CITED REFERENCES AND FURTHER READING:

Abramowitz, M., and Stegun, I.A. 1964, *Handbook of Mathematical Functions* (Washington: National Bureau of Standards); reprinted 1968 (New York: Dover); online at `http://www.nr.com/aands`.[1]

6.14 Statistical Functions

Certain special functions get frequent use because of their relation to common univariate statistical distributions, that is, probability densities in a single variable. In this section we survey a number of such common distributions in a unified way, giving, in each case, routines for computing the probability density function $p(x)$; the cumulative density function or *cdf*, written $P(< x)$; and the inverse of the cumulative density function $x(P)$. The latter function is needed for finding the values of x associated with specified *percentile points* or *quantiles* in significance tests, for example, the 0.5%, 5%, 95% or 99.5% points.

The emphasis of this section is on defining and computing these statistical functions. Section §7.3 is a related section that discusses how to generate random deviates from the distributions discussed here. We defer discussion of the actual use of these distributions in statistical tests to Chapter 14.

6.14.1 Normal (Gaussian) Distribution

If x is drawn from a *normal distribution* with mean μ and standard deviation σ, then we write

$$\begin{aligned} x &\sim \mathrm{N}(\mu,\sigma), \qquad \sigma > 0 \\ p(x) &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right) \end{aligned} \tag{6.14.1}$$

with $p(x)$ the probability density function. Note the special use of the notation "$\sim$" in this section, which can be read as "is drawn from a distribution." The variance of the distribution is, of course, σ^2.

The cumulative distribution function is the probability of a value $\le x$. For the normal distribution, this is given in terms of the complementary error function by

$$\mathrm{cdf} \equiv P(< x) \equiv \int_{-\infty}^{x} p(x')dx' = \frac{1}{2}\mathrm{erfc}\left(-\frac{1}{\sqrt{2}}\left[\frac{x-\mu}{\sigma}\right]\right) \tag{6.14.2}$$

The inverse cdf can thus be calculated in terms of the inverse of erfc,

$$x(P) = \mu - \sqrt{2}\sigma\,\mathrm{erfc}^{-1}(2P) \tag{6.14.3}$$

The following structure implements the above relations.

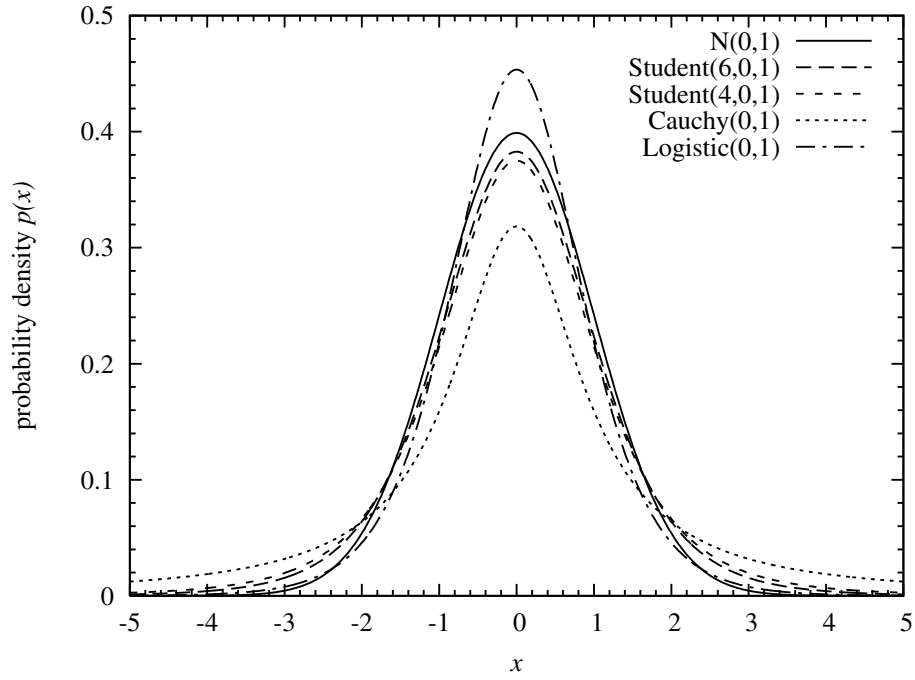

Figure 6.14.1. Examples of centrally peaked distributions that are symmetric on the real line. Any of these can substitute for the normal distribution either as an approximation or in applications such as robust estimation. They differ largely in the decay rate of their tails.

erf.h

```
struct Normaldist : Erf {
Normal distribution, derived from the error function Erf.
    Doub mu, sig;
    Normaldist(Doub mmu = 0., Doub ssig = 1.) : mu(mmu), sig(ssig) {
    Constructor. Initialize with μ and σ. The default with no arguments is N(0, 1).
        if (sig <= 0.) throw("bad sig in Normaldist");
    }
    Doub p(Doub x) {
    Return probability density function.
        return (0.398942280401432678/sig)*exp(-0.5*SQR((x-mu)/sig));
    }
    Doub cdf(Doub x) {
    Return cumulative distribution function.
        return 0.5*erfc(-0.707106781186547524*(x-mu)/sig);
    }
    Doub invcdf(Doub p) {
    Return inverse cumulative distribution function.
        if (p <= 0. || p >= 1.) throw("bad p in Normaldist");
        return -1.41421356237309505*sig*inverfc(2.*p)+mu;
    }
};
```

We will use the conventions of the above code for all the distributions in this section. A distribution's parameters (here, μ and σ) are set by the constructor and then referenced as needed by the member functions. The density function is always `p()`, the cdf is `cdf()`, and the inverse cdf is `invcdf()`. We will generally check the arguments of probability functions for validity, since many program bugs can show up as, e.g., a probability out of the range $[0, 1]$.

6.14.2 Cauchy Distribution

Like the normal distribution, the *Cauchy distribution* is a centrally peaked, symmetric distribution with a parameter μ that specifies its center and a parameter σ that specifies its width. *Unlike* the normal distribution, the Cauchy distribution has tails that decay very slowly at infinity, as $|x|^{-2}$, so slowly that moments higher than the zeroth moment (the area under the curve) don't even exist. The parameter μ is therefore, strictly speaking, not the mean, and the parameter σ is not, technically, the standard deviation. But these two parameters substitute for those moments as measures of central position and width.

The defining probability density is

$$x \sim \text{Cauchy}(\mu, \sigma), \qquad \sigma > 0$$
$$p(x) = \frac{1}{\pi\sigma}\left(1 + \left[\frac{x-\mu}{\sigma}\right]^2\right)^{-1} \tag{6.14.4}$$

If $x \sim \text{Cauchy}(0,1)$, then also $1/x \sim \text{Cauchy}(0,1)$ and also $(ax+b)^{-1} \sim \text{Cauchy}(-b/a, 1/a)$.

The cdf is given by

$$\text{cdf} \equiv P(<x) \equiv \int_{-\infty}^{x} p(x')dx' = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x-\mu}{\sigma}\right) \tag{6.14.5}$$

The inverse cdf is given by

$$x(P) = \mu + \sigma\tan\left(\pi[P - \tfrac{1}{2}]\right) \tag{6.14.6}$$

Figure 6.14.1 shows Cauchy$(0,1)$ as compared to the normal distribution N$(0,1)$, as well as several other similarly shaped distributions discussed below.

The Cauchy distribution is sometimes called the *Lorentzian distribution*.

distributions.h

```
struct Cauchydist {
```
Cauchy distribution.
```
    Doub mu, sig;
    Cauchydist(Doub mmu = 0., Doub ssig = 1.) : mu(mmu), sig(ssig) {
```
    Constructor. Initialize with μ and σ. The default with no arguments is Cauchy$(0,1)$.
```
        if (sig <= 0.) throw("bad sig in Cauchydist");
    }
    Doub p(Doub x) {
```
    Return probability density function.
```
        return 0.318309886183790671/(sig*(1.+SQR((x-mu)/sig)));
    }
    Doub cdf(Doub x) {
```
    Return cumulative distribution function.
```
        return 0.5+0.318309886183790671*atan2(x-mu,sig);
    }
    Doub invcdf(Doub p) {
```
    Return inverse cumulative distribution function.
```
        if (p <= 0. || p >= 1.) throw("bad p in Cauchydist");
        return mu + sig*tan(3.14159265358979324*(p-0.5));
    }
};
```

6.14.3 Student-t Distribution

A generalization of the Cauchy distribution is the Student-t distribution, named for the early 20th century statistician William Gosset, who published under the name "Student" because his employer, Guinness Breweries, required him to use a pseudonym. Like the Cauchy distribution, the Student-t distribution has power-law tails at infinity, but it has an additional parameter ν that specifies how rapidly they decay, namely as $|t|^{-(\nu+1)}$. When ν is an integer, the number of convergent moments, including the zeroth, is thus ν.

The defining probability density (conventionally written in a variable t instead of x) is

$$t \sim \text{Student}(\nu, \mu, \sigma), \qquad \nu > 0,\ \sigma > 0$$
$$p(t) = \frac{\Gamma(\frac{1}{2}[\nu+1])}{\Gamma(\frac{1}{2}\nu)\sqrt{\nu\pi}\sigma}\left(1 + \frac{1}{\nu}\left[\frac{t-\mu}{\sigma}\right]^2\right)^{-\frac{1}{2}(\nu+1)} \tag{6.14.7}$$

The Cauchy distribution is obtained in the case $\nu = 1$. In the opposite limit, $\nu \to \infty$, the normal distribution is obtained. In pre-computer days, this was the basis of various approximation schemes for the normal distribution, now all generally irrelevant. Figure 6.14.1 shows examples of the Student-t distribution for $\nu = 1$ (Cauchy), $\nu = 4$, and $\nu = 6$. The approach to the normal distribution is evident.

The mean of Student(ν, μ, σ) is (by symmetry) μ. The variance is not σ^2, but rather

$$\text{Var}\{\text{Student}(\nu, \mu, \sigma)\} = \frac{\nu}{\nu - 2}\sigma^2 \tag{6.14.8}$$

For additional moments, and other properties, see [1].

The cdf is given by an incomplete beta function. If we let

$$x \equiv \frac{\nu}{\nu + \left(\frac{t-\mu}{\sigma}\right)^2} \tag{6.14.9}$$

then

$$\text{cdf} \equiv P(<t) \equiv \int_{-\infty}^{t} p(t')dt' = \begin{cases} \frac{1}{2}I_x(\frac{1}{2}\nu, \frac{1}{2}), & t \le \mu \\ 1 - \frac{1}{2}I_x(\frac{1}{2}\nu, \frac{1}{2}), & t > \mu \end{cases} \tag{6.14.10}$$

The inverse cdf is given by an inverse incomplete beta function (see code below for the exact formulation).

In practice, the Student-t cdf is the above form is rarely used, since most statistical tests using Student-t are double-sided. Conventionally, the two-tailed function $A(t|\nu)$ is defined (only) for the case $\mu = 0$ and $\sigma = 1$ by

$$A(t|\nu) \equiv \int_{-t}^{+t} p(t')dt' = 1 - I_x(\tfrac{1}{2}\nu, \tfrac{1}{2}) \tag{6.14.11}$$

with x as given above. The statistic $A(t|\nu)$ is notably used in the test of whether two observed distributions have the same mean. The code below implements both equations (6.14.10) and (6.14.11), as well as their inverses.

ncgammabeta.h

```
struct Studenttdist : Beta {
```
Student-t distribution, derived from the beta function Beta.
```
    Doub nu, mu, sig, np, fac;
    Studenttdist(Doub nnu, Doub mmu = 0., Doub ssig = 1.)
    : nu(nnu), mu(mmu), sig(ssig) {
```
Constructor. Initialize with ν, μ and σ. The default with one argument is Student$(\nu, 0, 1)$.
```
        if (sig <= 0. || nu <= 0.) throw("bad sig,nu in Studentdist");
        np = 0.5*(nu + 1.);
        fac = gammln(np)-gammln(0.5*nu);
    }
    Doub p(Doub t) {
```
Return probability density function.
```
        return exp(-np*log(1.+SQR((t-mu)/sig)/nu)+fac)
            /(sqrt(3.14159265358979324*nu)*sig);
    }
    Doub cdf(Doub t) {
```
Return cumulative distribution function.
```
        Doub p = 0.5*betai(0.5*nu, 0.5, nu/(nu+SQR((t-mu)/sig)));
        if (t >= mu) return 1. - p;
        else return p;
    }
    Doub invcdf(Doub p) {
```
Return inverse cumulative distribution function.
```
        if (p <= 0. || p >= 1.) throw("bad p in Studentdist");
        Doub x = invbetai(2.*MIN(p,1.-p), 0.5*nu, 0.5);
        x = sig*sqrt(nu*(1.-x)/x);
        return (p >= 0.5? mu+x : mu-x);
    }
    Doub aa(Doub t) {
```
Return the two-tailed cdf $A(t|\nu)$.
```
        if (t < 0.) throw("bad t in Studentdist");
        return 1.-betai(0.5*nu, 0.5, nu/(nu+SQR(t)));
    }
    Doub invaa(Doub p) {
```
Return the inverse, namely t such that $p = A(t|\nu)$.
```
        if (p < 0. || p >= 1.) throw("bad p in Studentdist");
        Doub x = invbetai(1.-p, 0.5*nu, 0.5);
        return sqrt(nu*(1.-x)/x);
    }
};
```

6.14.4 Logistic Distribution

The *logistic distribution* is another symmetric, centrally peaked distribution that can be used instead of the normal distribution. Its tails decay exponentially, but still much more slowly than the normal distribution's "exponent of the square."

The defining probability density is

$$p(y) = \frac{e^{-y}}{(1+e^{-y})^2} = \frac{e^{y}}{(1+e^{y})^2} = \tfrac{1}{4}\operatorname{sech}^2\left(\tfrac{1}{2}y\right) \tag{6.14.12}$$

The three forms are algebraically equivalent, but, to avoid overflows, it is wise to use the negative and positive exponential forms for positive and negative values of y, respectively.

The variance of the distribution (6.14.12) turns out to be $\pi^2/3$. Since it is convenient to have parameters μ and σ with the conventional meanings of mean and standard deviation, equation (6.14.12) is often replaced by the *standardized logistic*

distribution,

$$x \sim \text{Logistic}(\mu, \sigma), \qquad \sigma > 0$$
$$p(x) = \frac{\pi}{4\sqrt{3}\sigma} \text{sech}^2 \left(\frac{\pi}{2\sqrt{3}} \left[\frac{x-\mu}{\sigma} \right] \right) \tag{6.14.13}$$

which implies equivalent forms using the positive and negative exponentials (see code below).

The cdf is given by

$$\text{cdf} \equiv P(< x) \equiv \int_{-\infty}^{x} p(x')dx' = \left[1 + \exp\left(-\frac{\pi}{\sqrt{3}} \left[\frac{x-\mu}{\sigma} \right] \right) \right]^{-1} \tag{6.14.14}$$

The inverse cdf is given by

$$x(P) = \mu + \frac{\sqrt{3}}{\pi} \sigma \log \left(\frac{P}{1-P} \right) \tag{6.14.15}$$

distributions.h

```
struct Logisticdist {
Logistic distribution.
    Doub mu, sig;
    Logisticdist(Doub mmu = 0., Doub ssig = 1.) : mu(mmu), sig(ssig) {
    Constructor. Initialize with μ and σ. The default with no arguments is Logistic(0, 1).
        if (sig <= 0.) throw("bad sig in Logisticdist");
    }
    Doub p(Doub x) {
    Return probability density function.
        Doub e = exp(-abs(1.81379936423421785*(x-mu)/sig));
        return 1.81379936423421785*e/(sig*SQR(1.+e));
    }
    Doub cdf(Doub x) {
    Return cumulative distribution function.
        Doub e = exp(-abs(1.81379936423421785*(x-mu)/sig));
        if (x >= mu) return 1./(1.+e);          Because we used abs to control over-
        else return e/(1.+e);                       flow, we now have two cases.
    }
    Doub invcdf(Doub p) {
    Return inverse cumulative distribution function.
        if (p <= 0. || p >= 1.) throw("bad p in Logisticdist");
        return mu + 0.5513288954217920 49*sig*log(p/(1.-p));
    }
};
```

The logistic distribution is cousin to the *logit transformation* that maps the open unit interval $0 < p < 1$ onto the real line $-\infty < u < \infty$ by the relation

$$u = \log \left(\frac{p}{1-p} \right) \tag{6.14.16}$$

Back when a book of tables and a slide rule were a statistician's working tools, the logit transformation was used to approximate processes on the interval by analytically simpler processes on the real line. A uniform distribution on the interval maps by the logit transformation to a logistic distribution on the real line. With the computer's ability to calculate distributions on the interval directly (beta distributions, for example), that motivation has vanished.

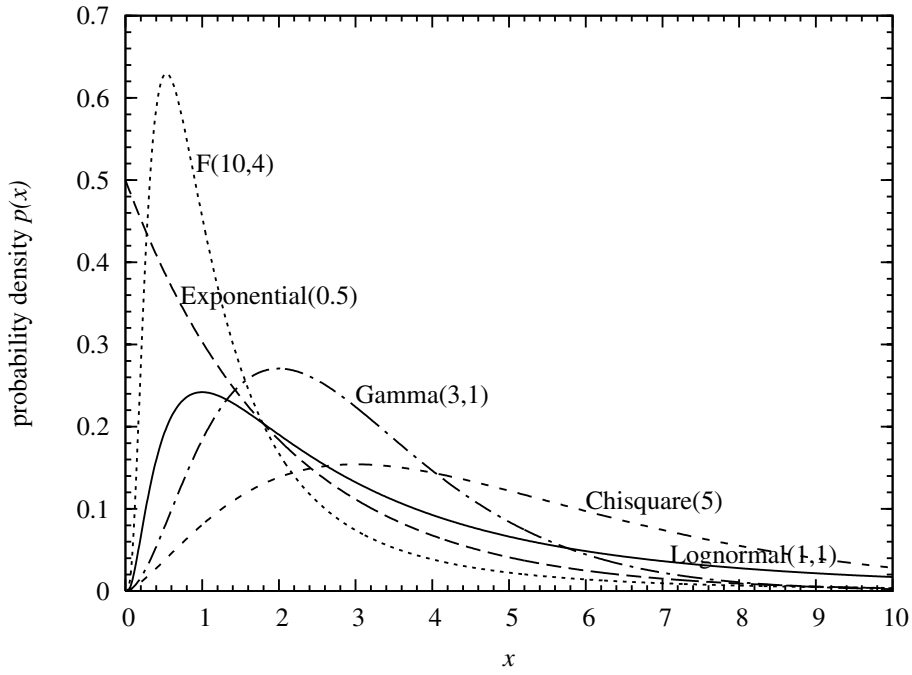

Figure 6.14.2. Examples of common distributions on the half-line $x > 0$.

Another cousin is the *logistic equation*,

$$\frac{dy}{dt} \propto y(y_{\max} - y) \tag{6.14.17}$$

a differential equation describing the growth of some quantity y, starting off as an exponential but reaching, asymptotically, a value $y_{\max}$. The solution of this equation is identical, up to a scaling, to the cdf of the logistic distribution.

6.14.5 Exponential Distribution

With the *exponential distribution* we now turn to common distribution functions defined on the positive real axis $x \geq 0$. Figure 6.14.2 shows examples of several of the distributions that we will discuss. The exponential is the simplest of them all. It has a parameter β that can control its width (in inverse relationship), but its mode is always at zero:

$$\begin{aligned} x &\sim \text{Exponential}(\beta), \qquad \beta > 0 \\ p(x) &= \beta \exp(-\beta x), \qquad x > 0 \end{aligned} \tag{6.14.18}$$

$$\text{cdf} \equiv P(< x) \equiv \int_0^x p(x')dx' = 1 - \exp(-\beta x) \tag{6.14.19}$$

$$x(P) = -\frac{1}{\beta}\log(1 - P) \tag{6.14.20}$$

The mean and standard deviation of the exponential distribution are both $1/\beta$. The median is $\log(2)/\beta$. Reference [1] has more to say about the exponential distribution than you would ever think possible.

distributions.h

```
struct Expondist {
Exponential distribution.
    Doub bet;
    Expondist(Doub bbet) : bet(bbet) {
    Constructor. Initialize with β.
        if (bet <= 0.) throw("bad bet in Expondist");
    }
    Doub p(Doub x) {
    Return probability density function.
        if (x < 0.) throw("bad x in Expondist");
        return bet*exp(-bet*x);
    }
    Doub cdf(Doub x) {
    Return cumulative distribution function.
        if (x < 0.) throw("bad x in Expondist");
        return 1.-exp(-bet*x);
    }
    Doub invcdf(Doub p) {
    Return inverse cumulative distribution function.
        if (p < 0. || p >= 1.) throw("bad p in Expondist");
        return -log(1.-p)/bet;
    }
};
```

6.14.6 Weibull Distribution

The Weibull distribution generalizes the exponential distribution in a way that is often useful in hazard, survival, or reliability studies. When the lifetime (time to failure) of an item is exponentially distributed, there is a constant probability per unit time that an item will fail, if it has not already done so. That is,

$$\text{hazard} \equiv \frac{p(x)}{P(>x)} \propto \text{constant} \tag{6.14.21}$$

Exponentially lived items don't age; they just keep rolling the same dice until, one day, their number comes up. In many other situations, however, it is observed that an item's hazard (as defined above) does change with time, say as a power law,

$$\frac{p(x)}{P(>x)} \propto x^{\alpha-1}, \qquad \alpha > 0 \tag{6.14.22}$$

The distribution that results is the Weibull distribution, named for Swedish physicist Waloddi Weibull, who used it as early as 1939. When $\alpha > 1$, the hazard increases with time, as for components that wear out. When $0 < \alpha < 1$, the hazard decreases with time, as for components that experience "infant mortality."

We say that

$$x \sim \text{Weibull}(\alpha, \beta) \qquad \text{iff} \qquad y \equiv \left(\frac{x}{\beta}\right)^{\alpha} \sim \text{Exponential}(1) \tag{6.14.23}$$

The probability density is

$$p(x) = \left(\frac{\alpha}{\beta}\right)\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^{\alpha}}, \qquad x > 0 \tag{6.14.24}$$

The cdf is

$$\text{cdf} \equiv P(< x) \equiv \int_0^x p(x')dx' = 1 - e^{-(x/\beta)^\alpha} \tag{6.14.25}$$

The inverse cdf is

$$x(P) = \beta\,[-\log(1-P)]^{1/\alpha} \tag{6.14.26}$$

For $0 < \alpha < 1$, the distribution has an infinite (but integrable) cusp at $x = 0$ and is monotonically decreasing. The exponential distribution is the case of $\alpha = 1$. When $\alpha > 1$, the distribution is zero at $x = 0$ and has a single maximum at the value $x = \beta\,[(\alpha-1)/\alpha]^{1/\alpha}$.

The mean and variance are given by

$$\begin{aligned} \mu &= \beta\,\Gamma(1+\alpha^{-1}) \\ \sigma^2 &= \beta^2\left\{\Gamma(1+2\alpha^{-1}) - \left[\Gamma(1+\alpha^{-1})\right]^2\right\} \end{aligned} \tag{6.14.27}$$

With correct normalization, equation (6.14.22) becomes

$$\text{hazard} \equiv \frac{p(x)}{P(>x)} = \left(\frac{\alpha}{\beta}\right)\left(\frac{x}{\beta}\right)^{\alpha-1} \tag{6.14.28}$$

6.14.7 Lognormal Distribution

Many processes that live on the positive x-axis are naturally approximated by normal distributions on the "$\log(x)$-axis," that is, for $-\infty < \log(x) < \infty$. A simple, but important, example is the multiplicative random walk, which starts at some positive value x_0, and then generates new values by a recurrence like

$$x_{i+1} = \begin{cases} x_i(1+\epsilon) & \text{with probability } 0.5 \\ x_i/(1+\epsilon) & \text{with probability } 0.5 \end{cases} \tag{6.14.29}$$

Here ϵ is some small, fixed, constant.

These considerations motivate the definition

$$x \sim \text{Lognormal}(\mu,\sigma) \qquad \text{iff} \qquad u \equiv \frac{\log(x)-\mu}{\sigma} \sim \text{N}(0,1) \tag{6.14.30}$$

or the equivalent definition

$$\begin{aligned} &x \sim \text{Lognormal}(\mu,\sigma), \qquad \sigma > 0 \\ &p(x) = \frac{1}{\sqrt{2\pi}\sigma x}\exp\left(-\frac{1}{2}\left[\frac{\log(x)-\mu}{\sigma}\right]^2\right), \qquad x > 0 \end{aligned} \tag{6.14.31}$$

Note the required extra factor of x^{-1} in front of the exponential: The density that is "normal" is $p(\log x)d\log x$.

While μ and σ are the mean and standard deviation in $\log x$ space, they are *not* so in x space. Rather,

$$\begin{aligned} \text{Mean}\{\text{Lognormal}(\mu,\sigma)\} &= e^{\mu+\frac{1}{2}\sigma^2} \\ \text{Var}\{\text{Lognormal}(\mu,\sigma)\} &= e^{2\mu}\,e^{\sigma^2}(e^{\sigma^2}-1) \end{aligned} \tag{6.14.32}$$

The cdf is given by

$$\text{cdf} \equiv P(< x) \equiv \int_0^x p(x')dx' = \frac{1}{2}\text{erfc}\left(-\frac{1}{\sqrt{2}}\left[\frac{\log(x)-\mu}{\sigma}\right]\right) \tag{6.14.33}$$

The inverse to the cdf involves the inverse complementary error function,

$$x(P) = \exp[\mu - \sqrt{2}\sigma\,\text{erfc}^{-1}(2P)] \tag{6.14.34}$$

erf.h

```
struct Lognormaldist : Erf {
Lognormal distribution, derived from the error function Erf.
    Doub mu, sig;
    Lognormaldist(Doub mmu = 0., Doub ssig = 1.) : mu(mmu), sig(ssig) {
        if (sig <= 0.) throw("bad sig in Lognormaldist");
    }
    Doub p(Doub x) {
    Return probability density function.
        if (x < 0.) throw("bad x in Lognormaldist");
        if (x == 0.) return 0.;
        return (0.398942280401432678/(sig*x))*exp(-0.5*SQR((log(x)-mu)/sig));
    }
    Doub cdf(Doub x) {
    Return cumulative distribution function.
        if (x < 0.) throw("bad x in Lognormaldist");
        if (x == 0.) return 0.;
        return 0.5*erfc(-0.707106781186547524*(log(x)-mu)/sig);
    }
    Doub invcdf(Doub p) {
    Return inverse cumulative distribution function.
        if (p <= 0. || p >= 1.) throw("bad p in Lognormaldist");
        return exp(-1.41421356237309505*sig*inverfc(2.*p)+mu);
    }
};
```

Multiplicative random walks like (6.14.29) and lognormal distributions are key ingredients in the economic theory of efficient markets, leading to (among many other results) the celebrated *Black-Scholes formula* for the probability distribution of the price of an investment after some elapsed time τ. A key piece of the Black-Scholes derivation is implicit in equation (6.14.32): If an investment's average return is zero (which may be true in the limit of zero risk), then its price cannot simply be a widening lognormal distribution with fixed μ and increasing σ, for its expected value would then diverge to infinity! The actual Black-Scholes formula thus defines both how σ increases with time (basically as $\tau^{1/2}$) *and* how μ correspondingly decreases with time, so as to keep the overall mean under control. A simplified version of the Black-Scholes formula can be written as

$$S(\tau) \sim S(0) \times \text{Lognormal}\left(r\tau - \tfrac{1}{2}\sigma^2\tau,\ \sigma\sqrt{\tau}\right) \tag{6.14.35}$$

where $S(\tau)$ is the price of a stock at time τ, r is its expected (annualized) rate of return, and σ is now redefined to be the stock's (annualized) volatility. The definition of volatility is that, for small values of τ, the fractional variance of the stock's price is $\sigma^2\tau$. You can check that (6.14.35) has the desired expectation value $E[S(\tau)] = S(0)$, for all τ, if $r = 0$. A good reference is [3].

6.14.8 Chi-Square Distribution

The *chi-square* (or χ^2) distribution has a single parameter $\nu > 0$ that controls both the location and width of its peak. In most applications ν is an integer and is referred to as the *number of degrees of freedom* (see §14.3).

The defining probability density is

$$\begin{aligned} \chi^2 &\sim \text{Chisquare}(\nu), \qquad \nu > 0 \\ p(\chi^2)d\chi^2 &= \frac{1}{2^{\frac{1}{2}\nu}\Gamma(\frac{1}{2}\nu)}(\chi^2)^{\frac{1}{2}\nu-1}\exp\left(-\tfrac{1}{2}\chi^2\right)d\chi^2, \qquad \chi^2 > 0 \end{aligned} \tag{6.14.36}$$

where we have written the differentials $d\chi^2$ merely to emphasize that χ^2, not χ, is to be viewed as the independent variable.

The mean and variance are given by

$$\begin{aligned} \text{Mean}\{\text{Chisquare}(\nu)\} &= \nu \\ \text{Var}\{\text{Chisquare}(\nu)\} &= 2\nu \end{aligned} \tag{6.14.37}$$

When $\nu \geq 2$ there is a single mode at $\chi^2 = \nu - 2$.

The chi-square distribution is actually just a special case of the gamma distribution, below, so its cdf is given by an incomplete gamma function $P(a, x)$,

$$\text{cdf} \equiv P(< \chi^2) \equiv P(\chi^2|\nu) \equiv \int_0^{\chi^2} p(\chi^{2\prime})d\chi^{2\prime} = P\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right) \tag{6.14.38}$$

One frequently also sees the complement of the cdf, which can be calculated either from the incomplete gamma function $P(a, x)$, or from its complement $Q(a, x)$ (often more accurate if P is very close to 1):

$$Q(\chi^2|\nu) \equiv 1 - P(\chi^2|n) = 1 - P\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right) \equiv Q\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right) \tag{6.14.39}$$

The inverse cdf is given in terms of the function that is the inverse of $P(a, x)$ on its second argument, which we here denote $P^{-1}(a, p)$:

$$x(P) = 2P^{-1}\left(\frac{\nu}{2}, P\right) \tag{6.14.40}$$

ncgammabeta.h

```
struct Chisqdist : Gamma {
```

χ^2 distribution, derived from the gamma function `Gamma`.

```
    Doub nu,fac;
    Chisqdist(Doub nnu) : nu(nnu) {
```

Constructor. Initialize with ν.

```
        if (nu <= 0.) throw("bad nu in Chisqdist");
        fac = 0.693147180559945309*(0.5*nu)+gammln(0.5*nu);
    }
    Doub p(Doub x2) {
```

Return probability density function.

```
        if (x2 <= 0.) throw("bad x2 in Chisqdist");
        return exp(-0.5*(x2-(nu-2.)*log(x2))-fac);
    }
    Doub cdf(Doub x2) {
```

```
    Return cumulative distribution function.
        if (x2 < 0.) throw("bad x2 in Chisqdist");
        return gammp(0.5*nu,0.5*x2);
    }
    Doub invcdf(Doub p) {
    Return inverse cumulative distribution function.
        if (p < 0. || p >= 1.) throw("bad p in Chisqdist");
        return 2.*invgammp(p,0.5*nu);
    }
};
```

6.14.9 Gamma Distribution

The *gamma distribution* is defined by

$$\begin{aligned} x &\sim \text{Gamma}(\alpha, \beta), \qquad \alpha > 0,\ \beta > 0 \\ p(x) &= \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \qquad x > 0 \end{aligned} \tag{6.14.41}$$

The exponential distribution is the special case with $\alpha = 1$. The chi-square distribution is the special case with $\alpha = \nu/2$ and $\beta = 1/2$.

The mean and variance are given by,

$$\begin{aligned} \text{Mean}\{\text{Gamma}(\alpha, \beta)\} &= \alpha/\beta \\ \text{Var}\{\text{Gamma}(\alpha, \beta)\} &= \alpha/\beta^2 \end{aligned} \tag{6.14.42}$$

When $\alpha \geq 1$ there is a single mode at $x = (\alpha - 1)/\beta$.

Evidently, the cdf is the incomplete gamma function

$$\text{cdf} \equiv P(< x) \equiv \int_0^x p(x')dx' = P(\alpha, \beta x) \tag{6.14.43}$$

while the inverse cdf is given in terms of the inverse of $P(a, x)$ on its second argument by

$$x(P) = \frac{1}{\beta} P^{-1}(\alpha, P) \tag{6.14.44}$$

incgammabeta.h

```
struct Gammadist : Gamma {
Gamma distribution, derived from the gamma function Gamma.
    Doub alph, bet, fac;
    Gammadist(Doub aalph, Doub bbet = 1.) : alph(aalph), bet(bbet) {
    Constructor. Initialize with α and β.
        if (alph <= 0. || bet <= 0.) throw("bad alph,bet in Gammadist");
        fac = alph*log(bet)-gammln(alph);
    }
    Doub p(Doub x) {
    Return probability density function.
        if (x <= 0.) throw("bad x in Gammadist");
        return exp(-bet*x+(alph-1.)*log(x)+fac);
    }
    Doub cdf(Doub x) {
    Return cumulative distribution function.
        if (x < 0.) throw("bad x in Gammadist");
        return gammp(alph,bet*x);
    }
```

```
    Doub invcdf(Doub p) {
    Return inverse cumulative distribution function.
        if (p < 0. || p >= 1.) throw("bad p in Gammadist");
        return invgammp(p,alph)/bet;
    }
};
```

6.14.10 F-Distribution

The *F-distribution* is parameterized by two positive values ν_1 and ν_2, usually (but not always) integers.

The defining probability density is

$$\begin{aligned} F &\sim \mathrm{F}(\nu_1,\nu_2), \qquad \nu_1>0,\ \nu_2>0 \\ p(F) &= \frac{\nu_1^{\frac{1}{2}\nu_1}\nu_2^{\frac{1}{2}\nu_2}}{B(\frac{1}{2}\nu_1,\frac{1}{2}\nu_2)}\frac{F^{\frac{1}{2}\nu_1-1}}{(\nu_2+\nu_1 F)^{(\nu_1+\nu_2)/2}}, \qquad F>0 \end{aligned} \tag{6.14.45}$$

where $B(a,b)$ denotes the beta function. The mean and variance are given by

$$\begin{aligned} \mathrm{Mean}\{\mathrm{F}(\nu_1,\nu_2)\} &= \frac{\nu_2}{\nu_2-2}, && \nu_2>2 \\ \mathrm{Var}\{\mathrm{F}(\nu_1,\nu_2)\} &= \frac{2\nu_2^2(\nu_1+\nu_2-2)}{\nu_1(\nu_2-2)^2(\nu_2-4)}, && \nu_2>4 \end{aligned} \tag{6.14.46}$$

When $\nu_1 \geq 2$ there is a single mode at

$$F = \frac{\nu_2(\nu_1-2)}{\nu_1(\nu_2+2)} \tag{6.14.47}$$

.

For fixed ν_1, if $\nu_2 \to \infty$, the F-distribution becomes a chi-square distribution, namely

$$\lim_{\nu_2\to\infty} \mathrm{F}(\nu_1,\nu_2) \cong \frac{1}{\nu_1}\mathrm{Chisquare}(\nu_1) \tag{6.14.48}$$

where "$\cong$" means "are identical distributions."

The F-distribution's cdf is given in terms of the incomplete beta function $I_x(a,b)$ by

$$\mathrm{cdf} \equiv P(<x) \equiv \int_0^x p(x')dx' = I_{\nu_1 F/(\nu_2+\nu_1 F)}\left(\tfrac{1}{2}\nu_1,\tfrac{1}{2}\nu_2\right) \tag{6.14.49}$$

while the inverse cdf is given in terms of the inverse of $I_x(a,b)$ on its subscript argument by

$$\begin{aligned} u &\equiv I_p^{-1}\left(\tfrac{1}{2}\nu_1,\tfrac{1}{2}\nu_2\right) \\ x(P) &= \frac{\nu_2 u}{\nu_1(1-u)} \end{aligned} \tag{6.14.50}$$

A frequent use of the F-distribution is to test whether two observed samples have the same variance.

incgammabeta.h

```
struct Fdist : Beta {
```
F distribution, derived from the beta function `Beta`.
```
    Doub nu1,nu2;
    Doub fac;
    Fdist(Doub nnu1, Doub nnu2) : nu1(nnu1), nu2(nnu2) {
```
Constructor. Initialize with ν_1 and ν_2.
```
        if (nu1 <= 0. || nu2 <= 0.) throw("bad nu1,nu2 in Fdist");
        fac = 0.5*(nu1*log(nu1)+nu2*log(nu2))+gammln(0.5*(nu1+nu2))
            -gammln(0.5*nu1)-gammln(0.5*nu2);
    }
    Doub p(Doub f) {
```
Return probability density function.
```
        if (f <= 0.) throw("bad f in Fdist");
        return exp((0.5*nu1-1.)*log(f)-0.5*(nu1+nu2)*log(nu2+nu1*f)+fac);
    }
    Doub cdf(Doub f) {
```
Return cumulative distribution function.
```
        if (f < 0.) throw("bad f in Fdist");
        return betai(0.5*nu1,0.5*nu2,nu1*f/(nu2+nu1*f));
    }
    Doub invcdf(Doub p) {
```
Return inverse cumulative distribution function.
```
        if (p <= 0. || p >= 1.) throw("bad p in Fdist");
        Doub x = invbetai(p,0.5*nu1,0.5*nu2);
        return nu2*x/(nu1*(1.-x));
    }
};
```

6.14.11 Beta Distribution

The *beta distribution* is defined on the unit interval $0 < x < 1$ by

$$\begin{aligned} x &\sim \text{Beta}(\alpha, \beta), \qquad \alpha > 0,\ \beta > 0 \\ p(x) &= \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}, \qquad 0 < x < 1 \end{aligned} \tag{6.14.51}$$

The mean and variance are given by

$$\begin{aligned} \text{Mean}\{\text{Beta}(\alpha, \beta)\} &= \frac{\alpha}{\alpha+\beta} \\ \text{Var}\{\text{Beta}(\alpha, \beta)\} &= \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \end{aligned} \tag{6.14.52}$$

When $\alpha > 1$ and $\beta > 1$, there is a single mode at $(\alpha-1)/(\alpha+\beta-2)$. When $\alpha < 1$ and $\beta < 1$, the distribution function is "U-shaped" with a minimum at this same value. In other cases there is neither a maximum nor a minimum.

In the limit that β becomes large as α is held fixed, all the action in the beta distribution shifts toward $x = 0$, and the density function takes the shape of a gamma distribution. More precisely,

$$\lim_{\beta\to\infty} \beta\, \text{Beta}(\alpha, \beta) \cong \text{Gamma}(\alpha, 1) \tag{6.14.53}$$

The cdf is the incomplete beta function

$$\text{cdf} \equiv P(< x) \equiv \int_0^x p(x')dx' = I_x(\alpha, \beta) \tag{6.14.54}$$

while the inverse cdf is given in terms of the inverse of $I_x(\alpha, \beta)$ on its subscript argument by

$$x(P) = I_p^{-1}(\alpha, \beta) \tag{6.14.55}$$

ncgammabeta.h

```
struct Betadist : Beta {
Beta distribution, derived from the beta function Beta.
    Doub alph, bet, fac;
    Betadist(Doub aalph, Doub bbet) : alph(aalph), bet(bbet) {
    Constructor. Initialize with α and β.
        if (alph <= 0. || bet <= 0.) throw("bad alph,bet in Betadist");
        fac = gammln(alph+bet)-gammln(alph)-gammln(bet);
    }
    Doub p(Doub x) {
    Return probability density function.
        if (x <= 0. || x >= 1.) throw("bad x in Betadist");
        return exp((alph-1.)*log(x)+(bet-1.)*log(1.-x)+fac);
    }
    Doub cdf(Doub x) {
    Return cumulative distribution function.
        if (x < 0. || x > 1.) throw("bad x in Betadist");
        return betai(alph,bet,x);
    }
    Doub invcdf(Doub p) {
    Return inverse cumulative distribution function.
        if (p < 0. || p > 1.) throw("bad p in Betadist");
        return invbetai(p,alph,bet);
    }
};
```

6.14.12 Kolmogorov-Smirnov Distribution

The *Kolmogorov-Smirnov* or *KS* distribution, defined for positive z, is key to an important statistical test that is discussed in §14.3. Its probability density function does not directly enter into the test and is virtually never even written down. What one typically needs to compute is the cdf, denoted $P_{KS}(z)$, or its complement, $Q_{KS}(z) \equiv 1 - P_{KS}(z)$.

The cdf $P_{KS}(z)$ is defined by the series

$$P_{KS}(z) = 1 - 2\sum_{j=1}^{\infty}(-1)^{j-1}\exp(-2j^2z^2) \tag{6.14.56}$$

or by the equivalent series (nonobviously so!)

$$P_{KS}(z) = \frac{\sqrt{2\pi}}{z}\sum_{j=1}^{\infty}\exp\left(-\frac{(2j-1)^2\pi^2}{8z^2}\right) \tag{6.14.57}$$

Limiting values are what you'd expect for cdf's named "P" and "Q":

$$\begin{aligned} P_{KS}(0) &= 0 \qquad P_{KS}(\infty) = 1 \\ Q_{KS}(0) &= 1 \qquad Q_{KS}(\infty) = 0 \end{aligned} \tag{6.14.58}$$

Both of the series (6.14.56) and (6.14.57) are convergent for all $z > 0$. Moreover, for any z, one or the other series converges extremely rapidly, requiring no more

than three terms to get to IEEE double precision fractional accuracy. A good place to switch from one series to the other is at $z \approx 1.18$. This renders the KS functions computable by a single exponential and a small number of arithmetic operations (see code below).

Getting the inverse functions $P_{KS}^{-1}(P)$ and $Q_{KS}^{-1}(Q)$, which return a value of z from a P or Q value, is a little trickier. For $Q \lesssim 0.3$ (that is, $P \gtrsim 0.7$), an iteration based on (6.14.56) works nicely:

$$\begin{aligned} x_0 &\equiv 0 \\ x_{i+1} &= \tfrac{1}{2}Q + x_i^4 - x_i^9 + x_i^{16} - x_i^{25} + \cdots \\ z(Q) &= \sqrt{-\tfrac{1}{2}\log(x_\infty)} \end{aligned} \tag{6.14.59}$$

For $x \lesssim 0.06$ you only need the first two powers of x_i.

For larger values of Q, that is, $P \lesssim 0.7$, the number of powers of x required quickly becomes excessive. A useful approach is to write (6.14.57) as

$$\begin{aligned} y\log(y) &= -\frac{\pi P^2}{8}\left(1 + y^4 + y^{12} + \cdots + y^{2j(j-1)} + \cdots\right)^{-1} \\ z(P) &= \frac{\pi/2}{\sqrt{-\log(y)}} \end{aligned} \tag{6.14.60}$$

If we can get a good enough initial guess for y, we can solve the first equation in (6.14.60) by a variant of Halley's method: Use values of y from the *previous* iteration on the right-hand side of (6.14.60), and use Halley only for the $y\log(y)$ piece, so that the first and second derivatives are analytically simple functions.

A good initial guess is obtained by using the inverse function to $y\log(y)$ (the function `invxlogx` in §6.11) with the argument $-\pi P^2/8$. The number of iterations within the `invxlogx` function and the Halley loop is never more than half a dozen in each, often less. Code for the KS functions and their inverses follows.

ksdist.h

```
struct KSdist {
Kolmogorov-Smirnov cumulative distribution functions and their inverses.
    Doub pks(Doub z) {
    Return cumulative distribution function.
        if (z < 0.) throw("bad z in KSdist");
        if (z == 0.) return 0.;
        if (z < 1.18) {
            Doub y = exp(-1.2337005501361698/SQR(z));
            return 2.25675833419102515*sqrt(-log(y))
                *(y + pow(y,9) + pow(y,25) + pow(y,49));
        } else {
            Doub x = exp(-2.*SQR(z));
            return 1. - 2.*(x - pow(x,4) + pow(x,9));
        }
    }
    Doub qks(Doub z) {
    Return complementary cumulative distribution function.
        if (z < 0.) throw("bad z in KSdist");
        if (z == 0.) return 1.;
        if (z < 1.18) return 1.-pks(z);
        Doub x = exp(-2.*SQR(z));
        return 2.*(x - pow(x,4) + pow(x,9));
    }
    Doub invqks(Doub q) {
```

```
    Return inverse of the complementary cumulative distribution function.
        Doub y,logy,yp,x,xp,f,ff,u,t;
        if (q <= 0. || q > 1.) throw("bad q in KSdist");
        if (q == 1.) return 0.;
        if (q > 0.3) {
            f = -0.3926990816987241 55*SQR(1.-q);
            y = invxlogx(f);                    Initial guess.
            do {
                yp = y;
                logy = log(y);
                ff = f/SQR(1.+ pow(y,4)+ pow(y,12));
                u = (y*logy-ff)/(1.+logy);      Newton's method correction.
                y = y - (t=u/MAX(0.5,1.-0.5*u/(y*(1.+logy))));  Halley.
            } while (abs(t/y)>1.e-15);
            return 1.57079632679489662/sqrt(-log(y));
        } else {
            x = 0.03;
            do {                                Iteration (6.14.59).
                xp = x;
                x = 0.5*q+pow(x,4)-pow(x,9);
                if (x > 0.06) x += pow(x,16)-pow(x,25);
            } while (abs((xp-x)/x)>1.e-15);
            return sqrt(-0.5*log(x));
        }
    }
    Doub invpks(Doub p) {return invqks(1.-p);}
    Return inverse of the cumulative distribution function.
};
```

6.14.13 Poisson Distribution

The eponymous *Poisson distribution* was derived by Poisson in 1837. It applies to a process where discrete, uncorrelated events occur at some mean rate per unit time. If, for a given period, λ is the mean expected number of events, then the probability distribution of seeing exactly k events, $k \geq 0$, can be written as

$$\begin{aligned} k &\sim \text{Poisson}(\lambda), \qquad \lambda > 0 \\ p(k) &= \frac{1}{k!}\lambda^k e^{-\lambda}, \qquad k = 0, 1, \ldots \end{aligned} \tag{6.14.61}$$

Evidently $\sum_k p(k) = 1$, since the k-dependent factors in (6.14.61) are just the series expansion of e^{λ}.

The mean and variance of Poisson(λ) are both λ. There is a single mode at $k = \lfloor \lambda \rfloor$, that is, at λ rounded down to an integer.

The Poisson distribution's cdf is an incomplete gamma function $Q(a, x)$,

$$P_\lambda(< k) = Q(k, \lambda) \tag{6.14.62}$$

Since k is discrete, $P_\lambda(< k)$ is of course different from $P_\lambda(\leq k)$, the latter being given by

$$P_\lambda(\leq k) = Q(k + 1, \lambda) \tag{6.14.63}$$

Some particular values are

$$P_\lambda(< 0) = 0 \qquad P_\lambda(< 1) = e^{-\lambda} \qquad P_\lambda(< \infty) = 1 \tag{6.14.64}$$

Some other relations involving the incomplete gamma functions $Q(a,x)$ and $P(a,x)$ are

$$\begin{aligned} P_\lambda(\geq k) &= P(k,\lambda) = 1 - Q(k,\lambda) \\ P_\lambda(> k) &= P(k+1,\lambda) = 1 - Q(k+1,\lambda) \end{aligned} \tag{6.14.65}$$

Because of the discreteness in k, the inverse of the cdf must be defined with some care: Given a value P, we define $k_\lambda(P)$ as the integer such that

$$P_\lambda(< k) \leq P < P_\lambda(\leq k) \tag{6.14.66}$$

In the interest of conciseness, the code below cheats a little bit and allows the right-hand $<$ to be $\leq$. If you may be supplying P's that are *exact* $P_\lambda(< k)$'s, then you will need to check both $k_\lambda(P)$ as returned, and $k_\lambda(P)+1$. (This will essentially never happen for "round" P's like 0.95, 0.99, etc.)

incgammabeta.h

```
struct Poissondist : Gamma {
```
Poisson distribution, derived from the gamma function `Gamma`.
```
    Doub lam;
    Poissondist(Doub llam) : lam(llam) {
```
Constructor. Initialize with λ.
```
        if (lam <= 0.) throw("bad lam in Poissondist");
    }
    Doub p(Int n) {
```
Return probability density function.
```
        if (n < 0) throw("bad n in Poissondist");
        return exp(-lam + n*log(lam) - gammln(n+1.));
    }
    Doub cdf(Int n) {
```
Return cumulative distribution function.
```
        if (n < 0) throw("bad n in Poissondist");
        if (n == 0) return 0.;
        return gammq((Doub)n,lam);
    }
    Int invcdf(Doub p) {
```
Given argument P, return integer n such that $P(< n) \leq P \leq P(< n+1)$.
```
        Int n,nl,nu,inc=1;
        if (p <= 0. || p >= 1.) throw("bad p in Poissondist");
        if (p < exp(-lam)) return 0;
        n = (Int)MAX(sqrt(lam),5.);
        if (p < cdf(n)) {
            do {
                n = MAX(n-inc,0);
                inc *= 2;
            } while (p < cdf(n));
            nl = n; nu = n + inc/2;
        } else {
            do {
                n += inc;
                inc *= 2;
            } while (p > cdf(n));
            nu = n; nl = n - inc/2;
        }
        while (nu-nl>1) {
            n = (nl+nu)/2;
            if (p < cdf(n)) nu = n;
            else nl = n;
        }
        return nl;
    }
};
```
Starting guess near peak of density.
Expand interval until we bracket.
Now contract the interval by bisection.

6.14.14 Binomial Distribution

Like the Poisson distribution, the *binomial distribution* is a discrete distribution over $k \geq 0$. It has two parameters, $n \geq 1$, the "sample size" or maximum value for which k can be nonzero; and p, the "event probability" (not to be confused with $p(k)$, the probability of a particular k). We write

$$\begin{aligned} k &\sim \text{Binomial}(n, p), \qquad n \geq 1,\ 0 < p < 1 \\ p(k) &= \binom{n}{k} p^k (1-p)^{n-k}, \qquad k = 0, 1, \ldots, n \end{aligned} \tag{6.14.67}$$

where $\binom{n}{k}$ is, of course, the binomial coefficient.

The mean and variance are given by

$$\begin{aligned} \text{Mean}\{\text{Binomial}(n, p)\} &= np \\ \text{Var}\{\text{Binomial}(n, p)\} &= np(1-p) \end{aligned} \tag{6.14.68}$$

There is a single mode at the value k that satisfies

$$(n+1)p - 1 < k \leq (n+1)p \tag{6.14.69}$$

The distribution is symmetrical iff $p = \frac{1}{2}$. Otherwise it has positive skewness for $p < \frac{1}{2}$ and negative for $p > \frac{1}{2}$. Many additional properties are described in [2].

The Poisson distribution is obtained from the binomial distribution in the limit $n \to \infty$, $p \to 0$ with the np remaining finite. More precisely,

$$\lim_{n\to\infty} \text{Binomial}(n, \lambda/n) \cong \text{Poisson}(\lambda) \tag{6.14.70}$$

The binomial distribution's cdf can be computed from the incomplete beta function $I_x(a, b)$,

$$P(< k) = 1 - I_p(k, n-k+1) \tag{6.14.71}$$

so we also have (analogously to the Poisson distribution)

$$\begin{aligned} P(\leq k) &= 1 - I_p(k+1, n-k) \\ P(> k) &= I_p(k+1, n-k) \\ P(\geq k) &= I_p(k, n-k+1) \end{aligned} \tag{6.14.72}$$

Some particular values are

$$P(< 0) = 0 \qquad P(< [n+1]) = 1 \tag{6.14.73}$$

The inverse cdf is defined exactly as for the Poisson distribution, above, and with the same small warning about the code.

ncgammabeta.h

```
struct Binomialdist : Beta {
```

Binomial distribution, derived from the beta function Beta.

```
    Int n;
    Doub pe, fac;
    Binomialdist(Int nn, Doub ppe) : n(nn), pe(ppe) {
```

Constructor. Initialize with n (sample size) and p (event probability).

```
        if (n <= 0 || pe <= 0. || pe >= 1.) throw("bad args in Binomialdist");
```

```
        fac = gammln(n+1.);
    }
    Doub p(Int k) {
    Return probability density function.
        if (k < 0) throw("bad k in Binomialdist");
        if (k > n) return 0.;
        return exp(k*log(pe)+(n-k)*log(1.-pe)
            +fac-gammln(k+1.)-gammln(n-k+1.));
    }
    Doub cdf(Int k) {
    Return cumulative distribution function.
        if (k < 0) throw("bad k in Binomialdist");
        if (k == 0) return 0.;
        if (k > n) return 1.;
        return 1. - betai((Doub)k,n-k+1.,pe);
    }
    Int invcdf(Doub p) {
    Given argument P, return integer n such that P(< n) ≤ P ≤ P(< n + 1).
        Int k,kl,ku,inc=1;
        if (p <= 0. || p >= 1.) throw("bad p in Binomialdist");
        k = MAX(0,MIN(n,(Int)(n*pe)));          Starting guess near peak of density.
        if (p < cdf(k)) {                       Expand interval until we bracket.
            do {
                k = MAX(k-inc,0);
                inc *= 2;
            } while (p < cdf(k));
            kl = k; ku = k + inc/2;
        } else {
            do {
                k = MIN(k+inc,n+1);
                inc *= 2;
            } while (p > cdf(k));
            ku = k; kl = k - inc/2;
        }
        while (ku-kl>1) {                       Now contract the interval by bisection.
            k = (kl+ku)/2;
            if (p < cdf(k)) ku = k;
            else kl = k;
        }
        return kl;
    }
};
```

CITED REFERENCES AND FURTHER READING:

Johnson, N.L. and Kotz, S. 1970, *Continuous Univariate Distributions*, 2 vols. (Boston: Houghton Mifflin).[1]

Johnson, N.L. and Kotz, S. 1969, *Discrete Distributions* (Boston: Houghton Mifflin).[2]

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. 2003, *Bayesian Data Analysis*, 2nd ed. (Boca Raton, FL: Chapman & Hall/CRC), Appendix A.

Lyuu, Y-D. 2002, *Financial Engineering and Computation* (Cambridge, UK: Cambridge University Press).[3]