

Package: TukeyGH77 (via r-universe)

August 20, 2024

Type Package

Title Tukey g-&-h Distribution

Version 0.1.3

Date 2024-08-19

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, cph], Inna Chervoneva [ctb, cph]

Maintainer Tingting Zhan <tingtingzhan@gmail.com>

Date/Publication 2024-08-19 15:30:06 UTC

Repository <https://tingtingzhan.r-universe.dev>

RemoteUrl <https://github.com/cran/TukeyGH77>

RemoteRef HEAD

RemoteSha f6ed081a0505cf7ba9de884416a4807141817cce

Contents

TukeyGH77-package	2
GH2z	3
letterValue	4
TukeyGH	5
vuniroot2	6
z2GH	8
Index	9

TukeyGH77-package	<i>Tukey g-&-h Distribution</i>
-------------------	-------------------------------------

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> [copyright holder]

Other contributors:

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

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](https://doi.org/10.1002/9781118150702.ch11)

GH2z

*Inverse of Tukey g-&-h Transformation***Description**

To transform Tukey *g*-&-*h* quantiles to standard normal quantiles.

Usage

GH2z(q, q0 = (q - A)/B, A = 0, B = 1, ...)

Arguments

q [double vector](#), quantiles *q*

q0 (optional) [double vector](#), standardized quantiles $q_0 = (q - A)/B$

A, B (optional) [double scalars](#), location and scale parameters of Tukey *g*-&-*h* transformation. Ignored if *q0* is provided.

... parameters of internal helper function [.GH2z](#)

Details

Unfortunately, function [GH2z](#), the inverse of Tukey *g*-&-*h* transformation, does not have a closed form and needs to be solved numerically.

For compute intensive jobs, use internal helper function [.GH2z](#).

Value

Function [GH2z](#) returns a [double vector](#) of the same length as input *q*.

Examples

```
z = rnorm(1e3L)
all.equal.numeric(.GH2z(z2GH(z, g = .3, h = .1), g = .3, h = .1), z)
all.equal.numeric(.GH2z(z2GH(z, g = 0, h = .1), g = 0, h = .1), z)
all.equal.numeric(.GH2z(z2GH(z, g = .2, h = 0), g = .2, h = 0), z)
```

 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](https://doi.org/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)

log, log.p	logical scalar, if TRUE, probabilities p are given as $\log(p)$.
...	other parameters of function <code>vuniroot2</code>
n	integer scalar, number of observations
p	double vector , probabilities
lower.tail	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)
```

Description

To solve a monotone function $y = f(x)$ for a given **vector** of y values.

Usage

```
vuniroot2(
  y,
  f,
  interval = stop("must provide a length-2 `interval`"),
  tol = .Machine$double.eps^0.25,
  maxiter = 1000L
)
```

Arguments

<code>y</code>	numeric vector of y values
<code>f</code>	monotone function $f(x)$ whose roots are to be solved
<code>interval</code>	length-2 numeric vector
<code>tol</code>	double scalar, desired accuracy, i.e., convergence tolerance
<code>maxiter</code>	integer scalar, maximum number of iterations

Details

Function `vuniroot2`, different from `vuniroot`, does

- accept `NA_real_` as element(s) of y
- handle the case when the analytic root is at lower and/or upper
- return a root of Inf (if $\text{abs}(f(\text{lower})) \geq \text{abs}(f(\text{upper}))$) or $-\text{Inf}$ (if $\text{abs}(f(\text{lower})) < \text{abs}(f(\text{upper}))$), when the function value $f(\text{lower})$ and $f(\text{upper})$ are not of opposite sign.

Value

Function `vuniroot2` returns a numeric vector x as the solution of $y = f(x)$ with given vector y .

Examples

```
library(rstpm2)

# ?rstpm2::vuniroot does not accept NA \eqn{y}
tryCatch(vuniroot(function(x) x^2 - c(NA, 2:9), lower = 1, upper = 3), error = identity)

# ?rstpm2::vuniroot not good when the analytic root is at `lower` or `upper`
f <- function(x) x^2 - 1:9
vuniroot(f, lower = .99, upper = 3.001) # good
tryCatch(vuniroot(f, lower = 1, upper = 3, extendInt = 'no'), warning = identity)
tryCatch(vuniroot(f, lower = 1, upper = 3, extendInt = 'yes'), warning = identity)
tryCatch(vuniroot(f, lower = 1, upper = 3, extendInt = 'downX'), error = identity)
tryCatch(vuniroot(f, lower = 1, upper = 3, extendInt = 'upX'), warning = identity)

vuniroot2(c(NA, 1:9), f = function(x) x^2, interval = c(1, 3)) # all good
```

z2GH

Tukey g -&- h Transformation

Description

To transform standard normal quantiles to Tukey g -&- h quantiles.

Usage

```
z2GH(z, A = 0, B = 1, g = 0, h = 0)
```

Arguments

`z` **double** scalar or **vector**, standard normal quantiles.
`A, B, g, h` **double** scalar or **vector**, parameters of Tukey g -&- h distribution

Details

Function `z2GH` transforms standard normal quantiles to Tukey g -&- h quantiles.

Value

Function `z2GH` returns a **double** scalar or **vector**.

Note

Function `gk::z2gh` is not fully vectorized, i.e., cannot take **vector** `z` and **vector** `A/B/g/h`, as of 2023-07-20 (package `gk` version 0.6.0)

Index

.GH2z, 3

all.equal, 5

character, 4

dGH, 6
dGH (TukeyGH), 5
double, 3–8

function, 7

GH2z, 3, 3

integer, 6, 7

length, 7
letterValue, 4, 4
logical, 6

matrix, 6

numeric, 7

pGH, 6
pGH (TukeyGH), 5

qGH, 6
qGH (TukeyGH), 5

rGH, 6
rGH (TukeyGH), 5

TukeyGH, 5
TukeyGH77 (TukeyGH77-package), 2
TukeyGH77-package, 2

vector, 3–8
vuniroot, 7
vuniroot2, 6, 6, 7

z2GH, 8, 8