Package 'GLDEX'

July 21, 2025

Version 2.0.0.9.3

Date 2023-08-21

Title Fitting Single and Mixture of Generalised Lambda Distributions

Maintainer Steve Su <allegro.su@gmail.com>

Depends cluster, grDevices, graphics, stats, spacefillr

Description

The fitting algorithms considered in this package have two major objectives. One is to provide a smoothing device to fit distributions to data using the weight and unweighted discretised approach based on the bin width of the histogram. The other is to provide a definitive fit to the data set using the maximum likelihood and quantile matching estimation. Other methods such as moment matching, starship method, L moment matching are also provided. Diagnostics on goodness of fit can be done via qqplots, KS-resample tests and comparing mean, variance, skewness and kurtosis of the data with the fitted distribution. References include the following: Karvanen and Nuutinen (2008) ``Characterizing the generalized lambda distribution by L-moments" <doi:10.1016/j.csda.2007.06.021>, King and MacGillivray (1999) ``A starship method for fitting the generalised lambda distributions" <doi:10.1111/1467-842X.00089>, Su (2005) ``A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data" <doi:10.22237/jmasm/1130803560>, Su (2007) ``Nmerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions" <doi:10.1016/j.csda.2006.06.008>, Su (2007) ``Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R" <doi:10.18637/jss.v021.i09>, Su (2009) ``Confidence Intervals for Quantiles Using Generalized Lambda Distributions" <doi:10.1016/j.csda.2009.02.014>, Su (2010) ``Chapter 14: Fitting GLDs and Mixture of GLDs to Data using Quantile Matching Method" <doi:10.1201/b10159>, Su (2010) ``Chapter 15: Fitting GLD to data using GLDEX 1.0.4 in R" <doi:10.1201/b10159>, Su (2015) ``Flexible Parametric Quantile Regression Model" <doi:10.1007/s11222-014-9457-1>, Su (2021) ``Flexible parametric accelerated failure time model"<doi:10.1080/10543406.2021.1934854>.

License GPL (>= 3)

NeedsCompilation yes

Author Steve Su [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-3368-4926>),
Martin Maechler [aut],

Juha Karvanen [aut],
Robert King [aut],
Benjamin Dean [ctb],
R Core Team [aut]

Repository CRAN

Date/Publication 2023-08-21 04:02:41 UTC

Contents

`GLDEX-package` *This package fits RS and FMKL generalised lambda distributions using various methods. It also provides functions for fitting bimodal distributions using mixtures of generalised lambda distributions.*

Description

The fitting algorithms considered in this package have two major objectives. One is to provide a smoothing device to fit distributions to data using the weight and unweighted discretised approach based on the bin width of the histogram. The other is to provide a definitive fit to the data set using the maximum likelihood estimation.

Copyright Information: To ensure the stability of this package, this package ports other functions from other open sourced packages directly so that any changes in other packages will not cause this package to malfunction.

All functions obtained from other sources have been acknowledged by the author in the authorship or the description sections of the help files and they are freely available online for all to use. Please contact the author for any copyright issues.

Specifically the following functions have been modified from R:

hist.su, ks.gof, pretty.su

The following functions are taken from other open source packages in R:

runif.pseudo, rnorm.pseudo, runif.halton, rnorm.halton, runif.sobol, rnorm.sobol by Diethelm Wuertz distributed under GPL.

digitsBase, QUnif and sHalton written by Martin Maechler distributed under GPL.

dgl, pgl, rgl, qgl, starship.adpativegrid, starship.obj and starship written by Robert King and some functions modified by Steve Su distributed under GPL.

Lmoments and t1lmoments written by Juha Karvanen distributed under GPL.

Details

This package allows a direct fitting method onto the data set using `fun.RMFMKL.ml`, `fun.RMFMKL.ml.m`, `fun.RMFMKL.hs`, `fun.RMFMKL.hs.nw`, `fun.RPRS.ml`, `fun.RPRS.ml.m`, `fun.RPRS.hs`, `fun.RPRS.hs.nw`, `fun.RMFMKL.qs`, `fun.RPRS.qs`, `fun.RMFMKL.mm`, `fun.RPRS.mm`, `fun.RMFMKL.lm`, `fun.RPRS.lm` and in the case of bimodal data set: `fun.auto.bimodal.qs`, `fun.auto.bimodal.ml`, `fun.auto.bimodal.pml` functions. The resulting fits can be graphically gauged by `fun.plot.fit` or `fun.plot.fit.bm` (for bimodal data), or examined by numbers using the Kolmogorov-Smirnoff resample tests (`fun.diag.ks.g`) and `fun.diag.ks.g.bimodal`). For unimodal data fits, it is also possible to compare the mean, variance, skewness and kurtosis of the fitted distribution with the data set using `fun.comp.moments.ml` and `fun.comp.moments.ml.2` functions. Similarly, for bimodal data fits, this is done via `fun.theo.bi.mv.gld` and `fun.moments.r`. Additionally, L moments for single generalised lambda distribution can be obtained using `fun.lm.theo.gld`. For graphical display of goodness of fit, quantile plots can be used, these can be done using `qqplot.gld` and `qqplot.gld.bi` for univariate and bimodal generalised lambda distribution fits respectively.

Author(s)

Steve Su <allegro.su@gmail.com>

References

Asquith, W. (2007), L-moments and TL-moments of the generalized lambda distribution, Computational Statistics and Data Analysis 51(9): 4484-4496.

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), A study of the generalized tukey lambda family, Communications in Statistics - Theory and Methods *17*, 3547-3567.

Gilchrist, Warren G. (2000), Statistical Modelling with Quantile Functions, Chapman & Hall

Karian, Z.A., Dudewicz, E.J., and McDonald, P. (1996), The extended generalized lambda distribution system for fitting distributions to data: history, completion of theory, tables, applications, the "Final Word" on Moment fits, Communications in Statistics - Simulation and Computation *25*, 611-642.

Karian, Zaven A. and Dudewicz, Edward J. (2000), Fitting statistical distributions: the Generalized Lambda Distribution and Generalized Bootstrap methods, Chapman & Hall

Karvanen, J. and A. Nuutinen (2008), Characterizing the generalized lambda distribution by L-moments, Computational Statistics and Data Analysis 52(4): 1971-1983. doi:10.1016/j.csda.2007.06.021

King, R.A.R. & MacGillivray, H. L. (1999), A starship method for fitting the generalised lambda distributions, Australian and New Zealand Journal of Statistics, 41, 353-374 doi:10.1111/1467-842X.00089

Ramberg, J. S. & Schmeiser, B. W. (1974), An approximate method for generating asymmetric random variables, Communications of the ACM *17*, 78-82.

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424. doi:10.22237/jmasm/1130803560

Su, S. (2007). Nmerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Computational Statistics and Data Analysis: *51*, 8, 3983-3998. doi:10.1016/j.csda.2006.06.008

Su, S. (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9. doi:10.18637/jss.v021.i09

Su, S. (2009). Confidence Intervals for Quantiles Using Generalized Lambda Distributions. Computational Statistics and Data Analysis *53*, 9, 3324-3333. doi:10.1016/j.csda.2009.02.014

Su, S. (2010). Chapter 14: Fitting GLDs and Mixture of GLDs to Data using Quantile Matching Method. Handbook of Distribution Fitting Methods with R. (Karian and Dudewicz) 557-583. doi:10.1201/b10159-3

Su, S. (2010). Chapter 15: Fitting GLD to data using GLDEX 1.0.4 in R. Handbook of Distribution Fitting Methods with R. (Karian and Dudewicz) 585-608. doi:10.1201/b10159-3

Su, S. (2015) "Flexible Parametric Quantile Regression Model" Statistics & Computing 25 (3). 635-650. doi:10.1007/s11222-014-9457-1

Su S. (2021) "Flexible parametric accelerated failure time model" J Biopharm Stat. 2021 Sep 31(5):650-667. doi:10.1080/10543406.2021.1934854

See Also

GLDreg package in R for GLD regression models.

Examples

```
###Univariate distributions example:

set.seed(1000)
```

```
junk<-rweibull(300,3,2)

##A faster ML estimtion

junk.fit1<-fun.RMFMKL.ml.m(junk)
junk.fit2<-fun.RPRS.ml.m(junk)

qqplot.gld(junk.fit1,data=junk,param="fmkl")
qqplot.gld(junk.fit2,data=junk,param="rs")

##Using discretised approach, with 50 classes

#Using discretised method with weights
obj.fit1.hs<-fun.data.fit.hs(junk)

#Plot the resulting fit. The fun.plot.fit function also works for singular fits
#such as those by fun.plot.fit(obj.fit1.ml,junk,nclass=50,
#param=c("rs","fmkl","fmkl"),xlab="x")

fun.plot.fit(obj.fit1.hs,junk,nclass=50,param=c("rs","fmkl"),xlab="x")

#Compare the mean, variance, skewness and kurtosis of the fitted distribution
#with actual data

fun.theo.mv.gld(obj.fit1.hs[1,1],obj.fit1.hs[2,1],obj.fit1.hs[3,1],
obj.fit1.hs[4,1],param="rs")
fun.theo.mv.gld(obj.fit1.hs[1,2],obj.fit1.hs[2,2],obj.fit1.hs[3,2],
obj.fit1.hs[4,2],param="fmkl")
fun.moments.r(junk)

#Conduct resample KS tests
fun.diag.ks.g(obj.fit1.hs[,1],junk,param="rs")
fun.diag.ks.g(obj.fit1.hs[,2],junk,param="fmkl")

##Try another fit, say 15 classes

obj.fit2.hs<-fun.data.fit.hs(junk,rs.default="N",fmkl.default="N",no.c.rs = 15,
no.c.fmkl = 15)

fun.plot.fit(obj.fit2.hs,junk,nclass=50,param=c("rs","fmkl"),xlab="x")

fun.theo.mv.gld(obj.fit2.hs[1,1],obj.fit2.hs[2,1],obj.fit2.hs[3,1],
obj.fit2.hs[4,1],param="rs")
fun.theo.mv.gld(obj.fit2.hs[1,2],obj.fit2.hs[2,2],obj.fit2.hs[3,2],
obj.fit2.hs[4,2],param="fmkl")
fun.moments.r(junk)

fun.diag.ks.g(obj.fit2.hs[,1],junk,param="rs")
fun.diag.ks.g(obj.fit2.hs[,2],junk,param="fmkl")

##Uses the maximum likelihood estimation method

#Fit the function using fun.data.fit.ml
```

```
obj.fit1.ml<-fun.data.fit.ml(junk)

#Plot the resulting fit
fun.plot.fit(obj.fit1.ml,junk,nclass=50,param=c("rs","fmkl","fmkl"),xlab="x",
name=".ML")

#Compare the mean, variance, skewness and kurtosis of the fitted distribution
#with actual data
fun.comp.moments.ml(obj.fit1.ml,junk)

#Do a quantile plot

qqplot.gld(junk,obj.fit1.ml[,1],param="rs",name="RS ML fit")

#Run a KS resample test on the resulting fit

fun.diag2(obj.fit1.ml,junk,1000)

#It is possible to use say fun.data.fit.ml(junk,rs.leap=409,fmkl.leap=409,
#FUN="QUnif") to find solution under a different set of low discrepancy number
#generators.

###Bimodal distributions example:

#Fitting mixture of generalised lambda distributions on the data set using both
#the maximum likelihood and partition maximum likelihood and plot the resulting
#fits

opar <- par()
par(mfrow=c(2,1))

junk<-fun.auto.bimodal.ml(faithful[,1],per.of.mix=0.01,clustering.m=clara,
init1.sel="rprs",init2.sel="rmfmkl",init1=c(-1.5,1.5),init2=c(-0.25,1.5),
leap1=3,leap2=3)
fun.plot.fit.bm(nclass=50,fit.obj=junk,data=faithful[,1],
name="Maximum likelihood using",xlab="faithful1",param.vec=c("rs","fmkl"))

#Do a quantile plot

qqplot.gld.bi(faithful[,1],junk$par,param1="rs",param2="fmkl",
name="RS FMKL ML fit",range=c(0.001,0.999))

par(opar)

junk<-fun.auto.bimodal.pml(faithful[,1],clustering.m=clara,init1.sel="rprs",
init2.sel="rmfmkl",init1=c(-1.5,1.5),init2=c(-0.25,1.5),leap1=3,leap2=3)
fun.plot.fit.bm(nclass=50,fit.obj=junk,data=faithful[,1],
name="Partition maximum likelihood using",xlab="faithful1",
param.vec=c("rs","fmkl"))

#Fit the faithful[,1] data from the dataset library using sobol sequence
#generator for the first distribution fit and halton sequence for the second
#distribution fit.
```

```
fit1<-fun.auto.bimodal.ml(faithful[,1],init1.sel="rmfmkl",init2.sel="rmfmkl",
init1=c(-0.25,1.5),init2=c(-0.25,1.5),leap1=3,leap2=3,fun1="runif.sobol",
fun2="runif.halton")

#Run diagnostic KS tests

fun.diag.ks.g.bimodal(fit1$par[1:4],fit1$par[5:8],prop1=fit1$par[9],
data=faithful[,1],param1="fmkl",param2="fmkl")

#Find the theoretical moments of the fit

fun.theo.bi.mv.gld(fit1$par[1],fit1$par[2],fit1$par[3],fit1$par[4],"fmkl",
fit1$par[5],fit1$par[6],fit1$par[7],fit1$par[8],"fmkl",fit1$par[9])

#Compare this with the empirical moments from the data set.

fun.moments.r(faithful[,1])

#Do a quantile plot

qqplot.gld.bi(faithful[,1],fit1$par,param1="fmkl",param2="fmkl",
name="FMKL FMKL ML fit")

#Quantile matching method

#Fitting faithful data from the dataset library, with the clara clustering
#regime. The first distribution is RS and the second distribution is fmkl.
#The percentage of data mix is 1%.

#Fitting normal(3,2) distriution using the default setting
junk<-rnorm(50,3,2)
fun.data.fit.qs(junk)

fun.auto.bimodal.qs(faithful[,1],per.of.mix=0.01,clustering.m=clara,
init1.sel="rprs",init2.sel="rmfmkl",init1=c(-1.5,1,5),init2=c(-0.25,1.5),
leap1=3,leap2=3)

#L Moment matching

#Fitting normal(3,2) distriution using the default setting
junk<-rnorm(50,3,2)
fun.data.fit.lm(junk)

# Moment matching method

#Fitting normal(3,2) distriution using the default setting
junk<-rnorm(50,3,2)
fun.data.fit.mm(junk)

# Example on fitting mixture of normal distributions

data1<-c(rnorm(1500,-1,2/3),rnorm(1500,1,2/3))
```

```
junk<-fun.auto.bimodal.ml(data1,per.of.mix=0.01,clustering.m=data1>0,
init1.sel="rprs",init2.sel="rmfmkl",init1=c(-1.5,1.5),init2=c(-0.25,1.5),
leap1=3,leap2=3)

fun.plot.fit.bm(nclass=50,fit.obj=junk,data=data1,
name="Maximum likelihood using",xlab="faithful1",param.vec=c("rs","fmkl"))

qqplot.gld.bi(data1,junk$par,param1="rs",param2="fmkl",
name="RS FMKL ML fit",range=c(0.001,0.999))

# Generate random observations from FMKL generalised lambda distributions with
# parameters (1,2,3,4) and (4,3,2,1) with 50% of data from each distribution.
fun.simu.bimodal(c(1,2,3,4),c(4,3,2,1),prop1=0.5,param1="fmkl",param2="fmkl")
```

`digitsBase` *Digit/Bit Representation of Integers in any Base*

Description

Integer number representations in other Bases.

Formally, for every element N =`x[i]`, compute the (vector of) "digits" A of the `base` b representation of the number N, $N = \sum_{k=0}^{M} A_{M-k} b^k$.
Revert such a representation to integers.

Usage

```
digitsBase(x, base = 2, ndigits = 1 + floor(log(max(x), base)))
```

Arguments

`x`	For `digitsBase()`: non-negative integer (vector) whose base `base` digits are wanted. For `as.intBase()`: a list of numeric vectors, a character vector, or an integer matrix as returned by `digitsBase()`, representing digits in base `base`.
`base`	integer, at least 2 specifying the base for representation.
`ndigits`	number of bits/digits to use.

Value

For `digitsBase()`, an object, say `m`, of class `"basedInt"` which is basically a (`ndigits` x `n`) `matrix` where `m[,i]` corresponds to `x[i]`, `n <- length(x)` and `attr(m,"base")` is the input `base`.

`as.intBase()` and the `as.integer` method for `basedInt` objects return an `integer` vector.

Note

`digits` and `digits.v` are now deprecated and will be removed from the **sfsmisc** package.

Author(s)

Martin Maechler, Dec 4, 1991 (for S-plus; then called `digits.v`).

Examples

```
digitsBase(0:12, 8) #-- octal representation

## This may be handy for just one number (and default decimal):
digits <- function(n, base = 10) as.vector(digitsBase(n, base = base))
digits(128, base = 8) # 2 0 0

## one way of pretty printing (base <= 10!)
b2ch <- function(db)
        noquote(gsub("^0+(.{1,})$"," \1", apply(db, 2, paste, collapse = "")))
b2ch(digitsBase(0:33, 2))  #->  0 1 10 11 100 101 ... 100001
b2ch(digitsBase(0:33, 4))  #->  0 1 2 3 10 11 12 13 20 ... 200 201

## Hexadecimal:
i <- c(1:20, 100:106)
M <- digitsBase(i, 16)
hexdig <- c(0:9, LETTERS[1:6])
cM <- hexdig[1 + M]; dim(cM) <- dim(M)
b2ch(cM) #->  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F 10 11 ... 6A
```

---

fun.auto.bimodal.ml *Fitting mixture of generalied lambda distribtions to data using maximum likelihood estimation via the EM algorithm*

Description

This function will fit mixture of generalised lambda distributions to dataset. It is restricted to two generalised lambda distributions. The method of fitting is maximum likelihood via EM algorithm. It is a two step optimization procedure, each unimodal part of the bimodal distribution is modelled using the maximum likelihood method or the starship method (FMKL GLD only), these initial values are the used to maximise the likelihood for the entire bimodal distribution using the EM algorithm. It fits mixture of the form p*(f1)+(1-p)*(f2) where f1 and f2 are pdfs of the generalised lambda distributions.

Usage

```
fun.auto.bimodal.ml(data, per.of.mix = 0.01, clustering.m = clara,
init1.sel = "rprs", init2.sel = "rprs", init1=c(-1.5, 1.5), init2=c(-1.5, 1.5),
leap1=3, leap2=3,fun1="runif.sobol",fun2="runif.sobol",no=10000, max.it=5000,
optim.further="Y")
```

Arguments

data	A numerical vector representing the dataset.
per.of.mix	Level of mix between two parts of the distribution, usually 1-2% of cross mix is sufficient.
clustering.m	Clustering method used in classifying the dataset into two parts. Valid arguments include clara, fanny and pam from the cluster library. Default is clara. Or a logical vector specifying how data should be split.
init1.sel	This can be "rprs", "rmfmkl" or "star", the initial method used to fit the first distribution.
init2.sel	This can be "rprs", "rmfmkl" or "star", the initial method used to fit the second distribution.
init1	Inititial values lambda3 and lambda4 for the first generalised lambda distribution.
init2	Inititial values lambda3 and lambda4 for the second generalised lambda distribution.
leap1	See scrambling argument in fun.gen.qrn.
fun1	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
leap2	See scrambling argument in fun.gen.qrn.
fun2	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.
max.it	Maximum number of iterations for numerical optimisation.
optim.further	Whether to optimise the function further using full maximum likelihood method, recommended setting is "Y"

Details

The initial values that work well for RPRS are c(-1.5,1.5) and for RMFMKL are c(-0.25,1.5). For scrambling, if 1, 2 or 3 the sequence is scrambled otherwise not. If 1, Owen type type of scrambling is applied, if 2, Faure-Tezuka type of scrambling, is applied, and if 3, both Owen+Faure-Tezuka type of scrambling is applied. The star method uses the same initial values as rmfmkl since it uses the FMKL generalised lambda distribution. Nelder-Simplex algorithm is used in the numerical optimization. rprs stands for revised percentile method for RS generalised lambda distribution and "rmfmkl" stands for revised method of moment for FMKL generalised lambda distribution. These acronyms represents the initial optimization algorithm used to get a reasonable set of initial values for the subsequent optimization procedues. This function is an improvement from Su (2007) in Journal of Statistical Software.

Value

par	The best set of parameters found, the first four corresponds to the first distribution fit, the second four corresponds to the second distribution fit, the last value correspond to p for the first distribution fit.

value	The value of -ML for the paramters obtained.
counts	A two-element integer vector giving the number of calls to fn and gr respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to fn to compute a finite-difference approximation to the gradient.
convergence	0 indicates successful convergence, 1 indicates the iteration limit maxit had been reached, 10 indicates degeneracy of the Nelder-Mead simplex.
message	A character string giving any additional information returned by the optimizer, or NULL.

Note

If the number of observations is small, rprs can sometimes fail as the percentiles may not exist for this data. Also, if the initial values do not span a valid generalised lambda distribution, try another set of initial values.

Author(s)

Steve Su

References

Bratley P. and Fox B.L. (1988) Algorithm 659: Implementing Sobol's quasi random sequence generator, ACM Transactions on Mathematical Software 14, 88-100.

Joe S. and Kuo F.Y. (1998) Remark on Algorithm 659: Implementing Sobol's quasi random Sequence Generator.

Nelder, J. A. and Mead, R. (1965) A simplex algorithm for function minimization. Computer Journal *7*, 308-313.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.auto.bimodal.pml, fun.plot.fit.bm, fun.diag.ks.g.bimodal

Examples

```
# Fitting faithful data from the dataset library, with the clara clustering
# regime. The first distribution is RS and the second distribution is fmkl.
# The percentage of data mix is 1%.

fun.auto.bimodal.ml(faithful[,1],per.of.mix=0.01,clustering.m=clara,
init1.sel="rprs",init2.sel="rmfmkl",init1=c(-1.5,1,5),init2=c(-0.25,1.5),
leap1=3,leap2=3)
```

`fun.auto.bimodal.pml` *Fitting mixture of generalied lambda distribtions to data using parition maximum likelihood estimation*

Description

This function will fit mixture of generalised lambda distributions to dataset. It is restricted to two generalised lambda distributions. The method of fitting is parition maximum likelihood. It is a two step optimization procedure, each unimodal part of the bimodal distribution is modelled using the maximum likelihood method or the starship method (FMKL GLD only). These initial values the used to "maximise" the complete log likelihood for the entire bimodal distribution. It fits mixture of the form p*(f1)+(1-p)*(f2) where f1 and f2 are pdfs of the generalised lambda distributions.

Usage

```
fun.auto.bimodal.pml(data, clustering.m = clara, init1.sel = "rprs",
init2.sel = "rprs", init1=c(-1.5, 1.5), init2=c(-1.5, 1.5), leap1=3, leap2=3,
fun1="runif.sobol", fun2="runif.sobol",no=10000,max.it=5000, optim.further="Y")
```

Arguments

`data`	A numerical vector representing the dataset.
`clustering.m`	Clustering method used in classifying the dataset into two parts. Valid arguments include clara, fanny and pam from the cluster library. Default is clara. Or a logical vector specifying how data should be split.
`init1.sel`	This can be `"rprs"`, `"rmfmkl"` or `"star"`, the initial method used to fit the first distribution.
`init2.sel`	This can be `"rprs"`, `"rmfmkl"` or `"star"`, the initial method used to fit the second distribution.
`init1`	Inititial values lambda3 and lambda4 for the first generalised lambda distribution.
`init2`	Inititial values lambda3 and lambda4 for the second generalised lambda distribution.
`leap1`	See scrambling argument in `fun.gen.qrn`.
`fun1`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`leap2`	See scrambling argument in `fun.gen.qrn`.
`fun2`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`no`	Number of initial random values to find the best initial values for optimisation.
`max.it`	Maximum number of iterations for numerical optimisation.
`optim.further`	Whether to optimise the function further using full maximum likelihood method, recommended setting is "Y"

Details

The initial values that work well for RPRS are `c(-1.5,1.5)` and for RMFMKL are `c(-0.25,1.5)`. For scrambling, if `1`, `2` or `3` the sequence is scrambled otherwise not. If `1`, Owen type type of scrambling is applied, if `2`, Faure-Tezuka type of scrambling, is applied, and if `3`, both Owen+Faure-Tezuka type of scrambling is applied. The `star` method uses the same initial values as `rmfmkl` since it uses the FMKL generalised lambda distribution. Nelder-Simplex algorithm is used in the numerical optimization. `rprs` stands for revised percentile method for RS generalised lambda distribution and "rmfmkl" stands for revised method of moment for FMKL generalised lambda distribution. These acronyms represents the initial optimization algorithm used to get a reasonable set of initial values for the subsequent optimization procedues. This function is an improvement from Su (2007) in Journal of Statistical Software.

Value

`par`	The best set of parameters found, the first four corresponds to the first distribution fit, the second four corresponds to the second distribution fit, the last value correspond to p for the first distribution fit.
`value`	The value of -PML for the paramters obtained.
`counts`	A two-element integer vector giving the number of calls to "fn" and "gr" respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to 'fn' to compute a finite-difference approximation to the gradient.
`convergence`	`0` indicates successful convergence, `1` indicates the iteration limit `maxit` had been reached, `10` indicates degeneracy of the Nelder-Mead simplex.
`message`	A character string giving any additional information returned by the optimizer, or `NULL`.

Note

If the number of observations is small, `rprs` can sometimes fail as the percentiles may not exist for this data. Also, if the initial values do not result in a valid generalised lambda distribution, try another set of initial values.

References

Bratley P. and Fox B.L. (1988) Algorithm 659: Implementing Sobol's quasi random sequence generator, ACM Transactions on Mathematical Software 14, 88-100.

Joe S. and Kuo F.Y. (1998) Remark on Algorithm 659: Implementing Sobol's quasi random Sequence Generator.

Nelder, J. A. and Mead, R. (1965) A simplex algorithm for function minimization. Computer Journal *7*, 308-313.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

`fun.auto.bimodal.ml`,`fun.plot.fit.bm`, `fun.diag.ks.g.bimodal`

Examples

```
# Fitting faithful data from the dataset library, with the clara clustering
# regime. The first distribution is RS and the second distribution is fmkl.

fun.auto.bimodal.pml(faithful[,1],clustering.m=clara,init1.sel="rprs",
init2.sel="rmfmkl",init1=c(-1.5,1,5),init2=c(-0.25,1.5),leap1=3,leap2=3)
```

`fun.auto.bimodal.qs` *Fitting mixtures of generalied lambda distribtions to data using quantile matching method*

Description

This function will fit mixture of generalised lambda distributions to dataset. It is restricted to two generalised lambda distributions. The method of fitting is quantile matching method. It is a two step optimization procedure, each unimodal part of the bimodal distribution is modelled using quantile matching method. The initial values obtained are then used to maximise the theoretical and empirical quantile match for the entire bimodal distribution. It fits mixture of the form p*(f1)+(1-p)*(f2) where f1 and f2 are pdfs of the generalised lambda distributions.

Usage

```
fun.auto.bimodal.qs(data, per.of.mix = 0.01, clustering.m = clara,
init1.sel = "rprs", init2.sel = "rprs", init1=c(-1.5, 1.5), init2=c(-1.5, 1.5),
leap1=3, leap2=3, fun1 = "runif.sobol", fun2 = "runif.sobol", trial.n = 100,
len = 1000, type = 7, no = 10000, maxit = 5000)
```

Arguments

`data`	A numerical vector representing the dataset.
`per.of.mix`	Level of mix between two parts of the distribution, usually 1-2% of cross mix is sufficient.
`clustering.m`	Clustering method used in classifying the dataset into two parts. Valid arguments include clara, fanny and pam from the cluster library. Default is clara. Or a logical vector specifying how data should be split.
`init1.sel`	This can be `"rprs"` or `"rmfmkl"`, representing the choice (RS or FMKL) of the first distribution
`init2.sel`	This can be `"rprs"` or `"rmfmkl"`, representing the choice (RS or FMKL) of the second distribution
`init1`	Inititial values lambda3 and lambda4 for the first generalised lambda distribution.
`init2`	Inititial values lambda3 and lambda4 for the second generalised lambda distribution.
`leap1`	See scrambling argument in `fun.gen.qrn`.

fun1	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
leap2	See scrambling argument in fun.gen.qrn.
fun2	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
trial.n	Number of evenly spaced quantile ranging from 0 to 1 to be used in the checking phase, to find the best set of initial values for optimisation, this is intended to be lower than len to speed up the fitting algorithm. Default is 100.
len	Number of evenly spaced quantile ranging from 0 to 1 to be used, default is 1000
type	Type of quantile to be used, default is 7, see quantile
no	Number of initial random values to find the best initial values for optimisation.
maxit	Maximum number of iterations for numerical optimisation. Default is 5000.

Details

The initial values that work well for RPRS are c(-1.5,1.5) and for RMFMKL are c(-0.25,1.5). For scrambling, if 1, 2 or 3 the sequence is scrambled otherwise not. If 1, Owen type type of scrambling is applied, if 2, Faure-Tezuka type of scrambling, is applied, and if 3, both Owen+Faure-Tezuka type of scrambling is applied. The star method uses the same initial values as rmfmkl since it uses the FMKL generalised lambda distribution. Nelder-Simplex algorithm is used in the numerical optimization. rprs stands for revised percentile method for RS generalised lambda distribution and "rmfmkl" stands for revised method of moment for FMKL generalised lambda distribution. These acronyms represents the initial optimization algorithm used to get a reasonable set of initial values for the subsequent optimization procedues.

Value

par	The best set of parameters found, the first four corresponds to the first distribution fit, the second four corresponds to the second distribution fit, the last value correspond to p for the first distribution fit.
value	The value of -ML for the paramters obtained.
counts	A two-element integer vector giving the number of calls to fn and gr respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to fn to compute a finite-difference approximation to the gradient.
convergence	0 indicates successful convergence, 1 indicates the iteration limit maxit had been reached, 10 indicates degeneracy of the Nelder-Mead simplex.
message	A character string giving any additional information returned by the optimizer, or NULL.

Note

If the number of observations is small, rprs can sometimes fail as the percentiles may not exist for this data. Also, if the initial values do not span a valid generalised lambda distribution, try another set of initial values.

Author(s)

Steve Su

References

Bratley P. and Fox B.L. (1988) Algorithm 659: Implementing Sobol's quasi random sequence generator, ACM Transactions on Mathematical Software 14, 88-100.

Joe S. and Kuo F.Y. (1998) Remark on Algorithm 659: Implementing Sobol's quasi random Sequence Generator.

Nelder, J. A. and Mead, R. (1965) A simplex algorithm for function minimization. Computer Journal *7*, 308-313.

Su (2008). Fitting GLD to data via quantile matching method. (Book chapter to appear)

See Also

fun.auto.bimodal.pml, fun.auto.bimodal.ml, fun.plot.fit.bm, fun.diag.ks.g.bimodal

Examples

```
# Fitting faithful data from the dataset library, with the clara clustering
# regime. The first distribution is RS and the second distribution is fmkl.
# The percentage of data mix is 1%.

fun.auto.bimodal.qs(faithful[,1],per.of.mix=0.01,clustering.m=clara,
init1.sel="rprs",init2.sel="rmfmkl",init1=c(-1.5,1,5),init2=c(-0.25,1.5),
leap1=3,leap2=3)
```

---

fun.bimodal.fit.ml *Finds the final fits using the maximum likelihood estimation for the bimodal dataset.*

Description

This is the secondary optimization procedure to evaluate the final bimodal distribution fits using the maximum likelihood. It usually relies on initial values found by `fun.bimodal.init` function.

Usage

```
fun.bimodal.fit.ml(data, first.fit, second.fit, prop, param1, param2, selc1,
selc2)
```

Arguments

data	Dataset to be fitted.
first.fit	The distribution parameters or the initial values of the first distribution fit.
second.fit	The distribution parameters or the initial values of the second distribution fit.
prop	The proportion of the data set, usually obtained from fun.bimodal.init.
param1	Can be either "rs" or "fmkl", depending on the type of first distribution used.
param2	Can be either "rs" or "fmkl", depending on the type of second distribution used.
selc1	Selection of initial values for the first distribution, can be either "rs", "fmkl" or "star". Choose initial values from RPRS (ML), RMFMKL (ML) or STAR method.
selc2	Selection of initial values for the second distribution, can be either "rs", "fmkl" or "star". Choose initial values from RPRS (ML), RMFMKL (ML) or STAR method.

Details

This function should be used in tandem with fun.bimodal.init.

Value

par	The first four numbers are the parameters of the first generalised lambda distribution, the second four numbers are the parameters of the second generalised lambda distribution and the last value is the proportion of the first generalised lambda distribution.
value	The objective value of negative likelihood obtained using the par above.
counts	A two-element integer vector giving the number of calls to functions. Gradient is not used in this case.
convergence	An integer code. 0 indicates successful convergence. Error codes are: 1 indicates that the iteration limit 'maxit' had been reached. 10 indicates degeneracy of the Nelder-Mead simplex.
message	A character string giving any additional information returned by the optimizer, or NULL.

Note

There is currently no guarantee of a global convergence.

Author(s)

Steve Su

References

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

link{fun.bimodal.fit.pml}, fun.bimodal.init

Examples

```
# Extract faithful[,2] into faithful2
faithful2<-faithful[,2]

# Uses clara clustering method
clara.faithful2<-fun.class.regime.bi(faithful2, 0.01, clara)

# Save into two different objects
qqqq1.faithful2.cc<-clara.faithful2$data.a
qqqq2.faithful2.cc<-clara.faithful2$data.b

# Find the initial values
result.faithful2.init<-fun.bimodal.init(data1=qqqq1.faithful2.cc,
data2=qqqq2.faithful2.cc, rs.leap1=3,fmkl.leap1=3,rs.init1 = c(-1.5, 1.5),
fmkl.init1 = c(-0.25, 1.5), rs.leap2=3,fmkl.leap2=3,rs.init2 = c(-1.5, 1.5),
fmkl.init2 = c(-0.25, 1.5))

# Find the final fits
result.faithful2.rsrs<-fun.bimodal.fit.ml(data=faithful2,
result.faithful2.init[[2]],result.faithful2.init[[3]],
result.faithful2.init[[1]], param1="rs",param2="rs",selc1="rs",selc2="rs")

# Output
result.faithful2.rsrs
```

---

fun.bimodal.fit.pml *Finds the final fits using partition maximum likelihood estimation for the bimodal dataset.*

Description

This is the secondary optimization procedure to evaluate the final bimodal distribution fits using the partition maximum likelihood. It usually relies on initial values found by fun.bimodal.init function.

Usage

```
fun.bimodal.fit.pml(data1, data2, first.fit, second.fit, prop, param1, param2,
selc1, selc2)
```

Arguments

data1	First data set, usually obtained by fun.class.regime.bi.
data2	Second data set, usually obtained by fun.class.regime.bi.
first.fit	The distribution parameters or the initial values of the first distribution fit.
second.fit	The distribution parameters or the initial values of the second distribution fit.
prop	The proportion of the data set, usually obtained from fun.bimodal.init.
param1	Can be either rs or fmkl, depending on the type of first distribution used.
param2	Can be either rs or fmkl, depending on the type of second distribution used.
selc1	Selection of initial values for the first distribution, can be either "rs", "fmkl" or "star". Choose initial values from RPRS (ML), RMFMKL (ML) or STAR method.
selc2	Selection of initial values for the second distribution, can be either "rs", "fmkl" or "star". Choose initial values from RPRS (ML), RMFMKL (ML) or STAR method.

Details

This function should be used in tandem with fun.bimodal.init function.

Value

par	The first four numbers are the parameters of the first generalised lambda distribution, the second four numbers are the parameters of the second generalised lambda distribution and the last value is the proportion of the first generalised lambda distribution.
value	The objective value of negative likelihood obtained.
counts	A two-element integer vector giving the number of calls to functions. Gradient is not used in this case.
convergence	An integer code. 0 indicates successful convergence. Error codes are: 1 indicates that the iteration limit 'maxit' had been reached. 10 indicates degeneracy of the Nelder-Mead simplex.
message	A character string giving any additional information returned by the optimizer, or NULL.

Note

There is currently no guarantee of a global convergence.

Author(s)

Steve Su

References

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.bimodal.fit.ml, fun.bimodal.init

Examples

```
# Extract faithful[,2] into faithful2
faithful2<-faithful[,2]

# Uses clara clustering method
clara.faithful2<-clara(faithful2,2)$clustering

# Save into two different objects
qqqq1.faithful2.cc<-faithful2[clara.faithful2==1]
qqqq2.faithful2.cc<-faithful2[clara.faithful2==2]

# Find the initial values
result.faithful2.init<-fun.bimodal.init(data1=qqqq1.faithful2.cc,
data2=qqqq2.faithful2.cc, rs.leap1=3,fmkl.leap1=3,rs.init1 = c(-1.5, 1.5),
fmkl.init1 = c(-0.25, 1.5), rs.leap2=3,fmkl.leap2=3,rs.init2 = c(-1.5, 1.5),
fmkl.init2 = c(-0.25, 1.5))

# Find the final fits
result.faithful2.rsrs<-fun.bimodal.fit.pml(data1=qqqq1.faithful2.cc,
data2=qqqq2.faithful2.cc, result.faithful2.init[[2]],
result.faithful2.init[[3]], result.faithful2.init[[1]],param1="rs",
param2="rs",selc1="rs",selc2="rs")

# Output
result.faithful2.rsrs
```

---

fun.bimodal.init *Finds the initial values for optimisation in fitting the bimodal generalised lambda distribution.*

Description

After classifying the data using fun.class.regime.bi, this function evaluates the temporary or initial solutions by estimating each part of the bimodal distribution using the maximum likelihood estimation and starship method. These initial solutions are then passed onto fun.bimodal.fit.ml or fun.bimodal.fit.pml to obtain the final fits.

Usage

```
fun.bimodal.init(data1, data2, rs.leap1, fmkl.leap1, rs.init1, fmkl.init1,
rs.leap2, fmkl.leap2, rs.init2, fmkl.init2,fun1="runif.sobol",
fun2="runif.sobol",no=10000)
```

Arguments

data1	The first data obtained by the clustering algorithm.
data2	The second data obtained by the clustering algorithm.
rs.leap1	See scrambling argument in fun.gen.qrn.
fmkl.leap1	See scrambling argument in fun.gen.qrn.
rs.init1	Inititial values (lambda3 and lambda4) for the first RS generalised lambda distribution. c(-1.5,1.5) tends to work well.
fmkl.init1	Inititial values (lambda3 and lambda4) for the first FMKL generalised lambda distribution. c(-0.25,1.5) tends to work well
rs.leap2	See scrambling argument in fun.gen.qrn.
fmkl.leap2	See scrambling argument in fun.gen.qrn.
rs.init2	Inititial values (lambda3 and lambda4) for the second RS generalised lambda distribution. c(-1.5,1.5) tends to work well.
fmkl.init2	Inititial values (lambda3 and lambda4) for the second FMKL generalised lambda distribution. c(-0.25,1.5) tends to work well
fun1	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
fun2	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

All three methods of fitting (RPRS, RMFMKL and STAR) will be given for each part of the bimodal distribution.

Value

prop	Proportion of the number of observations in the first data in relation to the entire data.
first.fit	A matrix comprising the parameters of GLD obtained from RPRS, RMFMKL and STAR for the first dataset.
second.fit	A matrix comprising the parameters of GLD obtained from RPRS, RMFMKL and STAR for the second dataset.

Note

This is not designed to be called by the end user explicitly, the difficulties with RPRS parameterisation should be noted by the users.

Author(s)

Steve Su

References

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.class.regime.bi,fun.bimodal.fit.pml, fun.bimodal.fit.ml

Examples

```
# Split the first column of the faithful data into two using

fun.class.regime.bi
faithful1.mod<-fun.class.regime.bi(faithful[,1], 0.1, clara)

# Save the datasets
qqqq1.faithful1.cc1<-faithful1.mod$data.a
qqqq2.faithful1.cc1<-faithful1.mod$data.b

# Find the initial values for secondary optimisation.

result.faithful1.init1<-fun.bimodal.init(data1=qqqq1.faithful1.cc1,
data2=qqqq2.faithful1.cc1, rs.leap1=3,fmkl.leap1=3,rs.init1 = c(-1.5, 1.5),
fmkl.init1 = c(-0.25, 1.5), rs.leap2=3,fmkl.leap2=3,rs.init2 = c(-1.5, 1.5),
fmkl.init2 = c(-0.25, 1.5))

# These initial values are then passed onto fun,bimodal.fit.ml to obtain the
# final fits.
```

---

fun.check.gld *Check whether the RS or FMKL/FKML GLD is a valid GLD for single values of L1, L2, L3 and L4*

Description

This function will return a single logical value showing whether a combination of L1, L2, L3 and L4 is a valid GLD.

Usage

```
fun.check.gld(lambda1, lambda2, lambda3, lambda4, param)
```

Arguments

lambda1	A numerical vector for L1 of GLD
lambda2	A numerical vector for L2 of GLD
lambda3	A numerical vector for L3 of GLD
lambda4	A numerical vector for L4 of GLD
param	Can be "rs", "fmkl", or "fkml"

Value

A single logical value indicating whether the specified GLD is a valid probability density function

Author(s)

Steve Su

See Also

fun.check.gld.multi

Examples

```
fun.check.gld(1,4,3,2,"rs")

fun.check.gld(1,4,3,2,"fkml")

fun.check.gld(1,4,3,-2,"rs")
```

---

fun.check.gld.multi *Check whether the RS or FMKL/FKML GLD is a valid GLD for vectors of L1, L2, L3 and L4*

Description

This function will return a logical vector showing whether vector combinations of L1, L2, L3 and L4 are valid GLDs.

Usage

```
fun.check.gld.multi(l1, l2, l3, l4, param)
```

Arguments

l1	A numerical vector for L1 of GLD
l2	A numerical vector for L2 of GLD
l3	A numerical vector for L3 of GLD
l4	A numerical vector for L4 of GLD
param	Can be "rs", "fmkl", or "fkml"

Details

This is an extension to fun.check.gld

Value

A logical vector indicating whether the specified parameters are valid GLDs

Author(s)

Steve Su

See Also

fun.check.gld

Examples

```
fun.check.gld.multi(c(1,2,3),c(4,5,6),c(7,8,9),c(10,11,12),param="rs")

fun.check.gld.multi(c(1,2,3),c(4,5,6),c(7,8,9),c(10,11,-12),param="rs")
```

---

fun.class.regime.bi *Classifies data into two groups using a clustering regime.*

Description

This function is primarily designed to split a bimodal data vector into two groups to allow the fitting of mixture generalised lambda distributions.

Usage

```
fun.class.regime.bi(data, perc.cross, fun.cross)
```

Arguments

data	Data to be classified into two groups.
perc.cross	Percentage of cross over from one data to the other, usually set at 1%
fun.cross	Any clustering function such as link{clara}, pam, fanny can be used here. Or a logical vector indicating how data should be split.

Details

This function is part of the routine mixture fitting procedure provided in this package. The perc.cross argument or percentage of cross over is designed to allow the use of maximum likelihood estimation via EM algorithm for fitting bimodal data. When this is invoked, it will ensure both part of the data will contain both the minmum and maximum of the data set as well as a proportion (specified in perc.cross argument) of observations from each other. If 1% is required, then data.a will contains 1% of the data.b and vice versa after the full data set has been classified into data.a and data.b by the fun.cross classification regime.

Value

data.a	First group of data obtained by the classification algorithm.
data.b	Second group of data obtained by the classification algorithm.

Author(s)

Steve Su

References

Kaufman, L. and Rousseeuw, P. J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

Su (2006) Maximum Log Likelihood Estimation using EM Algorithm and Partition Maximum Log Likelihood Estimation for Mixtures of Generalized Lambda Distributions. Working Paper.

See Also

link{clara}, pam, fanny

Examples

```
# Classify the faithful[,1] data into two categories with 10% cross over mix.
 fun.class.regime.bi(faithful[,1],0.1,clara)

# Classify the faithful[,1] data into two categories with no mixing:
 fun.class.regime.bi(faithful[,1],0,clara)
```

---

fun.comp.moments.ml *Compare the moments of the data and the fitted univariate generalised lambda distribution.*

Description

After fitting the distribution, it is often desirable to see whether the moments of the data matches with the fitted distribution. This function computes the theoretical and actual moments especially for fun.data.fit.ml function output.

Usage

```
fun.comp.moments.ml(theo.obj, data, name = "ML")
```

Arguments

theo.obj	Fitted distribution parameters, usually output from fun.data.fit.ml
data	Data set used
name	Naming the method used in fitting the distribution, by default this is "ML".

Value

r.mat	A matrix showing the mean, variance, skewness and kurtosis of the fitted distribution in comparison to the data set.
eval.mat	Absolute difference in each of the four moments from the data under each of the distibutional fits.

Note

Sometimes it is difficult to find RPRS type of fits to data set, so instead `fun.comp.moments.ml.2` is used to compare the theoretical moments of RMFMKL.ML and STAR methods with respect to the dataset fitted.

Author(s)

Steve Su

See Also

`fun.comp.moments.ml.2`

Examples

```
# Generate random normally distributed observations.
 junk<-rnorm(1000,3,2)

# Fit the dataset using fun.data.ml
 fit<-fun.data.fit.ml(junk)

# Compare the resulting fits. It is usually the case the maximum likelihood
# provides better estimation of the moments than the starship method.
 fun.comp.moments.ml(fit,junk)
```

`fun.comp.moments.ml.2` *Compare the moments of the data and the fitted univariate generalised lambda distribution. Specialised funtion designed for RMFMKL.ML and STAR methods.*

Description

After fitting the distribution, it is often desirable to see whether the moments of the data matches with the fitted distribution. This function computes the theoretical and actual moments for the FMKL GLD maximum likelihood estimation and starship method.

Usage

```
fun.comp.moments.ml.2(theo.obj, data, name = "ML")
```

Arguments

`theo.obj`	Fitted distribution parameters, there should be two sets, both FMKL GLD.
`data`	Data set used
`name`	Naming the method used in fitting the distribution, by default this is "ML".

Value

r.mat	A matrix showing the mean, variance, skewness and kurtosis of the fitted distribution in comparison to the data set.
eval.mat	Absolute difference in each of the four moments from the data under each of the distibutional fits.

Note

To compare all three fits under fun.data.fit.ml see fun.comp.moments.ml function.

Author(s)

Steve Su

See Also

fun.comp.moments.ml

Examples

```
## Generate random normally distributed observations.
 junk<-rnorm(1000,3,2)

## Fit the dataset using fun.data.ml
 fit<-cbind(fun.RMFMKL.ml(junk),starship(junk)$lambda)

## Compare the resulting fits. It is usually the case the maximum likelihood
## provides better estimation of the moments than the starship method.
 fun.comp.moments.ml.2(fit,junk)
```

---

fun.data.fit.hs *Fit RS and FMKL generalised distributions to data using discretised approach with weights.*

Description

This function fits RS and FMKL generalised distribution to data using discretised approach with weights. It is designed to act as a smoother device rather than a definitive fit.

Usage

```
fun.data.fit.hs(data, rs.default = "Y", fmkl.default = "Y", rs.leap = 3,
fmkl.leap = 3, rs.init = c(-1.5, 1.5), fmkl.init = c(-0.25, 1.5), no.c.rs = 50,
no.c.fmkl = 50,FUN="runif.sobol", no=10000)
```

Arguments

data	Dataset to be fitted
rs.default	If yes, this function uses the default method fun.nclass.e to calculate number of classes required for the RS distribution fits.
fmkl.default	If yes, this function uses the default method fun.nclass.e to calculate number of classes required for the FMKL distribution fits.
rs.leap	See scrambling argument in fun.gen.qrn.
fmkl.leap	See scrambling argument in fun.gen.qrn.
rs.init	Initial values for RS distribution optimization, c(-1.5,1.5) tends to work well.
fmkl.init	Initial values for FMKL distribution optimization, c(-0.25,1.5) tends to work well.
no.c.rs	Number of classes or bins of histogram to be optimized over for the RS GLD. This argument is ineffective if default="Y".
no.c.fmkl	Number of classes or bins of histogram to be optimized over for the FMKL GLD. This argument is ineffective if default="Y".
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function optimises the deviations of frequency of the bins to that of the theoretical so it has the effect of "fitting clothes" onto the data set. The user can decide the frequency of the bins they want the distribution to smooth over. The resulting fit may or may not be an adequate fit from a formal statistical point of view such as satisfying the goodness of fit for example, but it can be useful to suggest the range of different distributions exhibited by the data set. The default number of classes calculates the mean and variance after categorising the data into different bins and uses the number of classes that best matches the mean and variance of the original, ungrouped data.The weighting is designed to accentuate the peak or the dense part of the distribution and suppress the tails.

Value

A matrix showing the four parameters of the RS and FMKL distribution fit.

Note

In some cases, the resulting fit may not converge, there are currently no checking mechanism in place to ensure global convergence. The RPRS method can sometimes fail if there are no valid percentiles in the data set or if initial values do not give a valid distribution.

Author(s)

Steve Su

References

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.RPRS.hs.nw, fun.RMFMKL.hs.nw, fun.RMFMKL.hs, fun.RPRS.hs, fun.data.fit.hs.nw, fun.data.fit.ml

Examples

```
# Fitting normal(3,2) distriution using the default setting
 junk<-rnorm(1000,3,2)
 fun.data.fit.hs(junk)
```

---

fun.data.fit.hs.nw *Fit RS and FMKL generalised distributions to data using discretised approach without weights.*

Description

This function fits RS and FMKL generalised distribution to data using discretised approach without weights. It is designed to act as a smoother device rather than a definitive fit.

Usage

```
fun.data.fit.hs.nw(data, rs.default = "Y", fmkl.default = "Y", rs.leap = 3,
fmkl.leap = 3, rs.init = c(-1.5, 1.5), fmkl.init = c(-0.25, 1.5), no.c.rs = 50,
no.c.fmkl = 50,FUN="runif.sobol",no=10000)
```

Arguments

data	Dataset to be fitted
rs.default	If yes, this function uses the default method fun.nclass.e to calculate number of classes required for the RS distribution fits.
fmkl.default	If yes, this function uses the default method fun.nclass.e to calculate number of classes required for the FMKL distribution fits.
rs.leap	See scrambling argument in fun.gen.qrn.
fmkl.leap	See scrambling argument in fun.gen.qrn.
rs.init	Initial values for RS distribution optimization, c(-1.5,1.5) tends to work well.
fmkl.init	Initial values for FMKL distribution optimization, c(-0.25,1.5) tends to work well.
no.c.rs	Number of classes or bins of histogram to be optimized over for the RS GLD. This argument is ineffective if default="Y".

no.c.fmkl	Number of classes or bins of histogram to be optimized over for the FMKL GLD. This argument is ineffective if default="Y".
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function optimises the deviations of frequency of the bins to that of the theoretical so it has the effect of "fitting clothes" onto the data set. The user can decide the frequency of the bins they want the distribution to smooth over. The resulting fit may or may not be an adequate fit from a formal statistical point of view such as satisfying the goodness of fit for example, but it can be useful to suggest the range of different distributions exhibited by the data set. The default number of classes calculates the mean and variance after categorising the data into different bins and uses the number of classes that best matches the mean and variance of the original, ungrouped data.

Value

A matrix showing the four parameters of the RS and FMKL distribution fit.

Note

In some cases, the resulting fit may not converge, there are currently no checking mechanism in place to ensure global convergence. The RPRS method can sometimes fail if there are no valid percentiles in the data set or if initial values do not give a valid distribution.

Author(s)

Steve Su

References

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.RPRS.hs, fun.RMFMKL.hs, fun.RMFMKL.hs.nw, fun.RPRS.hs.nw, fun.data.fit.hs, fun.data.fit.ml

Examples

```
# Fitting normal(3,2) distriution using the default setting
 junk<-rnorm(1000,3,2)
 fun.data.fit.hs.nw(junk)
```

`fun.data.fit.lm` *Fit data using L moment matching estimation for RS and FMKL GLD*

Description

This function fits generalised lambda distributions to data using L moment matching method

Usage

```
fun.data.fit.lm(data, rs.leap = 3, fmkl.leap = 3, rs.init = c(-1.5, 1.5),
fmkl.init = c(-0.25, 1.5), FUN = "runif.sobol", no = 10000)
```

Arguments

`data`	Dataset to be fitted.
`rs.leap`	See scrambling argument in `fun.gen.qrn`.
`fmkl.leap`	See scrambling argument in `fun.gen.qrn`.
`rs.init`	Inititial values (lambda3 and lambda4) for the RS generalised lambda distribution.
`fmkl.init`	Inititial values (lambda3 and lambda4) for the FMKL generalised lambda distribution.
`FUN`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function consolidates `fun.RPRS.lm` and `fun.RMFMKL.lm` and gives all the fits in one output.

Value

A matrix showing the parameters of RS and FMKL generalised lambda distributions.

Author(s)

Steve Su

References

Asquith, W. (2007). "L-moments and TL-moments of the generalized lambda distribution." Computational Statistics and Data Analysis 51(9): 4484-4496.

Karvanen, J. and A. Nuutinen (2008). "Characterizing the generalized lambda distribution by L-moments." Computational Statistics and Data Analysis 52(4): 1971-1983.

See Also

fun.RPRS.qs, fun.RMFMKL.qs, fun.data.fit.ml, fun.data.fit.hs, fun.data.fit.hs.nw , fun.data.fit.qs, fun.data.fit.mm

Examples

```
# Fitting normal(3,2) distriution using the default setting
 junk<-rnorm(50,3,2)
 fun.data.fit.lm(junk)
```

`fun.data.fit.ml` *Fit data using RS, FMKL maximum likelihood estimation and the FMKL starship method.*

Description

This function fits generalised lambda distributions to data using RPRS, RMFMKL and starship methods.

Usage

```
fun.data.fit.ml(data, rs.leap = 3, fmkl.leap = 3, rs.init = c(-1.5, 1.5),
fmkl.init = c(-0.25, 1.5),FUN="runif.sobol",no=10000)
```

Arguments

`data`	Dataset to be fitted.
`rs.leap`	See scrambling argument in fun.gen.qrn.
`fmkl.leap`	See scrambling argument in fun.gen.qrn.
`rs.init`	Inititial values (lambda3 and lambda4) for the RS generalised lambda distribution.
`fmkl.init`	Inititial values (lambda3 and lambda4) for the FMKL generalised lambda distribution.
`FUN`	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function consolidates fun.RPRS.ml, fun.RMFMKL.ml and starship and gives all the fits in one output.

Value

A matrix showing the parameters of generalised lambda distribution for RPRS, FMFKL and STAR methods.

Note

RPRS can sometimes fail if it is not possible to calculate the percentiles of the data set. This usually happens when the number of data point is small.

Author(s)

Steve Su

References

King, R.A.R. & MacGillivray, H. L. (1999), A starship method for fitting the generalised lambda distributions, Australian and New Zealand Journal of Statistics, 41, 353-374

Su, S. (2007). Numerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Computational statistics and data analysis 51(8) 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.RPRS.ml, fun.RMFMKL.ml, starship, fun.data.fit.hs, fun.data.fit.hs.nw , fun.data.fit.qs , fun.data.fit.mm , fun.data.fit.lm

Examples

```
# Fitting normal(3,2) distriution using the default setting
 junk<-rnorm(50,3,2)
 fun.data.fit.ml(junk)
```

---

fun.data.fit.mm *Fit data using moment matching estimation for RS and FMKL GLD*

Description

This function fits generalised lambda distributions to data using moment matching method

Usage

```
fun.data.fit.mm(data, rs.leap = 3, fmkl.leap = 3, rs.init = c(-1.5, 1.5),
fmkl.init = c(-0.25, 1.5), FUN = "runif.sobol", no = 10000)
```

Arguments

data	Dataset to be fitted.
rs.leap	See scrambling argument in fun.gen.qrn.
fmkl.leap	See scrambling argument in fun.gen.qrn.

rs.init	Inititial values (lambda3 and lambda4) for the RS generalised lambda distribution.
fmkl.init	Inititial values (lambda3 and lambda4) for the FMKL generalised lambda distribution.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function consolidates fun.RPRS.mm and fun.RMFMKL.mm and gives all the fits in one output.

Value

A matrix showing the parameters of RS and FMKL generalised lambda distributions.

Note

RPRS can sometimes fail if it is not possible to calculate the percentiles of the data set. This usually happens when the number of data point is small.

References

Karian, Z. and E. Dudewicz (2000). Fitting Statistical Distributions: The Generalized Lambda Distribution and Generalised Bootstrap Methods. New York, Chapman and Hall.

See Also

fun.RPRS.qs, fun.RMFMKL.qs, fun.data.fit.ml, fun.data.fit.hs, fun.data.fit.hs.nw , fun.data.fit.qs, fun.data.fit.lm

Examples

```
# Fitting normal(3,2) distriution using the default setting
 junk<-rnorm(50,3,2)
 fun.data.fit.mm(junk)
```

---

fun.data.fit.qs *Fit data using quantile matching estimation for RS and FMKL GLD*

Description

This function fits generalised lambda distributions to data using quantile matching method

Usage

```
fun.data.fit.qs(data, rs.leap = 3, fmkl.leap = 3, rs.init = c(-1.5, 1.5),
fmkl.init = c(-0.25, 1.5), FUN = "runif.sobol", trial.n = 100, len = 1000,
type = 7, no = 10000)
```

Arguments

data	Dataset to be fitted.
rs.leap	See scrambling argument in fun.gen.qrn.
fmkl.leap	See scrambling argument in fun.gen.qrn.
rs.init	Inititial values (lambda3 and lambda4) for the RS generalised lambda distribution.
fmkl.init	Inititial values (lambda3 and lambda4) for the FMKL generalised lambda distribution.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
trial.n	Number of evenly spaced quantile ranging from 0 to 1 to be used in the checking phase, to find the best set of initial values for optimisation, this is intended to be lower than len to speed up the fitting algorithm. Default is 100.
len	Number of evenly spaced quantile ranging from 0 to 1 to be used, default is 1000
type	Type of quantile to be used, default is 7, see quantile
no	Number of initial random values to find the best initial values for optimisation.

Details

This function consolidates fun.RPRS.qs and fun.RMFMKL.qs and gives all the fits in one output.

Value

A matrix showing the parameters of RS and FMKL generalised lambda distributions.

Note

RPRS can sometimes fail if it is not possible to calculate the percentiles of the data set. This usually happens when the number of data point is small.

Author(s)

Steve Su

References

Su (2008). Fitting GLD to data via quantile matching method. (Book chapter to appear)

See Also

fun.RPRS.qs, fun.RMFMKL.qs, fun.data.fit.ml, fun.data.fit.hs, fun.data.fit.hs.nw , fun.data.fit.mm, fun.data.fit.lm

Examples

```
# Fitting normal(3,2) distriution using the default setting
 junk<-rnorm(50,3,2)
 fun.data.fit.qs(junk)
```

`fun.diag.ks.g` *Compute the simulated Kolmogorov-Smirnov tests for the unimodal dataset*

Description

This function counts the number of times the p-value exceed 0.05 for the null hypothesis that the observations simulated from the fitted distribution is the same as the observations simulated from the unimodal data set.

Usage

```
fun.diag.ks.g(result, data, no.test = 1000, len = floor(0.9 * length(data)),
param, alpha = 0.05)
```

Arguments

`result`	A vector representing the four parameters of the generalised lambda distribution.
`data`	The unimodal dataset.
`no.test`	Total number of tests required.
`len`	Number of data to sample.
`param`	Type of the generalised lambda distribution, `"rs"` or `"fmkl"`.
`alpha`	Significance level of KS test.

Value

A numerical value representing number of times the p-value exceeds alpha.

Note

If there are ties, jittering is used in `ks.gof`.

Author(s)

Steve Su

References

Stephens, M. A. (1986). Tests based on EDF statistics. In Goodness-of-Fit Techniques. D'Agostino, R. B. and Stevens, M. A., eds. New York: Marcel Dekker.

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su (2007). Nmerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Computational Statistics and Data Analysis: *51*, 8, 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.diag.ks.g.bimodal

Examples

```
# Generate 1000 random observations from Normal distribution with mean=100,
# standard deviation=10. Save this as junk
 junk<-rnorm(1000,100,10)

# Fit junk using RPRS method via the maxmum likelihood.
 fit1<-fun.RPRS.ml(junk, c(-1.5, 1.5), leap = 3)

# Calculate the simulated KS test result:
 fun.diag.ks.g(fit1,junk,param="rs")
```

---

fun.diag.ks.g.bimodal *Compute the simulated Kolmogorov-Smirnov tests for the bimodal dataset*

Description

This function counts the number of times the p-value exceed 0.05 for the null hypothesis that the observations simulated from the fitted distribution is the same as the observations simulated from the bimodal data set.

Usage

```
fun.diag.ks.g.bimodal(result1, result2, prop1, prop2, data, no.test = 1000,
len = floor(0.9 * length(data)), param1, param2, alpha = 0.05)
```

Arguments

result1	A vector representing the four parameters of the first generalised lambda distribution.
result2	A vector representing the four parameters of the second generalised lambda distribution.

prop1	Proportion of the first distribution fitted to the bimodal dataset.
prop2	Proportion of the second distribution fitted to the bimodal dataset.
data	The bimodal dataset.
no.test	Total number of tests required.
len	Number of data to sample.
param1	Type of first generalised lambda distribution, can be "rs" or "fmkl".
param2	Type of second generalised lambda distribution, can be "rs" or "fmkl".
alpha	Significance level of KS test.

Value

A numerical value representing number of times the p-value exceeds alpha.

Note

If there are ties, jittering is used in ks.gof.

Author(s)

Steve Su

References

Stephens, M. A. (1986). Tests based on EDF statistics. In Goodness- of-Fit Techniques. D'Agostino, R. B. and Stevens, M. A., eds. New York: Marcel Dekker.

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su (2007). Nmerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Computational Statistics and Data Analysis: *51*, 8, 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.diag.ks.g

Examples

```
# Fit the faithful[,1] data from the MASS library
 fit1<-fun.auto.bimodal.ml(faithful[,1],init1.sel="rprs",init2.sel="rmfmkl",
 init1=c(-1.5,1,5),init2=c(-0.25,1.5),leap1=3,leap2=3)
# Run diagnostic KS tests
 fun.diag.ks.g.bimodal(fit1$par[1:4],fit1$par[5:8],prop1=fit1$par[9],
 data=faithful[,1],param1="rs",param2="fmkl")
```

`fun.diag1` *Diagnostic function for theoretical distribution fits through the resample Kolmogorov-Smirnoff tests*

Description

This function is primarily designed to be used for testing the fitted distribution with reference to a theoretical distribution. It is also tailored for output obtained from the `fun.data.fit.ml` function.

Usage

```
fun.diag1(result, test, no.test = 1000, alpha = 0.05)
```

Arguments

`result`	Output from `fun.data.fit.ml` function.
`test`	Simulated observations from theoretical distribution, the length should be no.test~2.
`no.test`	Number of times to do the KS tests.
`alpha`	Significance level of KS test.

Value

A vector showing the number of times the KS p-value is greater than alpha for each of the distribution fit strategy.

Note

If there are ties, jittering is used in `ks.gof`.

Author(s)

Steve Su

References

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su, S. (2007). Numerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Journal of Computational statistics and data analysis 51(8) 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

`fun.diag2`, `fun.diag.ks.g`, `fun.diag.ks.g.bimodal`

Examples

```
# Fits a Weibull 5,2 distribution:
 weibull.approx.ml<-fun.data.fit.ml(rweibull(1000,5,2))

# Compute the resample K-S test results.
 fun.diag1(weibull.approx.ml, rweibull(100000, 5, 2))
```

`fun.diag2` *Diagnostic function for empirical data distribution fits through the resample Kolmogorov-Smirnoff tests*

Description

This function is primarily designed to be used for testing the fitted distribution with reference to an empirical data. It is also tailored for output obtained from the `fun.data.fit.ml` function.

Usage

```
fun.diag2(result, data, no.test = 1000, len=100, alpha = 0.05)
```

Arguments

`result`	Output from `fun.data.fit.ml` function.
`data`	Observations in which the distribution was fitted upon.
`no.test`	Number of times to do the KS tests.
`len`	Number of observations to sample from the data. This is also the number of observations sampled from the fitted distribution in each KS test.
`alpha`	Significance level of KS test.

Value

A vector showing the number of times the KS p-value is greater than alpha for each of the distribution fit strategy.

Note

If there are ties, jittering is used in `ks.gof`.

Author(s)

Steve Su

References

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su, S. (2007). Numerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Journal of Computational statistics and data analysis 51(8) 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.diag1, fun.diag.ks.g, fun.diag.ks.g.bimodal

Examples

```
# Fits a Normal 3,2 distribution:
 junk<-rnorm(1000,3,2)
 fit<-fun.data.fit.ml(junk)

# Compute the resample K-S test results.
 fun.diag2(fit,junk)
```

`fun.disc.estimation` *Estimates the mean and variance after cutting up a vector of variable into evenly spaced categories.*

Description

This function supplements fun.nclass.e and it is not intended to be used by the users directly.

Usage

```
fun.disc.estimation(x, nint)
```

Arguments

x	A vector of observations.
nint	Number of intervals to cut the vectors into.

Details

The function cuts the vector into evenly spaced categories and estimate the mean and variance of the actual data based on the categorisation.

Value

Two numerical values, the first being the mean and the second being the variance.

Author(s)

Steve Su

See Also

fun.nclass.e

Examples

```
## Cut up a randomly normally distributed observations into 5 evenly spaced
## categories and estimate the mean and variance based on this cateogorisation.
junk<-rnorm(1000,3,2)
fun.disc.estimation(junk,5)
```

---

fun.gen.qrn *Finds the low discrepancy quasi random numbers*

Description

This function calls the runif.sobol, runif.sobol.owen and runif.halton essentially from the **spacefillr** package.

Usage

```
fun.gen.qrn(n, dimension, scrambling, FUN = "runif.sobol")
```

Arguments

n	Number to generate.
dimension	Number of dimensions.
scrambling	seed used, or leap as in the case of QUnif.
FUN	This can be "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".

Value

A vector of values if dimension=1, otherwise a matrix of values between 0 and 1.

Author(s)

Steve Su

Examples

```
fun.gen.qrn(1000,5,3,"runif.sobol")

fun.gen.qrn(1000,5,409,"QUnif")
```

`fun.lm.theo.gld` *Find the theoretical first four L moments of the generalised lambda distribution.*

Description

This function computes the first four L moments for both RS and FMKL generalised lambda distributions.

Usage

```
fun.lm.theo.gld(L1, L2, L3, L4, param)
```

Arguments

`L1`	Lambda 1. Or c(Lambda 1,Lambda 2,Lambda 3,Lambda 4).
`L2`	Lambda 2.
`L3`	Lambda 3.
`L4`	Lambda 4.
`param`	"rs" or "fmkl" or "fkml"

Value

A vector listing the first four L moments

Note

Sometimes the theoretical moments may not exist, in those cases, `NA` is returned.

Author(s)

Steve Su

References

Asquith, W. (2007). "L-moments and TL-moments of the generalized lambda distribution." Computational Statistics and Data Analysis 51(9): 4484-4496.

Karvanen, J. and A. Nuutinen (2008). "Characterizing the generalized lambda distribution by L-moments." Computational Statistics and Data Analysis 52(4): 1971-1983.

See Also

`fun.theo.mv.gld`

Examples

```
fun.lm.theo.gld(1, 2, 3, 4, "rs")
fun.lm.theo.gld(1, 2, 3, 4, "fmkl")
```

`fun.mApply` *Applying functions based on an index for a matrix.*

Description

This is a generic function that can be used to find mean, variance, sum or other operations according to some index imposed on the matrix or vector.

Usage

```
fun.mApply(X, INDEX, FUN = NULL, ..., simplify = TRUE)
```

Arguments

`X`	Matrix with n rows.
`INDEX`	Vector or list of vectors of length n.
`FUN`	Function to operate on submatrices of `X` by `INDEX`
`...`	Arguments to function.
`simplify`	Set as `TRUE` by default, see `sapply` fo details.

Value

If `FUN` returns more than one number, `fun.mApply` returns a matrix with rows corresponding to unique values of `INDEX`.

Author(s)

Tony Plate

Examples

```
# Finding the row medians of a matrix (matrix(1:20,nrow=5))
fun.mApply(matrix(1:20,nrow=5),list(1:5),median)
```

`fun.minmax.check.gld` *Check whether the specified GLDs cover the minimum and the maximum values in a dataset*

Description

This function checks the lowest and highest quantiles of the specified GLDs against the specified dataset

Usage

```
fun.minmax.check.gld(data, lambdas, param, lessequalmin = 1,
greaterequalmax = 1)
```

Arguments

data	A vector of numerical dataset
lambdas	A matrix of four columns representing lambda 1 to lambda 4 of the GLD
param	Can be "rs", "fkml" or "fmkl"
lessequalmin	Can be 0 or 1
greaterequalmax	Can be 0 or 1

Details

lessequalmin==1 means the lowest value of GLD <= minimum value of data lessequalmin==0 means the lowest value of GLD < minimum value of data greaterequalmin==1 means the highest value of GLD >= maximum value of data greaterequalmin==0 means the highest value of GLD > maximum value of data

Value

A vector of logical values indicating whether the specified data the specified GLDs cover the minimum and the maximum values in a dataset

Author(s)

Steve Su

Examples

```
fun.minmax.check.gld(runif(100,.9,1),matrix(1:12,ncol=4),param="rs",0,0)
fun.minmax.check.gld(runif(100,.98,1),matrix(1:12,ncol=4),param="fkml",1,1)
```

---

fun.moments.bimodal *Finds the moments of fitted mixture of generalised lambda distribution by simulation.*

Description

This functions compute the mean, variance, skewness and kurtosis of the fitted generalised lambda distribution mixtures using Monte Carlo simulation.

Usage

```
fun.moments.bimodal(result1, result2, prop1, prop2, len = 1000,
no.test = 1000, param1, param2)
```

Arguments

result1	A vector comprising four values for the first generalised lambda distribution.
result2	A vector comprising four values for the second generalised lambda distribution.
prop1	Proportion of the first generalised lambda distribution
prop2	1-prop1, this can be left unspecified.
len	Length of object for each simulation run.
no.test	Number of simulation run.
param1	This can be "rs" or "fmkl", specifying the type of the first generalised lambda distribution.
param2	This can be "rs" or "fmkl", specifying the type of the second generalised lambda distribution.

Details

There is also a theoretical computation of the moments in fun.theo.bi.mv.gld, it should be noted that the theoretical moments may not exist. The length of object in len means how many observations should be generated in each simulation run, with the number of simulation runs governed by no.test.

Value

A matrix with four columns showing the mean, variance, skewness and kurtosis of the fitted generalised lambda distribution mixtures using Monte Carlo simulation. Each row represents a simulation run.

Author(s)

Steve Su

See Also

fun.theo.bi.mv.gld, fun.simu.bimodal, fun.rawmoments

Examples

```
# Fitting the first column of the Old Faithful Geyser data
 fit1<-fun.auto.bimodal.ml(faithful[,1],init1.sel="rmfmkl",init2.sel="rmfmkl",
 init1=c(-0.25,1.5),init2=c(-0.25,1.5),leap1=3,leap2=3)

# After fitting compute the monte carlo moments using fun.moments.bimodal
 fun.moments.bimodal(fit1$par[1:4],fit1$par[5:8],prop1=fit1$par[9],
 param1="fmkl",param2="fmkl")

# It is also possible to compare this with the moments of the original dataset:
 fun.moments(faithful[,1])
```

`fun.moments.r` *Calculate mean, variance, skewness and kurtosis of a numerical vector*

Description

This function evaluates the mean, variance, skewness and kurtosis of a numerical vector. Missing values are automatically removed.

Usage

```
fun.moments.r(x, normalise = "N")
```

Arguments

`x`	A numeric vector
`normalise`	"Y" if you want kurtosis to be calculated with reference to kurtosis = 0 under Normal distribution. Default is "N".

Value

A vector of mean, variance, skewness and kurtosis.

Note

Please contact the author directly if you find a bug!

Author(s)

Steve Su

See Also

`fun.theo.mv.gld`

Examples

```
fun.moments.r(rnorm(1000))
fun.moments.r(rnorm(1000),normalise="Y")
```

`fun.nclass.e` *Estimates the number of classes or bins to smooth over in the discretised method of fitting generalised lambda distribution to data.*

Description

Support function for discretised method of fitting distribution to data.

Usage

```
fun.nclass.e(x)
```

Arguments

x	Vector of data.

Details

This function calculates the mean and variance of the discretised data from 1 to the very last observation and chooses the best number of categories that represent the mean and variance of the actual data set through the criterion of squared deviations.

Value

A numerical value suggesting the best number of class that can be used to represent the mean and variane of the original data set.

Note

This is not designed to be called directly by end user.

Author(s)

Steve Su

See Also

`fun.disc.estimation`

Examples

```
fun.nclass.e(rnorm(100,3,2))
```

`fun.plot.fit` *Plotting the univariate generalised lambda distribution fits on the data set.*

Description

This function is designed for univariate generalised lambda distribution fits only.

Usage

```
fun.plot.fit(fit.obj, data, nclass = 50, xlab = "", name = "", param.vec,
ylab="Density", main="")
```

Arguments

`fit.obj`	Fitted object from `fun.data.fit.ml`, `fun.data.fit.hs`, `fun.data.fit.hs.nw`, `fun.RPRS.ml`, `fun.RMFMKL.ml`, `fun.RPRS.hs`, `fun.RMFMKL.hs`, `fun.RPRS.hs.nw`, `fun.RMFMKL.hs.nw`
`data`	Dataset to be plotted.
`nclass`	Number of class of histogram, the default is 50.
`xlab`	Label on the x axis.
`name`	Naming the type of distribution fits.
`param.vec`	A vector describing the type of generalised lambda distribution used in the `fit.obj`.
`ylab`	Label on the y axis.
`main`	Title of the graph.

Value

A graphical output showing the data and the resulting distributional fits.

Note

If the distribution fits over fits the peak of the distribution, it can be difficult to see the actual data set.

Author(s)

Steve Su

See Also

`fun.plot.fit.bm`, `fun.data.fit.ml`, `fun.data.fit.hs`, `fun.data.fit.hs.nw`, `fun.RPRS.ml`, `fun.RMFMKL.ml`, `fun.RPRS.hs`, `fun.RMFMKL.hs`, `fun.RPRS.hs.nw`, `fun.RMFMKL.hs.nw`

Examples

```
# Generate Normally distribute random numbers as dataset
 junk<-rnorm(1000,3,2)

# Fit the data set using fun.data.fit.ml.
# Also, fun.data.fit.hs or fun.data.fit.hs.nw can be used.
 obj.fit<-fun.data.fit.ml(junk)

# Plot the resulting fits
 fun.plot.fit(obj.fit,junk,xlab="x",name=".ML",param.vec=c("rs","fmkl","fmkl"))

# This function also works for singular fits such as those by fun.RPRS.ml,
# fun.RMFMKL.ml, fun.RPRS.hs, fun.RMFMKL.hs, fun.RPRS.hs.nw, fun.RMFMKL.hs.nw
 junk<-rnorm(1000,3,2)
 obj.fit<-fun.RPRS.ml(junk)
 fun.plot.fit(obj.fit,junk,xlab="x",name="RPRS.ML",param.vec=c("rs"))
```

`fun.plot.fit.bm` *Plotting mixture of two generalised lambda distributions on the data set.*

Description

This function is designed for mixture of two generalised lambda distributions only.

Usage

```
fun.plot.fit.bm(fit.obj, data, nclass = 50, xlab = "", name = "", main="",
param.vec, ylab="Density")
```

Arguments

`fit.obj`	Fitted object from `fun.auto.bimodal.ml`, `fun.auto.bimodal.pml`
`data`	Dataset to be plotted.
`nclass`	Number of class of histogram, the default is 50.
`xlab`	Label on the x axis.
`name`	Legend, usually used to identify type of GLD used if `main` is provided. If `main` is not provided, then this is used in the title.
`main`	Title of the graph.
`param.vec`	A vector describing the type of generalised lambda distribution used in the `fit.obj`.
`ylab`	Label on the y axis.

Value

A graphical output showing the data and the resulting distributional fits.

Note

If the distribution fits over fits the peak of the distribution, it can be difficult to see the actual data set.

Author(s)

Steve Su

See Also

fun.auto.bimodal.ml, fun.auto.bimodal.pml, fun.plot.fit

Examples

```
 opar <- par()
 par(mfrow=c(2,1))

# Fitting mixture of generalised lambda distributions on the data set using
# both the maximum likelihood and partition maximum likelihood and plot
# the resulting fits

 junk<-fun.auto.bimodal.ml(faithful[,1],per.of.mix=0.1,clustering.m=clara,
 init1.sel="rprs",init2.sel="rmfmkl",init1=c(-1.5,1,5),init2=c(-0.25,1.5),
 leap1=3,leap2=3)
 fun.plot.fit.bm(nclass=50,fit.obj=junk,data=faithful[,1],
 name="Maximum likelihood using",xlab="faithful1",param.vec=c("rs","fmkl"))

 junk<-fun.auto.bimodal.pml(faithful[,1],clustering.m=clara,init1.sel="rprs",
 init2.sel="rmfmkl",init1=c(-1.5,1,5),init2=c(-0.25,1.5),leap1=3,leap2=3)
 fun.plot.fit.bm(nclass=50,fit.obj=junk,data=faithful[,1],
 name="Partition maximum likelihood using",xlab="faithful1",
 param.vec=c("rs","fmkl"))

 junk<-fun.auto.bimodal.ml(faithful[,1],per.of.mix=0.1,clustering.m=clara,
 init1.sel="rprs",init2.sel="rmfmkl",init1=c(-1.5,1,5),init2=c(-0.25,1.5),
 leap1=3,leap2=3)
 fun.plot.fit.bm(nclass=50,fit.obj=junk,data=faithful[,1],
 main="Mixture distribution fit",
 name="RS and FMKL GLD",xlab="faithful1",param.vec=c("rs","fmkl"))

 par(opar)
```

---

fun.plot.many.gld *Plotting many univariate generalised lambda distributions on one page.*

Description

This is a variant of the fun.plot.fit function.

Usage

```
fun.plot.many.gld(fit.obj, data, xlab="", ylab="Density", main="", legd="",
param.vec)
```

Arguments

fit.obj	A matrix of generalised lambda distibutions parameters from fun.data.fit.ml, fun.data.fit.hs, fun.data.fit.hs.nw, fun.RPRS.ml, fun.RMFMKL.ml, fun.RPRS.hs, fun.RMFMKL.hs, fun.RPRS.hs.nw, fun.RMFMKL.hs.nw functions. Or a matrix of generalised lambda distribution parameters.
data	Dataset to be plotted or two values showing the ranges of value to be compared.
xlab	X-axis labels.
ylab	Y-axis labels.
main	Title for the plot.
legd	Legend for the plot.
param.vec	A vector showing the types of generalised lambda distributions. This can be "rs" or "fmkl", only needed if you want to put your own parameters for generalised lambda distributions which are not generated from a fitting algorithm in this package.

Value

A graph showing the different distributions on the same page.

Note

The data part of the function is not plotted, to see the dataset use the fun.plot.fit function.

Author(s)

Steve Su

See Also

fun.plot.fit, fun.plot.fit.bm

Examples

```
# Fit the dataset
 junk<-rnorm(1000,3,2)
 result.hs<-fun.data.fit.hs(junk,rs.default = "Y", fmkl.default = "Y",
 rs.leap=3, fmkl.leap=3,rs.init = c(-1.5, 1.5), fmkl.init = c(-0.25, 1.5),
 no.c.rs=50,no.c.fmkl=50)
```

```
 opar <- par()
 par(mfrow=c(2,2))

# Plot the entire data range
 fun.plot.many.gld(result.hs,junk,"x","density","",
 legd=c("RPRS.hs", "RMFMKL.hs"))

# Plot and compare parts of the distributions
 fun.plot.many.gld(result.hs,c(1,2),"x","density","",legd=c("RPRS.hs",
"RMFMKL.hs"))
 fun.plot.many.gld(result.hs,c(0.1,0,2),"x","density","",legd=c("RPRS.hs",
"RMFMKL.hs"))
 fun.plot.many.gld(result.hs,c(3,4),"x","density","",legd=c("RPRS.hs",
"RMFMKL.hs"))

 par(opar)
```

`fun.rawmoments` *Computes the raw moments of the generalised lambda distribution up to 4th order.*

Description

This function is of theoretical interest only.

Usage

```
fun.rawmoments(L1, L2, L3, L4, param = "fmkl")
```

Arguments

`L1`	Location parameter of the generalised lambda distribution.
`L2`	Scale parameter of the generalised lambda distribution.
`L3`	First shape parameter of the generalised lambda distribution.
`L4`	Second shape parameter of the generalised lambda distribution.
`param`	`"rs"` or `"fmkl"` specifying the type of the generalised lambda distribution.

Details

This function is the building block for `fun.theo.bi.mv.gld`.

Value

A vector showing the raw moments of the specified generalised lambda distribution up to the fourth order.

Author(s)

Steve Su

References

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), A study of the generalized tukey lambda family, Communications in Statistics - Theory and Methods *17*, 3547-3567.

Karian, Zaven A. and Dudewicz, Edward J. (2000), Fitting statistical distributions: the Generalized Lambda Distribution and Generalized Bootstrap methods, Chapman & Hall

Ramberg, J. S. & Schmeiser, B. W. (1974), An approximate method for generating asymmetric random variables, Communications of the ACM *17*, 78-82.

See Also

~~objects to See Also as help, ~~~

Examples

```
## Generate some random numbers using FMKL and RS generalised lambda
## distributions and then compute the empirical and theoretical
## E(X), E(X^2), E(X^3), E(X^4)

junk<-rgl(100000,1,2,3,4)
mean(junk)
mean(junk^2)
mean(junk^3)
mean(junk^4)

junk<-rgl(100000,1,2,3,4,"rs")
mean(junk)
mean(junk^2)
mean(junk^3)
mean(junk^4)

fun.rawmoments(1,2,3,4)
fun.rawmoments(1,2,3,4,"rs")
```

---

fun.RMFMKL.hs *Fit FMKL generalised distribution to data using discretised approach with weights.*

Description

This function fits FMKL generalised distribution to data using discretised approach with weights. It is designed to act as a smoother device rather than as a definitive fit.

Usage

```
fun.RMFMKL.hs(data, default = "Y", fmkl.init = c(-0.25, 1.5), no.c.fmkl = 50,
leap = 3,FUN="runif.sobol",no=10000)
```

Arguments

data	Dataset to be fitted
default	If yes, this function uses the default method fun.nclass.e to calculate number of classes required.
fmkl.init	Initial values for FMKL distribution optimization, c(-0.25,1.5) tends to work well.
no.c.fmkl	Number of classes or bins of histogram to be optimized over. This argument is ineffective if default="Y".
leap	See scrambling argument in fun.gen.qrn.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function optimises the deviations of frequency of the bins to that of the theoretical so it has the effect of "fitting clothes" onto the data set. The user can decide the frequency of the bins they want the distribution to smooth over. The resulting fit may or may not be an adequate fit from a formal statistical point of view such as satisfying the goodness of fit for example, but it can be useful to suggest the range of different distributions exhibited by the data set. The default number of classes calculates the mean and variance after categorising the data into different bins and uses the number of classes that best matches the mean and variance of the original, ungrouped data. The weighting is designed to accentuate the peak or the dense part of the distribution and suppress the tails.

Value

A vector representing four parametefmkl of the FMKL generalised lambda distribution.

Note

In some cases, the resulting fit may not converge, there are currently no checking mechanism in place to ensure global convergence.

Author(s)

Steve Su

References

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.RMFMKL.hs.nw, fun.RPRS.hs.nw, fun.RPRS.hs, fun.data.fit.hs, fun.data.fit.hs.nw

Examples

```
# Using the default number of classes
 fun.RMFMKL.hs(data=rnorm(1000,3,2),default="Y",fmkl.init=c(-0.25,1.5),leap=3)
# Using 20 classes
 fun.RMFMKL.hs(data=rnorm(1000,3,2),default="N",fmkl.init=c(-0.25,1.5),
 no.c.fmkl=20,leap=3)
```

---

fun.RMFMKL.hs.nw *Fit FMKL generalised distribution to data using discretised approach without weights.*

Description

This function fits FMKL generalised distribution to data using discretised approach without weights. It is designed to act as a smoother device rather than as a definitive fit.

Usage

```
fun.RMFMKL.hs.nw(data, default = "Y", fmkl.init = c(-0.25, 1.5),
no.c.fmkl = 50, leap = 3,FUN="runif.sobol",no=10000)
```

Arguments

data	Dataset to be fitted
default	If yes, this function uses the default method fun.nclass.e to calculate number of classes required.
fmkl.init	Initial values for FMKL distribution optimization, c(-0.25,1.5) tends to work well.
no.c.fmkl	Number of classes or bins of histogram to be optimized over. This argument is ineffective if default="Y".
leap	See scrambling argument in fun.gen.qrn.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function optimises the deviations of frequency of the bins to that of the theoretical so it has the effect of "fitting clothes" onto the data set. The user can decide the frequency of the bins they want the distribution to smooth over. The resulting fit may or may not be an adequate fit from a formal statistical point of view such as satisfying the goodness of fit for example, but it can be useful to suggest the range of different distributions exhibited by the data set. The default number of classes calculates the mean and variance after categorising the data into different bins and uses the number of classes that best matches the mean and variance of the original, ungrouped data.

Value

A vector representing four parameters of the FMKL generalised lambda distribution.

Note

In some cases, the resulting fit may not converge, there are currently no checking mechanism in place to ensure global convergence.

Author(s)

Steve Su

References

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.RPRS.hs.nw, fun.RMFMKL.hs, fun.RPRS.hs, fun.data.fit.hs, fun.data.fit.hs.nw

Examples

```
# Using the default number of classes
 fun.RMFMKL.hs.nw(data=rnorm(1000,3,2),default="Y",
 fmkl.init=c(-0.25,1.5),leap=3)
# Using 20 classes
 fun.RMFMKL.hs.nw(data=rnorm(1000,6,5),default="N",fmkl.init=c(-0.25,1.5),
 no.c.fmkl=20,leap=3)
```

---

fun.RMFMKL.lm *Fit FMKL generalised lambda distribution to data set using L moment matching*

Description

This function fits FMKL generalised lambda distribution to data set using L moment matching

Usage

```
fun.RMFMKL.lm(data, fmkl.init = c(-0.25, 1.5), leap = 3, FUN = "runif.sobol",
no = 10000)
```

Arguments

data	Dataset to be fitted
fmkl.init	Initial values for FMKL distribution optimization, c(-0.25,1.5) tends to work well.
leap	See scrambling argument in fun.gen.qrn.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function provides method of L moment fitting scheme for FMKL GLD. Note this function can fail if there are no defined percentiles from the data set or if the initial values do not lead to a valid FMKL generalised lambda distribution.

This function is based on scheme detailed in the literature below but it has been modified by the author (Steve Su).

Value

A vector representing four parametefmkl of the FMKL generalised lambda distribution.

Author(s)

Steve Su

References

Asquith, W. (2007). "L-moments and TL-moments of the generalized lambda distribution." Computational Statistics and Data Analysis 51(9): 4484-4496.

Karvanen, J. and A. Nuutinen (2008). "Characterizing the generalized lambda distribution by L-moments." Computational Statistics and Data Analysis 52(4): 1971-1983.

See Also

fun.RMFMKL.ml, fun.RMFMKL.mm, fun.RMFMKL.qs, fun.data.fit.ml fun.data.fit.lm, fun.data.fit.qs, fun.data.fit.mm

Examples

```
# Fitting the normal distribution
 fun.RMFMKL.lm(data=rnorm(1000,2,3),fmkl.init=c(-0.25,1.5),leap=3)
```

`fun.RMFMKL.ml` *Fit FMKL generalised lambda distribution to data set using maximum likelihood estimation*

Description

This function fits FMKL generalised lambda distribution to data set using maximum likelihood estimation.

Usage

```
fun.RMFMKL.ml(data, fmkl.init = c(-0.25, 1.5), leap = 3,FUN="runif.sobol",
no=10000)
```

Arguments

`data`	Dataset to be fitted
`fmkl.init`	Initial values for FMKL distribution optimization, `c(-0.25,1.5)` tends to work well.
`leap`	See scrambling argument in `fun.gen.qrn`.
`FUN`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function provides one of the definitive fit to data set using generalised lambda distributions.

Value

A vector representing four parameters of the FMKL generalised lambda distribution.

Author(s)

Steve Su

References

Su, S. (2007). Numerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Journal of Computational statistics and data analysis 51(8) 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

`fun.RPRS.ml`, `fun.data.fit.ml`, `fun.data.fit.qs`

Examples

```
# Fitting the normal distribution
 fun.RMFMKL.ml(data=rnorm(1000,2,3),fmkl.init=c(-0.25,1.5),leap=3)
```

`fun.RMFMKL.ml.m` *Fit RS generalised lambda distribution to data set using maximum likelihood estimation*

Description

This function fits FMKL generalised lambda distribution to data set using maximum likelihood estimation using faster implementation through C programming

Usage

```
fun.RMFMKL.ml.m(data, fmkl.init = c(-0.25, 1.5), leap = 3, FUN = "runif.sobol",
no = 10000)
```

Arguments

`data`	Dataset to be fitted
`fmkl.init`	Initial values for FMKL distribution optimization, `c(-0.25,1.5)` tends to work well.
`leap`	See scrambling argument in `fun.gen.qrn`.
`FUN`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function provides one of the definitive fit to data set using generalised lambda distributions.

Value

A vector representing four parameters of the FMKL generalised lambda distribution.

Author(s)

Steve Su

References

Su, S. (2007). Numerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Journal of Computational statistics and data analysis 51(8) 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.RMFMKL.ml

Examples

```
# Fitting the normal distribution
 fun.RMFMKL.ml.m(data=rnorm(1000,2,3),fmkl.init=c(-0.25,1.5),leap=3)
```

---

fun.RMFMKL.mm *Fit FMKL generalised lambda distribution to data set using moment matching*

Description

This function fits FMKL generalised lambda distribution to data set using moment matching

Usage

```
fun.RMFMKL.mm(data, fmkl.init = c(-0.25, 1.5), leap = 3, FUN = "runif.sobol",
no = 10000)
```

Arguments

data	Dataset to be fitted
fmkl.init	Initial values for FMKL distribution optimization, c(-0.25,1.5) tends to work well.
leap	See scrambling argument in fun.gen.qrn.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function provides method of moment fitting scheme for FMKL GLD. Note this function can fail if there are no defined percentiles from the data set or if the initial values do not lead to a valid FMKL generalised lambda distribution.

This function is based on scheme detailed in the literature below but it has been modified by the author (Steve Su).

Value

A vector representing four parametefmkl of the FMKL generalised lambda distribution.

Author(s)

Steve Su

References

Karian, Z. and E. Dudewicz (2000). Fitting Statistical Distributions: The Generalized Lambda Distribution and Generalised Bootstrap Methods. New York, Chapman and Hall.

See Also

`fun.RMFMKL.ml`, `fun.RMFMKL.lm`, `fun.RMFMKL.qs`, `fun.data.fit.ml` `fun.data.fit.lm`, `fun.data.fit.qs`, `fun.data.fit.mm`

Examples

```
# Fitting the normal distribution
 fun.RMFMKL.mm(data=rnorm(1000,2,3),fmkl.init=c(-0.25,1.5),leap=3)
```

`fun.RMFMKL.qs` *Fit FMKL generalised lambda distribution to data set using quantile matching*

Description

This function fits FMKL generalised lambda distribution to data set using quantile matching

Usage

```
fun.RMFMKL.qs(data, fmkl.init = c(-0.25, 1.5), leap = 3, FUN = "runif.sobol",
trial.n = 100, len = 1000, type = 7, no = 10000)
```

Arguments

`data`	Dataset to be fitted
`fmkl.init`	Initial values for FMKL distribution optimization, `c(-0.25,1.5)` tends to work well.
`leap`	See scrambling argument in `fun.gen.qrn`.
`FUN`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`trial.n`	Number of evenly spaced quantile ranging from 0 to 1 to be used in the checking phase, to find the best set of initial values for optimisation, this is intended to be lower than `len` to speed up the fitting algorithm. Default is 100.
`len`	Number of evenly spaced quantile ranging from 0 to 1 to be used, default is 1000
`type`	Type of quantile to be used, default is 7, see `quantile`
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function provides quantile matching fitting scheme for FMKL GLD. Note this function can fail if there are no defined percentiles from the data set or if the initial values do not lead to a valid FMKL generalised lambda distribution.

Value

A vector representing four parametefmkl of the FMKL generalised lambda distribution.

Author(s)

Steve Su

References

Su (2008). Fitting GLD to data via quantile matching method. (Book chapter to appear)

See Also

fun.RMFMKL.ml, fun.RMFMKL.lm, fun.RMFMKL.mm, fun.data.fit.ml fun.data.fit.lm, fun.data.fit.qs, fun.data.fit.mm

Examples

```
# Fitting the normal distribution
 fun.RMFMKL.qs(data=rnorm(1000,2,3),fmkl.init=c(-0.25,1.5),leap=3)
```

---

fun.RPRS.hs *Fit RS generalised distribution to data using discretised approach with weights.*

Description

This function fits RS generalised distribution to data using discretised approach with weights. It is designed to act as a smoother device rather than as a definitive fit.

Usage

```
fun.RPRS.hs(data, default = "Y", rs.init = c(-1.5, 1.5), no.c.rs = 50,
leap = 3,FUN="runif.sobol",no=10000)
```

Arguments

data	Dataset to be fitted
default	If yes, this function uses the default method fun.nclass.e to calculate number of classes required.
rs.init	Initial values for RS distribution optimization, c(-1.5,1.5) tends to work well.
no.c.rs	Number of classes or bins of histogram to be optimized over. This argument is ineffective if default="Y".
leap	See scrambling argument in fun.gen.qrn.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function optimises the deviations of frequency of the bins to that of the theoretical so it has the effect of "fitting clothes" onto the data set. The user can decide the frequency of the bins they want the distribution to smooth over. The resulting fit may or may not be an adequate fit from a formal statistical point of view such as satisfying the goodness of fit for example, but it can be useful to suggest the range of different distributions exhibited by the data set. The default number of classes calculates the mean and variance after categorising the data into different bins and uses the number of classes that best matches the mean and variance of the original, ungrouped data. The weighting is designed to accentuate the peak or the dense part of the distribution and suppress the tails.

Value

A vector representing four parameters of the RS generalised lambda distribution.

Note

In some cases, the resulting fit may not converge, there are currently no checking mechanism in place to ensure global convergence. The RPRS method can sometimes fail if there are no valid percentiles in the data set or if initial values do not give a valid distribution.

Author(s)

Steve Su

References

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.RPRS.hs.nw, fun.RMFMKL.hs.nw, fun.RMFMKL.hs, fun.data.fit.hs, fun.data.fit.hs.nw

Examples

```
# Using the default number of classes
 fun.RPRS.hs(data=rnorm(1000,2,3),default="Y",rs.init=c(-1.5,1.5),leap=3)
# Using 20 classes
 fun.RPRS.hs(data=rnorm(1000,2,3),default="N",rs.init=c(-1.5,1.5),
 no.c.rs=20,leap=3)
```

`fun.RPRS.hs.nw` *Fit RS generalised distribution to data using discretised approach without weights.*

Description

This function fits RS generalised distribution to data using discretised approach without weights. It is designed to act as a smoother device rather than as a definitive fit.

Usage

```
fun.RPRS.hs.nw(data, default = "Y", rs.init = c(-1.5, 1.5), no.c.rs = 50,
leap = 3,FUN="runif.sobol",no=10000)
```

Arguments

`data`	Dataset to be fitted
`default`	If yes, this function uses the default method `fun.nclass.e` to calculate number of classes required.
`rs.init`	Initial values for RS distribution optimization, `c(-1.5,1.5)` tends to work well.
`no.c.rs`	Number of classes or bins of histogram to be optimized over. This argument is ineffective if `default="Y"`.
`leap`	See scrambling argument in `fun.gen.qrn`.
`FUN`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function optimises the deviations of frequency of the bins to that of the theoretical so it has the effect of "fitting clothes" onto the data set. The user can decide the frequency of the bins they want the distribution to smooth over. The resulting fit may or may not be an adequate fit from a formal statistical point of view such as satisfying the goodness of fit for example, but it can be useful to suggest the range of different distributions exhibited by the data set. The default number of classes calculates the mean and variance after categorising the data into different bins and uses the number of classes that best matches the mean and variance of the original, ungrouped data.

Value

A vector representing four parameters of the RS generalised lambda distribution.

Note

In some cases, the resulting fit may not converge, there are currently no checking mechanism in place to ensure global convergence. The RPRS method can sometimes fail if there are no valid percentiles in the data set or if initial values do not give a valid distribution.

Author(s)

Steve Su

References

Su, S. (2005). A Discretized Approach to Flexibly Fit Generalized Lambda Distributions to Data. Journal of Modern Applied Statistical Methods (November): 408-424.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

fun.RPRS.hs.nw, fun.RPRS.hs, fun.RMFMKL.hs, fun.data.fit.hs, fun.data.fit.hs.nw

Examples

```
# Using the default number of classes
 fun.RPRS.hs.nw(data=rnorm(1000,3,2),default="Y",rs.init=c(-1.5,1.5),leap=3)
# Using 20 classes
 fun.RPRS.hs.nw(data=rnorm(1000,3,2),default="N",rs.init=c(-1.5,1.5),
 no.c.rs=20,leap=3)
```

---

fun.RPRS.lm *Fit RS generalised lambda distribution to data set using L moment matching*

Description

This function fits RS generalised lambda distribution to data set using L moment matching

Usage

```
fun.RPRS.lm(data, rs.init = c(-1.5, 1.5), leap = 3, FUN = "runif.sobol",
no = 10000)
```

Arguments

data	Dataset to be fitted
rs.init	Initial values for RS distribution optimization, c(-1.5,1.5) tends to work well.
leap	See scrambling argument in fun.gen.qrn.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
no	Number of initial random values to find the best initial values for optimisation.

Details

This function provides method of L moment fitting scheme for RS GLD. Note this function can fail if there are no defined percentiles from the data set or if the initial values do not lead to a valid RS generalised lambda distribution.

This function is based on scheme detailed in the literature below but it has been modified by the author (Steve Su).

Value

A vector representing four parameters of the RS generalised lambda distribution.

Author(s)

Steve Su

References

Asquith, W. (2007). "L-moments and TL-moments of the generalized lambda distribution." Computational Statistics and Data Analysis 51(9): 4484-4496.

Karvanen, J. and A. Nuutinen (2008). "Characterizing the generalized lambda distribution by L-moments." Computational Statistics and Data Analysis 52(4): 1971-1983.

See Also

fun.RPRS.ml, fun.RPRS.mm, fun.RPRS.qs, fun.data.fit.ml fun.data.fit.lm, fun.data.fit.qs, fun.data.fit.mm

Examples

```
# Fitting the normal distribution
 fun.RPRS.lm(data=rnorm(1000,2,3),rs.init=c(-1.5,1.5),leap=3)
```

`fun.RPRS.ml` *Fit RS generalised lambda distribution to data set using maximum likelihood estimation*

Description

This function fits RS generalised lambda distribution to data set using maximum likelihood estimation.

Usage

```
fun.RPRS.ml(data, rs.init = c(-1.5, 1.5), leap = 3,FUN="runif.sobol",no=10000)
```

Arguments

`data`	Dataset to be fitted
`rs.init`	Initial values for RS distribution optimization, `c(-1.5,1.5)` tends to work well.
`leap`	See scrambling argument in `fun.gen.qrn`.
`FUN`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function provides one of the definitive fit to data set using generalised lambda distributions. Note this function can fail if there are no defined percentiles from the data set or if the initial values do not lead to a valid RS generalised lambda distribution.

Value

A vector representing four parameters of the RS generalised lambda distribution.

Author(s)

Steve Su

References

Su, S. (2007). Numerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Computational statistics and data analysis 51(8) 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

`fun.RMFMKL.ml`, `fun.data.fit.ml`, `fun.data.fit.qs`

Examples

```
# Fitting the normal distribution
 fun.RPRS.ml(data=rnorm(1000,2,3),rs.init=c(-1.5,1.5),leap=3)
```

`fun.RPRS.ml.m` *Fit RS generalised lambda distribution to data set using maximum likelihood estimation*

Description

This function fits RS generalised lambda distribution to data set using maximum likelihood estimation using faster implementation through C programming

Usage

```
fun.RPRS.ml.m(data, rs.init = c(-1.5, 1.5), leap = 3, FUN = "runif.sobol",
no = 10000)
```

Arguments

`data`	Dataset to be fitted
`rs.init`	Initial values for RS distribution optimization, `c(-1.5,1.5)` tends to work well.
`leap`	See scrambling argument in `fun.gen.qrn`.
`FUN`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function provides one of the definitive fit to data set using generalised lambda distributions. Note this function can fail if there are no defined percentiles from the data set or if the initial values do not lead to a valid RS generalised lambda distribution.

Value

A vector representing four parameters of the RS generalised lambda distribution.

Author(s)

Steve Su

References

Su, S. (2007). Numerical Maximum Log Likelihood Estimation for Generalized Lambda Distributions. Computational statistics and data analysis 51(8) 3983-3998.

Su (2007). Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. Journal of Statistical Software: *21* 9.

See Also

`fun.RPRS.ml`

Examples

```
# Fitting the normal distribution
 fun.RPRS.ml.m(data=rnorm(1000,2,3),rs.init=c(-1.5,1.5),leap=3)
```

`fun.RPRS.mm` *Fit RS generalised lambda distribution to data set using moment matching*

Description

This function fits RS generalised lambda distribution to data set using moment matching

Usage

```
fun.RPRS.mm(data, rs.init = c(-1.5, 1.5), leap = 3, FUN = "runif.sobol",
no = 10000)
```

Arguments

`data`	Dataset to be fitted
`rs.init`	Initial values for RS distribution optimization, `c(-1.5,1.5)` tends to work well.
`leap`	See scrambling argument in `fun.gen.qrn`.
`FUN`	A character string of either `"runif.sobol"` (default), `"runif.sobol.owen"`, `"runif.halton"` or `"QUnif"`.
`no`	Number of initial random values to find the best initial values for optimisation.

Details

This function provides method of moment fitting scheme for RS GLD. Note this function can fail if there are no defined percentiles from the data set or if the initial values do not lead to a valid RS generalised lambda distribution.

This function is based on scheme detailed in the literature below but it has been modified by the author (Steve Su).

Value

A vector representing four parameters of the RS generalised lambda distribution.

Author(s)

Steve Su

References

Karian, Z. and E. Dudewicz (2000). Fitting Statistical Distributions: The Generalized Lambda Distribution and Generalised Bootstrap Methods. New York, Chapman and Hall.

See Also

fun.RPRS.ml, fun.RPRS.lm, fun.RPRS.qs, fun.data.fit.ml fun.data.fit.lm, fun.data.fit.qs, fun.data.fit.mm

Examples

```
# Fitting the normal distribution
 fun.RPRS.mm(data=rnorm(1000,2,3),rs.init=c(-1.5,1.5),leap=3)
```

---

fun.RPRS.qs *Fit RS generalised lambda distribution to data set using quantile matching*

Description

This function fits RS generalised lambda distribution to data set using quantile matching

Usage

```
fun.RPRS.qs(data, rs.init = c(-1.5, 1.5), leap = 3, FUN = "runif.sobol",
trial.n = 100, len = 1000, type = 7, no = 10000)
```

Arguments

data	Dataset to be fitted
rs.init	Initial values for RS distribution optimization, c(-1.5,1.5) tends to work well.
leap	See scrambling argument in fun.gen.qrn.
FUN	A character string of either "runif.sobol" (default), "runif.sobol.owen", "runif.halton" or "QUnif".
trial.n	Number of evenly spaced quantile ranging from 0 to 1 to be used in the checking phase, to find the best set of initial values for optimisation, this is intended to be lower than len to speed up the fitting algorithm. Default is 100.
len	Number of evenly spaced quantile ranging from 0 to 1 to be used, default is 1000
type	Type of quantile to be used, default is 7, see quantile
no	Number of initial random values to find the best initial values for optimisation.

Details

This function provides quantile matching fitting scheme for RS GLD. Note this function can fail if there are no defined percentiles from the data set or if the initial values do not lead to a valid RS generalised lambda distribution.

Value

A vector representing four parameters of the RS generalised lambda distribution.

Author(s)

Steve Su

References

Su (2008). Fitting GLD to data via quantile matching method. (Book chapter to appear)

See Also

fun.RPRS.ml, fun.RPRS.lm, fun.RPRS.mm, fun.data.fit.ml fun.data.fit.lm, fun.data.fit.qs, fun.data.fit.mm

Examples

```
# Fitting the normal distribution
 fun.RPRS.qs(data=rnorm(1000,2,3),rs.init=c(-1.5,1.5),leap=3)
```

---

fun.simu.bimodal *Simulate a mixture of two generalised lambda distributions.*

Description

This function allows the user to simulate observations from a mixture of two generalised lambda distributions. It can be very useful for sensitivity analysis.

Usage

```
fun.simu.bimodal(result1, result2, prop1, prop2, len = 1000,
no.test = 1000, param1, param2)
```

Arguments

result1	A vector comprising four values for the first generalised lambda distribution.
result2	A vector comprising four values for the second generalised lambda distribution.
prop1	Proportion of the first generalised lambda distribution
prop2	1-prop1, this can be left unspecified.
len	Length of object for each simulation run.
no.test	Number of simulation run.
param1	This can be "rs" or "fmkl", specifying the type of the first generalised lambda distribution.
param2	This can be "rs" or "fmkl", specifying the type of the second generalised lambda distribution.

Details

The length of object in `len` means how many observations should be generated in each simulation run, with the number of simulation runs governed by `no.test`.

Value

A list with length equal to the number of simulation runs. Each subset of the list has random observations equal to the the number specified in `len`.

Author(s)

Steve Su

See Also

`fun.theo.bi.mv.gld`, `fun.moments.bimodal`, `fun.rawmoments`

Examples

```
# Generate random observations from FMKL generalised lambda distributions with
# parameters (1,2,3,4) and (4,3,2,1) with 50% of data from each distribution.
junk<-fun.simu.bimodal(c(1,2,3,4),c(4,3,2,1),prop1=0.5,param1="fmkl",
param2="fmkl")

# Calculate the maximum number from each simulation run
sapply(junk,max)

# Calculate the median from each simulation run
sapply(junk,median)
```

---

fun.theo.bi.mv.gld *Calculates the theoretical mean, variance, skewness and kurtosis for mixture of two generalised lambda distributions.*

Description

This is the bimodal counterpart for `fun.comp.moments.ml.2` and `fun.comp.moments.ml`.

Usage

```
fun.theo.bi.mv.gld(L1, L2, L3, L4, param1, M1, M2, M3, M4, param2, p1,
normalise="N")
```

Arguments

L1	Location parameter of the first generalised lambda distribution. Or all the parameters of mixture distribution in the form of c(L1,L2,L3,L4,M1,M2,M3,M4,p), you still must specify param1 and param2.
L2	Scale parameter of the first generalised lambda distribution.
L3	First shape parameter of the first generalised lambda distribution.
L4	Second shape parameter of the first generalised lambda distribution.
param1	"rs" or "fmkl" specifying the type of the first generalised lambda distribution.
M1	Location parameter of the second generalised lambda distribution
M2	Scale parameter of the second generalised lambda distribution.
M3	First shape parameter of the second generalised lambda distribution.
M4	Second shape parameter of the second generalised lambda distribution.
param2	"rs" or "fmkl" specifying the type of the second generalised lambda distribution.
p1	Proportion of the first generalisd lambda distribution.
normalise	"Y" if you want kurtosis to be calculated with reference to kurtosis = 0 under Normal distribution.

Value

A vector showing the theoretical mean, variance, skewness and kurtosis for mixture of two generalised lambda distributions.

Note

The theoretical moments may not always exist for generalised lambda distributions.

Author(s)

Steve Su

References

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), A study of the generalized tukey lambda family, Communications in Statistics - Theory and Methods *17*, 3547-3567.

Karian, Zaven A. and Dudewicz, Edward J. (2000), Fitting statistical distributions: the Generalized Lambda Distribution and Generalized Bootstrap methods, Chapman & Hall

Ramberg, J. S. & Schmeiser, B. W. (1974), An approximate method for generating asymmetric random variables, Communications of the ACM *17*, 78-82.

See Also

fun.moments.bimodal, fun.simu.bimodal, fun.rawmoments

Examples

```
# Fits the Old Faithful geyser data (first column) using the maximum
# likelihood.
 fit1<-fun.auto.bimodal.ml(faithful[,1],init1.sel="rmfmkl",init2.sel="rmfmkl",
 init1=c(-0.25,1.5),init2=c(-0.25,1.5),leap1=3,leap2=3)

# Find the theoretical moments of the fit
 fun.theo.bi.mv.gld(fit1$par[1],fit1$par[2],fit1$par[3],fit1$par[4],"fmkl",
 fit1$par[5],fit1$par[6],fit1$par[7],fit1$par[8],"fmkl",fit1$par[9])

# Compare this with the empirical moments from the data set.
 fun.moments(faithful[,1])
```

`fun.theo.mv.gld` *Find the theoretical first four moments of the generalised lambda distribution.*

Description

Computes the "mean","variance","skewness","kurtosis" statistics from a given generalised lambda distribution.

Usage

```
fun.theo.mv.gld(L1, L2, L3, L4, param, normalise="N")
```

Arguments

`L1`	Lambda 1. Or c(Lambda 1,Lambda 2,Lambda 3,Lambda 4).
`L2`	Lambda 2.
`L3`	Lambda 3.
`L4`	Lambda 4.
`param`	"rs" or "fmkl" or "fkml"
`normalise`	"Y" if you want kurtosis to be calculated with reference to kurtosis = 0 under Normal distribution.

Value

A vector listing the values of mean, variance, skewness and kurtosis.

Note

Sometimes the theoretical moments may not exist, in those cases, `NA` is returned.

Author(s)

Steve Su

References

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), A study of the generalized tukey lambda family, Communications in Statistics - Theory and Methods *17*, 3547-3567.

Gilchrist, Warren G. (2000), Statistical Modelling with Quantile Functions, Chapman & Hall

Karian, Z.A., Dudewicz, E.J., and McDonald, P. (1996), The extended generalized lambda distribution system for fitting distributions to data: history, completion of theory, tables, applications, the "Final Word" on Moment fits, Communications in Statistics - Simulation and Computation *25*, 611-642.

Karian, Zaven A. and Dudewicz, Edward J. (2000), Fitting statistical distributions: the Generalized Lambda Distribution and Generalized Bootstrap methods, Chapman & Hall

See Also

`fun.comp.moments.ml`, `fun.comp.moments.ml.2`, `fun.lm.theo.gld`

Examples

```
fun.theo.mv.gld(1, 2, 3, 4, "rs")
fun.theo.mv.gld(1, 2, 3, 4, "fmkl")
```

`fun.which.zero` *Determine which values are zero.*

Description

Returns an integer vector showing the position of zero values in the data.

Usage

```
fun.which.zero(data)
```

Arguments

`data`	A vector of data.

Value

An integer vector showing the position of zero values in the data.

Note

Any missing values will be returned as missing.

Author(s)

Steve Su

See Also

fun.zero.omit

Examples

```
# Finding where the zeros are in this vector: c(0,1,2,3,4,0,2)
fun.which.zero(c(0,1,2,3,4,0,2))
# Finding where the zeros are in this vector: c(0,1,2,3,NA,0,2)
fun.which.zero(c(0,1,2,3,NA,0,2))
```

---

fun.zero.omit *Returns a vector after removing all the zeros.*

Description

This function returns a vector after removing all the zeros.

Usage

```
fun.zero.omit(object)
```

Arguments

object	A vector of data.

Value

Returns a vector after removing zeros and also give information on the number of zeros in the data removed.

Note

Missing value and Inf values are not removed in this zero removing process.

Author(s)

Steve Su

See Also

fun.which.zero

Examples

```
# Removing zero entries from the vector c(0,1,2,3,4,0,2)
fun.zero.omit(c(0,1,2,3,4,0,2))
```

`gl.check.lambda.alt` *Checks whether the parameters provided constitute a valid generalised lambda distribution.*

Description

An alternative to the `gl.check.lambda` function in **gld** package.

Usage

```
gl.check.lambda.alt(l1, l2, l3, l4, param = "fmkl")
```

Arguments

`l1`	Lambda 1.
`l2`	Lambda 2.
`l3`	Lambda 3.
`l4`	Lambda 4.
`param`	`"rs"` or `"fmkl"` generalised lambda distribution.

Details

This version differs from `gl.check.lambda` in **gld** library.

Value

A logical value, `TRUE` or `FALSE`. `TRUE` indicates the parameters given is a valid probability distribution.

Author(s)

Steve Su

References

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), A study of the generalized tukey lambda family, Communications in Statistics - Theory and Methods *17*, 3547-3567.

Karian, Z.E., Dudewicz, E.J., and McDonald, P. (1996), The extended generalized lambda distribution system for fitting distributions to data: history, completion of theory, tables, applications, the "Final Word" on Moment fits, Communications in Statistics - Simulation and Computation *25*, 611-642.

Ramberg, J. S. & Schmeiser, B. W. (1974), An approximate method for generating asymmetric random variables, Communications of the ACM *17*, 78-82.

See Also

`gl.check.lambda.alt1`

Examples

```
gl.check.lambda.alt(0,1,.23,4.5,param="fmkl") ## TRUE
gl.check.lambda.alt(0,-1,.23,4.5,param="fmkl") ## FALSE
gl.check.lambda.alt(0,1,0.5,-0.5,param="rs") ## FALSE
```

`gl.check.lambda.alt1` *Checks whether the parameters provided constitute a valid generalised lambda distribution.*

Description

A replacement to the gl.check.lambda function in **gld** package.

Usage

```
gl.check.lambda.alt1(l1, l2 = NULL, l3 = NULL, l4 = NULL,
param = "fmkl", vect = FALSE)
```

Arguments

`l1`	Lambda 1.
`l2`	Lambda 2.
`l3`	Lambda 3.
`l4`	Lambda 4.
`param`	`"rs"` or `"fmkl"` generalised lambda distribution.
`vect`	A logical, set this to `TRUE` if the parameters are given in the vector form (it turns off checking of the format of 'lambdas' and the other lambda arguments

Details

This is a modified `gl.check.lambda` function in replace of **gld** library's `gl.check.lambda` function to allow for 5 parameters FMKL distributions and vector input of parameter values into this function.

Value

A logical value, `TRUE` or `FALSE`. `TRUE` indicates the parameters given is a valid probability distribution.

Author(s)

Steve Su

References

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), A study of the generalized tukey lambda family, Communications in Statistics - Theory and Methods *17*, 3547-3567.

Karian, Z.E., Dudewicz, E.J., and McDonald, P. (1996), The extended generalized lambda distribution system for fitting distributions to data: history, completion of theory, tables, applications, the "Final Word" on Moment fits, Communications in Statistics - Simulation and Computation *25*, 611-642.

Ramberg, J. S. & Schmeiser, B. W. (1974), An approximate method for generating asymmetric random variables, Communications of the ACM *17*, 78-82.

See Also

`gl.check.lambda.alt`

Examples

```
gl.check.lambda.alt1(c(0,1,.23,4.5),param="fmkl",vect=TRUE)
## TRUE, Using vector input of parameter values.
gl.check.lambda.alt1(0,-1,.23,4.5,param="fmkl") ## FALSE
gl.check.lambda.alt1(0,1,0.5,-0.5,param="rs") ## FALSE
```

`GLD functions` *The Generalised Lambda Distribution Family*

Description

Density, quantile density, distribution function, quantile function and random generation for the generalised lambda distribution (also known as the asymmetric lambda, or Tukey lambda). Works for both the `"fmkl"` and `"rs"` parameterisations. These functions originate from the **gld** library by Robert King and they are modified in this package to allow greater functionality and adaptability to new fitting methods. It does not give an error message for invalid distributions but will return NAs instead. To allow comparability with the pkg(gld) package, this package uses the same notation and description as those written by Robert King.

Usage

```
dgl(x, lambda1 = 0, lambda2 = NULL, lambda3 = NULL, lambda4 = NULL,
  param = "fmkl", inverse.eps = 1e-08,
  max.iterations = 500)
pgl(q, lambda1 = 0, lambda2 = NULL, lambda3 = NULL, lambda4 = NULL,
  param = "fmkl",  inverse.eps = 1e-08,
  max.iterations = 500)
qgl(p, lambda1, lambda2 = NULL, lambda3 = NULL, lambda4 = NULL,
  param = "fmkl")
rgl(n, lambda1=0, lambda2 = NULL, lambda3 = NULL, lambda4 = NULL,
  param = "fmkl")
```

Arguments

x	Vector of actual values for dgl
p	Vector of probabilities for qgl or qdgl
q	Vector of quantiles for pgl
n	Number of observations to be generated for rgl
lambda1	This can be either a single numeric value or a vector. If it is a vector, it must be of length 4 for parameterisations "fmkl" or "rs" and of length 5 for parameterisation "fm5" and the other 'lambda' arguments must be left as NULL. The numbering of the lambda parameters for the "fmkl" parameterisation is different to that used by Freimer, Mudholkar, Kollia and Lin (1988).
lambda2	Scale parameter
lambda3	First shape parameter
lambda4	Second shape parameter
param	"fmkl" or "fkml" uses Freimer, Kollia, Mudholkar, and Lin (1988) and it is the default setting. "rs" uses Ramberg and Schmeiser (1974)
inverse.eps	Accuracy of calculation for the numerical determination of F(x), defaults to 1e-8
max.iterations	Maximum number of iterations in the numerical determination of F(x), defaults to 500

Details

The generalised lambda distribution, also known as the asymmetric lambda, or Tukey lambda distribution, is a distribution with a wide range of shapes. The distribution is defined by its quantile function, the inverse of the distribution function. The 'gld' package implements three parameterisations of the distribution. The default parameterisation (the FMKL) is that due to Freimer Mudholkar, Kollia and Lin (1988) (see references below), with a quantile function:

$$F^{-1}(u) = \lambda_1 + \frac{\frac{u^{\lambda_3}-1}{\lambda_3} - \frac{(1-u)^{\lambda_4}-1}{\lambda_4}}{\lambda_2}$$

for $\lambda_2 > 0$.

A second parameterisation, the RS, chosen by setting param="rs" is that due to Ramberg and Schmeiser (1974), with the quantile function:

$$F^{-1}(u) = \lambda_1 + \frac{u^{\lambda_3} - (1-u)^{\lambda_4}}{\lambda_2}$$

This parameterisation has a complex series of rules determining which values of the parameters produce valid statistical distributions. See gl.check.lambda for details.

Value

dgl	gives the density (based on the quantile density and a numerical solution to $F^{-1}(u) = x$,
qdgl	gives the quantile density,

pgl	gives the distribution function (based on a numerical solution to $F^{-1}(u) = x$,
qgl	gives the quantile function, and
rgl	generates random observations.

References

Freimer, M., Kollia, G., Mudholkar, G. S. & Lin, C. T. (1988), A study of the generalized tukey lambda family, Communications in Statistics - Theory and Methods *17*, 3547-3567.

Gilchrist, Warren G. (2000), Statistical Modelling with Quantile Functions, Chapman & Hall

Karian, Z.A., Dudewicz, E.J., and McDonald, P. (1996), The extended generalized lambda distribution system for fitting distributions to data: history, completion of theory, tables, applications, the "Final Word" on Moment fits, Communications in Statistics - Simulation and Computation *25*, 611-642.

Karian, Zaven A. and Dudewicz, Edward J. (2000), Fitting statistical distributions: the Generalized Lambda Distribution and Generalized Bootstrap methods, Chapman & Hall

Ramberg, J. S. & Schmeiser, B. W. (1974), An approximate method for generating asymmetric random variables, Communications of the ACM *17*, 78-82.

Examples

```
qgl(seq(0,1,0.02),0,1,0.123,-4.3)
pgl(seq(-2,2,0.2),0,1,-.1,-.2,param="fmkl",inverse.eps=.Machine$double.eps)
```

---

histsu *Histogram with exact number of bins specified by the user*

Description

The generic function histsu computes a histogram of the given data values.

Usage

```
histsu(x, breaks = "Sturges", freq = NULL, probability = !freq,
include.lowest = TRUE, right = TRUE, density = NULL, angle = 45, col = NULL,
border = NULL, main = paste("Histogram of", xname), xlim = range(breaks),
ylim = NULL, xlab = xname, ylab, axes = TRUE, plot = TRUE, labels = FALSE,
nclass = NULL, ...)
```

Arguments

x	A vector of values for which the histogram is desired.
breaks	Either: 1) A vector giving the breakpoints between histogram cells, OR 2) A single number giving the number of cells for the histogram, OR 3) A character string naming an algorithm to compute the number of cells (see Details), OR 4) A function to compute the number of cells.

freq	logical; if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component 'density', are plotted (so that the histogram has a total area of one). Defaults to TRUE iff 'breaks' are equidistant (and 'probability' is not specified).
probability	A logical value, TRUE means it is not a frequency graph.
include.lowest	If TRUE, an x[i] equal to the 'breaks' value will be included in the first (or last, for right=FALSE) bar. This will be ignored (with a warning) unless 'breaks' is a vector.
right	If TRUE, the histograms cells are right-closed (left open) intervals.
density	The density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. Non-positive values of 'density' also inhibit the drawing of shading lines.
angle	The slope of shading lines, given as an angle in degrees (counter-clockwise).
col	A colour to be used to fill the bars. The default of NULL yields unfilled bars.
border	The color of the border around the bars. The default is to use the standard foreground color.
main	Title of the graph.
xlim	A two valued vector specifying the lower and upper limits of the x axis.
ylim	A two valued vector specifying the lower and upper limits of the y axis.
xlab	X axis labels.
ylab	Y axis labels.
axes	Logical value, if TRUE, axis will be drawn.
plot	Logical value, if TRUE, plot will be drawn.
labels	Logical or character. Additionally draw labels on top of bars, if not FALSE; see plot.histogram.
nclass	Number of bins of the histogram.
...	Other graphical parameters, see par for details.

Details

See hist help file. This function forces the number of class of histogram to that as specified by the user.

Value

An object of class "histogram" which is a list with components:

breaks	The n+1 cell boundaries (=breaks if that was a vector).
counts	N integers; for each cell, the number of x[] inside.
density	Values as estimated density values. If all(diff(breaks) == 1), they are the relative frequencies counts/n.
intensities	Same as density, deprecated.
mids	The n cell midpoints.
xname	A character string with the actual x argument name.
equidist	Logical, indicating if the distances between breaks are all the same.

Note

Please see hist help file.

Author(s)

R development team with modifications by Steve Su

References

Venables, W. N. and Ripley. B. D. (2002) Modern Applied Statistics with S. Springer.

See Also

hist

Examples

```
# See hist for extended example:
junk<-rgamma(1000,5)
# Forcing the number of bins to be 10:
histsu(junk,nclass=10)
```

---

is.inf *Returns a logical vecto, TRUE if the value is Inf or -Inf.*

Description

This function works in similar fashion as in is.na and is.notinf.

Usage

```
is.inf(x)
```

Arguments

x	A numerical value or a vector of data.

Value

A logical vector, T if the value is Inf or -Inf.

Note

In the presence of missing value, the function will return a missing value.

Author(s)

Steve Su

See Also

is.na,is.notinf

Examples

```
is.inf(c(Inf,2,2,1,-Inf))
```

`is.notinf` *Returns a logical vector TRUE, if the value is not Inf or -Inf.*

Description

This function works in similar fashion as in `is.na` and `is.inf`.

Usage

```
is.notinf(x)
```

Arguments

x	A numerical value or a vector of data.

Value

A logical vector, `T` if the value is not Inf or -Inf.

Note

In the presence of missing value, the function will return a missing value.

Author(s)

Steve Su

See Also

is.na,is.inf

Examples

```
is.notinf(c(Inf,2,2,1,-Inf))
```

ks.gof *Kolmogorov-Smirnov test*

Description

Performs one or two sample Kolmogorov-Smirnov tests.

Usage

```
ks.gof(x, y, ..., alternative = c("two.sided", "less", "greater"),
exact = NULL)
```

Arguments

x	A numeric vector of data values.
y	Either a numeric vector of data values, or a character string naming a distribution function.
...	Parameters of the distribution specified (as a character string) by 'y'.
alternative	Indicates the alternative hypothesis and must be one of "two.sided" (default),"less", or "greater". You can specify just the initial letter.
exact	NULL or a logical indicating whether an exact p-value should be computed. See Details for the meaning of NULL. Not used for the one-sided two-sample case.

Details

If y is numeric, a two-sample test of the null hypothesis that x and y were drawn from the same continuous distribution is performed.

Alternatively, y can be a character string naming a continuous distribution function. In this case, a one-sample test is carried out of the null that the distribution function which generated x is distribution y with parameters specified by "...".

The possible values "two.sided" (default),"less" and "greater" of alternative specify the null hypothesis that the true distribution function of x is equal to, not less than or not greater than the hypothesized distribution function (one-sample case) or the distribution function of y (two-sample case), respectively.

Exact p-values are not available for the one-sided two-sample case, or in the case of ties. exact = NULL (the default), an exact p-value is computed if the sample size if less than 100 in the one-sample case, and if the product of the sample sizes is less than 10000 in the two-sample case. Otherwise, asymptotic distributions are used whose approximations may be inaccurate in small samples. In the one-sample two-sided case, exact p-values are obtained as described in Marsaglia, Tsang & Wang (2003). The formula of Birnbaum & Tingey (1951) is used for the one-sample one-sided case.

If a single-sample test is used, the parameters specified in "..." must be pre-specified and not estimated from the data. There is some more refined distribution theory for the KS test with estimated parameters (see Durbin, 1973), but that is not implemented in ks.gof.

Value

A list with class "htest" containing the following components:

statistic	Value of test statistics.
p.value	P-value.
alternative	Character string describing the alternative hypothesis.
method	Character string indicating what type of test was performed.
data.name	Character string giving the name(s) of the data.

Note

This function handle ties by jittering, adding a very small uniform random number generated from the minimal value of the data set divided by 1e+08 to minimal value divided by 1e+07.

Author(s)

R

References

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

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

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

George Marsaglia, Wai Wan Tsang & Jingbo Wang (2003), Evaluating Kolmogorov's distribution. Journal of Statistical Software, *8*/18. <URL: http://www.jstatsoft.org/v08/i18/>.

See Also

ks.test

Examples

```
x <- rnorm(50)
y <- runif(30)
# Do x and y come from the same distribution?
ks.gof(x, y)
# Does x come from a shifted gamma distribution with shape 3 and rate 2?
ks.gof(x+2, "pgamma", 3, 2) # two-sided, exact
ks.gof(x+2, "pgamma", 3, 2, exact = FALSE)
ks.gof(x+2, "pgamma", 3, 2, alternative = "gr")
```

Lmoments *L-moments*

Description

Calculates sample L-moments, L-coefficients and covariance matrix of L-moments.

Usage

```
Lmoments(data,rmax=4,na.rm=FALSE,returnobject=FALSE,trim=c(0,0))
Lcoefs(data,rmax=4,na.rm=FALSE,trim=c(0,0))
Lmomcov(data,rmax=4,na.rm=FALSE)
Lmoments_calc(data,rmax=4)
Lmomcov_calc(data,rmax=4)
```

Arguments

data	matrix or data frame.
rmax	maximum order of L-moments.
na.rm	a logical value indicating whether 'NA' values should be removed before the computation proceeds.
returnobject	a logical value indicating whether a list object should be returned instead of an array of L-moments.
trim	c(0,0) for ordinary L-moments and c(1,1) for trimmed (t=1) L-moments

Value

Lmoments	returns an array of L-moments containing a row for each variable in data, or if returnobject=TRUE, a list containing the following:
lambdas	an array of L-moments
ratios	an array of mean, L-scale and L-moment ratios
trim	the value of the parameter 'trim'
source	a string with value "Lmoments" or "t1lmoments"
Lcoefs	returns an array of L-coefficients (mean, L-scale, L-skewness, L-kurtosis, ...)
Lmomcov	returns the covariance matrix of L-moments or a list of covariance matrices if the input has multiple columns.
Lmoments_calc	is internal function.
Lmomcov_calc	is internal function.

Note

Functions Lmoments and Lcoefs calculate trimmed L-moments if you specify trim=c(1,1).

Author(s)

Juha Karvanen <<juha.karvanen@ktl.fi>>

References

Karvanen, J. and A. Nuutinen (2008). "Characterizing the generalized lambda distribution by L-moments." Computational Statistics and Data Analysis 52(4): 1971-1983.

Asquith, W. (2007). "L-moments and TL-moments of the generalized lambda distribution." Computational Statistics and Data Analysis 51(9): 4484-4496.

Elamir, E. A., Seheult, A. H. 2004. "Exact variance structure of sample L-moments" Journal of Statistical Planning and Inference 124 (2) 337-359.

Hosking, J. 1990. "L-moments: Analysis and estimation distributions using linear combinations of order statistics", Journal of Royal Statistical Society B 52, 105-124.

See Also

`t1lmoments` for trimmed L-moments

Examples

```
x<-rnorm(500)
Lmoments(x)
```

---

pretty.su *An alternative to the normal pretty function in R.*

Description

Divide a range of values into equally spaced divisions. End points are given as output.

Usage

```
pretty.su(x, nint = 5)
```

Arguments

`x`	A vector of values.
`nint`	Number of intervals required.

Details

This is also used for the plotting of histogram in the `histsu` function.

Value

A vector of endpoints dividing the data into equally spaced regions.

Author(s)

Steve Su

See Also

pretty

Examples

```
# Generate random numbers from normal distribution:
junk<-rnorm(1000,2,3)

# Cut them into 7 regions, 8 endpoints.
pretty.su(junk,7)
```

---

qqplot.gld *Do a quantile plot on the univariate distribution fits.*

Description

This plots the theoretical and actual data quantiles to allow the user to examine the adequacy of a single gld distribution fit.

Usage

```
qqplot.gld(data, fit, param, len = 10000, name = "",
corner = "topleft",type="",range=c(0,1),xlab="",main="")
```

Arguments

data	Data fitted.
fit	Parameters of distribution fit.
param	Can be either "rs" or "fmkl".
len	Precision of the quantile calculatons. Default is 10000. This means 10000 points are taken from 0 to 1.
name	Name of the data set, added to the title of plot if main is missing.
corner	Can be "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center" as in legend.
type	This can be "" or "str.qqplot", the first produces the raw quantiles and the second plot them on a straight line. Default is "".
range	This is the range for which the quantiles are to be plotted. Default is c(0,1).
xlab	x axis label, if left blank, then default is "Data".
main	Title of the plot, if left blank, a default title will be added.

Value

A plot is given.

Author(s)

Steve Su

See Also

qqplot.gld.bi

Examples

```
 set.seed(1000)

 junk<-rweibull(300,3,2)

# Fit the function using fun.data.fit.ml
 obj.fit1.ml<-fun.data.fit.ml(junk)

# Do a quantile plot on the raw quantiles
 qqplot.gld(junk,obj.fit1.ml[,1],param="rs",name="RS ML fit")

# Or a qq plot to examine deviation from straight line
 qqplot.gld(junk,obj.fit1.ml[,1],param="rs",name="RS ML fit",type="str.qqplot")
```

---

qqplot.gld.bi *Do a quantile plot on the bimodal distribution fits.*

Description

This plots the theoretical and actual data quantiles to allow the user to examine the adequacy of two gld distributions mixture fit.

Usage

```
qqplot.gld.bi(data, fit, param1, param2, len = 10000, name = "",
corner = "topleft",type="",range=c(0,1),xlab="",main="")
```

Arguments

data	Data fitted.
fit	Parameters of distribution fit.
param1	Can be either "rs" or "fmkl".
param2	Can be either "rs" or "fmkl".
len	Precision of the quantile calculatons. Default is 10000. This means 10000 points are taken from 0 to 1.

name	Name of the data set, added to the title of plot if main is missing.
corner	Can be "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center" as in legend.
type	This can be "" or "str.qqplot", the first produces the raw quantiles and the second plot them on a straight line. Default is "".
range	This is the range for which the quantiles are to be plotted. Default is c(0,1).
xlab	x axis label, if left blank, then default is "Data"
main	Title of the plot, if left blank, a default title will be added.

Value

A plot is given.

Author(s)

Steve Su

See Also

qqplot.gld

Examples

```
 set.seed(1000)

 junk<-rweibull(300,3,2)

## Fitting mixture of generalised lambda distributions on the data set using
## both the maximum likelihood and partition maximum likelihood and plot the
## resulting fits
 junk<-fun.auto.bimodal.ml(faithful[,1],per.of.mix=0.1,clustering.m=clara,
 init1.sel="rprs",init2.sel="rmfmkl",init1=c(-1.5,1.5),init2=c(-0.25,1.5),
 leap1=3,leap2=3)
 fun.plot.fit.bm(nclass=50,fit.obj=junk,data=faithful[,1],
 name="Maximum likelihood using",xlab="faithful1",param.vec=c("rs","fmkl"))

## Do a quantile plot on the raw quantiles
 qqplot.gld.bi(faithful[,1],junk$par,param1="rs",param2="fmkl",
 name="RS FMKL ML fit")

## Or a qq plot to examine deviation from straight line
 qqplot.gld.bi(faithful[,1],junk$par,param1="rs",param2="fmkl",
 name="RS FMKL ML fit",type="str.qqplot")
```

QUnif *Quasi Randum Numbers via Halton Sequences*

Description

These functions provide quasi random numbers or *space filling* or *low discrepancy* sequences in the p-dimensional unit cube.

Usage

```
sHalton(n.max, n.min = 1, base = 2, leap = 1)
 QUnif (n, min = 0, max = 1, n.min = 1, p, leap = 1)
```

Arguments

n.max	maximal (sequence) number.
n.min	minimal sequence number.
n	number of p-dimensional points generated in QUnif. By default, n.min = 1, leap = 1 and the maximal sequence number is n.max = n.min + (n-1)*leap.
base	integer ≥ 2: The base with respect to which the Halton sequence is built.
min, max	lower and upper limits of the univariate intervals. Must be of length 1 or p.
p	dimensionality of space (the unit cube) in which points are generated.
leap	integer indicating (if > 1) if the series should be leaped, i.e., only every leapth entry should be taken.

Value

sHalton(n,m) returns a numeric vector of length n-m+1 of values in $[0, 1]$.

QUnif(n, min, max, n.min, p=p) generates n-n.min+1 p-dimensional points in $[min, max]^p$ returning a numeric matrix with p columns.

Note

For leap Kocis and Whiten recommend values of $L = 31, 61, 149, 409$, and particularly the $L = 409$ for dimensions up to 400.

Author(s)

Martin Maechler

References

James Gentle (1998) *Random Number Generation and Monte Carlo Simulation*; sec.\ 6.3. Springer.

Kocis, L. and Whiten, W.J. (1997) Computationsl Investigations of Low-Discrepancy Sequences. *ACM Transactions of Mathematical Software* **23**, 2, 266–294.

Examples

```
32*sHalton(20, base=2)
QUnif(n=10,p=2,leap=409)
```

---

skewness and kurtosis *Compute skewness and kurtosis statistics*

Description

This uses the S+ version directly.

Usage

```
skewness(x, na.rm = FALSE, method = "fisher")
kurtosis(x, na.rm = FALSE, method = "fisher")
```

Arguments

x	Any numerical object. Missing values NA are allowed.
na.rm	Logical flag: if na.rm=TRUE, missing values are removed from x before doing the computations. If na.rm=FALSE and x contains missing values, then the return value is NA.
method	Character string specifying the computation method. The two possible values are fisher for Fisher's g1 (skewness) and g2 (kurtosis) versions, and moment for the functional forms of the statistics. Only the first character of the string needs to be supplied.

Details

The moment forms are based on the definitions of skewness and kurtosis for distributions; these forms should be used when resampling (bootstrap or jackknife). The "fisher" forms correspond to the usual "unbiased" definition of sample variance, though in the case of skewness and kurtosis exact unbiasedness is not possible.

Value

A single value of skewness or kurtotis.

If y = x - mean(x), then the "moment" method computes the skewness value as mean(y~3)/mean(y~2) ~1.5 and the kurtosis value as mean(y~4)/mean(y ~2)~2 - 3. To see the "fisher" calculations, print out the functions.

Author(s)

Splus

See Also

var

Examples

```
x <- runif(30)
skewness(x)
skewness(x, method="moment")
kurtosis(x)
kurtosis(x, method="moment")
```

---

starship — *Carry out the "starship" estimation method for the generalised lambda distribution*

Description

Calculates estimates for the FMKL parameterisation of the generalised lambda distribution on the basis of data, using the starship method. The starship method is built on the fact that the generalised lambda distribution is a transformation of the uniform distribution. This method finds the parameters that transform the data closest to the uniform distribution. This function uses a grid-based search to find a suitable starting point (using starship.adaptivegrid) then uses optim to find the parameters that do this.

Usage

```
starship(data, optim.method = "Nelder-Mead", initgrid = NULL, param="FMKL",
optim.control=NULL)
```

Arguments

data	Data to be fitted, as a vector
optim.method	Optimisation method for optim to use, defaults to Nelder-Mead
initgrid	Grid of values of λ_3 and λ_4 to try, in starship.adaptivegrid. This should be a list with elements, lcvect, a vector of values for λ_3, ldvect, a vector of values for λ_4 and levect, a vector of values for λ_5 (levect is only required if param is fm5). If it is left as NULL, the default grid depends on the parameterisation. For fmkl, both lcvect and ldvect default to: -1.5 -1 -.5 -.1 0 .1 .2 .4 .8 1 1.5 (levect is NULL). For rs, both lcvect and ldvect default to: .1 .2 .4 .8 1 1.5

(levect is NULL).

For fm5, both lcvect and ldvect default to:

-1.5 -1 -.5 -.1 0 .1 .2 .4 .8 1 1.5

and levect defaults to:

-0.5 0.25 0 0.25 0.5

param	choose parameterisation: fmkl uses *Freimer, Mudholkar, Kollia and Lin (1988)* (default). rs uses *Ramberg and Schmeiser (1974)* fm5 uses the 5 parameter version of the FMKL parameterisation (paper to appear)
optim.control	List of options for the optimisation step. See optim for details. If left as NULL, the parscale control is set to scale λ_1 and λ_2 by the absolute value of their starting points.

Details

The starship method is described in King and MacGillivray, 1999 (see references). It is built on the fact that the generalised lambda distribution is a transformation of the uniform distribution. Thus the inverse of this transformation is the distribution function for the gld. The starship method applies different values of the parameters of the distribution to the distribution function, calculates the depths q corresponding to the data and chooses the parameters that make the depths closest to a uniform distribution.

The closeness to the uniform is assessed by calculating the Anderson-Darling goodness-of-fit test on the transformed data against the uniform, for a sample of size length(data).

This is implemented in 2 stages in this function. First a grid search is carried out, over a small number of possible parameter values (see starship.adaptivegrid for details). Then the minimum from this search is given as a starting point for an optimisation of the Anderson-Darling value using optim, with method given by optim.method

See references for details on parameterisations.

Value

Returns a list, with

lambda	A vector of length 4, giving the estimated parameters, in order, λ_1 - location parameter λ_2 - scale parameter λ_3 - first shape parameter λ_4 - second shape parameter
grid.results	output from the grid search - see starship.adaptivegrid for details
optim	output from the optim search - optim for details

Author(s)

Robert King, Darren Wraith

References

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), *A study of the generalized tukey lambda family*, Communications in Statistics - Theory and Methods **17**, 3547–3567.

Ramberg, J. S. & Schmeiser, B. W. (1974), *An approximate method for generating asymmetric random variables*, Communications of the ACM **17**, 78–82.

King, R.A.R. & MacGillivray, H. L. (1999), *A starship method for fitting the generalised λ distributions*, Australian and New Zealand Journal of Statistics **41**, 353–374

Owen, D. B. (1988), *The starship*, Communications in Statistics - Computation and Simulation **17**, 315–323.

See Also

`starship.adaptivegrid`, `starship.obj`

Examples

```
data <- rgl(100,0,1,.2,.2)
starship(data,optim.method="Nelder-Mead",initgrid=list(lcvect=(0:4)/10,
ldvect=(0:4)/10))
```

---

starship.adaptivegrid *Carry out the "starship" estimation method for the generalised lambda distribution using a grid-based search*

Description

Calculates estimates for the generalised lambda distribution on the basis of data, using the starship method. The starship method is built on the fact that the generalised lambda distribution is a transformation of the uniform distribution. This method finds the parameters that transform the data closest to the uniform distribution. This function uses a grid-based search.

Usage

```
starship.adaptivegrid(data, initgrid=list(
lcvect = c(-1.5, -1, -0.5, -0.1, 0, 0.1, 0.2, 0.4, 0.8, 1, 1.5),
ldvect = c(-1.5, -1, -0.5, -0.1, 0, 0.1, 0.2, 0.4, 0.8, 1, 1.5),
levect = c(-0.5,-0.25,0,0.25,0.5)),param="FMKL")
```

Arguments

`data`	Data to be fitted, as a vector
`initgrid`	A list with elements, `lcvect`, a vector of values for λ_3, `ldvect`, a vector of values for λ_4 and `levect`, a vector of values for λ_5 (`levect` is only required if `param` is `fm5`). *Note:* if `param=rs`, the non-positive values are dropped from `lcvect` and `ldvect`.
`param`	choose parameterisation: `fmkl` uses *Freimer, Mudholkar, Kollia and Lin (1988)* (default). `rs` uses *Ramberg and Schmeiser (1974)* `fm5` uses the 5 parameter version of the FMKL parameterisation (paper to appear)

Details

The starship method is described in King and MacGillivray, 1999 (see references). It is built on the fact that the generalised lambda distribution is a transformation of the uniform distribution. Thus the inverse of this transformation is the distribution function for the gld. The starship method applies different values of the parameters of the distribution to the distribution function, calculates the depths q corresponding to the data and chooses the parameters that make the depths closest to a uniform distribution.

The closeness to the uniform is assessed by calculating the Anderson-Darling goodness-of-fit test on the transformed data against the uniform, for a sample of size `length(data)`.

This function carries out a grid-based search. This was the original method of King and MacGillivray, 1999, but you are advised to instead use `starship` which uses a grid-based search together with an optimisation based search.

See references for details on parameterisations.

Value

`response`	The minimum "response value" — the result of the internal goodness-of-fit measure. This is the return value of starship.obj. See King and MacGillivray, 1999 for more details
`lambda`	A vector of length 4 giving the values of λ_1 to λ_4 that produce this minimum response, i.e. the estimates

Author(s)

Robert King, Darren Wraith

References

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), *A study of the generalized tukey lambda family*, Communications in Statistics - Theory and Methods **17**, 3547–3567.

Ramberg, J. S. & Schmeiser, B. W. (1974), *An approximate method for generating asymmetric random variables*, Communications of the ACM **17**, 78–82.

King, R.A.R. & MacGillivray, H. L. (1999), *A starship method for fitting the generalised λ distributions*, Australian and New Zealand Journal of Statistics **41**, 353–374

Owen, D. B. (1988), *The starship*, Communications in Statistics - Computation and Simulation **17**, 315–323.

See Also

`starship`, `starship.obj`

Examples

```
data <- rgl(100,0,1,.2,.2)
starship.adaptivegrid(data,list(lcvect=(0:4)/10,ldvect=(0:4)/10))
```

`starship.obj` *Objective function that is minimised in starship estimation method*

Description

The starship is a method for fitting the generalised lambda distribution. See `starship` for more details.

This function is the objective funciton minimised in the methods. It is a goodness of fit measure carried out on the depths of the data.

Usage

```
starship.obj(par, data, param = "fmkl")
```

Arguments

`par`	parameters of the generalised lambda distribution, a vector of length 4, giving λ_1 to λ_4. See references or `qgl` for details on the definitions of these parameters
`data`	Data — a vector
`param`	choose parameterisation: `fmkl` uses *Freimer, Mudholkar, Kollia and Lin (1988)* (default). `rs` uses *Ramberg and Schmeiser (1974)*

Details

The starship method is described in King and MacGillivray, 1999 (see references). It is built on the fact that the generalised lambda distribution is a transformation of the uniform distribution. Thus the inverse of this transformation is the distribution function for the gld. The starship method applies different values of the parameters of the distribution to the distribution function, calculates the depths q corresponding to the data and chooses the parameters that make the depths closest to a uniform distribution.

The closeness to the uniform is assessed by calculating the Anderson-Darling goodness-of-fit test on the transformed data against the uniform, for a sample of size `length(data)`.

This function returns that objective function. It is provided as a seperate function to allow users to carry out minimisations using `optim` or other methods. The recommended method is to use the `link{starship}` function.

Value

The Anderson-Darling goodness of fit measure, computed on the transformed data, compared to a uniform distribution. *Note that this is NOT the goodness-of-fit measure of the generalised lambda distribution with the given parameter values to the data.*

Author(s)

Robert King, Darren Wraith

References

Freimer, M., Mudholkar, G. S., Kollia, G. & Lin, C. T. (1988), *A study of the generalized tukey lambda family*, Communications in Statistics - Theory and Methods **17**, 3547–3567.

Ramberg, J. S. & Schmeiser, B. W. (1974), *An approximate method for generating asymmetric random variables*, Communications of the ACM **17**, 78–82.

King, R.A.R. & MacGillivray, H. L. (1999), *A starship method for fitting the generalised λ distributions*, Australian and New Zealand Journal of Statistics **41**, 353–374

Owen, D. B. (1988), *The starship*, Communications in Statistics - Computation and Simulation **17**, 315–323.

See Also

`starship` `starship.adaptivegrid`,

Examples

```
data <- rgl(100,0,1,.2,.2)
starship.obj(c(0,1,.2,.2),data,"fmkl")
```

`t1lmoments` *Trimmed L-moments*

Description

Calculates sample trimmed L-moments with trimming parameter 1.

Usage

```
t1lmoments(data,rmax=4)
```

Arguments

`data`	matrix or data frame.
`rmax`	maximum order of trimmed L-moments.

Value

array of trimmed L-moments (trimming parameter = 1) up to order 4 containing a row for each variable in data.

Note

Functions `link{Lmoments}` and `link{Lcoefs}` calculate trimmed L-moments if you specify `trim=c(1,1)`.

Author(s)

Juha Karvanen <juha.karvanen@ktl.fi>

References

Karvanen, J. and A. Nuutinen (2008). "Characterizing the generalized lambda distribution by L-moments." Computational Statistics and Data Analysis 52(4): 1971-1983.

Asquith, W. (2007). "L-moments and TL-moments of the generalized lambda distribution." Computational Statistics and Data Analysis 51(9): 4484-4496.

Elamir, E. A., Seheult, A. H. 2004. "Exact variance structure of sample L-moments" Journal of Statistical Planning and Inference 124 (2) 337-359.

Hosking, J. 1990. "L-moments: Analysis and estimation distributions using linear combinations of order statistics", Journal of Royal Statistical Society B 52, 105-124.

See Also

`Lmoments` for L-moments

Examples

```
x<-rnorm(500)
t1lmoments(x)
```

`which.na` *Determine Missing Values*

Description

Returns a vector showing the position of missing values in a vector.

Usage

```
which.na(x)
```

Arguments

`x`	An object which should be of `logical`, `numeric`, or `complex`

Value

This returns the indices of values in `x` which are missing or "Not a Number".

Examples

```
# A non-zero number divided by zero creates infinity,
# zero over zero creates a NaN
weird.values <- c(1/0, -2.9/0, 0/0, NA)
which.na(weird.values)
# Produces:
#  [1] 3 4
```

Index