Beta {stats} R Documentation

The Beta Distribution

Description

Density, distribution function, quantile function and random generation for the Beta distribution with parameters `shape1` and `shape2` (and optional non-centrality parameter `ncp`).

Usage

```
dbeta(x, shape1, shape2, ncp = 0, log = FALSE)
pbeta(q, shape1, shape2, ncp = 0, lower.tail = TRUE, log.p = FALSE)
qbeta(p, shape1, shape2, ncp = 0, lower.tail = TRUE, log.p = FALSE)
rbeta(n, shape1, shape2, ncp = 0)
```

Arguments

`x, q`
: vector of quantiles.

`p`
: vector of probabilities.

`n`
: number of observations. If `length(n) > 1`, the length is taken to be the number required.

`shape1, shape2`
: non-negative parameters of the Beta distribution.

`ncp`
: non-centrality parameter.

`log, log.p`
: logical; if TRUE, probabilities p are given as log(p).

`lower.tail`
: logical; if TRUE (default), probabilities are $P[X \le x]$, otherwise, $P[X > x]$.

Details

The Beta distribution with parameters `shape1` = a and `shape2` = b has density

$$\Gamma(a+b)/(\Gamma(a)\Gamma(b))x^{(a-1)}(1-x)^{(b-1)}$$

for $a > 0$, $b > 0$ and $0 \le x \le 1$ where the boundary values at $x=0$ or $x=1$ are defined as by continuity (as limits).
The mean is $a/(a+b)$ and the variance is $ab/((a+b)^2 (a+b+1))$. These moments and all distributional properties can be defined as limits (leading to point masses at 0, 1/2, or 1) when a or b are zero or infinite, and the corresponding `[dpqr]beta()` functions are defined correspondingly.

`pbeta` is closely related to the incomplete beta function. As defined by Abramowitz and Stegun 6.6.1

$$B_x(a,b) = \int_0^x t^{(a-1)} (1-t)^{(b-1)} dt,$$

and 6.6.2 $I_x(a,b) = B_x(a,b) / B(a,b)$ where $B(a,b) = B_1(a,b)$ is the Beta function (beta).

$I_x(a,b)$ is `pbeta(x, a, b)`.

The noncentral Beta distribution (with `ncp` = λ) is defined (Johnson *et al*, 1995, pp. 502) as the distribution of $X/(X+Y)$ where $X \sim \chi^2_{2a}(\lambda)$ and $Y \sim \chi^2_{2b}$.

Value

`dbeta` gives the density, `pbeta` the distribution function, `qbeta` the quantile function, and `rbeta` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

The length of the result is determined by `n` for `rbeta`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

Supplying `ncp = 0` uses the algorithm for the non-central distribution, which is not the same algorithm used if `ncp` is omitted. This is to give consistent behaviour in extreme cases with values of `ncp` very near zero.

Source

- The central `dbeta` is based on a binomial probability, using code contributed by Catherine Loader (see dbinom) if either shape parameter is larger than one, otherwise directly from the definition. The non-central case is based on the derivation as a Poisson mixture of betas (Johnson *et al*, 1995, pp. 502–3).

- The central `pbeta` for the default (`log_p = FALSE`) uses a C translation based on

Didonato, A. and Morris, A., Jr, (1992) Algorithm 708: Significant digit computation of the incomplete beta function ratios, *ACM Transactions on Mathematical Software*, **18**, 360–373. (See also
Brown, B. and Lawrence Levy, L. (1994) Certification of algorithm 708: Significant digit computation of the incomplete beta, *ACM Transactions on Mathematical Software*, **20**, 393–397.)
We have slightly tweaked the original “TOMS 708” algorithm, and enhanced for `log.p = TRUE`. For that (log-scale) case, underflow to `-Inf` (i.e., $P = 0$) or `0`, (i.e., $P = 1$) still happens because the original algorithm was designed without log-scale considerations. Underflow to `-Inf` now typically signals a `warning`.

- The non-central `pbeta` uses a C translation of

  Lenth, R. V. (1987) Algorithm AS226: Computing noncentral beta probabilities. *Appl. Statist*, **36**, 241–244, incorporating
  Frick, H. (1990)'s AS R84, *Appl. Statist*, **39**, 311–2, and
  Lam, M.L. (1995)'s AS R95, *Appl. Statist*, **44**, 551–2.

  This computes the lower tail only, so the upper tail suffers from cancellation and a warning will be given when this is likely to be significant.

- The central case of `qbeta` is based on a C translation of

  Cran, G. W., K. J. Martin and G. E. Thomas (1977). Remark AS R19 and Algorithm AS 109, *Applied Statistics*, **26**, 111–114, and subsequent remarks (AS83 and correction).

- The central case of `rbeta` is based on a C translation of

  R. C. H. Cheng (1978). Generating beta variates with nonintegral shape parameters. *Communications of the ACM*, **21**, 317–322.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Abramowitz, M. and Stegun, I. A. (1972) *Handbook of Mathematical Functions.* New York: Dover. Chapter 6: Gamma and Related Functions.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 2, especially chapter 25. Wiley, New York.

See Also

Distributions for other standard distributions.

`beta` for the Beta function.

Examples

```
x <- seq(0, 1, length = 21)
dbeta(x, 1, 1)
```

```
pbeta(x, 1, 1)

## Visualization, including limit cases:
pl.beta <- function(a,b, asp = if(isLim) 1, ylim = if(isLim) c(0,1.1)) {
  if(isLim <- a == 0 || b == 0 || a == Inf || b == Inf) {
    eps <- 1e-10
    x <- c(0, eps, (1:7)/16, 1/2+c(-eps,0,eps), (9:15)/16, 1-eps, 1)
  } else {
    x <- seq(0, 1, length = 1025)
  }
  fx <- cbind(dbeta(x, a,b), pbeta(x, a,b), qbeta(x, a,b))
  f <- fx; f[fx == Inf] <- 1e100
  matplot(x, f, ylab="", type="l", ylim=ylim, asp=asp,
          main = sprintf("[dpq]beta(x, a=%g, b=%g)", a,b))
  abline(0,1,     col="gray", lty=3)
  abline(h = 0:1, col="gray", lty=3)
  legend("top", paste0(c("d","p","q"), "beta(x, a,b)"),
         col=1:3, lty=1:3, bty = "n")
  invisible(cbind(x, fx))
}
pl.beta(3,1)

pl.beta(2, 4)
pl.beta(3, 7)
pl.beta(3, 7, asp=1)

pl.beta(0, 0)   ## point masses at  {0, 1}

pl.beta(0, 2)   ## point mass at 0 ; the same as
pl.beta(1, Inf)

pl.beta(Inf, 2) ## point mass at 1 ; the same as
pl.beta(3, 0)

pl.beta(Inf, Inf)# point mass at 1/2
```

[Package *stats* version 3.6.0 Index]