NTNU

Kunnskap for ei betre verd

Institutt for matematiske fag

# Eksamensoppgåve i
# **ST2304 Statistisk modellering for biologar og bioteknologar**

**Fagleg kontakt under eksamen:** Jarle Tufto

**Tlf:** 99 70 55 19

**Eksamensdato:** 9. juni 2016

**Eksamenstid (frå–til):** 9–13

**Hjelpemiddelkode/Tillatne hjelpemiddel:** Tabeller og formler i statistikk, Tapir Forlag, K. Rottmann: Matematisk formelsamling, Kalkulator Casio fx-82ES PLUS, CITIZEN SR-270X, CITIZEN SR-270X College eller HP30S, eit gult A4-ark med egne håndskrevne notater.

**Annan informasjon:**
Hjelpesider for nokre R funksjonar som du kan få bruk for følgjer i vedlegget. Alle svar skal grunngjevast og innehalde naudsynt mellomrekning.

**Målform/språk:** nynorsk

**Sidetal:** 7

**Sidetal vedlegg:** 3

**Kontrollert av:**

Dato Sign

**Oppgåve 1** Gå ut i frå at den stokastiske variabelen $X$ er binomisk fordelt med parametere $n = 20$ og $p = 0.3$.

**a)** Skriv R-uttrykk som rekner ut sannsyna

$$P(X < 9), \qquad P(X \leq 9), \qquad P(X > 9), \qquad P(X \geq 9).$$

Eit vanleg brukt mål på kor skjeiv fordelinga til ein stokastisk variabel $Y$ er

$$\frac{E((Y-\mu)^3)}{(\sigma^2)^{3/2}}$$

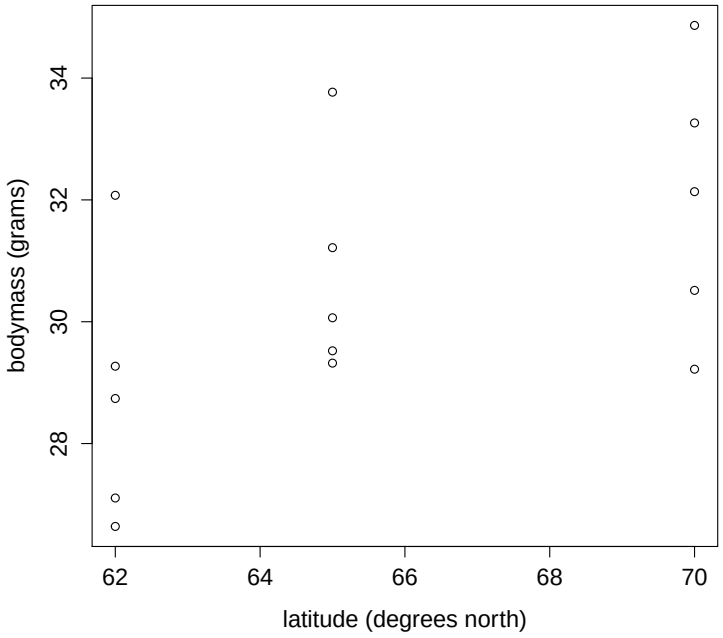
kor $\mu$ og $\sigma^2$ er forventing og varians til $Y$.

**b)** Rekn ut forventning og varians til $X$. Skriv eit R-uttrykk som simulerer 1000 realisasjoner av $X$ og tilordner dette til en vektor `x`. Skriv så eit nytt uttrykk som basert på dei simulerte verdene estimerar kor skjeiv fordelinga til $X$ er.

**Oppgåve 2** Bergmanns regel seier at gjennomsnittleg kroppstorleik innan ein art tenderar til å vere større i subpopulasjoner med kaldt klima enn i subpopulasjoner med varmt klima. I ein pilotstudie for å undersøke om dette gjeld for snømus (*Mustela nivalis*) samler ein forskar inn tilsaman 15 individ frå subpopulasjonar lokalisert ved tre ulike breiddegradar som vist i figur 1. Forskaren tilpassar så følgjande modell i R.

```
> linear <- lm(bodymass~latitude)
> summary(linear)

Call:
lm(formula = bodymass ~ latitude)

Residuals:
    Min      1Q  Median      3Q     Max
-2.9749 -1.3083 -0.1913  1.0138  3.5128

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.0439    10.4535   0.483   0.6375
latitude      0.3879     0.1590   2.440   0.0298 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.032 on 13 degrees of freedom
Multiple R-squared:  0.3141,Adjusted R-squared:  0.2613
F-statistic: 5.952 on 1 and 13 DF,  p-value: 0.02979
```

**a)** Skriv opp modellen vi har tilpassa ovanfor i matematisk notasjon og gjer rede for modellføresetnadene. Kva vert estimata av dei ukjende parameterane i modellen? Kva vert estimatet av forventa skilnad i kroppsstorleik (i gram) for to individ trukke frå 62. og 70. breiddegrad?




Figur 1: Observerte kroppsstorleikar (i gram) i subpopulasjonar ved tre ulike breddegradar.

**b)** Er effekten av breiddegrad statistisk signifikant dersom vi brukar$\alpha = 0.05$ som signifikansnivå? Skriv eit R-uttrykk som for samme hypotesetest rekner ut kritiske verde dersom vi i staden vel $\alpha = 0.01$ som signifikansnivå.

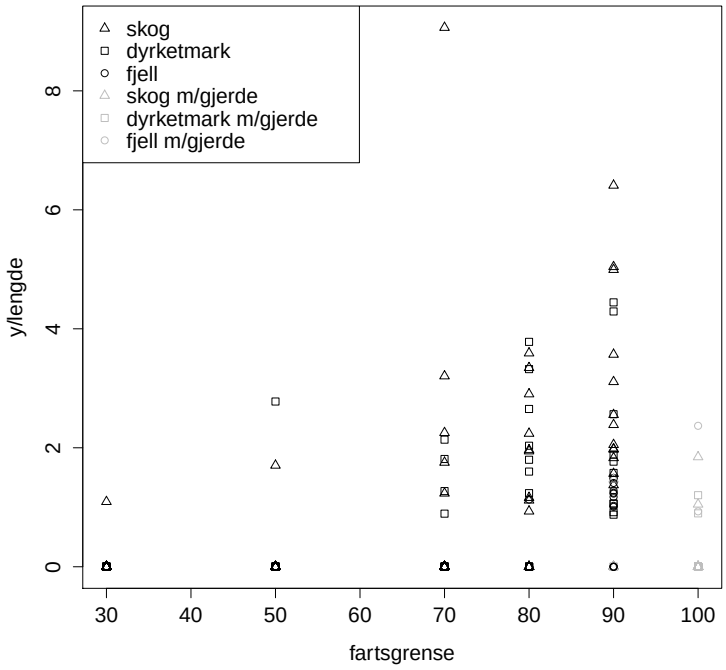
For å teste føresetnaden om linearitet tilpassar forskaren ein alternativ modell med breiddegrad i staden inkludert som ein kategorisk forklaringsvariabel (faktor) som følgjer.

```
> latfactor <- factor(latitude)
> latfactor
 [1] 62 62 62 62 62 65 65 65 65 65 70 70 70 70 70
Levels: 62 65 70
> nonlinear <- lm(bodymass~latfactor)
> summary(nonlinear)

Call:
lm(formula = bodymass ~ latfactor)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7790 -1.4715 -0.0266  0.8837  3.3096

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    28.7661     0.9272  31.024 7.91e-13 ***
latfactor65     2.0125     1.3113   1.535   0.1508
latfactor70     3.2336     1.3113   2.466   0.0297 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.073 on 12 degrees of freedom
Multiple R-squared:  0.3408,Adjusted R-squared:  0.2309
F-statistic: 3.101 on 2 and 12 DF,  p-value: 0.08209

> anova(linear,nonlinear)
Analysis of Variance Table

Model 1: bodymass ~ latitude
Model 2: bodymass ~ latfactor
  Res.Df    RSS Df Sum of Sq     F Pr(>F)
1     13 53.673
2     12 51.584  1    2.0891 0.486  0.499
```

**c)** Avgjer om modellane `linear` og `nonlinear` er nøsta. Er det grunnlag for å hevde at samanhengen mellom kroppstorleik og breiddegrad er ikkje-lineær om vi brukar $\alpha = 0.05$ som signifikansnivå? Kva for modell vel du?

**d)** Forskaren ønskjer å publisere resultata i ein journal som opererer med $\alpha = 0.01$ som signifikansnivå (sannsyn for type I feil). Ho treng difor å rekne ut kor mykje meir data

ho vil trengje før ho kan rekne med at effekten av breiddegrad vert statistisk signifikant ved dette signifikansnivået. Gå ut i frå at den lineære modellen er riktig og at dei reelle parameterverdene er som estimert over. Gå også ut i frå at det nye større datasettet samlast inn i form av kun to utval, begge av storleik $n$, trukke frå populasjonar lokalisera ved 62. og 70. breiddegrad. Skriv eit R-utrykk som rekner ut nødvendig utvalsstorleik $n$ under disse føresetnadene om vi krev at teststyrken skal vere minst 0.9 (sjå vedlegg).




Figur 2: Talet på kollisjonar mellom kjøretøy og elg dividert på vegsegmentlengde ($km^{-1}$) versus fartsgrense (km/time) for vegsegment innafor ulike vegetasjonstypar og med eller utan viltgjerding (sjå symbolforklaring).

**Oppgåve 3** Vegmyndigheitene ønskjer å analysere korleis talet på elgkollisjonar $y$ langs ulike vegsegmentar registrert i løpet av ein 10års-periode påverkas av fartsgrensen for vegsegmentene (km/time), vegetasjonstypen langs vegsegmentane (skog, dyrka mark, fjell), lengda på dei ulike vegsegmenta (målt i km), samt om vegsegmenta er gjærda inn eller ikkje (se figur 2). Dataene vert satt saman i følgjande dataframe i R (de første 30 av totalt 300 observasjoner er vist) som vi så analyserar med ein generalisert lineær modell.

```
   y vegetasjon fartsgrense gjerde lengde
1  0       skog          50    nei  0.699
2  0       skog          30    nei  0.948
3  0      fjell          90     ja  0.891
4  0 dyrketmark          70    nei  0.478
5  0      fjell          50    nei  0.384
6  0 dyrketmark          90     ja  1.089
7  0 dyrketmark          50    nei  0.411
8  0 dyrketmark          50    nei  0.382
9  1       skog          90     ja  0.680
10 0 dyrketmark          50    nei  1.043
11 0      fjell          50    nei  0.448
12 0 dyrketmark          80    nei  0.786
```

```
13 0 dyrketmark          50    nei  1.014
14 0       skog          30    nei  0.468
15 3       skog          80    nei  0.897
16 0 dyrketmark          30    nei  0.326
17 0      fjell          80    nei  0.752
18 0       skog          50    nei  0.644
19 0 dyrketmark         100     ja  1.134
20 0 dyrketmark         100     ja  0.522
21 0       skog          80    nei  0.846
22 0      fjell          80    nei  0.717
23 3       skog          90    nei  0.595
24 2       skog          50    nei  1.174
25 0       skog          50    nei  0.520
26 1       skog          70    nei  0.444
27 0      fjell         100     ja  0.805
28 0       skog         100     ja  1.126
29 0       skog          70    nei  0.574
30 0       skog          50    nei  0.880

Call:
glm(formula = y ~ log(fartsgrense) + vegetasjon + gjerde, family = poisson(link = "log"),
    offset = log(lengde))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.81143  -0.54300  -0.30002  -0.08992   3.07849

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)          -21.1970     3.3543  -6.319 2.63e-10 ***
log(fartsgrense)       4.9203     0.7604   6.471 9.74e-11 ***
vegetasjondyrketmark  -0.4956     0.2257  -2.196   0.0281 *
vegetasjonfjell       -1.7222     0.3286  -5.241 1.59e-07 ***
gjerdeja              -2.8762     0.4191  -6.863 6.73e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 319.28  on 299  degrees of freedom
Residual deviance: 161.71  on 295  degrees of freedom
AIC: 325.02

Number of Fisher Scoring iterations: 6
```

**a)** Forklar kvifor Poisson-føresetnaden, log link-funksjon, og bruken av log til vegsegmentlengda som offset-variabel kan vere rimelege føresetnader.

**b)** Vi har inkludert log til fartsgrensa som numerisk forklaringsvariabel i modellen. Med kor mange prosent reduserast forventa tal på kollisjonar i følgje den estimerte modellen dersom fartsgrensa reduserast frå 80 til 70 km/time gitt at andre forklaringsvariablar haldast konstant? For kva vegetasjonstype gir ein slik fartsgrensereduksjon størst forventa reduksjon i talet på kollisjonar?

**c)** Er det grunnlag for å tro at det er overdispersjon i dataene? Diskuter konkrete mekanismar som kan generere overdispersjon i den konkrete situasjonen vi har modellert.

**Oppgåve 4** Gå ut i frå at $x_1, x_2, \ldots, x_n$ er uavhengige observasjonar frå ein Gamma-fordeling med sannsynstettleiksfunksjon

$$f(x) = \frac{1}{\sigma^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\sigma}, \text{for } x > 0.$$

Sjå eventuellt hjelpesidane for informasjon om den matematiske funksjonen $\Gamma(\alpha)$.

**a)** Vi ønskjer å estimere dei ukjende parameterane $\alpha$ og $\sigma$. Skriv opp eit matematisk uttrykk for likelihood- og log-likelihoodfunksjonen. Skriv også ein R funksjon som rekner ut log likelihoodet for gitte parameterverde og for eit gjeve tilfeldig utval $x_1, x_2, \ldots, x_n$ representert i R på passande måte. Forklar kort med ord kva vi meiner med sannsynsmaksimeringsestimatorane av dei ukjende parameterane i modellen og korleis disse kan reknast ut numerisk i R.

`power.t.test` *Power calculations for one and two sample t tests*

### Description

Compute the power of the one- or two- sample t test, or determine parameters to obtain a target power.

### Usage

```
power.t.test(n = NULL, delta = NULL, sd = 1, sig.level = 0.05,
             power = NULL,
             type = c("two.sample", "one.sample", "paired"),
             alternative = c("two.sided", "one.sided"),
             strict = FALSE, tol = .Machine$double.eps^0.25)
```

### Arguments

| | |
|---|---|
| `n` | number of observations (per group) |
| `delta` | true difference in means |
| `sd` | standard deviation |
| `sig.level` | significance level (Type I error probability) |
| `power` | power of test (1 minus Type II error probability) |
| `type` | string specifying the type of t test. Can be abbreviated. |
| `alternative` | one- or two-sided test. Can be abbreviated. |
| `strict` | use strict interpretation in two-sided case |
| `tol` | numerical tolerance used in root finding, the default providing (at least) four significant digits. |

### Details

Exactly one of the parameters `n`, `delta`, `power`, `sd`, and `sig.level` must be passed as `NULL`, and that parameter is determined from the others. Notice that the last two have non-NULL defaults, so NULL must be explicitly passed if you want to compute them.

If `strict = TRUE` is used, the power will include the probability of rejection in the opposite direction of the true effect, in the two-sided case. Without this the power will be half the significance level if the true difference is zero.

### Value

Object of class `"power.htest"`, a list of the arguments (including the computed one) augmented with `method` and `note` elements.

### Note

`uniroot` is used to solve the power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

### Author(s)

Peter Dalgaard. Based on previous work by Claus Ekstroem

### See Also

`t.test`, `uniroot`

### Examples

```
power.t.test(n = 20, delta = 1)
power.t.test(power = .90, delta = 1)
power.t.test(power = .90, delta = 1, alternative = "one.sided")
```

`Binomial` *The Binomial Distribution*

### Description

Density, distribution function, quantile function and random generation for the binomial distribution with parameters `size` and `prob`.

This is conventionally interpreted as the number of 'successes' in `size` trials.

### Usage

```
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)
```

### 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. |
| `size` | number of trials (zero or more). |
| `prob` | probability of success on each trial. |
| `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 binomial distribution with `size` $= n$ and `prob` $= p$ has density

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

for $x = 0, \ldots, n$. Note that binomial *coefficients* can be computed by `choose` in R.

If an element of `x` is not integer, the result of `dbinom` is zero, with a warning.

$p(x)$ is computed using Loader's algorithm, see the reference below.

The quantile is defined as the smallest value $x$ such that $F(x) \ge p$, where $F$ is the distribution function.

### Value

`dbinom` gives the density, `pbinom` gives the distribution function, `qbinom` gives the quantile function and `rbinom` generates random deviates.

If `size` is not an integer, `NaN` is returned.

The length of the result is determined by `n` for `rbinom`, 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.

### Source

For `dbinom` a saddle-point expansion is used: see

Catherine Loader (2000). *Fast and Accurate Computation of Binomial Probabilities*; available from `http://www.herine.net/stat/software/dbinom.html`.

`pbinom` uses `pbeta`.

`qbinom` uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

`rbinom` (for `size < .Machine$integer.max`) is based on

Kachitvichyanukul, V. and Schmeiser, B. W. (1988) Binomial random variate generation. *Communications of the ACM*, **31**, 216–222.

For larger values it uses inversion.

### See Also

Distributions for other standard distributions, including `dnbinom` for the negative binomial, and `dpois` for the Poisson distribution.

### Examples

```
require(graphics)
# Compute P(45 < X < 55) for X Binomial(100,0.5)
sum(dbinom(46:54, 100, 0.5))

## Using "log = TRUE" for an extended range :
n <- 2000
k <- seq(0, n, by = 20)
plot (k, dbinom(k, n, pi/10, log = TRUE), type = "l", ylab = "log density",
      main = "dbinom(*, log=TRUE) is better than  log(dbinom(*))")
lines(k, log(dbinom(k, n, pi/10)), col = "red", lwd = 2)
## extreme points are omitted since dbinom gives 0.
mtext("dbinom(k, log=TRUE)", adj = 0)
mtext("extended range", adj = 0, line = -1, font = 4)
mtext("log(dbinom(k))", col = "red", adj = 1)
```

GammaDist *The Gamma Distribution*

### Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

### Usage

```
dgamma(x, shape, rate = 1, scale = 1/rate, log = FALSE)
pgamma(q, shape, rate = 1, scale = 1/rate, lower.tail = TRUE,
       log.p = FALSE)
qgamma(p, shape, rate = 1, scale = 1/rate, lower.tail = TRUE,
       log.p = FALSE)
rgamma(n, shape, rate = 1, scale = 1/rate)
```

### 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. |
| rate | an alternative way to specify the scale. |
| shape, scale | shape and scale parameters. Must be positive, scale strictly. |
| log, log.p | logical; if TRUE, probabilities/densities $p$ are returned as $log(p)$. |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \le x]$, otherwise, $P[X > x]$. |

### Details

If scale is omitted, it assumes the default value of 1.

The Gamma distribution with parameters shape $= \alpha$ and scale $= \sigma$ has density

$$f(x) = \frac{1}{\sigma^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\sigma}$$

for $x \ge 0$, $\alpha > 0$ and $\sigma > 0$. (Here $\Gamma(\alpha)$ is the function implemented by R's gamma() and defined in its help. Note that $\alpha = 0$ corresponds to the trivial distribution with all mass at point 0.)

The mean and variance are $E(X) = \alpha\sigma$ and $Var(X) = \alpha\sigma^2$.

The cumulative hazard $H(t) = -\log(1 - F(t))$ is

```
-pgamma(t, ..., lower = FALSE, log = TRUE)
```

Note that for smallish values of shape (and moderate scale) a large parts of the mass of the Gamma distribution is on values of $x$ so near zero that they will be represented as zero in computer arithmetic. So rgamma may well return values which will be represented as zero. (This will also happen for very large values of scale since the actual generation is done for scale = 1.)

### Value

dgamma gives the density, pgamma gives the distribution function, qgamma gives the quantile function, and rgamma generates random deviates.

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

The length of the result is determined by n for rgamma, 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

The S (Becker *et al* (1988) parametrization was via shape and rate: S had no scale parameter. In R 2.x.y scale took precedence over rate, but now it is an error to supply both.

pgamma is closely related to the incomplete gamma function. As defined by Abramowitz and Stegun 6.5.1 (and by 'Numerical Recipes') this is

$$P(a,x) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt$$

$P(a,x)$ is pgamma(x, a). Other authors (for example Karl Pearson in his 1922 tables) omit the normalizing factor, defining the incomplete gamma function $\gamma(a,x)$ as $\gamma(a,x) = \int_0^x t^{a-1} e^{-t} dt$, i.e., pgamma(x, a) * gamma(a). Yet other use the 'upper' incomplete gamma function,

$$\Gamma(a,x) = \int_x^\infty t^{a-1} e^{-t} dt,$$

which can be computed by pgamma(x, a, lower = FALSE) * gamma(a).

Note however that pgamma(x, a, ..) currently requires $a > 0$, whereas the incomplete gamma function is also defined for negative $a$. In that case, you can use gamma_inc(a,x) (for $\Gamma(a,x)$) from package **gsl**.

See also http://en.wikipedia.org/wiki/Incomplete_gamma_function, or http://dlmf.nist.gov/8.2#i.

### Source

dgamma is computed via the Poisson density, using code contributed by Catherine Loader (see dbinom).

pgamma uses an unpublished (and not otherwise documented) algorithm 'mainly by Morten Welinder'.

qgamma is based on a C translation of

Best, D. J. and D. E. Roberts (1975). Algorithm AS91. Percentage points of the chi-squared distribution. *Applied Statistics*, **24**, 385–388.

plus a final Newton step to improve the approximation.

rgamma for shape >= 1 uses

Ahrens, J. H. and Dieter, U. (1982). Generating gamma variates by a modified rejection technique. *Communications of the ACM*, **25**, 47–54,

and for 0 < shape < 1 uses

Ahrens, J. H. and Dieter, U. (1974). Computer methods for sampling from gamma, beta, Poisson and binomial distributions. *Computing*, **12**, 223–246.

### References


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

Shea, B. L. (1988) Algorithm AS 239, Chi-squared and incomplete Gamma integral, *Applied Statistics (JRSS C)* **37**, 466–473.

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

NIST Digital Library of Mathematical Functions. http://dlmf.nist.gov/, section 8.2.


### See Also

gamma for the gamma function.

Distributions for other standard distributions, including dbeta for the Beta distribution and dchisq for the chi-squared distribution which is a special case of the Gamma distribution.

### Examples

```
-log(dgamma(1:4, shape = 1))
p <- (1:9)/10
pgamma(qgamma(p, shape = 2), shape = 2)
1 - 1/exp(qgamma(p, shape = 1))

# even for shape = 0.001 about half the mass is on numbers
# that cannot be represented accurately (and most of those as zero)
pgamma(.Machine$double.xmin, 0.001)
pgamma(5e-324, 0.001)  # on most machines 5e-324 is the smallest
                       # representable non-zero number
table(rgamma(1e4, 0.001) == 0)/1e4
```

Special *Special Functions of Mathematics*

### Description

Special mathematical functions related to the beta and gamma functions.

### Usage

```
beta(a, b)
lbeta(a, b)

gamma(x)
lgamma(x)
psigamma(x, deriv = 0)
digamma(x)
trigamma(x)

choose(n, k)
lchoose(n, k)
factorial(x)
lfactorial(x)
```

### Arguments

| | |
|---|---|
| a, b | non-negative numeric vectors. |
| x, n | numeric vectors. |
| k, deriv | integer vectors. |

### Details

The functions beta and lbeta return the beta function and the natural logarithm of the beta function,

$$B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}.$$

The formal definition is

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

(Abramowitz and Stegun section 6.2.1, page 258). Note that it is only defined in R for non-negative a and b, and is infinite if either is zero.

The functions gamma and lgamma return the gamma function $\Gamma(x)$ and the natural logarithm of *the absolute value of* the gamma function. The gamma function is defined by (Abramowitz and Stegun section 6.1.1, page 255)

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

for all real x except zero and negative integers (when NaN is returned). There will be a warning on possible loss of precision for values which are too close (within about $10^{-8}$)) to a negative integer less than '-10'.

`factorial(x)` ($x!$ for non-negative integer `x`) is defined to be `gamma(x+1)` and `lfactorial` to be `lgamma(x+1)`.

The functions `digamma` and `trigamma` return the first and second derivatives of the logarithm of the gamma function. `psigamma(x, deriv)` (`deriv >= 0`) computes the `deriv`-th derivative of $\psi(x)$.

$$\texttt{digamma(x)} = \psi(x) = \frac{d}{dx}\ln\Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}$$

$\psi$ and its derivatives, the `psigamma()` functions, are often called the 'polygamma' functions, e.g. in Abramowitz and Stegun (section 6.4.1, page 260); and higher derivatives (`deriv = 2:4`) have occasionally been called 'tetragamma', 'pentagamma', and 'hexagamma'.

The functions `choose` and `lchoose` return binomial coefficients and the logarithms of their absolute values. Note that `choose(n, k)` is defined for all real numbers $n$ and integer $k$. For $k \geq 1$ it is defined as $n(n-1)\cdots(n-k+1)/k!$, as 1 for $k = 0$ and as 0 for negative $k$. Non-integer values of `k` are rounded to an integer, with a warning.
`choose(*, k)` uses direct arithmetic (instead of `[l]gamma` calls) for small `k`, for speed and accuracy reasons. Note the function `combn` (package **utils**) for enumeration of all possible combinations.

The `gamma`, `lgamma`, `digamma` and `trigamma` functions are internal generic primitive functions: methods can be defined for them individually or via the `Math` group generic.

### Source

`gamma`, `lgamma`, `beta` and `lbeta` are based on C translations of Fortran subroutines by W. Fullerton of Los Alamos Scientific Laboratory (now available as part of SLATEC).

`digamma`, `trigamma` and `psigamma` are based on

Amos, D. E. (1983). A portable Fortran subroutine for derivatives of the psi function, Algorithm 610, *ACM Transactions on Mathematical Software* **9(4)**, 494–502.

### References

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

Abramowitz, M. and Stegun, I. A. (1972) *Handbook of Mathematical Functions*. New York: Dover. http://en.wikipedia.org/wiki/Abramowitz_and_Stegun provides links to the full text which is in public domain.
Chapter 6: Gamma and Related Functions.

### See Also

`Arithmetic` for simple, `sqrt` for miscellaneous mathematical functions and `Bessel` for the real Bessel functions.

For the incomplete gamma function see `pgamma`.

### Examples

```
require(graphics)

choose(5, 2)
for (n in 0:10) print(choose(n, k = 0:n))

factorial(100)
lfactorial(10000)


## gamma has 1st order poles at 0, -1, -2, ...
## this will generate loss of precision warnings, so turn off
op <- options("warn")
options(warn = -1)
x <- sort(c(seq(-3, 4, length.out = 201), outer(0:-3, (-1:1)*1e-6, "+")))
plot(x, gamma(x), ylim = c(-20,20), col = "red", type = "l", lwd = 2,
     main = expression(Gamma(x)))
abline(h = 0, v = -3:0, lty = 3, col = "midnightblue")
options(op)

x <- seq(0.1, 4, length.out = 201); dx <- diff(x)[1]
par(mfrow = c(2, 3))
for (ch in c("", "l","di","tri","tetra","penta")) {
  is.deriv <- nchar(ch) >= 2
  nm <- paste0(ch, "gamma")
  if (is.deriv) {
    dy <- diff(y) / dx # finite difference
    der <- which(ch == c("di","tri","tetra","penta")) - 1
    nm2 <- paste0("psigamma(*, deriv = ", der,")")
    nm  <- if(der >= 2) nm2 else paste(nm, nm2, sep = " ==\n")
    y <- psigamma(x, deriv = der)
  } else {
    y <- get(nm)(x)
  }
  plot(x, y, type = "l", main = nm, col = "red")
  abline(h = 0, col = "lightgray")
  if (is.deriv) lines(x[-1], dy, col = "blue", lty = 2)
}
par(mfrow = c(1, 1))

## "Extended" Pascal triangle:
fN <- function(n) formatC(n, width=2)
for (n in -4:10) {
    cat(fN(n),":", fN(choose(n, k = -2:max(3, n+2))))
    cat("\n")
}

## R code version of choose()  [simplistic; warning for k < 0]:
mychoose <- function(r, k)
    ifelse(k <= 0, (k == 0),
           sapply(k, function(k) prod(r:(r-k+1))) / factorial(k))
k <- -1:6
cbind(k = k, choose(1/2, k), mychoose(1/2, k))

## Binomial theorem for n = 1/2 ;
## sqrt(1+x) = (1+x)^(1/2) = sum_{k=0}^Inf  choose(1/2, k) * x^k :
k <- 0:10 # 10 is sufficient for ~ 9 digit precision:
sqrt(1.25)
sum(choose(1/2, k)* .25^k)
```