Package ‘DistributionUtils’

August 28, 2023

Version 0.6-1
Date 2023-08-22
Title Distribution Utilities
Author David Scott <d.scott@auckland.ac.nz>
Maintainer David Scott <d.scott@auckland.ac.nz>
Depends R (>= 3.0.1)
Suggests GeneralizedHyperbolic, VarianceGamma, SkewHyperbolic, RUnit
Encoding UTF-8
Description Utilities are provided which are of use in the packages I have developed for dealing with distributions. Currently these packages are GeneralizedHyperbolic, VarianceGamma, and SkewHyperbolic and NormalLaplace. Each of these packages requires DistributionUtils. Functionality includes sample skewness and kurtosis, log-histogram, tail plots, moments by integration, changing the point about which a moment is calculated, functions for testing distributions using inversion tests and the Massart inequality. Also includes an implementation of the incomplete Bessel K function.
License GPL (>= 2)
NeedsCompilation yes
Repository CRAN
Date/Publication 2023-08-28 11:30:11 UTC

R topics documented:

  DistributionUtils-package .................................................. 2
  Bessel K Ratio ........................................................................ 3
  distCalcRange ................................................................. 4
  distIneqMassart ............................................................... 5
  distIneqMassartPlot .......................................................... 7
  distMode ................................................................. 8
  distStepSize ................................................................. 9
DistributionUtils-package

Utility functions useful for all distributions in packages following the standard approach developed in Scott, Würtz and Dong.

Description

Functionality includes sample skewness and kurtosis, log-histogram, tail plots, moments by integration, changing the point about which a moment is calculated, functions for testing distributions using inversion tests and the Massart inequality. Also includes an implementation of the incomplete Bessel K function.

Details

Contains functions which are useful for packages implementing distributions. Designed to work with my packages GeneralizedHyperbolic, VarianceGamma, SkewHyperbolic and Normal-Laplace.

Author(s)

David Scott <d.scott@auckland.ac.nz>
Maintainer: David Scott <d.scott@auckland.ac.nz>

References


See Also

GeneralizedHyperbolicDistribution
Description

Calculates the ratio of Bessel K functions of different orders, but the same value of the argument.

Usage

\[
\text{besselRatio}(x, \nu, \text{orderDiff}, \text{useExpScaled} = 700)
\]

Arguments

- **x**: Numeric, \( \geq 0 \). Value at which the numerator and denominator Bessel functions are evaluated.
- **nu**: Numeric. The order of the Bessel function in the denominator.
- **orderDiff**: Numeric. The order of the numerator Bessel function minus the order of the denominator Bessel function.
- **useExpScaled**: Numeric, \( \geq 0 \). The smallest value of \( x \) for which the ratio is calculated using the exponentially-scaled Bessel function values.

Details

Uses the function `besselK` to calculate the ratio of two modified Bessel function of the third kind whose orders are different. The calculation of Bessel functions will underflow if the value of \( x \) is greater than around 740. To avoid underflow the exponentially-scaled Bessel functions can be returned by `besselK`. The ratio is actually unaffected by exponential scaling since the scaling cancels across numerator and denominator.

The Bessel function ratio is useful in calculating moments of the generalized inverse Gaussian distribution, and hence also for the moments of the hyperbolic and generalized hyperbolic distributions.

Value

The ratio

\[
\frac{K_{\nu+k}(x)}{K_{\nu}(x)}
\]

of two modified Bessel functions of the third kind whose orders differ by \( k \).

Author(s)

David Scott <d.scott@auckland.ac.nz>

See Also

`besselK`, `gigMom`
Examples

```r
nus <- c(0:5, 10, 20)
x <- seq(1, 4, length.out = 11)
k <- 3

raw <- matrix(nrow = length(nus), ncol = length(x))
scaled <- matrix(nrow = length(nus), ncol = length(x))
compare <- matrix(nrow = length(nus), ncol = length(x))

for (i in 1:length(nus)){
  for (j in 1:length(x)) {
    raw[i,j] <- besselRatio(x[j], nus[i], orderDiff = k)
    scaled[i,j] <- besselRatio(x[j], nus[i], orderDiff = k, useExpScaled = 1)
    compare[i,j] <- raw[i,j]/scaled[i,j]
  }
}
raw
scaled
compare
```

---

**distCalcRange**

*Range of a Unimodal Distribution*

**Description**

Given the parameters of a unimodal distribution and the root of the density function name, this function determines the range outside of which the density function is negligible, to a specified tolerance.

**Usage**

```r
distCalcRange(densFn, param = NULL, tol = 10^(-5), ...)
```

**Arguments**

- **densFn**: Character. The name of the density function for which range calculation is required.
- **tol**: Tolerance.
- **param**: Numeric. A vector giving the parameter values for the distribution specified by densFn. If no param values are specified, then the default parameter values of each distribution are used instead.
- **...**: Passes arguments to `uniroot`. In particular, the parameters of the distribution.
distIneqMassart

Details
The name of the unimodal density function must be supplied as the characters of the root for that
density (e.g. norm, ghyp). The particular unimodal distribution being considered is specified by the
values of the parameters or of the param vector.
The function gives a range, outside of which the density is less than the given tolerance. It is used
in determining break points for the separate sections over which numerical integration is used to
determine the distribution function. The points are found by using unroot on the density function.

Value
A two-component vector giving the lower and upper ends of the range.

Author(s)
David Scott <d.scott@auckland.ac.nz>, Joyce Li <xli053@aucklanduni.ac.nz>

See Also
qDist

Examples

normRange <- distCalcRange("norm", tol = 10^(-7), mean = 4, sd = 1)
normRange
tRange <- distCalcRange("t", tol = 10^(-5), df = 4)
tRange

distIneqMassart Massart Inequality for Distributions

Description
This function implements a test of the random number generator and distribution function based on

Usage
distIneqMassart(densFn = "norm", n = 10000, probBound = 0.001, ...)

Arguments
densFn Character. The root name of the distribution to be tested.

n Numeric. The size of the sample to be used.
probBound Numeric. The value of the bound on the right hand side of the Massart inequality. See Details.

... Additional arguments to allow specification of the parameters of the distribution.
Details

Massart (1990) gave a version of the Dvoretsky-Kiefer-Wolfowitz inequality with the best possible constant:

\[
P \left( \sup_x |\hat{F}_n(x) - F(x)| > t \right) \leq 2 \exp(-2nt^2)
\]

where \( \hat{F}_n \) is the empirical distribution function for a sample of \( n \) independent and identically distributed random variables with distribution function \( F \). This inequality is true for all distribution functions, for all \( n \) and \( t \).

This test is used in base R to check the standard distribution functions. The code may be found in the file `p-r-random.tests.R` in the `tests` directory.

Value

- \( \sup \) Numerical. The supremum of the absolute difference between the empirical distribution and the true distribution function.
- \( \text{probBound} \) Numerical. The value of the bound on the right hand side of the Massart inequality.
- \( t \) Numerical. The lower bound which the supremum of the absolute difference between the empirical distribution and the true distribution function must exceed.
- \( \text{pVal} \) Numerical. The probability that the absolute difference between the empirical distribution and the true distribution function exceeds \( t \).
- \( \text{check} \) Logical. Indicates whether the inequality is satisfied or not.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

References


Examples

```r
## Normal distribution is the default
distIneqMassart()
## Specify parameter values
distIneqMassart(mean = 1, sd = 2)
## Gamma distribution has no default value for shape
distIneqMassart("gamma", shape = 1)
```
distIneqMassartPlot  Massart Inequality Plot Function

Description

Creates a Massart inequality plot for testing the empirical distribution and distribution function based on an inequality due to Massart (1990).

Usage

distIneqMassartPlot(densFn = "norm", param = NULL,
            nSamp = 50, n = 100, ...)

Arguments

densFn  Character. The root name of the distribution to be tested.
n      Numeric. The size of the sample to be used.
nSamp  Numeric. The number of samples used to approximate the LHS probability of the inequality.
param  Numeric. A vector giving the parameter values for the distribution specified by densFn. If no param values are specified, then the default parameter values of each distribution are used instead.
...    Passes the parameters of the distribution other than specified by param.

Details

Massart (1990) gave a version of the Dvoretsky-Kiefer-Wolfowitz inequality with the best possible constant:

\[ P\left(\sup_{x} |\hat{F}_n(x) - F(x)| > t\right) \leq 2 \exp(-2nt^2) \]

where \(\hat{F}_n\) is the empirical distribution function for a sample of \(n\) independent and identically distributed random variables with distribution function \(F\). This inequality is true for all distribution functions, for all \(n\) and \(t\).

The red curve in the plot shows the LHS probabilities and the black curve gives the RHS bound. The red curve should lie below the black curve in order that the empirical distribution represents a sample from the theoretical distribution.

Value

Returns NULL invisibly.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Xinxing Li <xli053@aucklanduni.ac.nz>
References


Examples

```r
## Not run:  
### Not run because of timing requirements of CRAN  
### The Massart Inequality plot for standard Normal Distribution  
distIneqMassartPlot()  
### The Massart Inequality plot for Gamma Distribution  
distIneqMassartPlot("gamma", shape = 1)  
## End(Not run)
```

---

**distMode**

*Mode of a Unimodal Distribution*

**Description**

Function to calculate the mode of a unimodal distribution which is specified by the root of the density function name and the corresponding parameters.

**Usage**

```r
distMode(densFn, param = NULL, ...)
```

**Arguments**

- `densFn` Character. The name of the density function for which the mode is required.
- `param` Numeric. A vector giving the parameter values for the distribution specified by `densFn`. If no `param` values are specified, then the default parameter values of each distribution are used instead.
- `...` Passes arguments to `optimize`. In particular, the parameters of the distribution.

**Details**

The name of the unimodal density function must be supplied as the characters of the root for that density (e.g. `norm`, `ghyp`). The particular unimodal distribution being considered is specified by the value of the argument `param`, or for base R distributions by specification in the ... arguments.

**Value**

The mode is found by a numerical optimization using `optimize`.
distStepSize

Author(s)

David Scott <d.scott@auckland.ac.nz>, Joyce Li <xli053@aucklanduni.ac.nz>

See Also
distStepSize, qDist.

Examples

normRange <- distCalcRange("norm", tol = 10^(-7), mean = 4, sd = 1)
curve(dnorm(x, mean = 4, sd = 1), normRange[1], normRange[2])
abline(v = distMode("norm", mean = 4, sd = 1), col = "blue")

Description

Given the parameters of a unimodal distribution and the root of the density function name, this function determines the step size when calculating the range of the specified unimodal distribution. The parameterization used is the one for the corresponding density function calculation.

Usage

distStepSize(densFn, dist, 
  param = NULL, side = c("right","left"), ...) 

Arguments

densFn Character. The name of the density function for which the step size needs to be calculated.
dist Numeric. Current distance value, for skew hyperbolic distribution only
param Numeric. A vector giving the parameter values for the distribution specified by densFn. If no param values are specified, then the default parameter values of each distribution are used instead.
side Character. "right" for a step to the right, "left" for a step to the left.
... Passes arguments in particular the parameters of the distribution to random sample generation function.

Details

This function is used for stepping to the right or the left to obtain an enclosing interval so uniroot can be used to search. The step size for the right tail is the absolute difference between the median and upper quantile and for the left tail is the absolute difference between the median and lower quantile. The skew hyperbolic distribution however needs a special step size. When the tail is...
declining exponentially the step is just a linear function of the current distance from the mode. If the
tail is declining only as a power of \(x\), an exponential step is used.

distStepSize is for internal use and is not expected to be called by users. It is documented here for
completeness.

**Value**

The size of the step.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>, Joyce Li <xli053@aucklanduni.ac.nz>

**See Also**

distCalcRange

**Examples**

```r
normRange <- distCalcRange("norm", tol = 10^(-7), mean = 4, sd = 1)
normRange
tRange <- distCalcRange("t", tol = 10^(-5), df = 4)
tRange
```

---

**incompleteBesselK**  
*The Incomplete Bessel K Function*

**Description**

Calculates the incomplete Bessel K function using the algorithm and code provided by Slavinsky
and Safouhi (2009).

**Usage**

```r
incompleteBesselK(x, y, nu, tol = .Machine$double.eps^0.85, nmax = 120)
incompleteBesselKR(x, y, nu, tol = .Machine$double.eps^0.85, nmax = 120)
SSFcoef(nmax, nu)
combinatorial(nu)
GDENOM(n, x, y, nu, An, nmax, Cnp)
GNUM(n, x, y, nu, Am, An, nmax, Cnp, GM, GN)
```

**Arguments**

- **x**  
  Numeric. Value of the first argument of the incomplete Bessel K function.

- **y**  
  Numeric. Value of the second argument of the incomplete Bessel K function.

- **nu**  
  Numeric. The order of the incomplete Bessel K function.
`incompleteBesselK`

`tol` Numeric. The tolerance for the difference between successive approximations of the incomplete Bessel K function.

`nmax` Integer. The maximum order allowed for the approximation of the incomplete Bessel K function.

`n` Integer. Current order of the approximation. Not required to be specified by users.

`An` Matrix of coefficients. Not required to be specified by users.

`Am` Matrix of coefficients. Not required to be specified by users.

`Cnp` Vector of elements of Pascal’s triangle. Not required to be specified by users.

`GN` Vector of denominators used for approximation. Not required to be specified by users.

`GM` Vector of numerators used for approximation. Not required to be specified by users.

**Details**

The function `incompleteBesselK` implements the algorithm proposed by Slavinsky and Safouhi (2010) and uses code provided by them.

The incomplete Bessel K function is defined by

\[ K_\nu(x, y) = \int_1^\infty t^{-\nu-1} \exp(-xt - y/t) \, dt \]

see Slavinsky and Safouhi (2010), or Harris (2008).

`incompleteBesselK` calls a Fortran routine to carry out the calculations. `incompleteBesselKR()` is a pure R version of the routine for computing the incomplete Bessel K function.

The functions `SSFcoef`, `combinatorial`, `GDENOM`, and `GNUM` are “subroutines”, i.e., auxiliary functions used in `incompleteBesselKR()`. They are not expected to be called by the user.

The approximation to the incomplete Bessel K function returned by `incompleteBesselK` is highly accurate. The default value of `tol` is about 10^-14 on a 32-bit computer. It appears that even higher accuracy is possible when \( x > y \). Then the tolerance can be taken as `Machine$double.eps` and the number of correct figures essentially coincides with the number of figures representable in the machine being used.

`incompleteBesselKR` is very slow compared to the Fortran version and is only included for those who wish to see the algorithm in R rather than Fortran.

**Value**

`incompleteBesselK` and `incompleteBesselKR` both return an approximation to the incomplete Bessel K function as defined above.

**Note**

The problem of calculation of the incomplete Bessel K function is equivalent to the problem of calculation of the cumulative distribution function of the generalized inverse Gaussian distribution. See Generalized Inverse Gaussian.
incompleteBesselK

Author(s)
David Scott <d.scott@auckland.ac.nz>, Thomas Tran, Richard Slevinsky, Hassan Safouhi.

References
Harris, Frank E. (2008) Incomplete Bessel, generalized incomplete gamma, or leaky aquifer func-


and accurate computation of incomplete Bessel functions. Applied Numerical Mathematics 60(12),
1411–1417.

See Also
besselK

Examples
### Harris (2008) gives accurate values (16 figures) for
### x = 0.01, y = 4, and nu = 0:9
### nu = 0, Harris value is 2.22531 07612 66469
options(digits = 16)
incompleteBesselK(0.01, 4, 0)
### nu = 9, Harris value is 0.00324 67980 03149
incompleteBesselK(0.01, 4, 9)

### Other values given in Harris (2008)
### x = 4.95, y = 5.00, nu = 2
incompleteBesselK(4.95, 5, 2) ## 0.00001 22499 87981
### x = 10, y = 2, nu = 6
### Slevinsky and Safouhi (2010) suggest Harris (2008) value
### is incorrect, give value 0.00000 04150 01064 21228
incompleteBesselK(10, 2, 6)
### x = 3.1, y = 2.6, nu = 5
incompleteBesselK(3.1, 2.6, 5) ## 0.00052 85043 25244

### Check values when x > y using numeric integration
(numIBF <- sapply(0:9, incompleteBesselK, x = 4, y = 0.01))

besselFn <- function(t, x, y, nu) {
  (t^(-nu - 1))*exp(-x*t - y/t)
}

(intIBF <- sapply(0:9, integrate, f = besselFn, lower = 1, upper = Inf,
  x = 4, y = 0.01))
intIBF <- as.numeric(intIBF[1, ])
numIBF - intIBF
max(abs(numIBF - intIBF)) ## 1.256649992398273e-11

options(digits = 7)
**integrateDens**

Integrates a Density Function

**Description**

Given a density function specified by the root of the density function name, returns the integral over a specified range, usually the whole real line. Used for checking that the integral over the whole real line is 1.

**Usage**

```r
tegrateDens(densFn = "norm", lower = -Inf, upper = Inf,
             subdivisions = 100, ...)
```

**Arguments**

- `densFn` Character. The name of the density function to be integrated.
- `lower` Numeric. The lower limit of the integration. Default is -Inf.
- `upper` Numeric. The upper limit of the integration. Default is Inf.
- `subdivisions` Numeric. The number of subdivisions to be passed to integrate.
- `...` Additional arguments to be passed to integrate. In particular, the parameters of the distribution.

**Details**

The name of the density function to be integrated must be supplied as the characters of the root for that density (e.g. `norm`, `gamma`). The density function specified is integrated numerically over the range specified via a call to `integrate`. The parameters of the distribution can be specified, otherwise the default parameters will be used.

**Value**

A list of class `integrate` with components:

- `value` The final estimate of the integral.
- `abs.error` Estimate of the modulus of the absolute error.
- `subdivisions` The number of subintervals produced in the subdivision process.
- `message` OK or a character string giving the error message.
- `call` The matched call to the integrate function.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>
See Also

momIntegrated

Examples

integrateDens("norm", mean = 1, sd = 1)
integrateDens("t", df = 4)
integrateDens("exp", rate = 2)
integrateDens("weibull", shape = 1)

inversionTests  Inversion Tests for Distributions

Description

Functions to check performance of distribution and quantile functions. Applying the distribution function followed by the quantile function to a set of numbers should reproduce the original set of numbers. Likewise applying the quantile function followed by the distribution function to numbers in the range (0,1) should produce the original numbers.

Usage

inversionTestpq(densFn = "norm", n = 10,
intTol = .Machine$double.eps^0.25,
uniTol = intTol, x = NULL, method = "spline", ...)
inversionTestqp(densFn = "norm",
qs = c(0.001, 0.01, 0.025, 0.05, 0.1, 0.2, 0.4, 0.5,
      0.6, 0.8, 0.9, 0.95, 0.975, 0.99, 0.999),
intTol = .Machine$double.eps^0.25,
uniTol = uniTol, method = "spline", ...)

Arguments

densFn  Character. The root name of the distribution to be tested.
qs  Numeric. Set of quantiles to which quantile function then distribution function will be applied. See Details.
n  Numeric. Number of values to be sampled from the distribution. See Details.
x  Numeric. Values at which the distribution function is to be evaluated. If NULL values are drawn at random from the distribution.
intTol  Value of rel.tol and hence abs.tol in calls to integrate. See integrate.
uniTol  Value of tol in calls to uniroot. See uniroot.
method  Character. If "spline" quantiles are found from a spline approximation to the distribution function. If "integrate", the distribution function used is always obtained by integration.
...  Additional arguments to allow specification of the parameters of the distribution.
inversionTests

Details

inversionTestpq takes a sample from the specified distribution of size \( n \) then applies the distribution function, followed by the quantile function. inversionTestqp applies the quantile function, followed by the distribution function to the set of quantiles specified by \( q_s \).

In both cases the starting and ending values should be the same.

These tests are used in base R to check the standard distribution functions. The code may be found in the file d-p-q-r.tests.R in the tests directory.

Value

inversionTestpq returns a list with components:

- \( q_{px} \): Numeric. The result of applying the distribution function ('p' function) then the quantile function ('q' function) to the randomly generated set of \( x \) values.
- \( x \): Numeric. The set of \( x \) values generated by the 'r' function.
- \( diffs \): Numeric. The differences \( q_{px} \) minus \( x \).
- \( n \): Numeric. Number of values sampled from the distribution.

inversionTestqp returns a list with components:

- \( pq_{qs} \): Numeric. The result of applying the quantile function ('q' function) then the distribution function ('p' function) to the quantiles \( q_s \).
- \( q_s \): Numeric. The set of quantiles.
- \( diffs \): Numeric. The differences \( pq_{qs} \) minus \( q_s \).

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

Examples

```r
## Default distribution is normal
inversionTestpq()
inversionTestqp()
## Supply parameters
inversionTestpq(mean = 1, sd = 2)
inversionTestqp(mean = 1, sd = 2)
## Gamma distribution, must specify shape
inversionTestpq("gamma", shape = 1)
inversionTestqp("gamma", shape = 1)
```
is.wholenumber

Is Object Numeric and Whole Numbers

Description

Checks whether an object is numeric and if so, are all the elements whole numbers, to a given
tolerance.

Usage

is.wholenumber(x, tolerance = .Machine$double.eps^0.5)

Arguments

x The object to be tested.

tolerance Numeric ≥ 0. Absolute differences greater than tolerance are treated as real
differences.

Details

The object x is first tested to see if it is numeric. If not the function returns 'FALSE'. Then if all the
elements of x are whole numbers to within the tolerance given by tolerance the function returns
'TRUE'. If not it returns 'FALSE'.

Value

Either 'TRUE' or 'FALSE' depending on the result of the test.

Author(s)

David Scott <d.scott@auckland.ac.nz>.

References

Based on a post by Tony Plate <tplate@acm.org> on R-help.

Examples

    is.wholenumber(-3:5)             # TRUE
    is.wholenumber(c(0,0.1,1.3,5))   # FALSE
    is.wholenumber(-3:5 + .Machine$double.eps)  # TRUE
    is.wholenumber(-3:5 + .Machine$double.eps^0.5)  # FALSE
    is.wholenumber(c(2L,3L))         # TRUE
    is.wholenumber(c("2L","3L"))  # FALSE
    is.wholenumber(0i ^ (-3:3))     # FALSE
    is.wholenumber(matrix(1:6, nrow = 3))  # TRUE
    is.wholenumber(list(-1:3,2:6))   # FALSE
    is.numeric(list(-1:3,2:6))      # FALSE
    is.wholenumber(unlist(list(-1:3,2:6)))  # TRUE
Description

Plots a log-histogram, as in for example Feiller, Flenley and Olbricht (1992).

The intended use of the log-histogram is to examine the fit of a particular density to a set of data, as an alternative to a histogram with a density curve. For this reason, only the log-density histogram is implemented, and it is not possible to obtain a log-frequency histogram.

The log-histogram can be plotted with histogram-like dashed vertical bars, or as points marking the tops of the log-histogram bars, or with both bars and points.

Usage

logHist(x, breaks = "Sturges",
       include.lowest = TRUE, right = TRUE,
       main = paste("Log-Histogram of", xName),
       xlim = range(breaks), ylim = NULL, xlab = xName,
       ylab = "Log-density", nclass = NULL, htype = "b", ...)

Arguments

\textbf{x} \hspace{1cm} \text{A vector of values for which the log-histogram is desired.}

\textbf{breaks} \hspace{1cm} \text{One of:}

\begin{itemize}
  \item a vector giving the breakpoints between log-histogram cells;
  \item a single number giving the number of cells for the log-histogram;
  \item a character string naming an algorithm to compute the number of cells (see \textbf{Details});
    \begin{itemize}
      \item a function to compute the number of cells.
    \end{itemize}
\end{itemize}

In the last three cases the number is a suggestion only.

\textbf{include.lowest} \hspace{1cm} \text{Logical. If TRUE, an 'x[i]' equal to the 'breaks' value will be included in the first (or last, for right = FALSE) bar.}

\textbf{right} \hspace{1cm} \text{Logical. If TRUE, the log-histograms cells are right-closed (left open) intervals.}

\textbf{main, xlab, ylab} \hspace{1cm} \text{These arguments to \textbf{title} have useful defaults here.}

\textbf{xlim} \hspace{1cm} \text{Sensible default for the range of x values.}

\textbf{ylim} \hspace{1cm} \text{Calculated by logHist, see \textbf{Details}.}

\textbf{nclass} \hspace{1cm} \text{Numeric (integer). For compatibility with hist only, nclass is equivalent to breaks for a scalar or character argument.}

\textbf{htype} \hspace{1cm} \text{Type of histogram. Possible types are:}

\begin{itemize}
  \item "h" for a *h*istogram only;
  \item "p" for *p*oints marking the top of the histogram bars only;
  \item "b" for *b*oth.
\end{itemize}

\ldots \hspace{1cm} \text{Further graphical parameters for calls to \textbf{plot} and \textbf{points}.}
Details

Uses `hist.default` to determine the cells or classes and calculate counts.

To calculate `ylim` the following procedure is used. The upper end of the range is given by the maximum value of the log-density, plus 25% of the absolute value of the maximum. The lower end of the range is given by the smallest (finite) value of the log-density, less 25% of the difference between the largest and smallest (finite) values of the log-density.

A log-histogram in the form used by Feiller, Flenley and Olbricht (1992) is plotted. See also Barndorff-Nielsen (1977) for use of log-histograms.

Value

Returns a list with components:

- `breaks`: The $n + 1$ cell boundaries (= `breaks` if that was a vector).
- `counts`: $n$ integers; for each cell, the number of $x[]$ inside.
- `logDensity`: Log of $\hat{f}(x_i)$, which are estimated density values.
  
  If all(diff(breaks) == 1), estimated density values are the relative frequencies $counts/n$ and in general satisfy $\sum_i \hat{f}(x_i)(b_{i+1} - b_i) = 1$, where $b_i = breaks[i]$.
- `mids`: The $n$ cell midpoints.
- `xName`: A character string with the actual $x$ argument name.
- `heights`: The location of the tops of the vertical segments used in drawing the log-histogram.
- `ylim`: The value of `ylim` calculated by `logHist`.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Richard Trendall, Thomas Tran

References


See Also

`hist`
Examples

```r
x <- rnorm(200)
hist(x)
### default
logHist(x)
### log histogram only
logHist(x, htype = "h")
### points only, some options
logHist(x, htype = "p", pch = 20, cex = 2, col = "steelblue")
```

momChangeAbout

**Obtain Moments About a New Location**

Description

Using the moments up to a given order about one location, this function either returns the moments up to that given order about a new location as a vector or it returns a moment of a specific order defined by users (order <= maximum order of the given moments) about a new location as a single number. A generalization of using raw moments to obtain a central moment or using central moments to obtain a raw moment.

Usage

```r
momChangeAbout(order = "all", oldMom, oldAbout, newAbout)
```

Arguments

- **order**: One of:
  - the character string "all", the default;
  - a positive integer less than the maximum order of `oldMom`.
- **oldMom**: Numeric. Moments of orders 1, 2, ..., about the point `oldAbout`.
- **oldAbout**: Numeric. The point about which the moments `oldMom` have been calculated.
- **newAbout**: Numeric. The point about which the desired moment or moments are to be obtained.

Details

Suppose \( m_k \) denotes the \( k \)-th moment of a random variable \( X \) about a point \( a \), and \( m_k^* \) denotes the \( k \)-th moment about \( b \). Then \( m_k^* \) may be determined from the moments \( m_1, m_2, \ldots, m_k \) according to the formula

\[
m_k^* = \sum_{i=0}^{k} (a - b)^i m^{k-i}
\]

This is the formula implemented by the function `momChangeAbout`. It is a generalization of the well-known formulae used to change raw moments to central moments or to change central moments to raw moments. See for example Kendall and Stuart (1989), Chapter 3.
momIntegrated

Moments Using Integration

Description

Calculates moments and absolute moments about a given location for any given distribution.

Value

The moment of order order about the location newAbout when order is specified. The vector of moments about the location newAbout from first order up to the maximum order of the oldMom when order takes the value "all" or is not specified.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

References


Examples

### Gamma distribution

```r
k <- 4
shape <- 2
old <- 0
new <- 1
sampSize <- 100000

### Calculate 1st to 4th raw moments
m <- numeric(k)
for (i in 1:k){
  m[i] <- gamma(shape + i)/gamma(shape)
}
m

### Calculate 4th moment about new
momChangeAbout(k, m, old, new)

### Calculate 3rd about new
momChangeAbout(3, m, old, new)

### Calculate 1st to 4th moments about new
momChangeAbout(oldMom = m, oldAbout = old, newAbout = new)

### Approximate kth moment about new using sampling
x <- rgamma(sampSize, shape)
mean((x - new)^k)
```
momIntegrated

Usage

momIntegrated(densFn = "ghyp", param = NULL, order, about = 0,
              absolute = FALSE, ...)

Arguments

densFn Character. The name of the density function whose moments are to be calculated. See Details.

param Numeric. A vector giving the parameter values for the distribution specified by densFn. If no param values are specified, then the default parameter values of the distribution are used instead.

order Numeric. The order of the moment or absolute moment to be calculated.

about Numeric. The point about which the moment is to be calculated.

absolute Logical. Whether absolute moments or ordinary moments are to be calculated. Default is FALSE.

... Passes arguments to integrate. In particular, the parameters of the distribution.

Details

Denote the density function by \( f \). Then if order = \( k \) and about = \( a \), momIntegrated calculates

\[
\int_{-\infty}^{\infty} (x - a)^k f(x) \, dx
\]

when absolute = FALSE and

\[
\int_{-\infty}^{\infty} |x - a|^k f(x) \, dx
\]

when absolute = TRUE.

The name of the density function must be supplied as the characters of the root for that density (e.g. norm, ghyp).

When densFn="ghyp", densFn="hyperb", densFn="gig" or densFn = "vg", the relevant package must be loaded or an error will result.

When densFn="invgamma" or "inverse gamma" the density used is the density of the inverse gamma distribution given by

\[
f(x) = \frac{\alpha^\alpha}{\Gamma(\alpha)} e^{-u} x^{-\alpha}, \quad u = \theta/x
\]

for \( x > 0, \alpha > 0 \) and \( \theta > 0 \). The parameter vector param = c(shape, rate) where shape = \( \alpha \) and rate = \( 1/\theta \). The default value for param is c(-1, 1).

Value

The value of the integral as specified in Details.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>, Xinxing Li <xli053@aucklanduni.ac.nz>
See Also
dghyp, dhyperb, dgamma, dgig, VarianceGamma

Examples

```r
require(GeneralizedHyperbolic)
### Calculate the mean of a generalized hyperbolic distribution
### Compare the use of integration and the formula for the mean
m1 <- momIntegrated("ghyp", param = c(0, 1, 3, 1, 1 / 2), order = 1, about = 0)
m1
ghypMean(param = c(0, 1, 3, 1, 1 / 2))
### The first moment about the mean should be zero
momIntegrated("ghyp", order = 1, param = c(0, 1, 3, 1, 1 / 2), about = m1)
### The variance can be calculated from the raw moments
m2 <- momIntegrated("ghyp", order = 2, param = c(0, 1, 3, 1, 1 / 2), about = 0)
m2
m2 - m1^2
### Compare with direct calculation using integration
momIntegrated("ghyp", order = 2, param = c(0, 1, 3, 1, 1 / 2), about = m1)
momIntegrated("ghyp", param = c(0, 1, 3, 1, 1 / 2), order = 2,
about = m1)
### Compare with use of the formula for the variance
ghypVar(param = c(0, 1, 3, 1, 1 / 2))
```
**Value**

The approximate standard error of the sample moment specified.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>

**References**


**See Also**

momChangeAbout

**Examples**

```r
### Moments of the normal distribution, mean 1, variance 4
mu <- 1
sigma <- 2
mom <- c(0,sigma^2,0,3*sigma^4,0,15*sigma^6,0,105*sigma^8)
### standard error of sample variance
momSE(2, 100, mom[1:4])
### should be
sqrt(2*sigma^4)/10
### standard error of sample central third moment
momSE(3, 100, mom[1:6])
### should be
sqrt(6*sigma^6)/10
### standard error of sample central fourth moment
momSE(4, 100, mom)
### should be
sqrt(96*sigma^8)/10
```

---

**moranTest**

*Moran's Log Spacings Test*

**Description**

This function implements a goodness-of-fit test using Moran’s log spacings statistic.

**Usage**

```r
moranTest(x, densFn, param = NULL, ...)
```
Arguments

- **densFn** Character. The root name of the distribution to be tested.
- **x** Numeric. Vector of data to be tested.
- **param** Numeric. A vector giving the parameter values for the distribution specified by **densFn**. If no **param** values are specified, then the default parameter values of the distribution are used instead.
- **...** Additional arguments to allow specification of the parameters of the distribution other than specified by **param**.

Details

Moran (1951) gave a statistic for testing the goodness-of-fit of a random sample of $x$-values to a continuous univariate distribution with cumulative distribution function $F(x, \theta)$, where $\theta$ is a vector of known parameters. This function implements the Cheng and Stephens (1989) extended Moran test for unknown parameters.

The test statistic is

$$T(\hat{\theta}) = (M(\hat{\theta}) + 1/2k - C_1)/C_2$$

Where $M(\theta)$, the Moran statistic, is

$$M(\theta) = -(\log(y_1 - y_0) + \log(y_2 - y_1) + \ldots + \log(y_m - y_{m-1}))$$

This test has null hypothesis: $H_0$: a random sample of $n$ values of $x$ comes from distribution $F(x, \theta)$, where $\theta$ is the vector of parameters. Here $\theta$ is expected to be the maximum likelihood estimate $\hat{\theta}$, an efficient estimate. The test rejects $H_0$ at significance level $\alpha$ if $T(\hat{\theta}) > \chi^2_n(\alpha)$.

Value

- **statistic** Numeric. The value of the Moran test statistic.
- **estimate** Numeric. A vector of parameter estimates for the tested distribution.
- **parameter** Numeric. The degrees of freedom for the Moran statistic.
- **p.value** Numeric. The p-value for the test.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Xinxing Li <xli053@aucklanduni.ac.nz>

References


Examples

### Normal Distribution
```r
x <- rnorm(100, mean = 0, sd = 1)
muhat <- mean(x)
sigmahat <- sqrt(var(x)*(100 - 1)/100)
result <- moranTest(x, "norm", mean = muhat, sd = sigmahat)
result
```

### Exponential Distribution
```r
y <- rexp(200, rate = 3)
lambdahat <- 1/mean(y)
result <- moranTest(y, "exp", rate = lambdahat)
result
```

---

**pDist**

*Distribution and Quantile Functions for Unimodal Distributions*

**Description**

Given the density function of a unimodal distribution specified by the root of the density function name, returns the distribution function and quantile function of the specified distribution.

**Usage**

```r
pDist(densFn = "norm", q, param = NULL, subdivisions = 100, lower.tail = TRUE, intTol = .Machine$double.eps^0.25, valueOnly = TRUE, ...)
qDist(densFn = "norm", p, param = NULL, lower.tail = TRUE, method = "spline", nInterpol = 501, uniTol = .Machine$double.eps^0.25, subdivisions = 100, intTol = uniTol, ...)
```

**Arguments**

- **densFn**: Character. The name of the density function for which the distribution function or quantile function is required.
- **q**: Vector of quantiles.
- **p**: Vector of probabilities.
- **param**: Numeric. A vector giving the parameter values for the distribution specified by densFn. If no param values are specified, then the default parameter values of each distribution are used instead.
- **method**: Character. If "spline" quantiles are found from a spline approximation to the distribution function. If "integrate", the distribution function used is always obtained by integration.
- **lower.tail**: Logical. If lower.tail = TRUE, the cumulative density is taken from the lower tail.
subdivisions  The maximum number of subdivisions used to integrate the density and determine the accuracy of the distribution function calculation.

intTol Value of rel.tol and hence abs.tol in calls to integrate. See integrate.

valueOnly Logical. If valueOnly = TRUE calls to pDist only return the value obtained for the integral. If valueOnly = FALSE an estimate of the accuracy of the numerical integration is also returned.

nInterpol Number of points used in qDist for cubic spline interpolation of the distribution function.

uniTol Value of tol in calls to uniroot. See uniroot.

... Passes additional arguments to integrate, distMode or distCalcRange. In particular, the parameters of the distribution.

Details

The name of the unimodal density function must be supplied as the characters of the root for that density (e.g. norm, ghyp).

pDist uses the function integrate to numerically integrate the density function specified. The integration is from -Inf to x if x is to the left of the mode, and from x to Inf if x is to the right of the mode. The probability calculated this way is subtracted from 1 if required. Integration in this manner appears to make calculation of the quantile function more stable in extreme cases.

qDist provides two methods to calculate quantiles both of which use uniroot to find the value of x for which a given q is equal to F(x) where F(.) denotes the distribution function. The difference is in how the numerical approximation to F is obtained. The more accurate method, which is specified as "integrate", is to calculate the value of F(x) whenever it is required using a call to pDist. It is clear that the time required for this approach is roughly linear in the number of quantiles being calculated. The alternative (and default) method is that for the major part of the distribution a spline approximation to F(x) is calculated and quantiles found using uniroot with this approximation. For extreme values of some heavy-tailed distributions (where the tail probability is less than $10^{-7}$), the integration method is still used even when the method specified as "spline".

If accurate probabilities or quantiles are required, tolerances (intTol and uniTol) should be set to small values, i.e $10^{-10}$ or $10^{-12}$ with method = "integrate". Generally then accuracy might be expected to be at least $10^{-9}$. If the default values of the functions are used, accuracy can only be expected to be around $10^{-4}$. Note that on 32-bit systems .Machine$double.eps^0.25 = 0.0001220703 is a typical value.

Value

pDist gives the distribution function, qDist gives the quantile function.

An estimate of the accuracy of the approximation to the distribution function can be found by setting valueOnly = FALSE in the call to pDist which returns a list with components value and error.

Author(s)

David Scott <d.scott@auckland.ac.nz> Joyce Li <xli053@aucklanduni.ac.nz>
safeIntegrate

Examples

pDist("norm", q = 2, mean = 1, sd = 1)
pDist("t", q = 0.5, df = 4)
require(GeneralizedHyperbolic)
pDist("ghyp", q = 0.1)
require(SkewHyperbolic)
qDist("skewhyp", p = 0.4, param = c(0, 1, 0, 10))
qDist("t", p = 0.2, df = 4)

Description

Adaptive quadrature of functions of one variable over a finite or infinite interval.

Usage

safeIntegrate(f, lower, upper, subdivisions=100,
rel.tol = .Machine$double.eps^0.25, abs.tol = rel.tol,
stop.on.error = TRUE, keep.xy = FALSE, aux = NULL, ...)

Arguments

f An R function taking a numeric first argument and returning a numeric vector of the same length. Returning a non-finite element will generate an error.
lower, upper The limits of integration. Can be infinite.
subdivisions The maximum number of subintervals.
rel.tol Relative accuracy requested.
abs.tol Absolute accuracy requested.
stop.on.error Logical. If true (the default) an error stops the function. If false some errors will give a result with a warning in the message component.
keep.xy Unused. For compatibility with S.
aux Unused. For compatibility with S.
... Additional arguments to be passed to f. Remember to use argument names not matching those of safeIntegrate(.)!

Details

This function is just a wrapper around integrate to check for equality of upper and lower. A check is made using all.equal. When numerical equality is detected, if lower (and hence upper) is infinite, the value of the integral and the absolute error are both set to 0. When lower is finite, the value of the integral is set to 0, and the absolute error to the average of the function values at upper and lower times the difference between upper and lower.

When upper and lower are determined to be different, the result is exactly as given by integrate.
Value

A list of class "integrate" with components:

- value: The final estimate of the integral.
- abs.error: Estimate of the modulus of the absolute error.
- subdivisions: The number of subintervals produced in the subdivision process.
- message: "OK" or a character string giving the error message.
- call: The matched call.

See Also

The function `integrate` and `all.equal`.

Examples

```r
integrate(dnorm, -1.96, 1.96)
safeIntegrate(dnorm, -1.96, 1.96) # Same as for integrate()
integrate(dnorm, -Inf, Inf)
safeIntegrate(dnorm, -Inf, Inf) # Same as for integrate()
integrate(dnorm, 1.96, 1.96) # OK here but can give an error
safeIntegrate(dnorm, 1.96, 1.96)
safeIntegrate(dnorm, -Inf, -Inf) # Avoids nonsense answer
integrate(dnorm, Inf, Inf)
safeIntegrate(dnorm, Inf, Inf) # Avoids nonsense answer
```

---

### Sample Moments

#### Sample Skewness and Kurtosis

Description

Computes the sample skewness and sample kurtosis.

Usage

```r
skewness(x, na.rm = FALSE)
kurtosis(x, na.rm = FALSE)
```

Arguments

- `x`: A numeric vector containing the values whose skewness or kurtosis is to be computed.
- `na.rm`: A logical value indicating whether NA values should be stripped before the computation proceeds.
Details

If \( N = \text{length}(x) \), then the skewness of \( x \) is defined as

\[
N^{-1} \text{sd}(x)^{-3} \sum_i (x_i - \text{mean}(x))^3.
\]

If \( N = \text{length}(x) \), then the kurtosis of \( x \) is defined as

\[
N^{-1} \text{sd}(x)^{-4} \sum_i (x_i - \text{mean}(x))^4 - 3.
\]

Value

The skewness or kurtosis of \( x \).

Note

These functions and the description of them are taken from the package \texttt{e1071}. They are included to avoid having to require an additional package.

Author(s)

Evgenia Dimitriadou, Kurt Hornik, Friedrich Leisch, David Meyer, and Andreas Weingessel

Examples

\begin{verbatim}
x <- rnorm(100)
skewness(x)
kurtosis(x)
\end{verbatim}
Arguments

x  A vector of values for which the tail plot is to be drawn.
log  A character string which contains "x" if the x-axis is to be logarithmic, "y" if the y-axis is to be logarithmic and "xy" or "yx" if both axes are to be logarithmic.
side  Character. "right" (the default) for a tail plot of the right-hand tail, "left" for a tail plot of the left-hand tail.
main  A main title for the plot.
xlab  A label for the x axis, defaults to NULL.
ylab  A label for the y axis, defaults to NULL.
distrFn  Character. The name of the distribution function to be added to the tail plot.
param  Vector specifying the parameters of the distribution, defaults to NULL.
mean  The mean of the normal distribution.
sd  The standard deviation of the normal distribution. Must be positive.
df  The degrees of freedom of the t-distribution, (> 0, may be non-integer). Defaults to Inf, corresponding to the standard normal distribution.
shape  The shape parameter of the gamma distribution. Must be positive.
scale  The scale parameter of the gamma distribution. Must be strictly positive, scale strictly.
rate  The rate parameter of the gamma distribution. An alternative way to specify the scale.
...  Other graphical parameters (see par).

Details
tailPlot draws either a left-hand or right-hand tail plot of the data x. See for example Resnick (2007), p.105. The left-hand tail plot plots the empirical distribution of the data against the order statistics, for order statistic values below the median. The right-hand tail plot plots one minus the empirical distribution of the data against the order statistics, for order statistic values above the median. The default is for the y-axis to be plotted on a log scale.
tailPlotLine adds a line for the specified distribution to an already drawn tail plot. The distribution can be any distribution which has default parameters, but if parameters need to be supplied the distribution must have an argument param which specifies the parameters. This is the case for all distributions in the form recommended in Scott et al (2009) and includes distributions from the packages GeneralizedHyperbolic, SkewHyperbolic, VarianceGamma and NormalLaplace (which is on R-Forge).

normTailPlotLine,tTailPlotLine and gammaTailPlotLine add the corresponding line derived respectively from the given normal, t, or gamma distribution to an already drawn tail plot.

Value

Returns NULL invisibly.
Author(s)

David Scott <d.scott@auckland.ac.nz>

References


Examples

```r
### Draw tail plot of some data
x <- rnorm(100, 1, 2)
tailPlot(x)
### Add normal distribution line
normTailPlotLine(x, mean = 1, sd = 2)
### Add t distribution line
tTailPlotLine(x, df = 5, lty = 2)
### Use fitted values
normTailPlotLine(x, mean = mean(x), sd = sd(x), lty = 3)

### Gamma distribution
x <- rgamma(100, shape = 1, scale = 1)
tailPlot(x)
### Add gamma distribution line
gammaTailPlotLine(x, shape = 1, scale = 1)
### Left tail example
tailPlot(x, side = "l")
### Add gamma distribution line
gammaTailPlotLine(x, shape = 1, scale = 1, side = "l")
### Log scale on both axes
tailPlot(x, side = "l", log = "xy")
### Add gamma distribution line
gammaTailPlotLine(x, shape = 1, scale = 1, side = "l")

### Add line from a standard distribution with default parameters
x <- rlnorm(100)
tailPlot(x)
tailPlotLine(x, distrFn = "lnorm")

### Add line from a distribution with 'param' argument
require(VarianceGamma)
param <- c(0, 0.5, 0, 0.5)
x <- rvg(100, param = param)
tailPlot(x)
tailPlotLine(x, distrFn = "vg", param = param)
```
tsHessian

Calculate Two-Sided Hessian Approximation

Description

Calculates an approximation to the Hessian of a function. Used for obtaining an approximation to
the information matrix for maximum likelihood estimation.

Usage

tsHessian(param, fun, ...)

Arguments

param Numeric. The Hessian is to be evaluated at this point.
fun A function of the parameters specified by param, and possibly other parameters.
... Values of other parameters of the function fun if required.

Details

As a typical statistical application, the function fun is the log-likelihood function, param specifies
the maximum likelihood estimates of the parameters of the distribution, and the data constitutes the
other parameter values required for determination of the log-likelihood function.

Value

The approximate Hessian matrix of the function fun where differentiation is with respect to the
vector of parameters param at the point given by the vector param.

Note

This code was borrowed from the fBasics function, in the file ‘utils-hessian.R’ with slight mod-
ification. This was in turn borrowed from Kevin Sheppard’s Matlab garch toolbox as implemented
by Alexios Ghalanos in his rgarch package.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

See Also

hyperbHessian and summary.hyperbFit in GeneralizedHyperbolic.
Examples

### Consider Hessian of log(1 + x + 2y)
### Example from Lang: A Second Course in Calculus, p.74
fun <- function(param){
  x <- param[1]
  y <- param[2]
  return(log(1 + x + 2*y))
}

### True value of Hessian at (0,0)
trueHessian <- matrix( c(-1,-2,
                           -2,-4), byrow = 2, nrow = 2)
trueHessian

### Value from tsHessian
approxHessian <- tsHessian(c(0,0), fun = fun)
approxHessian
maxDiff <- max(abs(trueHessian - approxHessian))
### Should be approximately 0.045
maxDiff
Index

* classes
  is.wholenumber, 16
* distribution
  distCalcRange, 4
  distIneqMassart, 5
  distIneqMassartPlot, 7
  distMode, 8
  distStepSize, 9
  incompleteBesselK, 10
  integrateDens, 13
  inversionTests, 14
  logHist, 17
  momChangeAbout, 19
  momIntegrated, 20
  momSE, 22
  moranTest, 23
  pDist, 25
  tailPlot, 29
* hplot
  logHist, 17
* math
  Bessel K Ratio, 3
  incompleteBesselK, 10
  integrateDens, 13
  inversionTests, 14
  logHist, 17
  momChangeAbout, 19
  momIntegrated, 20
  momSE, 22
  moranTest, 23
  pDist, 25
  tailPlot, 29
* package
  DistributionUtils-package, 2
  DistributionUtils-package, 2
* univariate
  distIneqMassart, 5
  inversionTests, 14
  moranTest, 23
* univar
  distCalcRange, 4
  distIneqMassartPlot, 7
  distMode, 8
  distStepSize, 9
  integrateDens, 13
  momChangeAbout, 19
  momIntegrated, 20
  momSE, 22
  pDist, 25
  Sample Moments, 28
  tailPlot, 29
* utilities
  safeIntegrate, 27
all.equal, 27, 28
Bessel K Ratio, 3
besselK, 3, 12
besselRatio(Bessel K Ratio), 3
combinatorial(incompleteBesselK), 10
dgamma, 22
dghyp, 22
dgig, 22
dhyperb, 22
distCalcRange, 4, 10, 26
distIneqMassart, 5
distIneqMassartPlot, 7
distMode, 8, 26
DistributionUtils
  (DistributionUtils-package), 2
DistributionUtils-package, 2
distStepSize, 9, 9
gammaTailPlotLine(tailPlot), 29
GDENOM(incompleteBesselK), 10
GeneralizedHyperbolicDistribution, 2
gigMom, 3
GNUm(incompleteBesselK), 10
hist, 18
hist.default, 18
incompleteBesselK, 10
incompleteBesselKR(incompleteBesselK), 10
integrate, 13, 14, 21, 26–28
integrateDens, 13
inversionTestpq (inversionTests), 14
inversionTestqp (inversionTests), 14
inversionTests, 14
is.wholenumber, 16
kurtosis (Sample Moments), 28
logHist, 17
momChangeAbout, 19, 23
momIntegrated, 14, 20
momSE, 22
moranTest, 23
normTailPlotLine (tailPlot), 29
optimize, 8
par, 30
pDist, 25
print.integrate (safeIntegrate), 27
qDist, 5, 9
qDist (pDist), 25
safeIntegrate, 27
Sample Moments, 28
skewness (Sample Moments), 28
SSFcoef (incompleteBesselK), 10
tailPlot, 29
tailPlotLine (tailPlot), 29
tsHessian, 32
tTailPlotLine (tailPlot), 29
uniroot, 4, 5, 14, 26
VarianceGamma, 22