge.
`message`	A character string giving any additional information on convergence returned by `optim`, or NULL.
`outer.iteration`	Number of outer iterations.
`lambda`	Values of the Lagrangian parameter. This is a vector of the same length as the total number of inequalities and equalities. It must be zero for inactive inequalities; non-negative for active inequalities; and can have any sign for equalities.
`sigma`	Value of augmented penalty parameter for the quadratic term.
`gradient`	Gradient of the augmented Lagrangian function at convergence. It should be small.
`hessian`	Hessian of the augmented Lagrangian function at convergence. It should be negative definite for maximization.
`ineq`	Values of inequality constraints at convergence. All of them must be non-negative.
`equal`	Values of equality constraints at convergence. All of them must be close to zero.
`kkt1`	A logical variable indicating whether or not the first-order KKT conditions were satisfied (printed 1 if conditions satisfied and 0 otherwise).
`kkt2`	A logical variable indicating whether or not the second-order KKT conditions were satisfied (printed 1 if conditions satisfied and 0 otherwise).
`scale`	$n \times D$ matrix, the estimated scale compositions, see `bval`.
`meanA`	Aitchison expectation at estimated parameters.
`fitted.values`	$(n \times (D-1))$ matrix, estimated log-ratio transforms.

`residuals`	Observed minus estimated log-ratio transforms.
`scores`	matrix $n \times npar$. Each row contains the (unweighted) derivatives of the log-density at a data point w.r.t the parameters.
`Rsquare`	ratio of total variation of `meanA` and total variation of compositions `u`.
`vcov`	The robust covariance matrix of parameters estimates, see `covest.SGB`.
`StdErr1`	Ordinary asymptotic standard errors of parameters.
`StdErr`	Robust asymptotic standard errors of parameters.
`fixed.par`	Indices of the fixed parameters.
`summary`	The summary from `covest.SGB`.
`AIC`	AIC criterion.
`V`	log-ratio transformation matrix (same as corresponding input parameter `V`)
`call`	Arguments for calling `regSGB`.
`Formula`	Expression for formula.

References

Graf, M. (2017). A distribution on the simplex of the Generalized Beta type. *In J. A. Martin-Fernandez (Ed.), Proceedings CoDaWork 2017*, University of Girona (Spain), 71-90.

Hijazi, R. H. and R. W. Jernigan (2009). Modelling compositional data using Dirichlet regression models. *Journal of Applied Probability and Statistics*, **4** (1), 77-91.

Kotz, S., N. Balakrishnan, and N. L. Johnson (2000). *Continuous Multivariate Distributions*, Volume 1, Models and Applications. John Wiley & Sons.

Madsen, K., H. Nielsen, and O. Tingleff (2004). Optimization With Constraints. *Informatics and Mathematical Modelling*, Technical University of Denmark.

Monti, G. S., G. Mateu-Figueras, and V. Pawlowsky-Glahn (2011). Notes on the scaled Dirichlet distribution. In *V. Pawlowsky-Glahn and A. Buccianti (Eds.), Compositional data analysis. Theory and applications.* Wiley.

Varadhan, R. (2015). alabama: Constrained Nonlinear Optimization. R package version 2015.3-1.

Wicker, N., J. Muller, R. K. R. Kalathur, and O. Poch (2008). A maximum likelihood approximation method for Dirichlet parameter estimation. Computational Statistics & Data Analysis 52 (3), 1315-1322.

Zeileis, A. and Y. Croissant (2010). Extended model formulas in R: Multiple parts and multiple responses. *Journal of Statistical Software* **34** (1), 1-13.

See Also

`stepSGB`, for an experimental stepwise descending regression, `initpar.SGB`, for the computation of initial parameters. This function uses `Formula`, `auglag`.

Examples

```
## Regression for car segment shares
## --------------------------------
data(carseg)
## Extract the compositions
uc <- as.matrix(carseg[,(1:5)])

## Extract the explanatory variables
attach(carseg)

## Example 1: without formula
## --------------------------
## Change some variables
dc <- data.frame(l.exp1=log(expend)*PAC,l.exp0=log(expend)*(1-PAC), l.sent=log(sent),
l.FBCF=log(FBCF), l.price=log(price), rates)

## Define the log-ratio transformation matrix
Vc <- matrix(c( 1,0,0,0,
               -1,1,0,0,
               0,-1,1,0,
               0,0,-1,1,
               0,0,0,-1),ncol=4,byrow=TRUE)
colnames(Vc) <- c("AB","BC","CD","DE")
rownames(Vc) <- colnames(uc)
Vc

# 2 next rows  only necessary when calling regSGB without a formula.
dc1 <- cbind("(Intercept)"= 1 , dc)
dc1 <- as.matrix(dc1)

object10 <- regSGB(dc1,uc, Vc,shape10=4.4)
summary(object10)

## Example 2: same with formula
## ----------------------------
## Define the formula
Form <- Formula(AB | BC | CD | DE ~  l.exp1 + l.exp0 + l.sent + l.FBCF + l.price +  rates)

## Regression with formula
object1 <- regSGB(Form, data= list(dc, uc, Vc),shape10=4.4)

summary(object1)

## Example 3: Usage of I()
## -----------------------
Form2 <- Formula(AB | BC | CD | DE ~  I(l.exp1 + l.exp0) + l.exp1 +l.sent +
                 l.FBCF + l.price + rates )
object2 <- regSGB(Form2,data= list(dc, uc, Vc),shape10=4.4)
object2

## Example 4: Usage of variable transformations on the original file
## -----------------------------------------------------------------
```

```
Form3 <- Formula(AB | BC | CD | DE ~  log(expend) + I(PAC*log(expend)) + log(sent) + log(FBCF) +
                 log(price) + rates)
object3 <- regSGB(Form3, data=list(carseg, uc, Vc),shape10=4.4)
object3
object2[["par"]]-object3[["par"]]    # same results

## Example 5: Fixing parameter values
## ------------------------------------
## 1. In the following regression we condition on shape1 = 2.36.
object4 <- regSGB(Form3,data=list(carseg, uc, Vc),
                  shape10 = 4.4,  bound = 2.0, ind = 1, shape1 = 2.36)
summary(object4)

## 2. In the following regression we condition on shape1 = 2.36 and the  coefficient of
## log(FBCF).BC = 0.  Notice that it is the 19th parameter.
object5 <- regSGB(Form3,data=list(carseg, uc, Vc),
                  shape10 = 4.4, bound = 2.0, ind = c(1,19) , shape1 = 2.36)
summary(object5)

object3[["AIC"]]
object4[["AIC"]]  # largest AIC
object5[["AIC"]]
```

SGBdistrib *Density and random generator for the SGB distribution*

Description

dSGB computes the density for a given argument u (a composition) and given parameters.
rSGB generates n compositions for given parameters.

Usage

```
dSGB(u, shape1, scale, shape2)
rSGB(n, shape1, scale, shape2)
```

Arguments

u	vector of length D containing the composition
shape1	overall shape parameter. shape1 = 1 for a Dirichlet composition.
scale	vector of the same length as u containing the scales of parts. If missing, scales are set to 1.
shape2	vector of length D containing the (Dirichlet) shapes for each part.
n	number of observations.

Details

The number of columns in u and the number of components in shape2 must match.

Value

dSGB gives the density, rSGB generates a $(n \times D$ - matrix with random compositions on each row.

See Also

bval,zval

Examples

```
   u1 <- c(0.2,0.3,0.5)
   scale1 <- c(0.25,0.33,0.32)
   shape1 <- 1
   shape2 <- c(0.8,3,0.9)
   dSGB(u1,shape1,scale1,shape2)
   rSGB(10,shape1,scale1,shape2)


## with equal scales
   dSGB(u1,shape1,shape2=shape2)
   rSGB(10,shape1,shape2=shape2)
```

SGBLik — *SGB log-likelihood and gradient*

Description

fn.SGB gives the log-likelihood and gr.SGB the gradient vector of the log-likelihood.

Usage

```
fn.SGB(x, d, u, V,  weight, ...)
gr.SGB(x, d, u, V, weight, ...)
```

Arguments

x	vector of parameters (shape1, coefi, shape2) where shape1 is the overall shape, coefi is the vector of regression coefficients (see initpar.SGB) and shape2 the vector of D Dirichlet shape parameters.
d	data matrix of explanatory variables (without constant vector) (nxm); n: sample size, m: number of auxiliary variables
u	data matrix of compositions (independent variables) (nxD); D: number of parts
V	full rank transformation of log(parts) into log-ratios, matrix $(Dx(D-1))$
weight	vector of length n; positive observation weights, default rep(1,n). Should be scaled to sum to n.
...	others parameters that might be introduced.

Details

The analytical expression for `fn.SGB` is found in the vignette "SGB regression", Section 3.2. More details in Graf(2017).

Value

fn.SGB: value of the log-likelihood at parameter `x`
gr.SGB: gradient vector at parameter `x`.

References

Graf, M. (2017). A distribution on the simplex of the Generalized Beta type. *In J. A. Martin-Fernandez (Ed.), Proceedings CoDaWork 2017*, University of Girona (Spain), 71-90.

See Also

regSGB

Examples

```
## Explanatory variable
da <- data.frame(l.depth=log(arc[["depth"]]))
damat <- as.matrix(da)

## Compositions
ua <- as.matrix(arc[,1:3])

## alr transforms
Va <- matrix(c(1,0,-1,0,1,-1),nrow=3)
colnames(Va) <- c("alr1","alr2")
Va

## Initial values
x <- initpar.SGB(damat,ua,Va)
fn.SGB(x, damat, ua, Va,weight=rep(1,dim(da)[1]))
gr.SGB(x, damat, ua, Va,weight=rep(1,dim(da)[1]))
```

SGButil *Computation of scales and z-vectors*

Description

bval computes the scale for each observed composition from the parameters and auxiliary variables for that observation.
zval computes the z-vector for each observed composition, i.e. the transform that is Dirichlet distributed under the SGB model for the observed composition.

Usage

```
bval(D, x, d, V)
zval(u, x, d, V)
```

Arguments

- `D` number of parts
- `x` vector of parameters (`shape1`,`coefi`,`shape2`) where `shape1` is the overall shape, `coefi` is the vector of regression coefficients (see `initpar.SGB`) and `shape2` the vector of D Dirichlet shape parameters
- `d` $(n \times m)$ - data matrix of explanatory variables (variables corresponding to `coefi`); n: sample size, m: number of auxiliary variables
- `u` $(n \times D)$ - data matrix of compositions (independent variables); D: number of parts
- `V` $D \times (D-1)$ - matrix specifying the full rank transformation of log(parts) into log-ratios

Details

See Graf (2017), Equation (8), or the vignette "SGB regression", Equation (1).

Value

transformed composition of length D.

References

Graf, M. (2017). A distribution on the simplex of the Generalized Beta type. *In J. A. Martin-Fernandez (Ed.), Proceedings CoDaWork 2017*, University of Girona (Spain), 71-90.

Examples

```
## Example with 2 compositions
u <- matrix(c(0.2,0.4,0.5,0.5,0.3,0.2),nrow=2,byrow=TRUE)
u
D <- NCOL(u)  # number of parts

## auxiliary variable
d <- matrix(c(3.2,4.6),ncol=1)

## log-ratio transformation
V <- matrix(c(c(1,-1,0)/sqrt(2),c(1,1,-2)/sqrt(6)),ncol=2)

## vector of parameters:
shape1 <- 2.00
coefi <- c(-0.78,  0.06,  0.96, -0.11)
shape2 <- c(1.80,  3.10,  4.00)
x <-c(shape1, coefi, shape2)
```

```
bval(D,x,d,V)
zval(u,x,d,V)
```

---

stepSGB *Stepwise backward elimination for SGB regression*

Description

Stepwise elimination of the non significant regression parameters. Possibility to assign a fixed value `shape1` to the overall shape parameter.

Usage

```
stepSGB(obj0, d, u,  weight = rep(1, dim(d)[1]), shape10 = obj0[["par"]][1],
  bound = 2.1, shape1 = NULL, Mean2 = TRUE, maxiter = 10,
  control.optim = list(fnscale = -1),
  control.outer = list(itmax = 1000, ilack.max = 200, trace = TRUE,
   kkt2.check = TRUE, method = "BFGS") )
```

Arguments

`obj0`	object of class regSGB, see `regSGB`.
`d`	data matrix of explanatory variables (without constant vector) $(n \times m)$; n: sample size, m: number of auxiliary variables.
`u`	data matrix of compositions (independent variables) $(n \times D)$; D: number of parts.
`weight`	vector of length n; positive observation weights, default `rep(1,n)`. Should be scaled to sum to n.
`shape10`	positive number, initial value of the overall shape parameter, default obj0[["par"]][1].
`bound`	inequality constraints on the estimates of shapes: `shape1*shape2[i]` > `bound,i=1,...,D`. By default bound = `2.1`, see `InequalityConstr`.
`shape1`	fixed value of the overall shape parameter. Default is NULL (no fixed value).
`Mean2`	logical, if TRUE (default), the initial shape2 parameters are each replaced by their average. See `initpar.SGB`.
`maxiter`	maximum number of iterations, i.e. attempts to set a parameter to 0.
`control.optim`	list of control parameters for optim, see `optim`. Default is from `auglag`, except `list(fnscale = -1)`. Always specify `fnscale = -1`.
`control.outer`	list of control parameters to be used by the outer loop in constrOptim.nl, see `auglag`. Default is from `auglag`, except `list(itmax = 1000,ilack.max = 200)`.
`...`	not used.

Details

This is an experimental procedure for searching a set of non-significant parameters that will be set to zero. The shape parameters are excluded from the elimination procedure. The algorithm starts with `obj0`, output of regSGB. The p-values for the regression parameters in `summary(obj0)` are taken in decreasing order. The parameter with the largest p-value is set to zero and `regSGB` computes the regression with this constraint. If the AIC value is smaller than the AIC in `obj0`, the parameter with the next largest p-value in `obj0` is set to zero and the regression with the two constraints is computed. The process iterates until either a larger AIC is found or `maxiter` is attained.
The initial value of the overall shape parameter is set to the estimated value in the full model `obj0`. The other initial values are computed as in `regSGB`.
There is the possibility to fix the value of the overvall shape parameter, if `shape1` is given a positive number a_0 (default NULL, no fixed value).
If `regSGB` was called without `Formula`, the data-frame with auxiliary variables for `stepSGB` follows the same rules as for the initial regSGB object, see Example 1 in `regSGB`.

Value

A list of class 'stepSGB' with the following 5 components:

`reg` A list with the following components:
`full` Object of class regSGB, same as `obj0`, see `regSGB`.
`iter1` Object of class regSGB obtained at iteration 1.
...
`iterk` Object of class regSGB obtained at iteration k.

`Formula` The original formula, or NULL

`iter` Value of k, the last iteration.

`tab` Data frame with `k+1` columns, overall results and `k` iterations. The rows are
`value` Log-likelihood
`n.par` Total number of parameters (including the shape2 param.)
`n.par.fixed` Number of fixed parameters.
`AIC` Value of the AIC criterion.
`convergence` 0 if converged.
`kkt1` 1 if first Karush-Kuhn-Tucker criterion fulfilled, zero otherwise.
`kkt2` 1 if second Karush-Kuhn-Tucker criterion fulfilled, zero otherwise.
`counts.function` Number of times the objective function (the log-likelihood) was evaluated.
`counts.gradient` Number of times the gradient was evaluated. \

`call` Arguments for calling `stepSGB`.

References

`vignette("SGB regression", package = "SGB")`

See Also

`regSGB`, `initpar.SGB`, `auglag`.

Examples

```
data(carseg)
## Extract the compositions
uc <- as.matrix(carseg[,(1:5)])
## Initial regression
data(ocar)

step_ocar <- stepSGB(ocar, carseg, uc, bound=2.1, control.outer=list(trace=FALSE))

summary(step_ocar[["reg"]][["full"]])
summary(step_ocar[["reg"]][["iter4"]])
step_ocar[["tab"]]
```

`summaryA.SGB` *Aitchison expectation and mode under the SGB distribution*

Description

The expectation and mode in the log-ratio space, transformed back to the simplex.

Usage

```
MeanA.SGB(shape1, scale, shape2)
ModeA.SGB(shape1, scale, shape2)
MeanAobj.SGB(obj)
ModeAobj.SGB(obj)
```

Arguments

`shape1`	overall shape parameter. `shape1` = 1 for a Dirichlet composition.
`scale`	vector of length D or matrix with D columns containing the scales of parts.
`shape2`	vector of length D containing the (Dirichlet) shapes for each part.
`obj`	list, result of regSGB. See `regSGB`.

Details

`MeanA`, `ModeA` compute Aitchison expectation and mode in function of the SGB distribution parameters, whereas `MeanAobj`, `ModeAobj` compute Aitchison expectation and mode in function of the model variables in an SGB regression object.

Value

A matrix or vector of dimensions $(n \times D)$. Each row gives the Aitchison expectation for compositions having the corresponding set of auxiliary variables.

References

Aitchison, J. (1986). *The Statistical Analysis of Compositional Data*. Monographs on Statistics and Applied Probability. Chapman and Hall Ltd (reprinted 2003 with additional material by the Blackburn Press, London (UK).

See Also

oilr.

Examples

```
   set.seed(1234)
   x <- c(2,rnorm(4,0,1),1.8,3.1,4.0)
   d <- c(3.2,4.6)
   V <- t(matrix(c(1/sqrt(2),-1/sqrt(2),0,
                 1/sqrt(6),1/sqrt(6),-2/sqrt(6)),
                 nrow=2,byrow=TRUE))
   D <- 3
   shape1 <- x[1]
   scale <- bval(D,x,d,V)
   shape2 <- x[(length(x)-D+1):length(x)]
# Expectation
   MeanA.SGB(shape1,scale,shape2)
# Mode
   ModeA.SGB(shape1, scale, shape2)

## Arctic lake data
# oilr is a SGB regression object
data(oilr)
MeanAobj.SGB(oilr)   # is the same as oilr[["meanA"]]
ModeAobj.SGB(oilr)
```

`Tabulation` *Tabulation of overall SGB regression results with AIC and matrix view of regression coefficients*

Description

table.regSGB: Value of the log-likelihood, number of parameters, AIC criterion, optimality tests and iterations counts.
coefmat: regression coefficients in matrix form with significance level.

Usage

```
table.regSGB(object)
coefmat(object,digits=3)
```

Arguments

`object`	an object of class regSGB
`digits`	number of decimal places for the coefficients

Value

`table.regSGB`: Data frame with one column, with the overall statistics results.

`value`	the maximum log-likelihood
`n.par`	the number of parameters
`n.par.fixed`	the number of fixed parameters
`AIC`	the AIC criterion
`Rsquare`	total variance of estimated over total variance of observed compositions
`convergence`	the convergence code (0: converged, others, see `auglag`).
`kkt1`	the first Karush-Kuhn-Tucker conditions (1=TRUE, 0=FALSE), see `auglag`.
`kkt2`	the second Karush-Kuhn-Tucker conditions (1=TRUE, 0=FALSE), see `auglag`.
`counts.function`	number of times the log-likelihood was evaluated.
`counts.gradient`	number of times the gradient was evaluated.

`coefmat`: character matrix with the regression coefficients arranged in columns, one for each log-ratio transform. Each ceofficient is followed by the significance level.

See Also

`regSGB`, `oilr`, `auglag`.

Examples

```
## Overall model statistics
table.regSGB(oilr)
##
print(coefmat(oilr),quote=FALSE)
## it is a subset of
summary(oilr)
```

Index