

Package ‘TukeyGH77’

July 22, 2025

Type Package

Title Tukey g-&-h Distribution

Version 0.1.4

Date 2025-03-15

Description Functions for density, cumulative density, quantile and simulation of Tukey g-and-h (1977) distributions. The quantile-based transformation (Hoaglin 1985 [doi:10.1002/9781118150702.ch11](https://doi.org/10.1002/9781118150702.ch11)) and its reverse transformation, as well as the letter-value based estimates (Hoaglin 1985), are also provided.

License GPL-2

Depends R (>= 4.4.0)

Imports rstpm2, stats

Suggests fitdistrplus

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

NeedsCompilation no

Author Tingting Zhan [aut, cre],
Inna Chervoneva [ctb]

Maintainer Tingting Zhan <tingtingzhan@gmail.com>

Repository CRAN

Date/Publication 2025-03-15 22:10:15 UTC

Contents

TukeyGH77-package	2
letterValue	2
TukeyGH	4

Index	6
--------------	----------

TukeyGH77-package *Tukey g-&-h Distribution*

Description

Density, cumulative density, quantile and simulation of the 4-parameter Tukey (1977) *g*-&-*h* distributions. The quantile-based transformation (Hoaglin 1985) and its reverse transformation, as well as the letter-value based estimates (Hoaglin 1985), are also provided.

Value

Returned values of individual functions are documented separately.

Author(s)

Maintainer: Tingting Zhan <tingtingzhan@gmail.com>

Other contributors:

- Inna Chervoneva <Inna.Chervoneva@jefferson.edu> [contributor]

References

Tukey, J.W. (1977): Modern Techniques in Data Analysis. In: NSF-sponsored Regional Research Conference at Southeastern Massachusetts University, North Dartmouth, MA.

Hoaglin, D.C. (1985): Summarizing shape numerically: The *g*-and-*h* distributions. Exploring data tables, trends, and shapes, pp. 461–513. John Wiley & Sons, Ltd, New York. doi:10.1002/9781118150702.ch11

letterValue *Letter-Value Estimation of Tukey g-&-h Distribution*

Description

Letter-value based estimation (Hoaglin, 1985) of Tukey *g*-, *h*- and *g*-&-*h* distribution. All equation numbers mentioned below refer to Hoaglin (1985).

Usage

```
letterValue(
  x,
  g_ = seq.int(from = 0.15, to = 0.25, by = 0.005),
  h_ = seq.int(from = 0.15, to = 0.35, by = 0.005),
  halfSpread = c("both", "lower", "upper"),
  ...
)
```

Arguments

<code>x</code>	double vector , one-dimensional observations
<code>g_</code>	double vector , probabilities used for estimating g parameter. Or, use <code>g_ = FALSE</code> to implement the constraint $g = 0$ (i.e., an h -distribution is estimated).
<code>h_</code>	double vector , probabilities used for estimating h parameter. Or, use <code>h_ = FALSE</code> to implement the constraint $h = 0$ (i.e., a g -distribution is estimated).
<code>halfSpread</code>	character scalar, either to use 'both' for half-spreads (default), 'lower' for half-spread, or 'upper' for half-spread.
<code>...</code>	additional parameters, currently not in use

Details

Unexported function `letterV_g()` estimates parameter g using equation (10) for g -distribution and the equivalent equation (31) for g -&- h distribution.

Unexported function `letterV_B()` estimates parameter B for Tukey g -distribution (i.e., $g \neq 0$, $h = 0$), using equation (8a) and (8b).

Unexported function `letterV_Bh_g()` estimates parameters B and h when $g \neq 0$, using equation (33).

Unexported function `letterV_Bh()` estimates parameters B and h for Tukey h -distribution, i.e., when $g = 0$ and $h \neq 0$, using equation (26a), (26b) and (27).

Function `letterValue()` plays a similar role as `fitdistrplus::start.arg.default`, thus extends `fitdistrplus::fitdist` for estimating Tukey g -&- h distributions.

Value

Function `letterValue()` returns a 'letterValue' object, which is **double vector** of estimates $(\hat{A}, \hat{B}, \hat{g}, \hat{h})$ for a Tukey g -&- h distribution.

Note

Parameter `g_` and `h_` does not have to be truly unique; i.e., **all.equal** elements are allowed.

References

Hoaglin, D.C. (1985). Summarizing Shape Numerically: The g -and- h Distributions. doi:10.1002/9781118150702.ch11

Examples

```
set.seed(77652); x = rGH(n = 1e3L, g = -.3, h = .1)
letterValue(x, g_ = FALSE, h_ = FALSE)
letterValue(x, g_ = FALSE)
letterValue(x, h_ = FALSE)
(m3 = letterValue(x))

library(fitdistrplus)
fit = fitdist(x, distr = 'GH', start = as.list.default(m3))
```

```
plot(fit) # fitdistrplus:::plot.fitdist
```

 TukeyGH

Tukey g-&-h Distribution

Description

Density, distribution function, quantile function and simulation for Tukey *g*-&-*h* distribution with location parameter *A*, scale parameter *B*, skewness *g* and elongation *h*.

Usage

```
dGH(x, A = 0, B = 1, g = 0, h = 0, log = FALSE, ...)

rGH(n, A = 0, B = 1, g = 0, h = 0)

qGH(p, A = 0, B = 1, g = 0, h = 0, lower.tail = TRUE, log.p = FALSE)

pGH(q, A = 0, B = 1, g = 0, h = 0, lower.tail = TRUE, log.p = FALSE, ...)
```

Arguments

<code>x, q</code>	double vector , quantiles
<code>A</code>	double scalar, location parameter $A = 0$ by default
<code>B</code>	double scalar, scale parameter $B > 0$. Default $B = 1$
<code>g</code>	double scalar, skewness parameter $g = 0$ by default (i.e., no skewness)
<code>h</code>	double scalar, elongation parameter $h \geq 0$. Default $h = 0$ (i.e., no elongation)
<code>log, log.p</code>	logical scalar, if TRUE, probabilities p are given as $\log(p)$.
<code>...</code>	other parameters of function vuniroot2()
<code>n</code>	integer scalar, number of observations
<code>p</code>	double vector , probabilities
<code>lower.tail</code>	logical scalar, if TRUE (default), probabilities are $Pr(X \leq x)$ otherwise, $Pr(X > x)$.

Value

Function [dGH\(\)](#) returns the density and accommodates [vector](#) arguments *A*, *B*, *g* and *h*. The quantiles *x* can be either [vector](#) or [matrix](#). This function takes about 1/5 time of `gk::dgh`.

Function [pGH\(\)](#) returns the distribution function, only taking scalar arguments and [vector](#) quantiles *q*. This function takes about 1/10 time of function `gk::pgh`.

Function [qGH\(\)](#) returns the quantile function, only taking scalar arguments and [vector](#) probabilities *p*.

Function [rGH\(\)](#) generates random deviates, only taking scalar arguments.

Examples

```
(x = c(NA_real_, rGH(n = 5L, g = .3, h = .1)))  
dGH(x, g = c(0,.1,.2), h = c(.1,.1,.1))  
  
p0 = seq.int(0, 1, by = .2)  
(q0 = qGH(p0, g = .2, h = .1))  
range(pGH(q0, g = .2, h = .1) - p0)  
  
q = (-2):3; q[2L] = NA_real_; q  
(p1 = pGH(q, g = .3, h = .1))  
range(qGH(p1, g = .3, h = .1) - q, na.rm = TRUE)  
(p2 = pGH(q, g = .2, h = 0))  
range(qGH(p2, g = .2, h = 0) - q, na.rm = TRUE)  
  
curve(dGH(x, g = .3, h = .1), from = -2.5, to = 3.5)
```

Index

`all.equal`, [3](#)

`character`, [3](#)

`dGH (TukeyGH)`, [4](#)

`dGH()`, [4](#)

`double`, [3](#), [4](#)

`integer`, [4](#)

`letterValue`, [2](#)

`letterValue()`, [3](#)

`logical`, [4](#)

`matrix`, [4](#)

`pGH (TukeyGH)`, [4](#)

`pGH()`, [4](#)

`qGH (TukeyGH)`, [4](#)

`qGH()`, [4](#)

`rGH (TukeyGH)`, [4](#)

`rGH()`, [4](#)

`TukeyGH`, [4](#)

`TukeyGH77 (TukeyGH77-package)`, [2](#)

`TukeyGH77-package`, [2](#)

`vector`, [3](#), [4](#)

`vuniroot2()`, [4](#)