Package 'SuppDists'

September 30, 2009

Version 1.1-6

Date 2009/09/29

Title Supplementary distributions

Author Bob Wheeler <bwheelerg@gmail.com>

Maintainer Bob Wheeler <bwheelerg@gmail.com>

Description Ten distributions supplementing those built into R. Inverse Gauss, Kruskal-Wallis, Kendall's Tau, Friedman's chi squared, Spearman's rho, maximum F ratio, the Pearson product moment correlation coefficiant, Johnson distributions, normal scores and generalized hypergeometric distributions. In addition two random number generators of George Marsaglia are included.

License GPL (>= 2)

URL http://www.bobwheeler.com/stat

Repository CRAN

Date/Publication 2009-09-30 18:56:07

R topics documented:

`Friedman` *Friedman's chi-square*

Description

Density, distribution function, quantile function, random generator and summary function for Friedman's chi square.

Usage

```
dFriedman(x, r, N, log=FALSE)
pFriedman(q, r, N, lower.tail=TRUE, log.p=FALSE)
qFriedman(p, r, N, lower.tail=TRUE, log.p=FALSE)
rFriedman(n, r, N)
sFriedman(r, N)
```

Arguments

`x,q`	vector of non-negative quantities
`p`	vector of probabilities
`n`	number of values to generate. If n is a vector, length(n) values will be generated
`r`	vector of number of treatments
`N`	(N >= 2) vector of number of replications of each treatment
`log, log.p`	logical vector; if TRUE, probabilities p are given as log(p)
`lower.tail`	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

The Freidman chi-squared is used for nonparametric ANOVA. The data in N rows of an $N \times r$ table are ranked separately such that the ranks take the values from 1 to r in the N different rows. The distributions are obtained on the assumption that there is no relationship between the N rows.

Formulae:

Let R_j be the sum of ranks for treatment $j (j = 1 \ldots r)$, then the Friedman statistic is

$$x = \frac{12}{Nr(r+1)} \sum_{j=1}^{r} R_j^2 - 3N(r+1)$$

this is asymptotically equivalent to a χ^2 random variable. One may also calculate the chi squared statistic for the usual analysis of variance which gives

$$F = \frac{(N-1)x}{N(r-1) - x}$$

which may be used with the F distribution functions in R for degrees of freedom $(r-1)$ and $(N-1)(r-1)$.

Value

The output values conform to the output from other such functions in R. `dFriedman()` gives the density, `pFriedman()` the distribution function and `qFriedman()` its inverse. `rFriedman()` generates random numbers. `sFriedman()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.

Note

Exact calculations are made for the following values of the parameters:

r	N
2	100
3	30
4	15
5	8

These exact calculations are made using the algorithm of Kendall and Smith (1939).

The incomplete beta, with continuity correction, is used for calculations outside these ranges. Some appreciation for the accuracy of the approximation may be obtained by comparing the calculated values with exact tables such as Odeh (1977). Iman and Davenport (1980) have studied the accuracy of various approximations.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Kendall, M. and Smith, B.B. (1939). The problem of m rankings. *Ann. Math. Stat.* **10.** 275-287.

Iman, R.L. and Davenport, J.M. (1980). Approximations of the critical region of the Friedman statistic. *Comm. Stat. Theor. Meth.* **A9(6).** 571-595.

Odeh, R.E. (1977). Extended tables of the distribution of Friedman's S-statistic in the two-way layout. *Commun. Statist.-Simula. Computa.* **B6(1).** 29-48.

Examples

```
pFriedman(2, r=5, N=10)
pFriedman(c(.8,3.5,9.3), r=5, N=10) ## approximately 5% 50% and 95%
sFriedman(r=5, N=10)
plot(function(x)dFriedman(x, r=5, N=10),0,10)
```

ghyper *Generalized hypergeometric distributions*

Description

Density, distribution function, quantile function, random generator and summary function for generalized hypergeometric distributions.

Usage

```
dghyper(x, a, k, N, log=FALSE)
pghyper(q, a, k, N, lower.tail=TRUE, log.p=FALSE)
qghyper(p, a, k, N, lower.tail=TRUE, log.p=FALSE)
rghyper(n, a, k, N)
sghyper(a, k, N)
tghyper(a, k, N) ## scalar arguments only
```

Arguments

x,q,n	vector of non-negative **integer** quantities
p	vector of probabilities
a	vector of real values giving the first column total
k	vector of real values giving the first row total
N	vector of real values giving the grand total
log, log.p	logical vector; if TRUE, probabilities p are given as log(p)
lower.tail	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

The basic representation is in terms of a two-way table:

x	k-x	k
a-x	b-k+x	N-k
a	b	N

and the associated hypergeometric probability $P(x) = C_x^a C_{k-x}^b / C_k^N$.

The table is constrained so that rows and columns add to the margins. In all cases x is an integer or zero, but meaningful probability distributions occur when the other parameters are real. Johnson, Kotz and Kemp (1992) give a general discussion.

Kemp and Kemp (1956) classify the possible probability distributions that can occur when real values are allowed, into eight types. The *classic* hypergeometric with integer values forms a ninth type. Five of the eight types correspond to known distributions used in various contexts. Three of the eight types, appear to have no practical applications, but for completeness they have been

implemented.

The Kemp and Kemp types are defined in terms of the ranges of the a, k, and N parameters and are given in `ghyper.types`. The function `tghyper()` will give details for specific values of a, k, and N.

These distributions apply to many important problems, which has lead to a variety of names:

The Kemp and Kemp types IIA and IIIA are known as:

- Negative hypergeometric
- Inverse hypergeometric
- Hypergeometric waiting time
- Beta-binomial

The advantages of the conditional argument are considerable. Consider a few examples:

1. Future event: Consider two events which have occurred u and v times respectively. The distribution function of x occurrences of the first event in a sample of k new trials is calculated. Here a = -u-1, and N = -u-v-2.

   Example: Suppose Toronto has won 3 games and Atlanta 1 in the World Series. What is the probability that Toronto will win the series by taking 2 or more of the remaining 3 games?
2. Exceedance: Consider two samples of size m and k, then the distribution function of x, the number of elements out of k which exceed the r th largest element in the size m sample is calculated. Here a = -r, and N = -m-1.

   Example: Suppose that only once in the last century has the high-water mark at the St. Joe bridge exceeded 12 feet, what is the probability that it will not do so in the next ten years?
3. Waiting time: Consider an urn with T balls, m of which are white, and that drawing without replacement is continued until w white balls are obtained, then the distribution function of x, the number of balls in excess of w that must be drawn is desired. Here a = -w , N = -m-1, and k = T - m.

   Example: Suppose a lot of 100 contains 5 defectives. What is the mean number of items that must be inspected before a defective item is found?
4. Mixture: Suppose x has a binomial distribution with parameter p, and number of trials k. Suppose that p is not fixed, but itself distributed like a beta variable with parameters A and B, then the distribution of x is calculated with a = -A and N = -A -B.

Names for Kemp and Kemp type IV are:

- Beta-negative-binomial
- Beta-Pascal
- Generalized Waring

One application is accidents:

Suppose accidents follow a Poisson distribution with mean L, and suppose L varies with individuals according to accident proneness, m. In particular, suppose L follows a gamma distribution with parameter r and scale factor m , and that the scale factor n itself follows a beta distribution with parameters A and B, then the distribution of accidents, x, is beta-negative-binomial with a = -B, k = -r , and N = A -1. See Xekalki (1983) for a discussion of this as well as a discussion of accident models for proneness, contagion and spells.

Value

The output values conform to the output from other such functions in R. `dghyper()` gives the density, `pghyper()` the distribution function and `qghyper()` its inverse. `rghyper()` generates random numbers. `sghyper()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third central moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.
The function `tghyper()` returns the hypergeometric type and the range of values for x.

Note

The parameters of these functions differ from those of the hypergeometric functions of R. To translate between the two use the following as a model: phyper(x,m,n,k) = pghyper(x,k,m,m+n).

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Johnson, N.L., Kotz, S. and Kemp, A. (1992) *Univariate discrete distributions*. Wiley, N.Y.

Kemp, C.D., and Kemp, A.W. (1956). Generalized hypergeometric distributions. *Jour. Roy. Statist. Soc. B.* **18.** 202-211.

Xekalaki, E. (1983). The univariate generalized Waring distribution in relation to accident theory: proneness, spells or contagion. *Biometrics.* **39.** 887-895.

Examples

```
tghyper(a=4, k=4, N=10)                 ## classic
tghyper(a=4.1, k=5, N=10)               ## type IA(i) Real classic
tghyper(a=5, k=4.1, N=10)               ## type IA(ii) Real classic
tghyper(a=4.2, k=4.6, N=12.2)           ## type IB
tghyper(a=-5.1, k=10, N=-7)             ## type IIA
tghyper(a=-0.5, k=5.9, N=-0.7)          ## type IIB
tghyper(a=10, k=-5.1, N=-7)             ## type IIIA Negative hypergeometric
tghyper(a=5.9, k=-0.5, N=-0.7)          ## type IIIB
tghyper(a=-1, k=-1, N=5)                ## type IV Generalized Waring

sghyper(a=-1, k=-1, N=5)
plot(function(x)dghyper(x,a=-1,k=-1,N=5),0,5)

#Fisher's exact test: contingency table with rows (1,3),(3,1)
pghyper(1,4,4,8)
pghyper(3,4,4,8,lower.tail=FALSE)


#Beta-binomial applications:

#Application examples:
tghyper(-4,3,-6)
```

```
pghyper(2,-4,3,-6,lower=FALSE)
pghyper(0,-2,10,-101)
sghyper(-1,95,-6)$Mean+1
```

`ghyper.types` *Kemp and Kemp generalized hypergeometric types*

Description

Generalized hypergeometric types as given by Kemp and Kemp

Two-way table

The basic representation is in terms of a two-way table:

x	k-x	k
a-x	b-k+x	N-k
a	b	N

and the associated hypergeometric probability $P(x) = C_x^a C_{k-x}^b / C_k^N$.

The types are classified according to ranges of a, k, and N.

Kemp and Kemp types

Minor modifications in the definition of three of the types have been made to avoid numerical difficulties. Note, J denotes a nonnegative integer.

[Classic]	
	$0 < a, 0 < N, 0 < k$
	integers: a, N, k.
	$max(0, a + k - N) \leq x \leq min(a, k)$
[IA(i)] (Real classic)	at least one noninteger parameter
	$0 < a, 0 < N, 0 < k, k - 1 < a < N - (k - 1)$
	integer: k
	$0 \leq x \leq a$
[IA(ii)] (Real classic)	at least one noninteger parameter
	$0 < a, 0 < N, 0 < k, a - 1 < k < N - (a - 1)$
	integer: a
	$0 \leq x \leq a$
	Interchanging a and k transforms this to type IA(i)
[IB]	
	$0 < a, 0 < N, 0 < k, a + k - 1 < N, J < (a, k) < J + 1$
	integer: $0 \leq J$
	non-integer: a, k
	$0 <= x \ldots$

NOTE: Kemp and Kemp specify $-1 < N$.
No practical applications for this distribution.

[IIA] (negative hypergeometric)

$a < 0, N < a - 1, 0 < k$
integer: k
$0 \le x \le k$
NOTE: Kemp and Kemp specify $N < a, N \ne a - 1$

[IIB]

$a < 0, -1 < N < k + a - 1, 0 < k, J < (k, k + a - 1 - N) < J + 1$
non-integer: k
integer: $0 \le J$
$0 \le x$....
This is a very strange distribution. Special calculations were used.
Note: No practical applications.

[IIIA] (negative hypergeometric)

$0 < a, N < k - 1, k < 0$
integer: a
$0 \le x \le a$
Interchanging a and k transforms this to type IIA
NOTE: Kemp and Kemp specify $N < k, N \ne k - 1$

[IIIB]

$0 < a, -1 < N < a + k - 1, k < 0, J < (a, a + k - 1 - N) < J + 1$
non integer: a
integer: $0 \le J$
$0 \le x \ldots$
Interchanging a and k transforms this to type IIB
Note: No practical applications

[IV] (Generalized Waring)

$a < 0, -1 < N, k < 0$
$0 \le x \ldots$

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Kemp, C.D., and Kemp, A.W. (1956). Generalized hypergeometric distributions. *Jour. Roy. Statist. Soc. B.* **18.** 202-211. **39.** 887-895.

`invGauss` *The inverse Gaussian and Wald distributions*

Description

Density, distribution function, quantile function, random generator and summary function for the inverse Gaussian and Wald distributions.

Usage

```
dinvGauss(x, nu, lambda, log=FALSE)
pinvGauss(q, nu, lambda, lower.tail=TRUE, log.p=FALSE)
qinvGauss(p, nu, lambda, lower.tail=TRUE, log.p=FALSE)
rinvGauss(n, nu, lambda)
sinvGauss(nu, lambda)
```

Arguments

`x,q`	vector of non-negative quantities
`p`	vector of probabilities
`n`	vector of numbers of observations
`nu`	vector real and non-negative parameter – the Wald distribution results when nu=1
`lambda`	vector real and non-negative parameter
`log, log.p`	logical vector; if TRUE, probabilities p are given as log(p)
`lower.tail`	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

Probability functions:

$$f(x,\nu,\lambda) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left[-\lambda\frac{(x-\nu)^2}{2\nu^2 x}\right] \text{ – the density}$$

$$F(x,\nu,\lambda) = \Phi\left[\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\nu}-1\right)\right] + e^{2\lambda/\nu}\Phi\left[\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\nu}+1\right)\right] \text{ – the distribution function}$$

where $\Phi[]$ is the standard normal distribution function.

The calculations are accurate to at least seven significant figures over an extended range - much larger than that of any existing tables. We have tested them for $\lambda/\nu = 10e-20$, and $\lambda/\nu = 10e4$. Accessible tables are those of Wassan and Roy (1969), which unfortunately, are sometimes good to only two significant digits. Much better tables are available in an expensive CRC Handbook (1989), which are accurate to at least 7 significant digits for $\lambda/\nu \geq 0.02$ to $\lambda/\nu \leq 4000$.

These are first passage time distributions of Brownian motion with positive drift. See Whitmore and Seshadri (1987) for a heuristic derivation. The Wald (1947) form represents the average sample number in sequential analysis. The distribution has a non-monotonic failure rate, and is of considerable interest in lifetime studies: Ckhhikara and Folks (1977). A general reference is Seshadri (1993).

This is an extremely difficult distribution to treat numerically, and it would not have been possible without some extraordinary contributions. An elegant derivation of the distribution function is to be found in Shuster (1968). The first such derivation seems to be that of Zigangirov (1962), which because of its inaccessibility, the author has not read. The method of generating random numbers is

due to Michael, Schucany, and Haas (1976). The approximation of Whitmore and Yalovsky (1978) makes it possible to find starting values for inverting the distribution. All three papers are short, elegant, and non- trivial.

Value

The output values conform to the output from other such functions in R. `dinvGauss()` gives the density, `pinvGauss()` the distribution function and `qinvGauss()` its inverse. `rinvGauss()` generates random numbers. `sinvGauss()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Ckhhikara, R.S. and Folks, J.L. (1977) The inverse Gaussian distribution as a lifetime model. *Technometrics.* **19-4.** 461-468.

CRC Handbook. (1989). *Percentile points of the inverse Gaussian distribution.* J.A. Koziol (ed.) Boca Raton, FL.

Michael, J.R., Schucany, W.R. and Haas, R.W. (1976). Generating random variates using transformations with multiple roots. *American Statistician.* **30-2.** 88-90.

Seshadri, V. (1993). *The inverse Gaussian distribution.* Clarendon, Oxford

Shuster, J. (1968). On the inverse Gaussian distribution function. Jour. *Am. Stat. Assoc.* **63.** 1514-1516.

Wasan, M.T. and Roy, L.K. (1969). Tables of inverse Gaussian percentage points. *Technometrics.* **11-3.** 591-604.

Wald, A. (1947). *Sequential analysis.* Wiley, NY

Whitmore, G.A. and Seshadri, V. (1987). A heuristic derivation of the inverse Gaussian distribution. *American Statistician.* **41-4.** 280-281.

Whitmore, G.A. and Yalovsky, M. (1978). A normalizing logarithmic transformation for inverse Gaussian random variables. *Technometrics.* **20-2.** 207-208.

Zigangirov, K.S. (1962). Expression for the Wald distribution in terms of normal distribution. *Radiotech.Electron.* **7.** 164-166.

Examples

```
pinvGauss(1, 1, 16)
pinvGauss(c(.65,1,1.45), 1, 16) ## approximately 5% 50% and 95%
pars<-sinvGauss(1, 16)
plot(function(x)dinvGauss(x,1, 16),pars$Mean-3*pars$SD,pars$Mean+3*pars$SD)
```

`Johnson` *The Johnson distributions*

Description

Density, distribution function, quantile function, random generator and summary function for the Johnson distributions.

Usage

```
dJohnson(x, parms, log=FALSE)
pJohnson(q, parms, lower.tail=TRUE, log.p=FALSE)
qJohnson(p, parms, lower.tail=TRUE, log.p=FALSE)
rJohnson(n, parms)
sJohnson(parms)
JohnsonFit(t,moment="quant")
moments(x)
```

Arguments

`x,q`	vector of quantities
`t`	observation vector, t=x, or moment vector, t=[mean,m2,m3,m4]
`p`	vector of probabilities
`n`	vector of numbers of observations
`parms`	list or list of lists each containing output of JohnsonFit()
`moment`	character scalar specifying t: "quant" (default), or "use," or "find"
`log, log.p`	logical vector; if TRUE, probabilities p are given as log(p)
`lower.tail`	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

The Johnson system (Johnson 1949) is a very flexible system for describing statistical distributions. It is defined by

$$z = \gamma + \delta \log f(u), u = (x - \xi)/\lambda$$

and where $f()$ has four possible forms:

SL: $f(u) = u$ the log normal
SU: $f(u) = u + \sqrt{1 + u^2}$ an unbounded distribution
SB: $f(u) = u/(1 - u)$ a bounded distribution
SN: $\exp(u)$ the normal

Estimation of the Johnson parameters may be done from quantiles. The procedure of Wheeler (1980) is used.

They may also be estimated from the moments. Applied Statistics algorithm 99, due to Hill, Hill, and Holder (1976) has been translated into C for this implementation.

Value

The output values conform to the output from other such functions in R. `dJohnson()` gives the density, `pJohnson()` the distribution function and `qJohnson()` its inverse. `rJohnson()` generates random numbers. `sJohnson()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.
`moments()` calculates the moment statistics of x as a vector with elements (mu, sigma, skew, kurt), where mu is the mean of x, sigma the SD of x with divisor `length(x)`, skew is the skewness and kurt the kurtosis.
`JohnsonFit()` outputs a list containing the Johnson parameters (gamma, delta, xi, lambda, type), where type is one of the Johnson types: "SN", "SL", "SB", or "SU". `JohnsonFit()` does this using 5 order statistics when moment="quant", when moment="find" it does this by using the first four moments of t calculated by the function `moments()`, when moment="use" it assumes that the vector t is [mean,m2,m3,m4], where mi is the ith moment about the mean.
Fitting by moments is difficult numerically and often `JohnsonFit()` will report an error.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Hill, I.D., Hill, R., and Holder, R.L. (1976). Fitting Johnson curves by moments. *Applied Statistics.* AS99.

Johnson, N.L. (1949). Systems of frequency curves generated by methods of translation. *Biometrika,* **36.** 149-176.

Wheeler, R.E. (1980). Quantile estimators of Johnson curve parameters. *Biometrika.* **67-3** 725-728

Examples

```
xx<-rnorm(500)
parms<-JohnsonFit(xx)
sJohnson(parms)
plot(function(xx)dJohnson(xx,parms),-2,2)
pJohnson(1,parms)
parms2<-JohnsonFit(rexp(50))
qJohnson(p=0.5,list(parms,parms2))

## JohnsonFit with moment="find" and moment="use" is not always possible,
## and even when possible, may produce odd results.
## parms<-JohnsonFit(x,moment="find")

parms<-JohnsonFit(c(0,1,-.5,4),moment="use")
```

```
sJohnson(parms)

# Fit illustration
data(cars)
xx<-cars$speed
parms<-JohnsonFit(xx)
hist(xx,freq=FALSE)
plot(function(x)dJohnson(x,parms),0,25,add=TRUE)
```

`Kendall` *The distribution of Kendall's tau*

Description

Density, distribution function, quantile function, random generator and summary function for Kendall's tau.

Usage

```
dKendall(x, N, log=FALSE)
pKendall(q, N, lower.tail=TRUE, log.p=FALSE)
qKendall(p, N, lower.tail=TRUE, log.p=FALSE)
rKendall(n, N)
sKendall(N)
```

Arguments

`x,q`	vector of non-negative quantities
`p`	vector of probabilities
`n`	number of values to generate. If n is a vector, length(n) values will be generated
`N`	vector number of treatments
`log, log.p`	logical vector; if TRUE, probabilities p are given as log(p)
`lower.tail`	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

There are two categories with N treatments each. The treatments are ranked for each category, and then sorted according to the ranks for the first category. This produces a 2 by N array in which the numbers in the first row are increasing from 1 to N. The array is scanned, and every time two adjacent ranks in the second row are not in order, they are exchanged. The scanning is repeated until the second row is in increasing order. Let s denote the number of exchanges, then Kendall's tau is given by

$$\tau = 1 - \frac{4s}{N(N-1)}$$

This too is a product-moment correlation coefficient. See Kendall (1975), Chapter 2. Other methods for calculating the statistic are also discussed there.

The calculated values are exact for $N < 13$, thereafter an Edgeworth expansion is used.

Value

The output values conform to the output from other such functions in R. `dKendall()` gives the density, `pKendall()` the distribution function and `qKendall()` its inverse. `rKendall()` generates random numbers. `sKendall()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Kendall, M. (1975). *Rank Correlation Methods.* Griffin, London.

Examples

```
pKendall(0, N=10)
pKendall(c(-.42,0.02,.42), N=10) ## approximately 5% 50% and 95%
qKendall(.95,N=c(10,20))
sKendall(N=10)
plot(function(x)dKendall(x, N=10),-0.5,0.5)
```

KruskalWallis *Kruskall-Wallis distribution*

Description

Density, distribution function, quantile function, random generator and summary function for the Kruskal-Wallis test.

Usage

```
dKruskalWallis(x, c, N, U, log=FALSE)
pKruskalWallis(q, c, N, U, lower.tail=TRUE, log.p=FALSE)
qKruskalWallis(p, c, N, U, lower.tail=TRUE, log.p=FALSE)
rKruskalWallis(n, c, N, U)
sKruskalWallis(c, N, U)
```

Arguments

x,q	vector of non-negative quantities
p	vector of probabilities
n	number of values to generate. If n is a vector, length(n) values will be generated
c	vector number of treatments
N	vector total number of observations
U	vector sum of reciprocals of the number of the c sample sizes
log, log.p	logical vector; if TRUE, probabilities p are given as log(p)
lower.tail	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

This is a one-way layout with, perhaps, unequal sample sizes for each treatment. There are c treatments with sample sizes $n_j, j = 1 \ldots c$. The total sample size is $N = \sum_1^c n_j$. The distribution depends on c, N, and U, where $U = \sum_1^c (1/n_j)$.

Let R_j be the sum of the ranks for treatment $j (j = 1 \ldots c)$ then the Kruskal-Wallis statistic is

$$x = \frac{12}{N(N+1)} \sum_{j=1}^{c} \frac{R_j^2}{n_j} - 3(N+1)$$

This is asymptotically equivalent to a chi-squared variable with c-1 degrees of freedom.

The original paper is Kruskal and Wallis (1952) with errata appearing in Kruskal and Wallis (1953). No attempt is made to calculate exact values, rather an incomplete beta approximation is used following Wallace (1959).

Value

The output values conform to the output from other such functions in R. `dKruskalWallis()` gives the density, `pKruskalWallis()` the distribution function and `qKruskalWallis()` its inverse. `rKruskalWallis()` generates random numbers. `sKruskalWallis()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Kruskal, W.H. and Wallis, W.A. (1952) Use of ranks in one-criterion variance analysis. *Jour. Am. Stat. Assoc.* **47.** 583-634

Kruskal, W.H. and Wallis, W.A. (1953) Errata to Use of ranks in one-criterion variance analysis. *Jour. Am. Stat. Assoc.* **48.** 907-911.

Wallace, D.L. (1959). Simplified beta-approximations to the Kruskal-Wallis H test. *Jour. Am. Stat. Assoc.* **54.** 225-230.

Examples

```
# Assuming three treatments, each with a sample size of 5.
pKruskalWallis(1, 3, 15, 0.6)
pKruskalWallis(c(.1,1.5,5.7), 3, 15, 0.6) ## approximately 5% 50% and 95%
sKruskalWallis(3, 15, 0.6)
plot(function(x)dKruskalWallis(x, 3, 15, 0.6),0.5,8)
```

`maxFratio` *The maximum F-ratio distribution*

Description

Density, distribution function, quantile function, random generator and summary function for the maximum F-ratio.

Usage

```
dmaxFratio(x, df, k, log=FALSE)
pmaxFratio(q, df, k, lower.tail=TRUE, log.p=FALSE)
qmaxFratio(p, df, k, lower.tail=TRUE, log.p=FALSE)
rmaxFratio(n, df, k)
smaxFratio(df, k)
```

Arguments

`x,q`	vector of non-negative quantities
`p`	vector of probabilities
`n`	number of values to generate. If n is a vector, length(n) values will be generated
`df`	vector non-negative, integer degrees of freedom
`k`	vector non-negative, integer number of mean squares
`log, log.p`	logical vector; if TRUE, probabilities p are given as log(p)
`lower.tail`	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

The maximum F-ratio is the ratio of the largest to the smallest of k independent mean squares, all with the same df. The usual use is to test for homogeneity of normal variances.

Value

The output values conform to the output from other such functions in R. `dmaxFratio()` gives the density, `pmaxFratio()` the distribution function and qmaxFratio its inverse. `rmaxFratio()` generates random numbers. `smaxFratio()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.

Limitations

The literature contains no information on numerical procedures for this distribution, with the result that all calculations are slow.

Finding p from x should give results for almost any values of df and k – of course absolutely enormous values will take a while.

Finding x from p is an iterative calculation dependent on a good starting guess. Such good guesses have been made for $df \leq 24$ and $k \leq 160$. NA will be returned if larger values are attempted.

Note

The maximum F-ratio was introduced by Hartley (1950) as a shortcut test of the homogeneity of variances from normal data. Given a set of k mean squares, each based on the same number of degrees of freedom, df, the test statistic is the ratio of the largest to the smallest. Several tables have been constructed. The first by David, H.A. (1952). Currently the most extensive are those by Nelson (1987).

It is important to note that tests of this sort are substantially dependent on the assumption of normality, and can not be used robustly as can variance ratios in the analysis of variance.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Hartley, H.O. (1950) The maximum F-ratio as a short cut test for heterogeneity of variance. *Biometrika.* **37.** 308-312.

David, H.A. (1952). Upper 5 and 1% points of the maximum F-ratio. *Biometrika.* **38.** 422-424.

Nelson, L.S. (1987). Upper 10%, 5% and 1% points of the maximum F-ratio, *Jour. Qual. Tech.* **19-3.** 165-167.

Examples

```
pmaxFratio(4, 10, 10)
pmaxFratio(c(2.3, 4, 8.5), 10, 10)      ## approximately 5% 50% and 95%
qmaxFratio(p=.95,df=c(10,20), k=10)
smaxFratio(10, 10) ## Wait for this, it may take a while
plot(function(x)dmaxFratio(x, 10, 10),1,10)
```

MWC1019 *A very long period pseudo-random generator*

Description

A pseudo-random number generator with a period exceeding "10e9824" due to George Marsaglia

Usage

```
rMWC1019(n, new.start=FALSE, seed=556677)
```

Arguments

n	number of values to generate. If n is a vector, length(n) values will be generated
new.start	logical scalar. If TRUE the generator will be started afresh using the seed
seed	scalar 32 bit integer used as the generator seed

Details

The following is an email from George Marsaglia on this generator:

"MCW1019 will provide random 32-bit integers at the rate of 300 million per second (on a 850MHz PC).

The period of MWC1029 exceeds 10e9824, making it billions and billions ... and billions times as long as the highly touted longest-period RNG, the Mersenne twister. It is also several times as fast and takes a few lines rather than several pages of code. (This is not to say that the Mersenne twister is not a good RNG; it is. I just do not equate complexity of code with randomness. It is the complexity of the underlying randomness that counts.)

As for randomness, it passes all tests in The Diehard Battery of Tests of Randomness http://stat.fsu.edu/pub/diehard as well as three new tough tests I have developed with the apparent property that a RNG that passes tuftsts.c will pass all the tests in Diehard.

MWC1019 has the property that every possible sequence of 1018 successive 32-bit integers will appear somewhere in the full period, for those concerned with the "equi-distribution" in dimensions 2,3,...1016,1017,1018."

NOTE: This function does not work correctly on LP64 bit machines which use 64 bit longs.

The generator requires 1019 initial 32 bit random values to start. These are provided by using Marsaglia's MWC generator. By setting new.start=FALSE, the sequence may be sampled in blocks. R specifies an internal array dimension of 625 for seed length, and thus at the present time, it is not possible to implement this generator using the .Random.seed mechanism. In this implementation, rMWC1019() seems to be about twice as fast as the Mersenne-Twister implemented in R.

Value

generates a vector of random numbers.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

Examples

```
rMWC1019(50,new.start=TRUE,seed=492166)
rMWC1019(50)
```

`NormalScore` *Normal Scores distribution*

Description

Density, distribution function, quantile function, random generator and summary function for the normal scores test. A function to calculate expected values of normal order statistics is included.

Usage

```
dNormScore(x, c, N, U, log=FALSE)
pNormScore(q, c, N, U, lower.tail=TRUE, log.p=FALSE)
qNormScore(p, c, N, U, lower.tail=TRUE, log.p=FALSE)
rNormScore(n, c, N, U)
sNormScore(c, N, U)
normOrder(N)
```

Arguments

`x,q`	vector of non-negative quantities
`p`	vector of probabilities
`n`	number of values to generate. If n is a vector, length(n) values will be generated
`c`	vector number of treatments
`N`	vector total number of observations
`U`	vector sum of reciprocals of the number of the c sample sizes
`log, log.p`	logical vector; if TRUE, probabilities p are given as log(p)
`lower.tail`	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

This is the Kruskal-Wallis statistic with ranks replaced by the expected values of normal order statistics. There are c treatments with sample sizes $n_j, j = 1 \dots c$. The total sample size is $N = \sum_1^c n_j$. The distribution depends on c, N, and U, where $U = \sum_1^c (1/n_j)$.

Let $e_N(k)$ be the expected value of the k_{th} smallest observation in a sample of N independent normal variates. Rank all observations together, and let R_{ij} denote the rank of observation X_{ij}, $i = 1 \dots n_j$ for treatment $j = 1 \dots c$, then the normal scores test statistic is

$$x = (N-1)\frac{1}{\sum_{k=1}^{N} e_N(k)^2} \sum_{j=1}^{c} \frac{S_j^2}{n_j}$$

where $S_j = \sum_{i=1}^{n_j} (e_N(R_{ij}))$.

See Lu and Smith (1979) for a thorough discussion and some exact tables for small r and n. The calculations made here use an incomplete beta approximation – the same one used for Kruskal-Wallis, only differing in the calculation of the variance of the statistic.

The expected values of the normal order statistics use a modification of M.Maechler's C version of the Fortran algorithm given by Royston (1982). Spot checking the values against Harter (1969) confirms the accuracy to 4 decimal places as claimed by Royston.

Value

The output values conform to the output from other such functions in R. `dNormScore()` gives the density, `pNormScore()` the distribution function and `qNormScore()` its inverse. `rNormScore()` generates random numbers. `sNormScore()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis. `normOrder()` gives the expected values of the normal order statistics for a sample of size N.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Harter, H.L. (1969). *Order statistics and their use in testing and estimation, volume 2.* U.S. Supp. of Doc.

Lu, H.T. and Smith, P.J. (1979) Distribution of normal scores statistic for nonparametric one-way analysis of variance. *Jour. Am Stat. Assoc.* **74.** 715-722.

Royston, J.P. (1982). Expected normal order statistics (exact and approximate) AS 177. *Applied Statistics.* **31.** 161-165.

Examples

```
#Assuming three treatments, each with a sample size of 5
pNormScore(2, 3, 15, 0.6)
pNormScore(c(0.11,1.5,5.6), 3, 15, 0.6) ## approximately 5% 50% and 95%
```

```
sNormScore(3, 15, 0.6)
plot(function(x)dNormScore(x,c=5, N=15, U=0.6),0,5)
```

`Pearson` *The Pearson product moment correlation coefficient*

Description

Density, distribution function, quantile function, random generator and summary function for the distribution of Pearson's product moment correlation.

Usage

```
dPearson(x, N, rho=0.0, log=FALSE)
pPearson(q, N, rho=0.0, lower.tail=TRUE, log.p=FALSE)
qPearson(p, N, rho=0.0, lower.tail=TRUE, log.p=FALSE)
rPearson(n, N, rho=0.0)
sPearson(N, rho=0.0)
```

Arguments

`x,q`	vector of sample correlations
`p`	vector of probabilities
`rho`	vector of population correlations
`N`	vector of numbers of observations, $(N > 3)$
`n`	number of values to generate. If n is a vector, length(n) values will be generated
`log, log.p`	logical vector; if TRUE, probabilities p are given as log(p)
`lower.tail`	logical vector; if TRUE (default), probabilities are $P[R <= r]$, otherwise, $P[R > r]$

Value

The output values conform to the output from other such functions in R. `dPearson()` gives the density, `pPearson()` the distribution function and `qPearson()` its inverse. `rPearson()` generates random numbers. `sPearson()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

Examples

```
pPearson(0.5, N=10)
pPearson(q=0.5, N=10, rho=0.3)
sPearson(N=10)
plot(function(x)dPearson(x,N=10,rho=0.7),-1,1)
```

`Spearman` *Spearman's rho*

Description

Density, distribution function, quantile function, random generator and summary function for Spearman's rho.

Usage

```
dSpearman(x, r, log=FALSE)
pSpearman(q, r,  lower.tail=TRUE, log.p=FALSE)
qSpearman(p, r,  lower.tail=TRUE, log.p=FALSE)
rSpearman(n, r)
sSpearman(r)
```

Arguments

`x,q`	vector of non-negative quantities
`p`	vector of probabilities
`n`	number of values to generate. If n is a vector, length(n) values will be generated
`r`	(r >= 3) vector of number of observations
`log, log.p`	logical vector; if TRUE, probabilities p are given as log(p)
`lower.tail`	logical vector; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$

Details

Spearman's rho is the rank correlation coefficient between r pairs of items. It ranges from -1 to 1. Denote by d, the sum of squares of the differences between the matched ranks, then x is given by:

$$1 - \frac{6d}{r(r^2 - 1)}$$

This is, in fact, the product-moment correlation coefficient of rank differences. See Kendall (1975), Chapter 2. It is identical to Friedman's chi-squared for two treatments scaled to the -1, 1 range – if X is the Friedman statistic, then $\rho = fracXr - 1) - 1$.

Exact calculations are made for $r \leq 100$

These exact calculations are made using the algorithm of Kendall and Smith (1939).

The incomplete beta, with continuity correction, is used for calculations outside this range.

Value

The output values conform to the output from other such functions in R. `dSpearman()` gives the density, `pSpearman()` the distribution function and `qSpearman()` its inverse. `rSpearman()` generates random numbers. `sSpearman()` produces a list containing parameters corresponding to the arguments – mean, median, mode, variance, sd, third cental moment, fourth central moment, Pearson's skewness, skewness, and kurtosis.

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Kendall, M. (1975). *Rank Correlation Methods.* Griffin, London.

Kendall, M. and Smith, B.B. (1939). The problem of m rankings. *Ann. Math. Stat.* **10.** 275-287.

Examples

```
pSpearman(.95, 10)
pSpearman(c(-0.55,0,0.55), 10) ## approximately 5% 50% and 95%
sSpearman(10)
plot(function(x)dSpearman(x, 10),-.9,.9)
```

`ziggurat` *The Ziggurat normal and exponential generator*

Description

Generates normal and exponential random pseudo-random numbers by the method of Marsaglia and Tsang.

Usage

```
rziggurat(n, normal=TRUE, new.start=FALSE, seed=556677)
```

Arguments

`n`	number of values to generate. If n is a vector, length(n) values will be generated
`normal`	logical scalar; if TRUE normal values are produced, otherwise exponential values
`new.start`	logical scalar. If TRUE the generator will be started afresh using the seed
`seed`	scalar 32 bit starting integer

Value

Generates a vector of real pseudo random numbers.

Note

This function does not work properly on LP64 machines which use 64 bit longs.

This implementation running in R is approximately three times as fast as rnorm().

Author(s)

Bob Wheeler ⟨bwheelerg@gmail.com⟩

References

Marsaglia, George, and Tsang, Wai Wan. 2000. The Ziggurat method for generating random variables. *Journal of Statistical software.* **5-8.** http://www.jstatsoft.org/v05/i08/

Examples

```
rziggurat(50,new.start=TRUE)
rziggurat(50)
rziggurat(50,new.start=TRUE)
```

Index