The auxiliary function log1pmx() ("log 1 plus minus x"), had been introduced when R’s pgamma() (incomplete Γ function) had been numerically improved by Morten Welinder’s contribution to R’s PR#7307, in Jan. 2005\(^1\), it is mathematically defined as

\[
\log1pmx(x) := \log(1 + x) - x \approx -\frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} \pm \ldots, \tag{1}
\]

and for numerical evaluation, suffers from two levels of cancellations for small \(x\), i.e., using \log1p(x) for \log(1 + x) is not sufficient.

In 2000 already, Catherine Loader’s contributions for more accurate computation of binomial, Poisson and negative binomial probabilities, \cite{Loader2000}, had introduced auxiliary functions bd0() and stirlerr(), see below.

Much later, in R’s PR#15628, in Jan. 2014\(^2\), Welinder noticed that in spite of Loader’s improvements, Poisson probabilities were not perfectly accurate (only ca. 13 accurate digits instead of 15\(\approx \log_{10}(2^{52})\)), relating the problem to somewhat imperfect computations in bd0(), which he proposed to address using log1pmx() on one hand, and additionally addressing cancellation by using two double precision numbers to store the result (his proposal of an ebd0() function).

Here, I address the problem of providing more accurate bd0() (and stirlerr() as well), applying Welinder’s proposal to use log1pmx(), but otherwise diverging from the proposal.

1 Introduction

According to R’s reference documentation, help(dbinom), the binomial (point-mass) probabilities of the binomial distribution with size = \(n\) and prob = \(p\) has “density” (point probabilities)

\[
p(x) := p(x; n, p) := \binom{n}{x} p^x (1 - p)^{n-x}
\]

for \(x = 0, \ldots, n\), and these are (in R function dbinom()) computed via Loader’s algorithm (\cite{Loader2000}) which had improved accuracy considerably, also for R’s internal dpois_raw() function which is used further directly in dpois(), dbinom(), dgamma(), the non-central dbeta() and dchisq() and even the cumulative Γ() probabilities pgamma()
and hence indirectly e.g., for cumulative central and non-central chisquare probabilities (\texttt{pchisq()}).

Loader noticed that for large \(n\), the usual way to compute \(p(x; n, p)\) via its logarithm 
\[
\log(p(x; n, p)) = \log(n!) - \log(x!) - \log((n-x)!) + x \log(p) + (n-x) \log(1-p)
\] was inaccurate, even when accurate \(\log \Gamma(x) = \texttt{lgamma}(x)\) values are available to get 
\(\log(x!) = \log \Gamma(x+1)\), e.g., for \(x = 10^6, n = 2 \times 10^6, p = 1/2\), about 7 digits accuracy were lost from cancellation 
(in substraction of the log factorials).

Instead, she wrote 
\[
p(x; n, p) = p(x; n, x/n) \cdot e^{-D(x; n, p)}, 
\]
where the “Deviance” \(D(.)\) is defined as 
\[
D(x; n, p) = \log p(x; n, x/n) - \log p(x; n, p)
\] 
\[
= x \log \left(\frac{x}{np}\right) + (n-x) \log \left(\frac{n-x}{n(1-p)}\right),
\]
and to avoid cancellation, \(D()\) has to be computed somewhat differently, namely – correcting notation wrt the original – using a two-argument version \(D_0()\):
\[
D(x; n, p) = npD_0\left(\frac{x}{np}\right) + nqD_0\left(\frac{n-x}{np}\right)
\]
\[
= D_0(x, np) + D_0(n-x, nq),
\]
where \(q := 1-p\) and 
\[
\tilde{D}_0(r) := r \log(r) + 1 - r \quad \text{and}
\]
\[
D_0(x, M) := M \cdot \tilde{D}_0(x/M)
\]
\[
= M \cdot \left(\frac{x}{M} \log \left(\frac{x}{M}\right) + 1 - \frac{x}{M} \right) = x \log \left(\frac{x}{M}\right) + M - x
\]
Note that since \(\lim_{x \to 0} x \log x = 0\), setting
\[
\tilde{D}_0(0) := 1 \quad \text{and}
\]
\[
D_0(0, M) := M \tilde{D}_0(0) = M \cdot 1 = M
\]
defines \(D_0(x, M)\) for all \(x \geq 0\), \(M > 0\).

The careful C function implementation of \(D_0(x, M)\) is called \texttt{bd0(x, np)} in Loader’s C code and now R’s Mathlib at \url{https://svn.r-project.org/R/trunk/src/nmath/bd0.c}, mirrored, e.g., at Winston Chen’s github mirror². In 2014, Morten Welinder suggested in 
R’s PR\#15628⁴ that the current \texttt{bd0()} implementation is still inaccurate in some regions 
(mostly not in the one it has been carefully implemented to be accurate, i.e., when \(x \approx M\) 
notably for computing Poisson probabilities, \texttt{dpois()} in R; see more below.

Evaluating of \(p(x; n, p)\) in (2), in addition to \(D(x; n, p)\) in (4) also needs \(p(x; n, x/n)\) 
where in turn, the Stirling De Moivre series is used:
\[
\log n! = \frac{1}{2} \log(2\pi n) + n \log(n) - n + \delta(n), \quad \text{where the “Stirling error” } \delta(n) \text{ is}
\]
\[
\delta(n) := \log n! - \frac{1}{2} \log(2\pi n) - n \log(n) + n =
\]
\[
= \frac{1}{12n} - \frac{1}{360n^3} + \frac{1}{1260n^5} - \frac{1}{1680n^7} + \frac{1}{1188n^9} + O(n^{-11}).
\]

³\url{https://github.com/wch/r-source/blob/trunk/src/nmath/bd0.c}
⁴\url{https://bugs.r-project.org/bugzilla/show_bug.cgi?id=15628}
Note that \( \delta(n) \equiv \text{stirlerr}(n) \) in the C code provided by Loader and now in R’s Mathlib at https://svn.r-project.org/R/trunk/src/nmath/stirlerr.c.

Note that for the binomial, \( x \) is an integer in \( \{0, 1, \ldots, n\} \) and \( M = np \geq 0 \), but the formulas (6), (7) for \( D_0(x, M) \) apply and are needed, e.g., for \( \text{pgamma()} \) computations for general non-negative \((x, M > 0)\) where even \( x = 0 \) is well defined, see (8) above.

Further (see Loader), such a saddle point approach is needed for Poisson probabilities, as well, where

\[
p_\lambda(x) = e^{-\lambda} \frac{\lambda^x}{x!}
\]

\[
\log p_\lambda(x) = -\lambda + x \log \lambda - \log(x!) - \log(1/\sqrt{2\pi x}) - (x \log x - x + \delta(x)) = \log \frac{1}{\sqrt{2\pi x}} - x \log \frac{x}{\lambda} - x - \lambda - \delta(x),
\]

is re-expressed using \( \delta(x) \) and from (7) \( D_0(x, \lambda) \) as

\[
p_\lambda(x) = \frac{1}{\sqrt{2\pi x}} e^{-\delta(x)} D_0(x, \lambda)
\]

Also, negative binomial probabilities, \( \text{dnbinom()} \), TODO ......

Even for the \( t_\nu \) density, \( \text{dt()} \), .......

...but there have a direct approximations in package DPQ, currently functions \( \text{c_dt(nu)} \) and even more promisingly, \( \text{lb_chu(nu)} \). TODO ......

2 Loader’s “Binomial Deviance” \( D_0(x, M) = \text{bd0}(x, M) \)

Loader’s “Binomial Deviance” function \( D_0(x, M) = \text{bd0}(x, M) \) has been defined for \( x, M > 0 \) where the limit \( x \to 0 \) is allowed (even though not implemented in the original \( \text{bd0()} \)), here repeated from (6):

\[
D_0(x, M) := M \cdot \tilde{D}_0 \left( \frac{x}{M} \right), \quad \text{where} \quad \tilde{D}_0(u) := u \log(u) + 1 - u = u(\log(u) - 1) + 1.
\]

Note the graph of \( \tilde{D}_0(u) \),

![Graph of \( \tilde{D}_0(u) \)](image-url)
has a double zero at \( u = 1 \), such that for large \( M \) and \( x \approx M \), i.e., \( \frac{x}{M} \approx 1 \), the direct computation of \( D_0(x, M) = M \cdot D_0\left(\frac{x}{M}\right) \) is numerically problematic. Further,

\[
D_0(x, M) = M \cdot \left(\frac{x}{M}\log\left(\frac{x}{M}\right) - 1\right) = x \log\left(\frac{x}{M}\right) - x + M. \tag{15}
\]

We can rewrite this, originally by e-mail from Martyn Plummer, then also indirectly from Morten Welinder’s mentioning of \( \log_1p\text{mx}() \) in his PR notably for the important situation when \(|x - M| \ll M\). Setting \( t := (x - M)/M \), i.e., \(|t| \ll 1 \) for that situation, or equivalently, \( \frac{x}{M} = 1 + t \). Using \( t \),

\[
t := \frac{x - M}{M} \tag{16}
\]

\[
D_0(x, M) = M \cdot \left(1 + t\right) \log(1 + t) - t \cdot M = M \cdot \left(\left(1 + t\right) \log(1 + t) - t\right) =
\]

\[
= M \cdot p_1l_1(t), \tag{17}
\]

where

\[
p_1l_1(t) := (t + 1) \log(1 + t) - t = \frac{t^2}{2} - \frac{t^3}{6} \pm \cdots, \tag{18}
\]

\[
= (\log(1 + t) - t) + t \cdot \log(1 + t)
\]

\[
= \log_1\text{p}x(t) + t \cdot \log_1p(t) \tag{19}
\]

where the Taylor series expansion is useful for small \(|t|\),

\[
p_1l_1(t) = \frac{t^2}{2} - \frac{t^3}{6} + \frac{t^4}{12} \pm \cdots = \sum_{n=2}^{\infty} \frac{(-t)^n}{n(n-1)} = \frac{t^2}{2} \sum_{n=2}^{\infty} \frac{(-t)^{n-2}}{n(n-1)/2} = \frac{t^2}{2} \sum_{n=0}^{\infty} \frac{(-t)^n}{(n+2)} \tag{20}
\]

\[
= \frac{t^2}{2} \left(1 - t\left(\frac{1}{3} - t\left(\frac{1}{6} - t\left(\frac{1}{10} - \cdots\right)\right)\right)\right), \tag{21}
\]

which we provide in DPQ via function \( \text{p1liser}(t, k) \) getting the first \( k \) terms, and the corresponding series approximation for

\[
D_0(x, M) = \lim_{k \to \infty} \text{p1liser}\left(\frac{x - M}{M}, k, F = \frac{(x - M)^2}{M}\right), \tag{22}
\]

where the approximation of course uses a finite \( k \) instead of the limit \( k \to \infty \).

This Taylor series expansion is useful and nice, but may not even be needed typically, as both utility functions \( \log_1\text{p}x(t) \) and \( \log_1p(t) \) are available implemented to be fully accurate for small \( t \), \( t \ll 1 \), and (19), indeed, with \( t = (x - M)/M \) the evaluation of

\[
D_0(x, M) = M \cdot p_1l_1(t) = M \cdot \left(\log_1\text{p}x(t) + t \cdot \log_1p(t)\right), \tag{23}
\]

seems quite accurate already on a wide range of \((x, M)\) values.

Note that \( x \log_1p(x) \) and \( \log_1\text{p}x() \) have different signs, but also note that for small \(|x|\), are well approximated by \( x^2/2 \) and \(-x^2/2\), so their sum \( p_1l_1(x) = \log_1\text{p}x(x) + x \cdot \log_1p(x) \) is approximately \( x^2/2 \) and numerically computing \( x^2 - x^2/2 \) should only lose 1 or 2 bits of precision.

References

Figure 1: `p1l1()` and its constituents; on the right, zoomed in 4 and 8 orders of magnitude, where the Taylor approximations $x^2/2$ and $x^2/2 - x^3/6$ are visually already perfect.