

Package ‘LambertW’

February 14, 2012

Type Package

Title Analyze and Gaussianize skewed, heavy-tailed data

Version 0.2.9.9

Date 2011-06-25

Author Georg M. Goerg <gmg@stat.cmu.edu>

Maintainer Georg M. Goerg <gmg@stat.cmu.edu>

URL <http://www.stat.cmu.edu/~gmg> <http://arxiv.org/abs/0912.4554>
<http://arxiv.org/abs/1010.2265>

Description The Lambert W framework is a new generalized way to analyze skewed, heavy-tailed data. Lambert W random variables (RV) are based on an input/output framework where the input is a RV X with distribution $F(x)$, and the output $Y = \text{func}(X)$ has similar properties as X (but slightly skewed or heavy-tailed). Then this transformed RV Y has a Lambert W x F distribution - for details see References. This package contains functions to perform a Lambert W analysis of skewed and heavy-tailed data: data can be simulated, parameters can be estimated from real world data, quantiles can be computed, and results plotted/printed in a ‘nice’ way. Probably the most important function is ‘Gaussianize’, which works the same way as the R function ‘scale’ but actually makes your data Gaussian. An optional modular toolkit implementation allows users to define their own Lambert W x ‘my favorite distribution’ and use it for their analysis.

Depends moments, gsl, MASS, nortest, maxLik

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2012-01-08 01:07:40

R topics documented:

LambertW-package	3
AA	5
beta-methods	6
create_LambertW_input	7
create_LambertW_output	9
delta_01	12
delta_GMM	13
delta_Taylor	14
G	15
gamma_01	16
gamma_GMM	17
gamma_Taylor	18
Gaussianize	19
get.input	20
G_delta_alpha	21
H	22
H_gamma	23
IGMM	24
ks.test.t	26
LambertW-methods	27
LambertW_fit-methods	31
LambertW_input-methods	33
LambertW_output-methods	33
loglik-utils	34
mc	36
MLE_LambertW	38
normfit	39
p_1	41
skewness_test	42
SolarFlares	43
support	44
theta-utils	45
U-methods	47
vec.norm	49
W	50
W_delta_alpha	51
W_gamma	52

Description

The Lambert W framework is a new generalized way to analyze skewed, heavy-tailed data. Lambert W random variables (RV) are based on an input/output framework where the input is a RV X with distribution $F_X(x | \beta)$, and the output $Y = \text{func}(X)$ has similar properties as X (but slightly skewed or heavy-tailed). Then Y has a 'Lambert W $\times F_X(\cdot | \beta)$ ' distribution - for details see the References.

Implemented distributions (`distname = ...`) are: "cauchy", "chisq", "exp", "normal", "t", "unif". To be extended.

If your current empirical work requires another distribution, then a modular toolkit implementation (`create_LambertW_input` and `create_LambertW_output`) allows you to define your own Lambert W \times 'my favorite distribution' and use it in the analysis of real world data.

This package contains many functions to perform an adequate analysis of skewed and heavy-tailed data: data can be simulated (`rLambertW`), parameters can be estimated (`IGMM` and `MLE_LambertW`), quantiles can be computed (`qLambertW`), and results presented/plotted in a 'nice' way (`plot.LambertW_fit` and `print.LambertW_fit`).

Probably the most important function is `Gaussianize`, which works the same way as the R function `scale` but actually makes your data Gaussian.

This package is based on the notation and definitions of the papers listed in the References (Goerg (2011a) and Goerg (2011b)); until their final publication these functions, definitions, and names may be subject to change. I will not include these 2 in the descriptions of each single function, unless they are explicitly referenced to in the function's description.

If you use this package in your work please cite it. If you want, you can send me your 'Lambert W \times my favorite distribution' (with a corresponding paper reference which will be included.) implementation and I will make it part of the standard LambertW package.

Comments, suggestions, code improvements/accelerations, implementation of new input distributions, bug reports, etc. are more than welcome: feel free to contact me.

Author(s)

Author and maintainer: Georg M. Goerg <gmg@stat.cmu.edu>

References

Goerg, G.M. (2011a). "Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation". Forthcoming in the Annals of Applied Statistics (<http://arxiv.org/abs/0912.4554>).

Goerg, G.M. (2011b). "The Lambert Way to Gaussianize skewed, heavy-tailed data with the inverse of Tukey's h transformation as a special case.". In preparation for submission (<http://arxiv.org/abs/1010.2265>).

Examples

```

## Replicate parts of the statistical analysis in the References (2011a)
data(AA)
attach(AA)
X=AA[AA$sex=="f",]
y=X$bmi

op=par(no.readonly=TRUE)
normfit(y)

fit.gmm=IGMM(y, type="s")
summary(fit.gmm) # gamma is significant and positive
plot(fit.gmm)
# Comparison of estimated theoretical and sample moments
TAB = rbind(rbind(mLambertW(beta = fit.gmm$tau[1:2], gamma = fit.gmm$tau["gamma"], distname="normal")), cbind(mean(y),
rownames(TAB) = c("Theoretical (IGMM)", "Sample")
TAB

x=get.input(y, fit.gmm$tau)
normfit(x) # input is normal -> fit a Lambert W x Gaussian by maximum likelihood

fit.ml=MLE_LambertW(y, type="s", distname="normal")
summary(fit.ml)
plot(fit.ml)

## a finance example

M = ts(EuStockMarkets[201:1802,])
R = ts(diff(log(M))*100)

plot(R)

skewness(R[,"SMI"]) ## negative skewness
kurtosis(R[,"SMI"]) ## very high excess kurtosis

normfit(R[,"SMI"], vol = TRUE)

fit.SMI=IGMM(R[,"SMI"], type="s")
summary(fit.SMI) ## negative but small gamma

x = get.input(R[,"SMI"], fit.SMI$tau)
normfit(x, vol = TRUE) # symmetric, but still not normal -> try a skewed Lambert W x t distribution

## MLE for skewed Lambert W x t
fitMLt.SMI = MLE_LambertW(R[,"SMI"], distname = "t", type="s")
summary(fitMLt.SMI)
plot(fitMLt.SMI)

x = get.input(R[,"SMI"], fitMLt.SMI$tau)
plot(ts(cbind(R[,"SMI"], x)), main="")

## MLE for heavy-tail Lambert W x Gaussian (= Tukey's h)

```

```

fitMLh.SMI = MLE_LambertW(R[, "SMI"], distname = "normal", type="h")
summary(fitMLh.SMI)
plot(fitMLh.SMI)

## MLE for skewed heavy-tail Lambert W x Gaussian (= Tukey's hh)
fitMLhh.SMI = MLE_LambertW(R[, "SMI"], distname = "normal", type="hh")
summary(fitMLhh.SMI)
plot(fitMLhh.SMI)
x = get.input(R[, "SMI"], tau = fitMLhh.SMI$tau)
normfit(x) # Gaussianized the returns

```

AA

Australian athletes data set

Description

These data were collected in a study of how data on various characteristics of the blood varied with sport body size and sex of the athlete. Identical to the `ais` dataset in the DAAG package.

Usage

```
data(AA)
```

Format

A data frame with 202 observations on the following 13 variables.

rcc red blood cell count, in $10^{12} l^{-1}$

wcc white blood cell count, in 10^{12} per liter

hc hematocrit, percent

hg hemoglobin concentration, in g per decaliter

ferr plasma ferritins, $ng dl^{-1}$

bmi Body mass index, $kg cm^{-2} 10^2$

ssf sum of skin folds

pcBfat percent Body fat

lbm lean body mass, kg

ht height, cm

wt weight, kg

sex a factor with levels f m

sport a factor with levels B_Ball Field Gym Netball Row Swim T_400m T_Sprnt Tennis W_Polo

Details

Do blood hemoglobin concentrations of athletes in endurance-related events differ from those in power-related events?

Source

These data were the basis for the analyses that are reported in Telford and Cunningham (1991).

References

Telford, R.D. and Cunningham, R.B. 1991. Sex, sport and body-size dependency of hematology in highly trained athletes. *Medicine and Science in Sports and Exercise* 23: 788-794.

Examples

```
# See first example in 'LambertW-package'
```

beta-methods

Functions for parameter vector beta of the input distribution

Description

The parameter vector β describes the input distribution $F_X(x | \beta)$. Depending on the distribution β has different length and names: for example, for a "normal" distribution beta is a vector of length 2 describing "mu" and "sigma"; for an exponential distribution beta is a vector of length 1 (i.e. a number) describing the rate "lambda".

However, any β implies different mean and variance of X . Thus here are two functions that - depending on the input distribution - assign β its typical names given the distribution, and converts the parameter values to the implied τ .

Usage

```
beta2tau(beta, distname = c("normal"), gamma = 0, delta = 0, alpha = 1)
beta_names(distname = c("normal"))
```

Arguments

distname	name of the distribution $F_X(x \beta)$ (follows mostly the suffix of the R functions; exception "normal" instead of "norm"); default: "normal"
beta	parameter vector β of the input distribution; specifications as they are for the R implementation of this distribution. For example, if distname = "exp", then beta = 2 means that the rate of the exponential distribution equals 2; if distname = "normal" then beta = c(1, 2) means that the mean and standard deviation of the input Gaussian distribution are 1 and 2, respectively.
delta	heavy-tail parameter (= Tukey's h in case of a normal distribution); default: 0.
gamma	skewness parameter; default: 0.
alpha	exponent; default: 1.

Value

The function `beta2tau` returns the τ vector implied by `beta` and `distname`.

The function `beta_names` returns a list of names for each component of `beta`.

Author(s)

Georg M. Goerg

Examples

```
beta2tau(c(1,1), distname="normal") # the same as the input, as delta = gamma = 0 and alpha = 1
beta2tau(c(1,4,1), distname="t") # problem (see message)
beta2tau(c(1,4,3), distname="t") # no problem
beta_names("normal")
beta_names("exp")
```

`create_LambertW_input` *Modular toolkit for generation of input distribution object*

Description

(IMPORTANT: This functionality of the package is still in development; function names, functionality, etc. may be subject to change. If speed is your main concern, then use the individual function necessary to carry out your specific task; if convenience and functionality are important, then use this function and then [create_LambertW_output](#).)

This function creates an object of class `LambertW_input` for standard distribution (see [LambertW-package](#) for a list of supported distributions) as well as user-defined distributions. This will then be passed on to [create_LambertW_output](#) to create the Lambert W version of the input.

See Details and Values below for instructions.

Usage

```
create_LambertW_input(distname = NULL, beta = NULL, beta2tau = NULL, dU = NULL,
  pU = NULL, rU = NULL, qU = NULL)
```

Arguments

<code>distname</code>	name of the input distribution $F_X(x \beta)$; default: <code>NULL</code> .
<code>beta</code>	parameter vector β of the input distribution $F_X(x \beta)$
<code>beta2tau</code>	function to convert β to τ for distribution <code>distname</code> .
<code>dU</code>	function: probability density function (pdf) of the (zero-mean and) unit variance U version of $X \sim F_X(x \beta)$ (see References for details).
<code>pU</code>	function: cumulative distribution function (cdf) of the (zero-mean and) unit variance U
<code>rU</code>	function: random number generator for the (zero-mean and) unit variance U
<code>qU</code>	function: quantile function for the (zero-mean and) unit variance U

Details

In a first step one generates a LambertW_input object representing the input RV $X \sim F_X(x | \beta)$. In the second step it will be transformed to a Lambert W \times F distribution, where F can be any well-defined distribution - see [create_LambertW_output](#). Standard distributions such as "normal", "t" or "exp"onential are already implemented, but users can define any LambertW_input with their favorite distribution and then use [create_LambertW_output](#) to create a Lambert W \times 'favorite distribution' RV with the standard R functionality such as random number generation, pdf/cdf, quantiles, MLE, etc.\ for this newly generated Lambert W \times 'favorite distribution'.

This modular toolkit implementation intends to promote a wide-spread usage of Lambert W \times F distributions in practice.

If "distname" is not one of the already implemented (see [LambertW-package](#) for a list of supported distributions), then create_LambertW_input uses dU, rU, etc.\ supplied by the user. For the correct format of these, type create_LambertW_input in the console and see the structure/format for the already implemented distributions; it should be self-explanatory.

Value

A list of class LambertW_input; the values of an LambertW_input object are (for the most part) functions themselves (see Examples):

distname	name of the distribution $F_X(x \beta)$
beta	parameter vector β of the input distribution
beta2tau	function to convert β to the τ
tau	tau vector
dU	pdf of U
pU	cdf of U
qU	quantile function for U
rU	random number generator for U
dX	pdf of $X = U \cdot \sigma_x(\beta) + \mu_x(\beta)$
pX	cdf of X
qX	quantile function for X
rX	random number generator for X

Author(s)

Georg M. Goerg

Examples

```
Gauss_input = create_LambertW_input("normal", beta = c(2,1))
```

```
Gauss_input$dX # this returns a function itself
```

```
Gauss_input$dX(1) # evaluate the pdf of a N(2,1) at x = 1
```

```
plot(Gauss_input$dX(), -2,6) # by default it will take the beta from the LambertW_input object
```

```

plot(Gauss_input$dX(beta = c(3,4)), -2,6) # change the beta to c(3,4)

plot(Gauss_input$pX(), -3,6)
integrate(Gauss_input$dX(), -10,10)
Gauss_input$rX()(n = 100) # generate 100 samples from a N(2,1)

```

create_LambertW_output

Modular toolkit for generation of output distribution object

Description

(IMPORTANT: This functionality of the package is still in development; function names, functionality, etc. may be subject to change.)

To use this function you have to use [create_LambertW_input](#) first.

This function takes an object of class `LambertW_input` and creates an object of class `LambertW_output` for standard distribution as well as the user-defined distribution. This `LambertW_output` represents the RV $Y \sim \text{Lambert } W \times \text{'my favorite distribution'}$ with all its properties and R functionality, such as random number generation (`rY`), pdf (`dY`) and cdf (`pY`), etc. for this newly generated `Lambert W` \times 'favorite distribution'.

Please see Details and Values below for instructions.

Usage

```
create_LambertW_output(input = NULL, theta = NULL, distname = input$distname)
```

Arguments

<code>input</code>	an object of class <code>LambertW_input</code>
<code>theta</code>	a list containing (not necessarily all) parameters (alpha, beta, gamma, delta). The function complete_theta fills in missing standard values for the parameters. If not provided, the function will print out a warning since the transformation is the identity and so the output RV $Y \equiv \text{input } X$.
<code>distname</code>	name of the input distribution $F_X(x \beta)$; default: <code>input\$distname</code>

Details

This function converts a `LambertW_input` list into a `LambertW_output` list.

`create_LambertW_output` is based on the definition of `Lambert W` \times F RVs by the input/output approach (see References), and transports this very flexible and powerful framework to the code level. This allows users to create their own `Lambert W` \times 'my favorite distribution' RVs and then use R's standard functionality for distributions such as random number generation (`rY`), pdf (`dY`) and cdf (`pY`), quantile function (`qY`), etc. for this newly generated `Lambert W` \times 'favorite distribution' in empirical analysis.

Thus users can start estimating parameters, estimate quantiles, plot density functions within minutes of a newly defined Lambert $W \times F$ RV, which should promote a wide-spread use of Lambert $W \times F$ distributions in practice.

If the distribution you intend to use is not one of the already implemented (`distname`), then 1) use `create_LambertW_input` with your favorite distribution, and 2) pass it as an input argument to the `create_LambertW_output` along with the parameters.

Value

An object of class `LambertW_output`; the values of an `LambertW_output` object are (for the most part) functions themselves (see Examples):

<code>dY</code>	pdf of $Y \sim \text{Lambert } W \times \text{'my favorite distribution'}$ (see References)
<code>pY</code>	cdf of Y
<code>rY</code>	quantile function for Y
<code>rY</code>	random number generator for Y
<code>distname</code>	character string with the name of the new distribution. Format: "Lambert $W \times$ 'my favorite distribution'"
<code>beta</code>	parameter vector β of the input distribution $F_X(x \beta)$
<code>parameters</code>	a list containing all parameters (alpha, beta, gamma, delta).
<code>theta</code>	theta vector
<code>distname_with_beta</code>	name of the new distribution including the parameter beta. Format: "Lambert $W \times$ 'my favorite distribution(beta)'"

Author(s)

Georg M. Goerg

Examples

```
# create a Gaussian N(1,2) input
Gauss_input = create_LambertW_input("normal", beta = c(1,2))

# create a heavy-tailed version of a normal -> Tukey's h with h = 0.3
params = list(delta = c(0.3)) # gamma = 0, alpha = 1 will be assigned automatically if missing; beta comes from the input
LW.Gauss = create_LambertW_output(input = Gauss_input, theta = params)
LW.Gauss

op = par(no.readonly=TRUE)
par(mfrow=c(2,1), mar=c(3,3,2,1))
plot(LW.Gauss$dY(params), -7, 10, col="red")
plot(LW.Gauss$dY(), -7, 10, col="blue") # same in blue;
# actual information here: parameter for the specified model is not necessary to give again as input to the returned object

params.0 = params # compare to the input case (i.e. set delta = 0)
params.0$delta = 0
```

```

plot(LW.Gauss$dY(params.0), -7,10, add=TRUE, col=1) # to evaluate the RV at a different parameter value, it is neces
par(op)

plot(LW.Gauss$pY(params), -7,10, col="red")
plot(LW.Gauss$pY(params.0), -7,10, add=TRUE, col=1)

normfit(LW.Gauss$rY(params)(n=100))

## generate a positively skewed version of the decentralized scaled t_3 distribution
t_input = create_LambertW_input("t", beta = c(2,1,3))
t_input
params = list(gamma = 0.05) # skew it
LW.t = create_LambertW_output(input = t_input, theta = params)
LW.t

plot(LW.t$dY(params),-7,11, col = 2)
plot(t_input$dX(), -7,11, add=TRUE, col=1)
abline(v = params$beta[1])

# draw samples from the skewed t_3
yy = LW.t$rY()(n=100)
normfit(yy)

## generate a positively skewed version a Cauchy distribution
cauchy_input = create_LambertW_input("cauchy", beta = c(0,1))
params = list(alpha = 1, beta = cauchy_input$beta, gamma = 0.05, delta = 0) # the skewed version
LW.cauchy = create_LambertW_output(input = cauchy_input,distname=cauchy_input$distname, theta = params)
LW.cauchy

exp_input = create_LambertW_input("exp", beta = 1)
plot(exp_input)

# create a skewed exponential distribution
params = list("gamma" = 0.2)
LW.exp = create_LambertW_output(exp_input, theta = params)
plot(LW.exp)

# create a heavy-tail exponential distribution
params = list("delta" = 0.2)
LW.exp = create_LambertW_output(exp_input, theta = params)
plot(LW.exp)

# create a skewed chi-square distribution
chi_input = create_LambertW_input("chisq", beta = 5) # a Chi^2 with 5 df
plot(chi_input)
params = list("gamma" = sqrt(2)*0.2)
LW.chi = create_LambertW_output(chi_input, theta = params)
plot(LW.chi)

```

delta_01	<i>Input parameters to get a zero mean, unit variance output for a given delta</i>
----------	--

Description

Given δ , this function computes the input mean $\mu_x(\delta)$ and standard deviation $\sigma_x(\delta)$ for Gaussian input $X \sim N(\mu_x(\delta), \sigma_x^2(\delta))$, such that the resulting heavy-tail Lambert W Gaussian RV Y has zero mean and unit variance.

The function works for any output mean and variance, but default values are $\mu_y = 0$ and $\sigma_y = 1$ since they are the most useful, e.g. to generate a standardized LambertW white noise sequence.

So far only for Gaussian input and one-dimensional δ .

Usage

```
delta_01(delta, mu_y=0, sigma_y=1)
```

Arguments

delta	heavy-tail parameter (= Tukey's h for Gaussian input)
mu_y	output mean; default: 0
sigma_y	output standard deviation; default: 1

Value

The 3-dimensional vector $(\mu_x(\delta), \sigma_x(\delta), \delta)$.

Author(s)

Georg M. Goerg

Examples

```
delta_01(0) # for delta = 0, input == output, therefore (0,1,0)
delta_01(0.1) # delta > 0 (heavy-tails skewed): since Y is symmetric for all delta: mean = 0; however, sd must be sm
delta_01(1/3) # only moments up to order 2 exist
delta_01(1) # variance does not exist anymore for this delta, thus NaN
```

delta_GMM	<i>Estimate optimal delta</i>
-----------	-------------------------------

Description

This function minimizes the Euclidean distance between the theoretical kurtosis $\gamma_2(X)$, and the sample kurtosis of the back-transformed data $W_\delta(z)$ as a function of δ (see References). Note that only an iterative application of this function will give a good estimate of $\delta \rightarrow$ see [IGMM](#).

Usage

```
delta_GMM(z, kurtosis_x = 3, skewness_x = 0, type = "h", delta.0 = delta_Taylor(z), tol=.Machine$double.
```

Arguments

z	a numeric vector of data values.
kurtosis_x	theoretical kurtosis of the input X; default: 3 ($X \sim$ Gaussian)
skewness_x	theoretical skewness of the input X. This will be only used if type = "hh"; default: 0 ($X \sim$ symmetric)
type	what type of distribution should be estimated? heavy-tail Lambert W "h"; heavy-tails Lambert W "hh".
delta.0	starting value for optimization; default: delta_Taylor .
tol	a positive scalar giving the tolerance at which the distance is considered close enough to zero to terminate the algorithm; default: <code>.Machine\$double.eps^0.25</code>
restricted	indicator if the estimate for δ should be restricted to the positive reals; default: TRUE
bounds	bounds for the estimate of δ ; in practice δ will rarely exceed 2; here it is set to a maximum of 10. On the lower end it is set to -1. If restricted = TRUE then the first element will be overwritten and set to 0.

Value

A list with two elements:

delta	optimal δ for data z
iterations	number of iterations

Author(s)

Georg M. Goerg

See Also

[gamma_GMM](#) for the skewed version of this function; [IGMM](#) for an iterative method to estimate all parameters jointly.

Examples

```

set.seed(2)
y = rLambertW(n=1000, beta=c(1,2), delta = 1) ## very heavy-tails (like a Cauchy)

delta_GMM(y) # after the first iteration
IGMM(y, type="h")$TAU # all iterations
## note the fast settling down to a neighborhood of the final solution

```

delta_Taylor

Estimate of delta by Taylor approximation

Description

Computes an initial estimate of δ based on the Taylor approximation of the kurtosis of Lambert $W \times$ Gaussian RVs. See Details for the formula.

This is the initial estimate for [IGMM](#).

Usage

```
delta_Taylor(y = NULL, kurtosis_y = kurtosis(y))
```

Arguments

`y` a numeric vector of data values.
`kurtosis_y` kurtosis of `y`; default: empirical kurtosis of data `y`.

Value

Estimate of δ .

Computed using the second order Taylor approximation around $\delta = 0$

$$\gamma_2(\delta) = 3 + 12\delta + 66\delta^2 + \mathcal{O}(\delta^3).$$

Ignoring higher order terms and using the empirical estimate on the LHS we can solve for δ (only using the positive root) to get

$$\hat{\delta}_{Taylor} = \frac{1}{66} \cdot \left(\sqrt{66\widehat{\gamma_2(\mathbf{y})} - 162 - 6} \right),$$

where $\widehat{\gamma_2(\mathbf{y})}$ is the empirical kurtosis of the data `y`.

Since the kurtosis is only defined for $\delta < 1/4$, the output of `delta_Taylor` is upper bounded by $1/4$.

Author(s)

Georg M. Goerg

See Also

[IGMM](#) for an iterative method to estimate all parameters jointly.

Examples

```
set.seed(2)
```

```
y = rLambertW(n=1000, beta=c(0,1), delta = 0.2) ## a little heavy-tailed (kurtosis does exist)
delta_Taylor(y) # initial estimate is good because true delta = 0.2 close to 0, and empirical kurtosis well-defined.
delta_GMM(y) # best estimate
```

```
y = rLambertW(n=1000, beta=c(0,1), delta = 1) ## very heavy-tails (like a Cauchy)
delta_Taylor(y) # initial estimate very bad, because delta = 1 far from 0, and empirical kurtosis not well defined
delta_GMM(y) # best estimate
```

G

Tukey's h transformation with $h = 1$

Description

Tukey's h transformation with $h = 1$ equals $G(u) = u \exp(\frac{1}{2} \cdot u^2) = z$.

Usage

```
G(u)
```

Arguments

`u` a numeric vector of real/complex values.

Value

Returns $z = u \exp(\frac{1}{2} \cdot u^2)$ for $u \in C$. If u is a vector, so is z .

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011b). "The Lambert Way to Gaussianize skewed, heavy-tailed data with the inverse of Tukey's h transformation as a special case.". In preparation for submission (<http://arxiv.org/abs/1010.2265>).

See Also

[G_delta_alpha](#)

Examples

```
G(0)
G(5)

plot(G, -2,2, type="l", xlab="u", ylab="G(u)", lwd=2)
abline(h=0)
abline(v=0)
```

gamma_01	<i>Input parameters to get a zero mean, unit variance output for a given gamma</i>
----------	--

Description

Given γ , this function computes the input mean $\mu_x(\gamma)$ and standard deviation $\sigma_x(\gamma)$ for Gaussian input $X \sim N(\mu_x(\gamma), \sigma_x^2(\gamma))$, such that the resulting skewed Lambert W Gaussian RV Y has zero mean and unit variance.

The function works for any output mean and variance, but default values are $\mu_y = 0$ and $\sigma_y = 1$ since they are the most useful, e.g. to generate a standardized LambertW white noise sequence.

So far only for Gaussian input.

Usage

```
gamma_01(gamma, mu_y=0, sigma_y=1)
```

Arguments

gamma	skewness parameter
mu_y	output mean; default: 0
sigma_y	output standard deviation; default: 1

Value

The 3-dimensional vector $(\mu_x(\gamma), \sigma_x(\gamma), \gamma)$.

Author(s)

Georg M. Goerg

Examples

```
gamma_01(0) # for gamma = 0, input == output, therefore (0,1,0)
gamma_01(0.1) # gamma>0 (positively skewed): input mean must be slightly negative to get a zero-mean output
gamma_01(1)
```

gamma_GMM

*Estimate optimal gamma***Description**

This function minimizes the Euclidean distance between the theoretical skewness $\gamma_2(X)$, and the sample skewness of the back-transformed data $W_\delta(z)$ as a function of γ (see References). Note that only an iterative application of this function will give a good estimate of $\gamma \rightarrow$ see [IGMM](#).

A robust measure of the asymmetry can also be used (see MedCouple estimator: [mc](#)).

Usage

```
gamma_GMM(z, skewness_x = 0, gamma.0 = gamma_Taylor(z), robust = FALSE, tol=.Machine$double.eps^0.25, re
```

Arguments

<code>z</code>	a numeric vector of data values.
<code>gamma.0</code>	starting value for γ ; default: $(\text{skewness}(z) - \text{skewness}_x)/6$ (the Taylor approximation - see References)
<code>skewness_x</code>	theoretical skewness of the input X . default: 0
<code>robust</code>	indicator if robust estimation of the sample skewness (mc) should be used; default: FALSE
<code>tol</code>	a positive scalar giving the tolerance at which the distance is considered close enough to zero to terminate the algorithm; default: $.Machine\$double.eps^0.5$
<code>restricted</code>	indicator if the estimate for γ should be restricted to the positive reals; default: FALSE

Value

A list with two elements:

<code>gamma</code>	optimal γ for data z
<code>iterations</code>	number of iterations

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011a). "Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation". Forthcoming in the Annals of Applied Statistics (arxiv.org/abs/0912.4554).

See Also

[delta_GMM](#) for the heavy-tail version of this function; [mc](#) for a robust measure of asymmetry; [IGMM](#) for an iterative method to estimate all parameters accurately

Examples

```
set.seed(1)
y = rLambertW(n=1000, beta = c(1,2), gamma = 0.1, distname="normal") ## very highly skewed

gamma_GMM(y) # after the first iteration
IGMM(y)$theta # after the final iteration; convergence has been reached.

IGMM(y)$TAU
```

gamma_Taylor

Estimate of gamma by Taylor approximation

Description

Computes an initial estimate of γ based on the Taylor approximation of the skewness of Lambert $W \times$ Gaussian RVs around $\gamma = 0$. See Details for the formula.

This is the initial estimate for [IGMM](#).

Usage

```
gamma_Taylor(y = NULL, skewness_y = skewness(y), skewness_x = 0)
```

Arguments

`y` a numeric vector of data values.
`skewness_y` skewness of y ; default: empirical skewness of data y .
`skewness_x` skewness for input X ; default: 0 (symmetric input).

Value

Estimate of γ .

Computed using the second order Taylor approximation around $\delta = 0$

$$\gamma_1(\gamma) = 6\gamma + \mathcal{O}(\gamma^3).$$

Ignoring higher order terms and using the empirical estimate on the LHS we can solve for δ (only using the positive root) to get

$$\hat{\gamma}_{Taylor} = \frac{1}{6}\widehat{\gamma_1(\mathbf{y})},$$

where $\widehat{\gamma_1(\mathbf{y})}$ is the empirical skewness of the data \mathbf{y} .

As the Taylor approximation is only good in a neighborhood of $\gamma = 0$, the output of `gamma_Taylor` is restricted to the interval $-(0.25, 0.25)$.

Author(s)

Georg M. Goerg

See Also[IGMM](#) for an iterative method to estimate all parameters jointly.**Examples**

```
set.seed(2)

yy = rLambertW(n=1000, beta=c(0,1), gamma = 0.1) ## a little skewness
gamma_Taylor(yy) # initial estimate is good because true delta = 0.2 close to 0, and empirical kurtosis well-defined
gamma_GMM(yy) # best estimate
```

Gaussianize

*Gaussianizing matrix-like objects***Description**

'Gaussianize' is probably the most useful function in this package. It works the same way as the [scale](#) function in the R base package, but instead of just centering and scaling the data, it actually "Gaussianizes" the data. See Goerg (2011b) and Examples.

Usage

```
Gaussianize(y, type = "h", method = "MLE")
```

Arguments

y	a numeric matrix-like object
type	what type of non-normality: symmetric heavy-tails "h", skewed heavy-tails "hh", or just skewed "s".
method	what estimator should be used: "MLE" or "IGMM". "IGMM" gives exactly Gaussian characteristics (kurtosis $\equiv 3$ for "h" or skewness $\equiv 0$ for "s"), "MLE" comes close to this.

Value

the Gaussianized matrix.

The numeric parameters of mean, scale, and skewness/heavy-tail parameters that were used in the Gaussianizing transformation are returned as attributes 'Gaussianized:mu', 'Gaussianized:sigma', 'Gaussianized:delta' (for "h") or 'Gaussianized:delta_l' and 'Gaussianized:delta_r' (for "hh"), or 'Gaussianized:gamma' (for "s").

Note that mean and scale in the Lambert $W \times$ Gaussian framework are in general not identical to the center and scale of the [scale](#) function.

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011b). "The Lambert Way to Gaussianize skewed, heavy-tailed data with the inverse of Tukey's h transformation as a special case.". In preparation for submission (<http://arxiv.org/abs/1010.2265>).

Examples

```

set.seed(20)
y1 = rcauchy(n = 1000)
x1 = Gaussianize(y1)
normfit(x1) # Gaussianized a Cauchy!

start_from = 20
y_cum_avg = (cumsum(y1)/(1:length(y1)))[-c(1:start_from)]
x_cum_avg = (cumsum(x1)/(1:length(x1)))[-c(1:start_from)]

plot(c((start_from+1): length(y1)),y_cum_avg, type="l" , lwd = 2, main="Central limit theorem in practice", xlab = "x", ylab = "y")
lines(c((start_from+1): length(y1)), x_cum_avg, col=2, lwd=2)
abline(h = 0)

plot(x1, y1, xlab="Gaussian-like input", ylab = "Cauchy - output")

y2 = rLambertW(n = 1000, beta=c(3,1,6), delta = c(0,0.1), distname="t")
YY = cbind(y1, y2)
plot(YY)

XX = Gaussianize(YY, type="hh")
plot(XX)

```

get.input

Back-transform Y to X

Description

Given the data y and the parameter vector $\tau = (\mu_x(\beta), \sigma_x(\beta), \gamma, \alpha)$ (or $\tau = (\mu_x(\beta), \sigma_x(\beta), \delta, \alpha)$), this function computes the back-transformed input data \hat{x}_τ .

Usage

```
get.input(y, tau, return.u=FALSE)
```

Arguments

y a numeric vector of data values or an object of class LambertW_fit.
tau parameter vector
return.u should the normalized input be returned; default: FALSE

Value

The approximated input data vector \hat{x}_τ .

For $\gamma \neq 0$ it uses the principal branch solution `W_gamma` to get a unique input. For $\gamma = 0$ the back-transformation is bijective (for any $\delta \geq 0$).

If `return.u = TRUE` then it returns a list with 2 vectors

u centered and normalized input u
x input data \hat{x}_θ

Author(s)

Georg M. Goerg

Examples

```

set.seed(12)

# unskew very skewed data
y=rLambertW(n=1000, beta=c(0,1), gamma = 0.3)
normfit(y)
fit.gmm=IGMM(y, type="s")
summary(fit.gmm)

x=get.input(y, fit.gmm$tau)
# the same as
x = get.input(fit.gmm)
normfit(x) # symmetric Gaussian
  
```

G_delta_alpha

Tukey's h transformation with delta

Description

Heavy-tail Lambert W RV transformation: $H_\delta(u) = u \exp(\frac{\delta}{2}(u^2)^\alpha)$.

Usage

```

G_delta_alpha(u, delta = 0, alpha = 1)
G_2delta_alpha(u, delta=c(0,1/5), alpha = c(1,1))
  
```

Arguments

u	a numeric vector of real values
delta	heavy tail parameter; default delta = 0, which implies $G_{\text{delta_alpha}}(u) = u$.
alpha	exponent in $(u^2)^\alpha$; default alpha = 1.

Value

$u \exp(\frac{\delta}{2}(u^2)^\alpha)$.

Author(s)

Georg M. Goerg

Examples

```
G_delta_alpha(1, delta=1)
```

```
G_delta_alpha(2) ## default value is delta = 0: hence, G_delta(u) = u
G_delta_alpha(0, delta=1) ## G_delta(0) = 0 for all delta
```

H

Original function defining the Lambert W function

Description

The Lambert W function $W(z)$ is the inverse of this function: $H(u) = u \exp(u) = z$.

Usage

$H(u)$

Arguments

u a numeric vector of real/complex values.

Value

Returns $z = u \exp(u)$ for $u \in C$. If u is a vector, so is z .

Author(s)

Georg M. Goerg

References

Corless, R. M., G. H. Gonnet, D. E. G. Hare, and D. J. Jeffrey (1993). "On the Lambert W function". preprint.

See Also[H_gamma](#)**Examples**

```
H(0)
H(10)

plot(H, -5, 0.5, type="l", xlab="u", ylab="z")
abline(h=0)
abline(v=0)
```

H_gamma

H transformation with gamma

Description

Skewed Lambert W RV transformation: $H_\gamma(u) = u \exp(\gamma u)$.

This function is a wrapper for $H(\gamma u)/\gamma$.

Usage

```
H_gamma(u, gamma = 0)
```

Arguments

u a numeric vector of real values
gamma skewness parameter; default gamma = 0, which implies $H_\gamma(u) = u$.

Value

$u \exp(\gamma u)$.

Author(s)

Georg M. Goerg

Examples

```
H_gamma(1, gamma=1)

H_gamma(2) ## default value is gamma = 0: hence, H(u) = u
H_gamma(0, gamma=1) ## H_gamma(0)= 0 for all gamma
```

Description

An iterative methods of moments estimator finds this $\tau = (\mu_x, \sigma_x, \gamma, 0)$ (or $\tau = (\mu_x, \sigma_x, 0, \delta)$) which minimizes the distance between the sample and theoretical skewness (or kurtosis) of \mathbf{x} and \mathbf{X} .

For details of the Algorithm see the References.

Usage

```
IGMM(y,type="s", skewness_x = 0, kurtosis_x = 3, tau.0 = c(median(y), sd(y), gamma_Taylor(y), delta_Taylor(y)),
## Default S3 method:
IGMM(y, type="s", skewness_x = 0, kurtosis_x = 3, tau.0 = c(median(y), sd(y), gamma_Taylor(y), delta_Taylor(y))
```

Arguments

<code>y</code>	a numeric vector of real values.
<code>skewness_x</code>	theoretical skewness of input \mathbf{X} ; default 0
<code>type</code>	what type of distribution should be estimated? skewed Lambert W "s"; heavy-tail Lambert W "h"; heavy-tails Lambert W "hh".
<code>kurtosis_x</code>	theoretical kurtosis of input \mathbf{X} ; default 3 (Gaussian)
<code>tau.0</code>	starting values for IGMM algorithm; default: $(\text{median}(y), \text{sd}(y), \text{gamma_Taylor}(y), \text{delta_Taylor}(y))$. The standard deviation estimate gets changed inside IGMM to an improved initial estimate (given the estimate for γ and/or δ . See also gamma_Taylor and delta_Taylor .
<code>robust</code>	indicator if robust estimation of the sample skewness (see mc) should be used; default: FALSE (only for type = "s")
<code>tol</code>	a positive scalar giving the tolerance at which the distance is considered close enough to zero to terminate the algorithm; default: <code>.Machine\$double.eps^0.25</code>
<code>location_family</code>	is the underlying input distribution $F_X(x \beta)$ from a location family (for example, Gaussian input); default: TRUE. If FALSE, then $\theta_1 \equiv 0$ throughout the optimization, for example for exponential input.
<code>restricted</code>	indicator if the estimate for γ or δ should be restricted to the positive reals. If it is set to NULL (default) then it will be set internally to TRUE for heavy-tail(s) Lambert W x F distributions (type = "h" or "hh"; for skewed Lambert W x F (type = "s") it will be set to FALSE, unless it is not a scale family location_family = FALSE: in this case it is set to TRUE.

Value

A list of class LambertW_fit:

tol	tolerance level; see Arguments for a detailed description
data	data y
n	number of observations
type	type of Lambert W distribution; see Arguments for a detailed description
tau.0	starting value for θ
tau	IGMM estimate for θ
TAU	entire iteration sequence of $\tau^{(k)}$
sub_iterations	number of iterations only performed in GMM algorithm to find optimal γ (or δ)
iterations	number of iterations performed by the IGMM algorithm to update μ_x and σ_x . See References for details.
theta	a list containing γ or δ of IGMM.
hessian	Hessian matrix (obtained from simulations; see References)
call	function call
distname	a character string stating the theoretical skewness (or kurtosis) of the input distribution. Same information as skewness_x (or kurtosis_x)
skewness_x	theoretical skewness of the input (numeric value); default: 0
kurtosis_x	theoretical kurtosis of the input (numeric value); default: 3
location_family	see Arguments above.
message	message from the optimization method. What kind of convergence?
method	Estimation method; here: "IGMM"

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011a). "Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation". Forthcoming in the Annals of Applied Statistics (arxiv.org/abs/0912.4554).

Examples

```
# estimate theta for the skewed version of a Normal
y=rLambertW(n=1000, beta = c(2,1), distname = "normal", gamma = 0.2)
fity=IGMM(y, type="s")
fity
summary(fity)
plot(fity)
```

```
# estimate theta for the heavy-tailed version of a Normal = Tukey's h
y=rLambertW(n=1000, beta = c(2,1), distname = "normal", delta = 0.2)
fity=IGMM(y, type="h")
fity
summary(fity)
plot(fity)
```

ks.test.t

One-sample Kolmogorov-Smirnov test for student-t distribution

Description

Performs a two-sided KS test for $H_0 : X \sim t_\nu$ with ν degrees of freedom, mean μ_x , and standard deviation σ_x (not scale!). If the parameters are not specified, the ML estimates given the data are used (see [fitdistr](#)).

Usage

```
ks.test.t(y, theta = NULL)
```

Arguments

y	a numeric vector of data values.
theta	parameter vector including the degrees of freedom parameter: $\theta = (\mu_x, \sigma_x, \nu, \gamma)$; default: theta = NULL, thus θ is estimated from y.

Details

For estimated parameters of the t-distribution the p-values are slightly off, and should be corrected. For large sample sizes this should become ignoreable however. See [ks.test](#) and the references therein (Durbin (1973)).

Value

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

statistic	the value of the Kolmogorov-Smirnov statistic.
p.value	the p-value for the test.
alternative	a character string describing the alternative hypothesis.
method	the character string "One-sample Kolmogorov-Smirnov test student-t" plus rounded parameter values.
data.name	a character string giving the name(s) of the data.

Author(s)

Georg M. Goerg

See Also

[fitdistr](#), [ks.test](#)

LambertW-methods

The Lambert W x F Distribution

Description

Density, distribution function, quantile function and random number generation for the Lambert $W \times F_X(x | \beta)$ distribution with parameter vector $(\alpha, \beta, \gamma, \delta)$.

Implemented distributions (`distname = ...`) are: "cauchy", "chisq", "exp", "normal", "t", "unif". To be extended.

`qqLambertW` computes and plots the sample quantiles of the data `y` versus the theoretical Lambert $W \times F$ theoretical quantiles implied by the parameter estimate $\hat{\theta}$.

`mLambertW` computes the first 4 central moments of a Lambert $W \times$ Gaussian.

Usage

```
dLambertW(y, beta = c(0,1), gamma = 0, delta = 0, alpha = 1, distname = c("normal"), input.U = NULL, theta = NULL)
pLambertW(q, beta = c(0,1), gamma = 0, delta = 0, alpha = 1, distname = c("normal"), input.U = NULL, theta = NULL)
qLambertW(p, beta = c(0,1), gamma = 0, delta = 0, alpha = 1, distname = c("normal"), input.U = NULL, theta = NULL)
rLambertW(n = 1000, beta = c(0,1), gamma = 0, delta = 0, alpha = 1, distname = "normal", return.input = FALSE)
mLambertW(beta = c(0, 1), distname = c("normal"), gamma = 0, delta = 0, alpha = 1, theta = NULL)
qqLambertW(y, beta = c(0,1), gamma = 0, delta = 0, alpha = 1, distname = c("normal"), plot.it = TRUE, theta = NULL)
```

Arguments

<code>y, q</code>	vector of quantiles.
<code>p</code>	vector of probability levels
<code>n</code>	number of observations
<code>distname</code>	input distribution. default "normal".
<code>beta</code>	parameter vector β of the input distribution
<code>gamma</code>	skewness parameter; default: 0
<code>delta</code>	heavy-tail parameter (= Tukey's h in case of a normal distribution); default: 0
<code>alpha</code>	exponent; default: 1
<code>theta</code>	an optional argument to pass on in case parameters are available as a list. If <code>theta</code> is supplied, don't supply parameters <code>alpha</code> , <code>beta</code> , <code>gamma</code> , <code>delta</code> before; they are ignored. Instead <code>theta\$alpha</code> , etc is used.
<code>return.input</code>	indicator for returning the simulated input <code>x</code> that gets transformed to the actual observations <code>y</code> ; default: FALSE. Warning: if TRUE then the output is not a vector anymore, but a list containing the input <code>x</code> and the output <code>y</code> .
<code>plot.it</code>	Should the QQ plot be displayed or only the (sorted) theoretical and empirical quantiles be returned? default: TRUE.

`input.U` you can supply your own version of `U` (either a vector of simulated values or a function defining the pdf/cdf/quantile function of `U`); default: `NULL`.

`...` further arguments passed to or from other methods.

Details

See the references for the analytic expressions of the pdf and cdf. Quantiles are computed numerically for the `type = "s"` case (no analytic expression yet); for `"h"` or `"hh"` quantiles can be computed analytically given the input quantile function and θ .

All functions have the optional argument of user-supplied versions of the zero-mean, unit variance input `U`. For example, if you have your own density function as `dmydist(u) = function(u) ...` then you can pass this function to `dLambertW(..., input.U = dmydist(u))` and it will use this pdf to generate plots, estimates, etc. Important: for user-defined input, all `input.U` in `dLambertW`, `pLambertW`, ... must be supplied correctly.

Value

Following the standard distributions in R (e.g. `rnorm`, `rexp`, ...) the notation and functionality of LambertW distribution work as expected: `dLambertW` computes the value of the pdf at `y`, `pLambertW` the cdf at `y`, `qLambertW` is the quantile function, and `rLambertW` generates random deviates from a Lambert $W \times F_X(x | \beta)$ distribution.

`mLambertW` returns a vector with the 4 theoretical (central/standardized) moments of Y implied by β and `distname` (works so far only for `distname = "normal"`):

<code>mu</code>	mean
<code>sigma</code>	standard deviation
<code>skewness</code>	skewness
<code>kurtosis</code>	kurtosis (i.e. = 3 for a Gaussian)

`qqLambertW` returns a list of 2 vectors (in accordance to the output of `qqnorm`):

<code>x</code>	theoretical quantiles (sorted)
<code>y</code>	empirical quantiles (sorted)

By default `rLambertW` returns a sampled Lambert $W \times F_X(x | \beta)$ data vector. If `return.input = TRUE`, then the output is a list of 2 vectors:

<code>x</code>	simulated input
<code>y</code>	Lambert W random sample (given by <code>x</code> and the functional relation $y = func(x)$ - see References.

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011a). "Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation". Forthcoming in the Annals of Applied Statistics (arxiv.org/abs/0912.4554).

Goerg, G.M. (2011b). "The Lambert Way to Gaussianize skewed, heavy-tailed data with the inverse of Tukey's h transformation as a special case.". In preparation for submission (<http://arxiv.org/abs/1010.2265>).

Examples

```
#####
##### mLambertW #####
mLambertW(beta = c(0,1), gamma = 0.1)
mLambertW(gamma=0.1) # the same as by default: mean=0, standard deviation =1
mLambertW(beta = c(1,1), gamma = 0.1) # mean shifted by 1
mLambertW(gamma=0) # N(0,1)

#####
##### rLambertW #####
set.seed(1)
x=rLambertW(n=1000, beta=c(0,1)) # same as rnorm(1000)
skewness(x) # very small skewness
mc(x) # also close to zero

y = rLambertW(n=1000, beta=c(1,3), gamma = 0.1)
skewness(y) # high positive skewness (in theory equal to 3.70)
mc(y) # also the robust measure gives a high value

op=par(no.readonly=TRUE)
par(mfrow=c(2,2), mar=c(2,4,3,1))
plot(x)
hist(x, prob=TRUE, 15)
lines(density(x))

plot(y)
hist(y, prob=TRUE, 15)
lines(density(y))
par(op)
#####
##### dLambertW #####
beta.s = c(0,1)
gamma.s = 0.1

aux.dlambert=function(y){
  dLambertW(y, beta=beta.s, gamma = gamma.s, distname="normal")
}

x11(width=10, height=5)
par(mfrow=c(1,2), mar=c(3,3,3,1))
plot(aux.dlambert, -3.5,5, ylab="", main="Density function")
plot(dnorm, -3.5,5, add=TRUE, lty=2)
legend("topright" , c("Lambert W - Gaussian" , "Gaussian"), lty=1:2)
```

```

abline(h=0)

#####
##### pLambertW #####

aux.plambert=function(y){
pLambertW(y, beta=beta.s, gamma = gamma.s, distname="normal")
}

plot(aux.plambert, -3.5,3.5, ylab="", main="Distribution function")
plot(pnorm, -3.5,3.5, add=TRUE, lty=2)
legend("topleft" , c("Lambert W x Gaussian" , "Gaussian"), lty=1:2)
par(op)

##### Animation
gamma.v = seq(-0.15,0.15, length=31) # typical, empirical range of gamma
b = support(gamma_01(min(gamma.v)))[2]*1.1
a = support(gamma_01(max(gamma.v)))[1]*1.1

for (ii in 1:length(gamma.v)) {
aux.dlambert=function(y){
dLambertW(y, beta = gamma_01(gamma.v[ii])[1:2], gamma = gamma.v[ii], distname="normal") # zero-mean, unit variance
#dLambertW(y, beta = c(0,1), gamma = gamma.v[ii], distname="normal") # mean and variance change with gamma; only for
}

plot(aux.dlambert, a,b, ylab="", lty=2, col=2, lwd=2, main="pdf", ylim = c(0,0.45))
plot(dnorm, a,b, add=TRUE, lty=1, lwd=2)
legend("topright" , c("Lambert W x Gaussian" , "Gaussian"), lty=2:1, lwd=2, col=2:1)
abline(h=0)
legend("topleft", cex=1.3, c(as.expression(bquote(gamma == .(round(gamma.v[ii],3))))))
Sys.sleep(0.05)
}

#####
##### qLambertW #####

p.v=c(0.01, 0.05, 0.5, 0.9, 0.95,0.99)

qnorm(p.v)
qLambertW(p.v, beta = c(0,1), distname="normal", gamma = 0, delta = 0, alpha = 1) # the same as above except for nume

# positively skewed data -> quantiles are higher
qLambertW(p.v, beta = c(0,1), distname="normal", gamma = 0.1, delta = 0, alpha = 1)

#####
##### qqLambertW #####

y=rLambertW(n=500, distname="normal", beta = c(0,1), gamma = 0.1)
x11(height=6, width=9)
par(mfrow=c(1,2), mar=c(4,4,3,1))
qqnorm(y)
qqline(y)
qqLambertW(y, beta = c(0,1), gamma = 0.1)

```

```
par(op)
```

LambertW_fit-methods *Methods for output of LambertW estimators*

Description

Displays the output (class `LambertW_fit`) of the `MLE_LambertW` or `IGMM` estimator in a well-arranged and informative way: parameter estimates $\hat{\theta}$, standard errors (including 'significance stars'), theoretical support (only for type="s"), skewness tests, etc.; nice formatting (`print`), and plots (see `Details` and `Examples`).

Usage

```
## S3 method for class 'LambertW_fit'
summary(object, ...)
## S3 method for class 'summary.LambertW_fit'
print(x, ...)
## S3 method for class 'LambertW_fit'
print(x, ...)
## S3 method for class 'LambertW_fit'
plot(x, QQ=FALSE, a= NULL, b = NULL, ...)
```

Arguments

<code>object</code>	object of class <code>LambertW_fit</code>
<code>x</code>	object of class <code>LambertW_fit</code>
<code>a</code>	left limit from where cdf and pdf are plotted
<code>b</code>	right limit until cdf and pdf are plotted
<code>QQ</code>	should a LambertW QQ plot be displayed? default FALSE
<code>...</code>	further arguments passed to or from other methods.

Details

`print.LambertW_fit` prints only very basic information about $\hat{\theta}$ (to prevent an overload of data/information in the console when executing an estimator).

`print.summary.LambertW_fit` tries to be smart about formatting the coefficients, standard errors, etc. and additionally gives "significance stars".

`plot.LambertW_fit` plots a (1) histogram, (2) empirical density of the data y . These are compared (3) to the theoretical $F_X(x | \hat{\beta})$ and (4) Lambert W $\times F_X(\cdot | \hat{\beta})$ densities.

Value

summary gives an object of class `summary.LambertW_fit`. A list containing

<code>call</code>	function call
<code>coefmat</code>	matrix with 4 columns: $\hat{\theta}$, its standard errors, t-statistic, and corresponding (two-sided) p-values
<code>distname</code>	name of the input distribution
<code>n</code>	number of observations
<code>data</code>	original data set (<code>y</code>)
<code>input</code>	back-transformed data
<code>support</code>	support of <code>Y</code> (only for <code>type = "s"</code>)
<code>data.range</code>	empirical data range
<code>method</code>	estimation method
<code>hessian</code>	Hessian at the optimum. Numerically obtained for <code>method = "MLE"</code> ; for <code>method = "IGMM"</code> only a diagonal-matrix approximation taken from the covariance matrix obtained by simulations for $n = 1000$ observations.
<code>p_1</code>	Probability that one observation was caused by input lying on the non-principal branch; see <code>p_1</code> ; only for <code>type = "s"</code> .
<code>p_1n</code>	Probability that one or more of the n observation was caused by input lying on the non-principal branch; see <code>p_1</code> ; only for <code>type = "s"</code> .
<code>symmetry_pval</code>	the p-value for the test of identical left and right tail parameters by the Wald test (see <code>skewness_test</code>); only for <code>type = "hh"</code> .

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011a). "Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation". Forthcoming in the Annals of Applied Statistics (arxiv.org/abs/0912.4554).

Examples

```
data(AA)
attach(AA)

X=AA[AA$sex=="f",]
y=X$bmi

fit.ml=MLE_LambertW(y)
A=summary(fit.ml)
print(A)

plot(fit.ml)
plot(fit.ml, a = 12, b = 40)
```

 LambertW_input-methods

Methods for LambertW input objects

Description

plot.LambertW_input plots the theoretical (1) pdf and (2) cdf of the input $X \sim F_X(x | \beta)$
 print.LambertW_input prints an overview of the generated object.

Usage

```
## S3 method for class 'LambertW_input'
plot(x, a = NULL, b = NULL, ...)
## S3 method for class 'LambertW_input'
print(x, ...)
```

Arguments

x	object of class LambertW_input
a	left limit from where cdf and pdf are plotted; not necessary to specify
b	right limit until cdf and pdf are plotted; not necessary to specify
...	further arguments passed to or from other methods.

Author(s)

Georg M. Goerg

Examples

```
# create a heavy-tailed version of a normal -> Tukey's h with h = 0.1
Gauss_input = create_LambertW_input("normal", beta = c(1,2))
Gauss_input
plot(Gauss_input)
```

 LambertW_output-methods

Methods for LambertW output objects

Description

plot.LambertW_output plots the theoretical (1) pdf and (2) cdf of the output RV $Y \sim \text{Lambert W} \times F_X(x | \beta)$. It also adds the plot of the input RV $X \sim F_X(x | \beta)$ with no transformation ($\gamma = \delta = 0, \alpha = 1$).

print.LambertW_output prints an overview of the generated object.

Usage

```
## S3 method for class 'LambertW_output'
plot(x, a = NULL, b = NULL, ...)
## S3 method for class 'LambertW_output'
print(x, ...)
```

Arguments

x	object of class LambertW_output
a	left limit from where cdf and pdf are plotted; not necessary to specify
b	right limit until cdf and pdf are plotted; not necessary to specify
...	further arguments passed to or from other methods.

Author(s)

Georg M. Goerg

Examples

```
# create a heavy-tailed version delta = 0.3 of a Gaussian -> Tukey's h with h = 0.3
Gauss_input = create_LambertW_input("normal", beta = c(1,2))
params = list(delta = c(0.3))
LW.Gauss = create_LambertW_output(input = Gauss_input, theta = params)
LW.Gauss # print
plot(LW.Gauss)
```

loglik-utils

Log-Likelihood of the data for Lambert W x F_X RVs

Description

The log-likelihood of Lambert $W \times F$ RVs in most cases decomposes into two additive terms: i) the log-likelihood of the input, and ii) a penalty term for transforming the data.

loglik_input computes the log-likelihood for data x for various distributions given the parameter vector β . It can therefore be used independently of the Lambert W framework, just in general to compute log-likelihood of common distributions and parameters.

loglik_penalty computes the penalt of transforming the data back to the input (not necessarily Gaussian; any distribution specified by distname).

loglik_LambertW simply adds the two values to get the actual log-likelihood of the data y under Lambert $W \times F$ distributions and parameter vector θ .

theta2params and params2theta are internal functions that are used to simplify the conversion between the list-type theta and the vector-style argument of all optimization routines (which all require a vector-type argument of the function to be optimized).

Usage

```

loglik_input(beta = NULL, x = NULL, distname = "normal", f_X = NULL, logf_X = function(x) log(f_X(x)))
loglik_penalty(theta, y = NULL, type = "h", distname = "normal")
theta2params(theta)
params2theta(params, distname="normal", type ="h")
loglik_LambertW(theta, y = NULL, distname = "normal", type = "h", return.neg.value = FALSE)

```

Arguments

y	a numeric vector of real values (the observed data).
x	a numeric vector of real values (the 'input' data). Can be used also for general data values to evaluate the log-likelihood (not necessarily in the Lambert W framework).
distname	input distribution. default "normal".
beta	parameter vector β for the input distribution. Please see specifications in the code how to write beta for each distribution.
f_X	density function of x. Common distributions are already built-in (see distname). If you want to supply your own density, you must supply a function of x and beta.
logf_X	a function that returns the logarithm of the density function of x. Often the log of $f_X(x)$ has a simpler form (which is also faster to evaluate).
theta	the parameter vector in form of a list containing beta, gamma, delta etc.
type	what type of transformation? skewed Lambert W "s"; heavy-tail Lambert W "h"; heavy-tails Lambert W "hh".
params	a vector containing the parameter values of θ in an unlisted way.
return.neg.value	an indicator to return only the negative log-likelihood of the data (which is useful for optimization routines); otherwise it returns the list of input, penalty, and their sum = full likelihood. Default: FALSE.

Details

See the references for the detailed expressions of log-likelihood and penalty terms.

Generally, for the heavy-tail versions (type = "h" or type = "hh") such a decomposition always exists. For skewed Lambert W distributions it only exists for non-negative input (e.g. exponential, gamma, F, ...), not (!) for "normal", "t", "unif", "cauchy", ... In this case loglik_input and loglik_penalty will return NA, but the full log-likelihood will be returned correctly.

Value

loglik_input and loglik_penalty return one value.

loglik_LambertW Returns a list with 3 values:

loglik_input the loglikelihood of the transformed variable

loglik_penalty the penalty for transforming the data

loglik_LambertW

the total log-likelihood of the data given θ = sum of the two values before.

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011a). “Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation”. Forthcoming in the Annals of Applied Statistics (arxiv.org/abs/0912.4554).

Goerg, G.M. (2011b). “The Lambert Way to Gaussianize skewed, heavy-tailed data with the inverse of Tukey’s h transformation as a special case.”. In preparation for submission (<http://arxiv.org/abs/1010.2265>).

Examples

```
###
set.seed(1)
yy = rLambertW(n = 1000, beta = c(0,1), delta = 0.2)
loglik_penalty(list(beta = c(1,1), delta = c(0.2,0.2)), y = yy, type = "hh", distname = "normal")
loglik_penalty(list(beta = c(1,1), delta = c(0.2), gamma = 0.03), y = yy, type = "h", distname = "normal") # here suc

### computing the Gaussian log-likelihood
loglik_input(beta = c(0,1), x = yy, distname = "normal") # built-in version

loglik_input(beta = c(0,1), x = yy, distname = "user", logf_X = function(xx, beta = beta) dnorm(xx, mean = beta[1], s

### loglik_LambertW
loglik_LambertW(theta = list(beta = c(1,1), delta = c(0.09,0.07)), y = yy, type = "hh", distname = "normal") # all th
```

mc

MedCouple Estimator

Description

A robust measure of skewness. Details see references.

Usage

```
mc(x)
```

Arguments

x data

Value

a numerical value, indicating the amount of asymmetry

Author(s)

Georg M. Goerg

References

Brys, G., M. Hubert, and A. Struyf (2004). "A robust measure of skewness". *Journal of Computational and Graphical Statistics* 13 (4), 996 - 1017.

Examples

```
## a little simulation/demonstration
n.sim = 100

##### Gaussian (Symmetric) #####
A = cbind(0,0)
for (ii in 1:n.sim) {
  x = rnorm(100)
  A = rbind(A, c(mc(x), skewness(x)))
}
A = A[-1,]

##### skewed LambertW x Gaussian #####
theta.s = gamma_01(0.2) # zero mean, unit variance, but positive skewness
rbind(mLambertW(beta=theta.s[1:2], gamma = theta.s[3], distname="normal"))

B = cbind(0,0)
for (i in 1:n.sim) {
  y = rLambertW(n=100, beta=theta.s[1:2], gamma = theta.s[3], distname="normal")
  B = rbind(B, c(mc(y), skewness(y)))
}
B = B[-1,]

colnames(A) = colnames(B) = c("MedCouple", "Pearson Skewness")
op=par(no.readonly=TRUE)

par(mfrow=c(2,2), mar=c(4,4,3,1))
plot(A, main="Gaussian")
boxplot(A)
abline(h=0)

plot(B, main="Lambert W x Gaussian")
boxplot(B)
abline(h=mLambertW(beta=theta.s[1:2], gamma = theta.s[3], distname="normal")[3])
par(op)

apply(A, 2, mean)
apply(A, 2, sd)

apply(B, 2, mean)
apply(B, 2, sd)
```

MLE_LambertW

*Maximum Likelihood Estimation for Lambert W x F distributions***Description**

Maximum Likelihood Estimation (MLE) for Lambert W $\times F_X(\cdot \mid \beta)$ distributions; computes $\hat{\theta}_{MLE}$.

For type = "s" the parameter γ is estimated and $\delta = 0$ is held fixed; for type = "h" the one-dimensional δ is estimated and $\gamma = 0$ is held fixed; and finally for type = "hh" the 2-dimensional δ is estimated and $\gamma = 0$ is held fixed.

By default $\alpha = 1$ is fixed for any type. If you want to also estimate α from the data pass `fixed_theta = list()` to MLE_LambertW.

Usage

```
MLE_LambertW(y, distname = c("normal"), type = "s", theta.0 = list(), fixed_theta = list(alpha = 1), hes
## Default S3 method:
MLE_LambertW(y, distname = c("normal"), type = "s", theta.0 = list(), fixed_theta = list(alpha = 1), hes
```

Arguments

y	a numeric vector of real values.
distname	input distribution; default: "normal".
theta.0	a list containing the starting values of $(\alpha, \beta, \gamma, \delta)$ for the numerical optimization; default: see starting_theta .
hessian	indicator for returning the (numerically obtained) Hessian at the optimum; default: TRUE.
fixed_theta	a list of fixed parameters in the optimization; default only alpha = 1.
type	what type of distribution should be estimated? skewed Lambert W "s"; heavy-tail Lambert W "h"; heavy-tails Lambert W "hh".
estimate_only	an indicator if only $\hat{\theta}_{MLE}$ should be returned (or the entire output); for simulations, for example, it is not necessary to give a nicely formatted, organized output, because the only interest lies in $\hat{\theta}_{MLE}$; default: FALSE.

Value

A list of class LambertW_fit:

data	the data y
loglik	function: log-likelihood function
loglik.opt	value: log-likelihood evaluated at the optimum
theta.0	a list containing the starting values for numerical optimization
parameters	a list containing the estimated values of $(\alpha, \beta, \gamma, \delta)$

beta	estimated β of the input distribution via Lambert W MLE ($\neq \hat{\beta}$ of the input data)
theta	MLE for θ
type	see Arguments above
hessian	Hessian matrix; used to calculate standard errors (only if <code>hessian = TRUE</code> , otherwise NULL)
call	function call
distname	character string describing the input distribution.
message	message from the optimization method. What kind of convergence?
method	Estimation method; here "MLE"

Author(s)

Georg M. Goerg

Examples

```
data(AA)
attach(AA)

X=AA[AA$sex=="f",]
y=X$bmi

fit.ml=MLE_LambertW(y, type="s") # skewed only
summary(fit.ml)
plot(fit.ml)

fit.ml = MLE_LambertW(y, type="hh") # skewed heavy-tails (not too good fit because delta_1 = 0)
plot(fit.ml)
```

normfit

Graphical and statistical Gaussianity check

Description

Graphical and statistical check if data is Gaussian (3 common Normality tests, QQ-plots, histograms, etc).

These tests are from the `nor test` package. See 'See Also'.

Usage

```
normfit(data, volatility = FALSE, plot.it = TRUE, pch = 1, legend = TRUE)
```

Arguments

<code>data</code>	a numeric vector of data values.
<code>plot.it</code>	Should graphical inference be plotted (histogram, densities, qqplot, ...); default TRUE; otherwise only test results are returned.
<code>volatility</code>	Should the squared data and its autocorrelation be plotted? Useful for (financial) time series to see if squares exhibit dependence (for financial data they typically do); default: FALSE
<code>pch</code>	argument for the plot of the original data; default <code>pch = 1</code>
<code>legend</code>	should legends be placed in the histogram/density plot; default: TRUE

Value

A list containing 3 normality tests (each of class `htest`)

<code>ad</code>	Anderson Darling
<code>sf</code>	Shapiro-Francia
<code>sw</code>	Shapiro-Wilk

Author(s)

Georg M. Goerg

References

Thode Jr., H.C. (2002): "Testing for Normality". Marcel Dekker, New York.

See Also

[ad.test](#), [shapiro.test](#), [sf.test](#) in the `nortest` package

Examples

```
y=rLambertW(n = 1000, beta=c(3,4), gamma = 0.1, distname="normal")
normfit(y)

x = rnorm(n=1000)
normfit(x)
```

p_1 *Non-principal branch probability*

Description

Computes the probability that (at least) one (out of n) observation(s) of the latent variable U lie in the non-principal branch region. See Details.

Usage

```
p_1(gamma = 0, beta = c(0,1), distname="normal", n = 1)
```

Arguments

gamma	skewness parameter
beta	parameter vector of the input distribution
distname	name of the input distribution; default: "normal".
n	number of RVs/observations; default n=1.

Details

The probability that one observation of the latent U lies in the non-principal region is at most (see References)

$$p_{-1}(\gamma, n = 1) = P\left(U < -\frac{1}{|\gamma|}\right),$$

where U is the zero-mean, unit variance version of the input $X \sim F_X(x | \beta)$.

For N independent RVs U_1, \dots, U_N , we want to know the probability that at least one lies in the non-principal region.

$$p_{-1}(\gamma, n = N) := P\left(U_i < -\frac{1}{|\gamma|} \text{ for at least one } i\right)$$

By the rules of probability this equals (assuming independence)

$$\begin{aligned} P\left(U_i < -\frac{1}{|\gamma|} \text{ for at least one } i\right) &= 1 - P\left(U_i \geq -\frac{1}{|\gamma|}, \forall i\right) = 1 - \prod_{i=1}^N P\left(U_i \geq -\frac{1}{|\gamma|}\right) \\ &= 1 - \prod_{i=1}^N (1 - p_{-1}(\gamma, n = 1)) = 1 - (1 - p_{-1}(\gamma, n = 1))^N. \end{aligned}$$

Due to numerical stability the cdf of a geometric RV ([pgeom](#)) is used to evaluate the last expression.

Note that $1 - (1 - p_{-1}(\gamma, n = 1))^N$ reduces to the original definition $p_{-1}(\gamma, n = 1)$ for $N = 1$.

Value

p_{-1} for n observations (in practice very small given empirical evidence). Numerical problems can occur for $\gamma < 0.03$ (always returns 0 due to rounding errors).

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011a). "Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation". Forthcoming in the Annals of Applied Statistics (arxiv.org/abs/0912.4554)

Examples

```
# for n=1 observation
p_1(0) # this probability is identical to 0
p_1(0.01) # in theory not identical to 0; but machine precision too low
p_1(0.05) # extremely small
p_1(0.1) # not 0, but very small; gamma = 0.1 typical value for sample data
p_1(1.5) # 1 out of 4 samples is a non-principal input; however, gamma=1.5 is not common in practice

# for n=100 observations
p_1(0, n=100) # this probability is identical to 0
p_1(0.1, n=100) # still small
p_1(0.3, n=100) # a bit more likely
p_1(1.5, n=100) # we can be almost 100% sure (rounding errors) that at least one of the y_i observations was caused by
```

skewness_test

One-sample Kolmogorov-Smirnov test for student-t distribution

Description

Performs a test for the null hypothesis of symmetry $H_0 : \delta_l = \delta_r$ versus the alternative of asymmetry. This can be done by a Wald test of the linear restriction $H_0 : \delta_l - \delta_r = 0$ or a likelihood ratio test.

Usage

```
skewness_test(obj, method = "Wald")
```

Arguments

obj an object of class LambertW_fit with type = "hh" or sequence of observation. If data is passed to the function, then an asymmetric Lambert W \times Gaussian distribution (distname = "normal") with two tail parameters ("hh") is fit to the data internally and then used as the new obj.

method what kind of test: the "Wald" or the likelihood "ratio" test

Value

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

statistic	the value of the test statistic.
p.value	the p-value for the test.
method	the character string describing the test.
data.name	a character string giving the name(s) of the data.

Author(s)

Georg M. Goerg

Examples

```
# skewed data
yy = rLambertW(n = 1000, delta = c(0.1, 0.2), beta=c(2,1), distname="normal")
fit.ml = MLE_LambertW(yy, type="hh", distname="normal")
summary(fit.ml)
skewness_test(fit.ml, "ratio")
skewness_test(fit.ml, "Wald")

# symmetric data
yy = rLambertW(n = 1000, delta = c(0.2, 0.2), beta=c(2,1), distname="normal")
fit.ml = MLE_LambertW(yy, type="hh", distname="normal")
summary(fit.ml)
skewness_test(fit.ml, "ratio")
skewness_test(fit.ml, "Wald")
```

SolarFlares

Solar Flares data set

Description

This dataset represents the peak gamma-ray intensity of solar flares recorded from Feb, 1980 - Dec, 1989. It was analyzed for power-law properties in Clauset et al. (2009) and comes originally from Dennis et al. (1991).

Thanks to both authors and co-authors it could be made available in this package.

Usage

```
data(SolarFlares)
```

Format

A numeric vector of with $T = 12,773$ observations.

Source

<http://tuvalu.santafe.edu/~aaronc/powerlaws/data.htm>

<http://adsabs.harvard.edu/abs/1991chxb.book.....D>

See also References.

References

Dennis, B. R.; Orwig, L. E.; Kennard, G. S.; Labow, G. J.; Schwartz, R. A.; Shaver, A. R.; Tolbert, A. K. (1991). "The Complete Hard X Ray Burst Spectrometer Event List, 1980-1989." NASA Technical Memorandum 4332. See also <http://adsabs.harvard.edu/abs/1991chxb.book.....D>

Clauset, A., C.-R. Shalizi, and M.-E.-J. Newman (2009). "Power-law distributions in empirical data". SIAM Review 51, 661-703 (2009). See also <http://tuvalu.santafe.edu/~aaronc/powerlaws/>

Examples

```
data(SolarFlares)

plot(density(SolarFlares[SolarFlares < 350]), main = "Original Data (truncated to < 350)")

plot(SolarFlares, pch = ".", cex = 2)
plot(SolarFlares[SolarFlares < 350], pch = ".", cex = 2)

fit = MLE_LambertW(SolarFlares, type = "h", distname = "normal")
summary(fit)

xx = get.input(fit)
plot(xx, pch = ".", cex = 2, main = "Heavy-tails removed")
plot(density(xx), main = "Heavy-tails removed")
```

support

Compute the truncated support for skewed Lambert W distributions

Description

If the input $X \sim F$ has support on the entire real line $(-\infty, \infty)$, then the skewed Lambert $W \times F$ distribution has truncated support $[a, b]$, $a, b \in R \cup \pm\infty$ depending on β and (the sign of) γ .

Usage

```
support(theta, gamma = NULL)
```

Arguments

theta parameter vector with mean and variance (and γ as the third entry)

gamma optional argument to pass γ directly to the function, instead of the indirect theta.

Value

Half-open interval on the real line (if $\gamma \neq 0$) for input with support on the entire real line. For $\gamma = 0$ the support of Y is the same as for X . Heavy-tail Lambert W RVs are not affected by truncated support (since $\delta \geq 0$ by definition).

Author(s)

Georg M. Goerg

Examples

```
support(c(0,1,0)) # as gamma = 0
support(c(0,1,0.1)) # truncated on the left since gamma > 0
```

theta-utils

Checks/completes/restricts/bounds/finds starting values for theta

Description

`check_theta` checks if parameters $\alpha, \beta, \gamma, \delta$ describe a well-defined Lambert W distribution. Prints out a warning if they do not, or if number of parameters given is too large/small for the specified distribution.

`complete_theta` completes missing values in a parameters list so users don't have to specify everything in detail. If not supplied, then the following will be set: $\alpha = 1$, $\gamma = 0$, and $\delta = 0$. If β is not specified, but an object of class `LambertW_input` is passed on, then β will be set to the value of the input β .

`starting_theta` provides starting parameters for α, β, γ , and δ to be used in ML (`MLE_LambertW`) estimation. These values are obtained by first finding the input $\hat{x}_{\hat{\theta}}$ generating the output \mathbf{y} by IGMM, and then getting the MLE of β for this input data $\hat{x}_{\hat{\theta}} \sim F_X(x | \beta)$.

`bounds_theta` returns lower and upper bounds for θ which are necessary to specify for `MLE_LambertW` estimation.

`restrict_theta` restricts θ to transform the bounded search space to an unrestricted search space (by log-transformation). Thus by the log transformation we can convert the bounded optimization problem to an unbounded one, optimize on this space, and then go back again.

`theta2tau` converts θ to the implied parameter vector τ . See also `beta2tau`.

Usage

```
check_theta(alpha = 1, beta = NULL, gamma = 0, delta=0, distname = NULL)
complete_theta(theta = list(beta = c(0,1)), input = NULL)
starting_theta(y, type = "h", distname="normal", fixed_theta = list(alpha = 1))
bounds_theta(distname = "normal", beta = NULL, type = "s", fixed_theta = list(alpha = 1))
restrict_theta(theta, distname = "normal", type = "h", inverse = FALSE)
theta2tau(theta = list(beta = c(0,1)), distname = "normal")
```

Arguments

alpha	exponent; default: 1
beta	parameter vector β of the input distribution; default: NULL
gamma	skewness parameter; default: 0
delta	heavy-tail parameter (= Tukey's h in case of a normal distribution); default: 0
distname	name of the input distribution $F_X(x \beta)$.
theta	a (probably incomplete) parameter list; default: list(beta = c(0, 1))
input	an optional argument of LambertW_input
y	a numeric vector of real values.
type	type of distribution to be estimated: skewed Lambert W "s"; one-parameter heavy-tail Lambert W "h" or two-parameter heavy-tail Lambert W "hh" distribution.
fixed_theta	a list of fixed parameters in the optimization; default only alpha = 1
inverse	should the transformation be undone, i.e. use exp instead of log. Default: TRUE.

Value

check_theta prints out a warning if θ does not define a proper Lambert W \times F distribution, or if number of parameter vector β given is too large/small for the specified distribution; does nothing if parameters define a proper distribution.

complete_theta returns a list containing the arguments

alpha	exponent; default: 1
beta	parameter vector β of the input distribution; default: (0, 1)
gamma	skewness parameter; default: 0
delta	heavy-tail parameter (= Tukey's h in case of a normal distribution); default: 0

starting_theta returns a list containing the arguments

alpha	exponent; default: 1
beta	parameter vector β of the input distribution; estimated from the recovered input data $\widehat{\mathbf{x}}_{IGMM}$
gamma	skewness parameter; if type is "h" or "hh" gamma = 0; estimated from IGMM
delta	heavy-tail parameter (= Tukey's h in case of a normal distribution); if type = "s" delta = 0 ; estimated from IGMM

bounds_theta returns a list containing two vectors

lowerbounds	lower bounds for estimating θ
upperbounds	upper bounds for estimating θ

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011a). "Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation". Forthcoming in the Annals of Applied Statistics (arxiv.org/abs/0912.4554)

Examples

```

check_theta(beta= c(1,1,-1), distname = "t")
check_theta(beta= c(1,1,-1), distname = "normal")

check_theta(beta= c(1,1), distname = "normal") # ok

params = list(beta = c(2,1), delta = c(0.3)) # here alpha and gamma are missing
complete_theta(params) # plugged in with default values

###
x=rnorm(1000)
starting_theta(x, distname="normal", type="h")
starting_theta(x, distname="normal", type="s")

# starting values for the skewed version of a Normal
y = rLambertW(n=1000, distname="exp", beta = 2, gamma = 0.1)
starting_theta(y, distname="exp", type="s")

# starting values for the heavy-tailed version of a Normal = Tukey's h
y=rLambertW(n=1000, beta = c(2,1), distname = "normal", delta = 0.2)
starting_theta(y, distname="normal", type = "h") #
summary(MLE_LambertW(y, distname="normal", type="h")) # quite close

###
bounds_theta(type="hh", distname="normal", beta = c(0,1), fixed_theta = list())

###
theta_restr = restrict_theta(list(beta = c(-1,0.1), delta = c(0.2,0.2)))
theta_restr
restrict_theta(theta_restr, inverse = TRUE) # returns again the beta and delta from above

```

Description

Density, distribution function, quantile function and random number generation for the (zero-mean and) unit variance version U of the (location-)scale family input $X \sim F_X(x | \beta)$ - see References.

Usage

```
dU(u, beta = NULL, distname=c("normal"))
rU(n = NULL, beta = NULL, distname=c("normal"))
pU(u, beta = NULL, distname = c("normal"))
qU(p, beta = NULL, distname = c("normal"))
```

Arguments

u	vector of quantiles.
n	number of observations
p	vector of probability levels
beta	parameter vector of input distribution F_X (appropriate centering/scaling is done internally).
distname	input distribution. default "normal".

Details

Since the zero-mean, unit-variance (or only unit-variance) version U is one of the main building blocks of Lambert W \times F distributions, these functions are wrappers to be used by other functions such as [dLambertW](#) or [rLambertW](#).

Value

dU evaluates the pdf at y, pU evaluates the cdf, qU is the quantile function, and rU generates random deviates of U.

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011a). "Lambert W Random Variables - A New Family of Generalized Skewed Distributions with Applications to Risk Estimation". Forthcoming in the Annals of Applied Statistics (<http://arxiv.org/abs/0912.4554>).

Examples

```
# a zero-mean, unit variance version of the t_3 distribution.
plot(seq(-4, 4, length = 100),dU(seq(-4, 4, length = 100), beta = c(1,1,3), distname = "t"), type="l")
# the cdf of the unit-variance version of an exp(3) -> just an exp(1)
plot(seq(0, 4, length = 100),pU(seq(0, 4, length = 100), beta = 3, distname = "exp"), type="l")

# all have variance 1
var(rU(n = 1000, distname = "chisq", beta = 2))
var(rU(n = 1000, distname = "normal", beta = c(3,3)))
var(rU(n = 1000, distname = "exp", beta = 1))
var(rU(n=1000, distname="unif", beta = c(0,10)))
```

vec.norm

*L_p norm of a vector***Description**

Computes the L^p norm of an n-dimensional (real/complex) vector.

Usage

```
vec.norm(x, p=2)
```

Arguments

x n-dimensional vector (possibly complex values)
p which norm? default: p=2 (Euclidean norm)

Details

The L^p norm of a vector $\mathbf{x} \in \mathbb{C}^n$ equals

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p},$$

where $|x_i|$ is the absolute value of x_i . For $p = 2$ we get the Euclidean norm; for $p = 1$ the Manhattan distance.

Value

the norm of **x**

Author(s)

Georg M. Goerg

Examples

```
a = c(3,-4)

# Pythagoras
vec.norm(a)
# did not know Manhattan.
vec.norm(a, p=1)

### unit circle?
x = exp(1i*runif(20, -pi,pi))
plot(x)
sapply(x, vec.norm) # yes, indeed!
```

Description

The Lambert W function $W(z)$, $W_{-1}(z)$ and its first derivative ($d1W(z)$, $d1W_{-1}(z)$); in both cases the principal and non-principal branch. (These are wrapper functions for [lambert_W0](#) and [lambert_Wm1](#) in the `gsl` package.)

Usage

```
W(z)
W_1(z)
d1W(z, W_z = W(z))
d1W_1(z, W_1_z = W_1(z))
```

Arguments

<code>z</code>	a numeric vector of real values.
<code>W_z</code>	the principal branch of the Lambert W function evaluated at <code>z</code> ; see Details below for why this is useful.
<code>W_1_z</code>	the non-principal branch of the Lambert W function evaluated at <code>z</code> ; see Details below for why this is useful.

Details

The Lambert W function $W(z)$ is implicitly defined as the inverse function of

$$W(z) \exp(W(z)) = z.$$

For $z \geq 0$ the solution is unique $W(z)$.

For $-1/e \leq z < 0$ it has two solutions: the principal ($W(z)$) and non-principal branch ($W_{-1}(z)$).

For $z < -1/e$ no solution exists in the reals.

The derivative can be expressed as a function of $W(z)$:

$$W'(z) = \frac{1}{(1 + W(z)) \exp(W(z))} = \frac{W(z)}{z(1 + W(z))}, \quad z \neq 0, -1/e.$$

For this reason it is numerically often better to supply the value of $W(z)$ in the function, as often these values are computed already in a previous step of the computation.

For details see the References.

Value

Function value or NaN if $z < -1/e$.

Author(s)

Georg M. Goerg

References

Corless, R. M., G. H. Gonnet, D. E. G. Hare, and D. J. Jeffrey (1993). "On the Lambert W function". preprint.

See Also

[lambert_W0](#) and [lambert_Wm1](#) in the gsl package.

Examples

```

W(5) # exists in R
W(-5) # does not exist in R

W(-0.25) # the "reasonable" input event
W_1(-0.25) # the "extreme" input event

plot(W, -1, 3, type="l")
plot(W_1, -1, 3, type="l")

# the invsere of H
H(0)
W(0)

H(10)
W(H(10))

## at the critical point z = -1, both branches give the same inverse.
H(-1)
W(H(-1))
W_1(H(-1))

## going further to the left, the principal branch gives the 'wrong' solution;
## the non-principal must be used
H(-10)
W(H(-10))
W_1(H(-10))

```

W_delta_alpha

Inverse transformation for heavy-tail Lambert W RVs

Description

Inverse transformation for heavy-tail Lambert W RVs; inverse of Tukey's h transformation as a special case.

Usage

```

W_delta(z, delta = 0)
W_delta_alpha(z, delta = 0, alpha = 1)
W_2delta(z, delta = c(0,1/5))
W_2delta_alpha(z, delta = c(0,1/5), alpha = 1)
d1W_delta(z, delta = 1)
d1W_delta_alpha(z, delta = 1, alpha = 1)

```

Arguments

z a numeric vector of real values.

delta heavy-tail parameter; by default delta = 0.1, which implies $W_{\text{delta}}(z) = z$. If a vector of length 2 is supplied, then delta[1] on the left, and delta[2] on the right (of the center) will be used.

alpha exponent in $(u^2)^\alpha$; default alpha = 1.

Value

Computes $\text{sgn}(z) \left(\frac{1}{\delta} W(\delta(z^2)^\alpha)\right)^{1/2\alpha}$. If z is a vector, so is the output.

Author(s)

Georg M. Goerg

References

Goerg, G.M. (2011b). “The Lambert Way to Gaussianize skewed, heavy-tailed data with the inverse of Tukey’s h transformation as a special case.”. In preparation for submission (<http://arxiv.org/abs/1010.2265>).

Examples

```

G(0)
W(0)

G(10)
W_delta_alpha(G(10), delta = 1, alpha=1) # the inverse

```

W_gamma

Inverse transformation for skewed Lambert W RVs

Description

Inverse transformation for skewed Lambert W RVs. Principal and non-principal branch.

Usage

```

W_gamma(z, gamma = 0)
W_gamma_1(z, gamma = 0)

```

Arguments

<code>z</code>	a numeric vector of real values.
<code>gamma</code>	skewness parameter; by default <code>gamma = 0</code> , which implies <code>W_gamma(z) = W_gamma_1(z) = z</code> .

Details

A skewed Lambert W RV is defined by the transformation

$$z = u \exp(\gamma u) =: H_\gamma(u), \quad \gamma \in \mathbf{R}.$$

The function `W_gamma(z)` (and `W_gamma_1(z)`) are the inverse functions of this transformation. If $\gamma = 0$, then $z = u$ and the inverse transformation also equals the identity.

If $\gamma \neq 0$, the inverse transformation can be computed by

$$W_\gamma(z) = \frac{1}{\gamma} W(\gamma z).$$

Same holds for `W_gamma_1(z)`.

Value

Computes $\frac{1}{\gamma} W(\gamma z)$. If z is a vector, so is the output.

Author(s)

Georg M. Goerg

Index

- *Topic **datagen**
 - create_LambertW_input, 7
 - create_LambertW_output, 9
 - LambertW-methods, 27
 - loglik-utils, 34
 - U-methods, 47
- *Topic **datasets**
 - AA, 5
 - SolarFlares, 43
- *Topic **distribution**
 - create_LambertW_input, 7
 - create_LambertW_output, 9
 - LambertW-methods, 27
 - loglik-utils, 34
 - U-methods, 47
- *Topic **hplot**
 - LambertW_input-methods, 33
 - LambertW_output-methods, 33
 - normfit, 39
- *Topic **htest**
 - ks.test.t, 26
 - normfit, 39
 - skewness_test, 42
- *Topic **iteration**
 - IGMM, 24
- *Topic **manip**
 - get.input, 20
- *Topic **math**
 - beta-methods, 6
 - delta_01, 12
 - G, 15
 - G_delta_alpha, 21
 - gamma_01, 16
 - H, 22
 - H_gamma, 23
 - support, 44
 - theta-utils, 45
 - vec.norm, 49
 - W, 50
 - W_delta_alpha, 51
 - W_gamma, 52
- *Topic **models**
 - create_LambertW_output, 9
- *Topic **optimize**
 - delta_GMM, 13
 - delta_Taylor, 14
 - gamma_GMM, 17
 - gamma_Taylor, 18
 - IGMM, 24
 - MLE_LambertW, 38
- *Topic **package**
 - LambertW-package, 3
- *Topic **plot**
 - LambertW_fit-methods, 31
- *Topic **print**
 - LambertW_fit-methods, 31
 - LambertW_input-methods, 33
 - LambertW_output-methods, 33
- *Topic **univar**
 - create_LambertW_input, 7
 - create_LambertW_output, 9
 - Gaussianize, 19
 - LambertW-methods, 27
 - loglik-utils, 34
 - mc, 36
 - p_1, 41
 - U-methods, 47
- AA, 5
- ad.test, 40
- beta-methods, 6
- beta2tau, 45
- beta2tau (beta-methods), 6
- beta_names (beta-methods), 6
- bounds_theta (theta-utils), 45
- check_theta (theta-utils), 45
- complete_theta, 9

- complete_theta (theta-utils), 45
- create_LambertW_input, 3, 7, 9, 10
- create_LambertW_output, 3, 7, 8, 9

- d1W (W), 50
- d1W_1 (W), 50
- d1W_delta (W_delta_alpha), 51
- d1W_delta_alpha (W_delta_alpha), 51
- delta_01, 12
- delta_GMM, 13, 18
- delta_Taylor, 13, 14, 24
- dLambertW, 48
- dLambertW (LambertW-methods), 27
- dU (U-methods), 47

- fitdistr, 26, 27

- G, 15
- G_2delta_alpha (G_delta_alpha), 21
- G_delta_alpha, 15, 21
- gamma_01, 16
- gamma_GMM, 13, 17
- gamma_Taylor, 18, 24
- Gaussianize, 3, 19
- get.input, 20

- H, 22
- H_gamma, 23, 23

- IGMM, 3, 13–15, 17–19, 24, 31, 45, 46

- ks.test, 26, 27
- ks.test.t, 26

- lambert_W0, 50, 51
- lambert_Wm1, 50, 51
- LambertW (LambertW-package), 3
- LambertW-package, 7, 8
- LambertW-methods, 27
- LambertW-package, 3
- LambertW_fit-methods, 31
- LambertW_input-methods, 33
- LambertW_output-methods, 33
- loglik-utils, 34
- loglik_input (loglik-utils), 34
- loglik_LambertW (loglik-utils), 34
- loglik_penalty (loglik-utils), 34

- mc, 17, 18, 24, 36
- mLambertW (LambertW-methods), 27

- MLE_LambertW, 3, 31, 38, 45
- normfit, 39

- p_1, 32, 41
- params2theta (loglik-utils), 34
- pgeom, 41
- pLambertW (LambertW-methods), 27
- plot.LambertW_fit, 3
- plot.LambertW_fit
 - (LambertW_fit-methods), 31
- plot.LambertW_input
 - (LambertW_input-methods), 33
- plot.LambertW_output
 - (LambertW_output-methods), 33
- print.LambertW_fit, 3
- print.LambertW_fit
 - (LambertW_fit-methods), 31
- print.LambertW_input
 - (LambertW_input-methods), 33
- print.LambertW_output
 - (LambertW_output-methods), 33
- print.summary.LambertW_fit
 - (LambertW_fit-methods), 31
- pU (U-methods), 47

- qLambertW, 3
- qLambertW (LambertW-methods), 27
- qqLambertW (LambertW-methods), 27
- qU (U-methods), 47

- restrict_theta (theta-utils), 45
- rLambertW, 3, 48
- rLambertW (LambertW-methods), 27
- rU (U-methods), 47

- scale, 3, 19
- sf.test, 40
- shapiro.test, 40
- skewness_test, 32, 42
- SolarFlares, 43
- starting_theta, 38
- starting_theta (theta-utils), 45
- summary.LambertW_fit
 - (LambertW_fit-methods), 31
- support, 44

- theta-utils, 45
- theta2params (loglik-utils), 34
- theta2tau (theta-utils), 45

U-methods, [47](#)

vec.norm, [49](#)

W, [50](#)

W_1 (W), [50](#)

W_2delta (W_delta_alpha), [51](#)

W_2delta_alpha (W_delta_alpha), [51](#)

W_delta (W_delta_alpha), [51](#)

W_delta_alpha, [51](#)

W_gamma, [21](#), [52](#)

W_gamma_1 (W_gamma), [52](#)