

`log1pmx()`, `bd0()`, `stirlerr()` – Computing Poisson, Binomial, Gamma Probabilities in R

Martin Mächler
Seminar für Statistik
ETH Zurich

April 2021 ff (L^AT_EX'ed August 22, 2026)

Abstract

The auxiliary function `log1pmx()` (“log 1 plus minus x”), had been introduced by Morten Welinder in his proposal to improve R’s `pgamma()` (incomplete Γ function) numerically, in R’s [PR#7307 comment #6](#)¹, in Jan. 2005. `log1pmx()` has also been added to the R’s C API Mathlib (aka `libRmath`, `r-mathlib`, or `nmath`) library in March 2005. It is defined as $\text{log1pmx}(x) := \log(1+x) - x$ 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, [Loader \(2000\)](#), had introduced auxiliary functions `bd0()` and `stirlerr()`, see below.

Much later, in R’s [PR#15628](#), in Jan. 2014², Welinder noticed that in spite of Loader’s improvements, Poisson probabilities were not perfectly accurate (only ca. 13 accurate digits instead of $15.6 \approx \log_{10}(2^{52})$), 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.

Notably, I noticed that `ebd0()` currently suffers from accuracy loss, when `bd0(x,M)` is large and $x/M \approx 1$.

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)

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

for $x = 0, \dots, n$, and these are (in R function `dbinom()`) computed via Loader’s algorithm ([Loader \(2000\)](#)) which had improved accuracy considerably, also for R’s internal `dpois_raw()` function which is used further directly in `dpois()`, `dnbinom()`, `dgamma()`, the non-central `dbeta()` and `dchisq()` and even the *cumulative* $\Gamma()$ probabilities `pgamma()`

¹https://bugs.R-project.org/show_bug.cgi?id=7307#c6

²https://bugs.r-project.org/show_bug.cgi?id=15628

and hence indirectly e.g., for cumulative central and non-central chisquare probabilities (`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 $\log \Gamma(x) = \text{lgamma}(x)$ 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 subtraction of the log factorials).

Instead, she wrote

$$p(x; n, p) = p(x; n, \frac{x}{n}) \cdot e^{-D(x; n, p)}, \quad (2)$$

where the “Deviance” $D(\cdot)$ is defined as

$$\begin{aligned} D(x; n, p) &= \log p(x; n, \frac{x}{n}) - \log p(x; n, p) \\ &= x \log\left(\frac{x}{np}\right) + (n-x) \log\left(\frac{n-x}{n(1-p)}\right), \end{aligned} \quad (3)$$

and to avoid cancellation, $D(\cdot)$ has to be computed somewhat differently, namely – correcting notation wrt the original – using a *two*-argument version $D_0(\cdot)$:

$$\begin{aligned} D(x; n, p) &= np d_0\left(\frac{x}{np}\right) + nq d_0\left(\frac{n-x}{nq}\right) \\ &= D_0(x, np) + D_0(n-x, nq), \end{aligned} \quad (4)$$

where $q := 1 - p$ and

$$d_0(r) := r \log(r) + 1 - r \quad \text{and} \quad (5)$$

$$\begin{aligned} D_0(x, M) &:= M \cdot d_0(x/M) \\ &= M \cdot \left(\frac{x}{M} \log\left(\frac{x}{M}\right) + 1 - \frac{x}{M} \right) = x \log\left(\frac{x}{M}\right) + M - x \end{aligned} \quad (6)$$

Note that since $\lim_{x \downarrow 0} x \log x = 0$, setting

$$\begin{aligned} d_0(0) &:= 1 \text{ and} \\ D_0(0, M) &:= M d_0(0) = M \cdot 1 = M \end{aligned} \quad (7)$$

defines $D_0(x, M)$ for all $x \geq 0$, $M > 0$.

The careful C function implementation of $D_0(x, M)$ is called `bd0(x, np)` in Loader’s C code and now R’s Mathlib at <https://svn.r-project.org/R/trunk/src/nmath/bd0.c>, mirrored, e.g., at [Winston Chen’s github mirror](#)³. In 2014, Morten Welinder suggested in [R’s PR#15628](#)⁴ that the current `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, `dpois()` in R; see more in [A.1](#) below.

Evaluating of $p(x; n, p)$ in (1) and (2), in addition to $D(x; n, p)$ in (4) also needs $p(x; n, \frac{x}{n})$ where in turn, the Stirling De Moivre series is used:

$$\log n! = \frac{1}{2} \log(2\pi n) + n \log(n) - n + \delta(n), \quad \text{where the “Stirling error” } \delta(n) \text{ is} \quad (8)$$

$$\delta(n) := \log n! - \frac{1}{2} \log(2\pi n) - n \log(n) + n = \quad (9)$$

$$= \frac{1}{12n} - \frac{1}{360n^3} + \frac{1}{1260n^5} - \frac{1}{1680n^7} + \frac{1}{1188n^9} + O(n^{-11}). \quad (10)$$

³<https://github.com/wch/r-source/blob/trunk/src/nmath/bd0.c>

⁴https://bugs.r-project.org/show_bug.cgi?id=15628

See appendix C how $\delta(n) \equiv \text{stirlerr}(n)$ is computed and implemented in the C code of R, and can be improved.

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

Summarizing, using (1) and (6), the binomial probabilities in R, `dbinom(x, n, p)` have been computed as

$$p(x; n, p) = p(x; n, \frac{x}{n}) \cdot e^{-D(x; n, p)} = \quad (11)$$

$$= \sqrt{\frac{n}{2\pi x(n-x)}} e^{\delta(n) - \delta(x) - \delta(n-x) - D(x; n, p)}, \quad (12)$$

the second line from replacing $p(x; n, \frac{x}{n})$ by eq. (5) of Loader, derived by using Stirling's (8) three times, viz. for n , x , and $n-x$, and noticing that many log terms cancel and the three $\log(2\pi^*)/2$ terms simplify to $\log(\frac{n}{2\pi x(n-x)})/2$.

Further, Loader showed that such a saddle point approach is needed for Poisson probabilities, as well, where

$$p_\lambda(x) = e^{-\lambda} \frac{\lambda^x}{x!} \quad (13)$$

$$\begin{aligned} \log p_\lambda(x) &= -\lambda + x \log \lambda + \underbrace{-\log(x!)}_{\log(1/\sqrt{2\pi x}) - (x \log x - x + \delta(x))} \\ &= \log \frac{1}{\sqrt{2\pi x}} - x \log \frac{x}{\lambda} + x - \lambda - \delta(x), \end{aligned} \quad (14)$$

is re-expressed using $\delta(x)$ and from (6) $D_0(x, \lambda)$ as

$$p_\lambda(x) = \frac{1}{\sqrt{2\pi x}} e^{-\delta(x) - D_0(x, \lambda)} \quad (15)$$

Also, negative binomial probabilities, `dnbinom()`, TODO

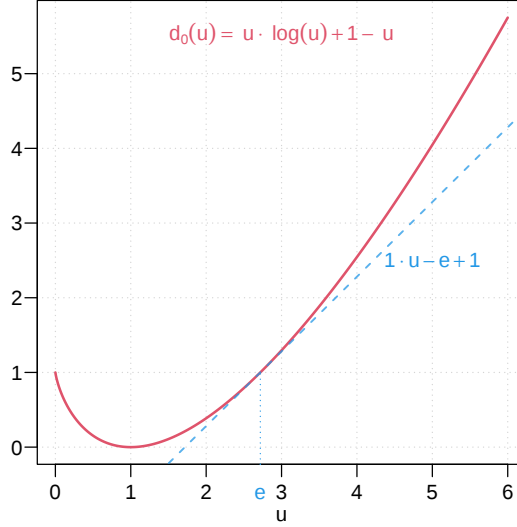
Even for the t_ν density, `dt()`,
... but there have a direct approximations in package **DPQ**, currently functions `c_dt(nu)` and even more promisingly, `lb_chi(nu)`. TODO

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 $x \rightarrow 0$ is allowed (even though not implemented in the original `bd0()`), here repeated from (5), (6) :

$$\begin{aligned} D_0(x, M) &:= M \cdot d_0\left(\frac{x}{M}\right), \quad \text{where} \\ d_0(u) &:= u \log(u) + 1 - u = u(\log(u) - 1) + 1. \end{aligned}$$

Note the graph of $d_0(u)$ ($= p_1 l_1(u-1)$, see (18) below),



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 $D_0(x, M) = M \cdot d_0(\frac{x}{M})$ is numerically problematic. Further,

$$D_0(x, M) = M \cdot \left(\frac{x}{M} \left(\log\left(\frac{x}{M}\right) - 1 \right) + 1 \right) = x \log\left(\frac{x}{M}\right) - x + M. \quad (16)$$

We can rewrite this, originally by e-mail from Martyn Plummer, then also indirectly from Morten Welinder's mentioning of `log1pmx()` in his PR#15628 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, $\frac{x}{M} = 1 + t$.

$$\text{With } t := \frac{x - M}{M} \quad (17)$$

$$\begin{aligned} D_0(x, M) &= \overbrace{M \cdot (1+t)}^x \log(1+t) - \overbrace{t \cdot M}^{x-M} = M \cdot ((t+1) \log(1+t) - t) = \\ &= M \cdot p_1 l_1(t) \stackrel{!}{=} M \cdot d_0(t+1), \end{aligned} \quad (18)$$

where

$$p_1 l_1(t) := (t+1) \log(1+t) - t = \frac{t^2}{2} - \frac{t^3}{6} \pm \dots, \quad (19)$$

$$\begin{aligned} &= (\log(1+t) - t) + t \cdot \log(1+t) \\ &= \log1pmx(t) + t \cdot \log1p(t) \end{aligned} \quad (20)$$

and

$$\log1pmx(t) := \log(1+t) - t \approx -t^2/2 + t^3/3 - t^4/4 \pm \dots \quad (21)$$

The Taylor series expansions for $\log1pmx(t)$ and $p_1 l_1(t)$ are useful for small $|t|$,

$$\begin{aligned} p_1 l_1(t) &= \frac{t^2}{2} - \frac{t^3}{6} + \frac{t^4}{12} \pm \dots = \sum_{n=2}^{\infty} \frac{(-t)^n}{n(n-1)} = \frac{t^2}{2} \sum_{n=2}^{\infty} \frac{(-t)^{n-2}}{n(n-1)/2} = \frac{t^2}{2} \sum_{n=0}^{\infty} \frac{(-t)^n}{\binom{n+2}{2}} = \\ &= \frac{t^2}{2} \left(1 - t \left(\frac{1}{3} - t \left(\frac{1}{6} - t \left(\frac{1}{10} - t \left(\frac{1}{15} - \dots \right) \right) \right) \right) \right), \end{aligned} \quad (22)$$

which we provide in **DPQ** via function `p1l1ser(t, k)` getting the first k terms, and by (18), the corresponding series approximation for

$$p_1 l_1(t) = \lim_{k \rightarrow \infty} \frac{t^2}{2} \sum_{n=0}^k \frac{(-t)^n}{\binom{n+2}{2}} =: \text{p1l1ser}(t, k), \text{ where } t = \frac{x - M}{M}. \quad (23)$$

This Taylor series expansion is useful and nice, but may not even be needed typically, as both utility functions $\log1pmx(t)$ and $\log1p(t)$ are available, implemented to be fully accurate for small t , $t \ll 1$, and (20), indeed, with $t = (x - M)/M$ the evaluation of

$$D_0(x, M) = M \cdot p_1 l_1(t) = M \cdot (\log1pmx(t) + t \cdot \log1p(t)), \quad (24)$$

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

```
> par(mfcol=1:2, mar = 0.1 + c(2.5, 3, 1, 2), mgp = c(1.5, 0.6, 0), las=1)
> p.p1l1( -1, 2, ylim = c(-1,2))
> zoomTo <- function(x,y=x, tx,ty){ arrows(x,-y, tx, ty)
+                                     text(x,-y, "zoom in", adj=c(1/3,9/8)) }
> zoomTo0 <- function(x,y=x) zoomTo(x,y, 0,0)
> zoomTo0(.3)
> p.p1l1(-1e-4, 1.5e-4, ylim=1e-8*c(-.6, 1), do.legend=FALSE)
```

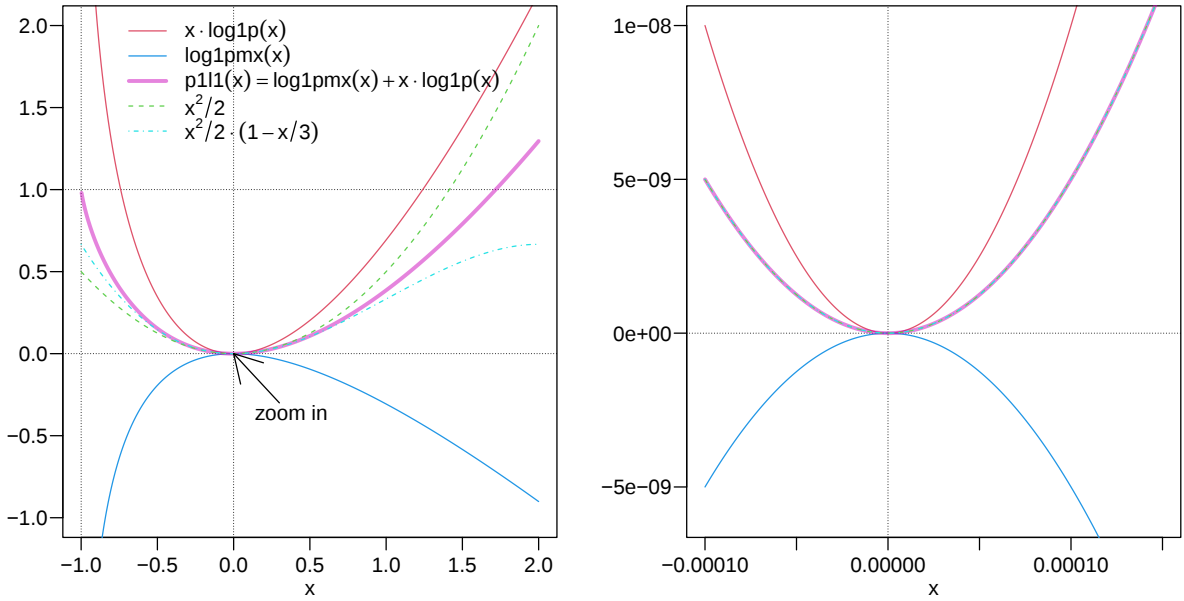


Figure 1: $p_1 l_1(t) = \text{p1l1}()$ and its constituents, $x \cdot \log1p(x)$ and $\log1pmx() = \log1pmx()$, with R functions from our **DPQ** package. On the right, zoomed in 4 and 8 orders of magnitude, where the Taylor approximations $x^2/2$ and $x^2/2 - x^3/6$ are visually already perfect.

Note that $x \cdot \log1p(x)$ and $\log1pmx()$ have different signs, but also note that for small $|x|$, are well approximated by x^2 and $-x^2/2$, so their sum $p_1 l_1(x) = \log1pmx(x) + x \cdot \log1p(x)$ is approximately $x^2/2$ and numerically computing $x^2 - x^2/2$ should only lose 1 or 2 bits of precision.

Note that in Appendix A.1, we show how using different versions of `bd0()` computations for computing Poisson density values, `dpois()`, i.e., our **DPQ** package's `dpois_raw()` leads to differing accurate results.

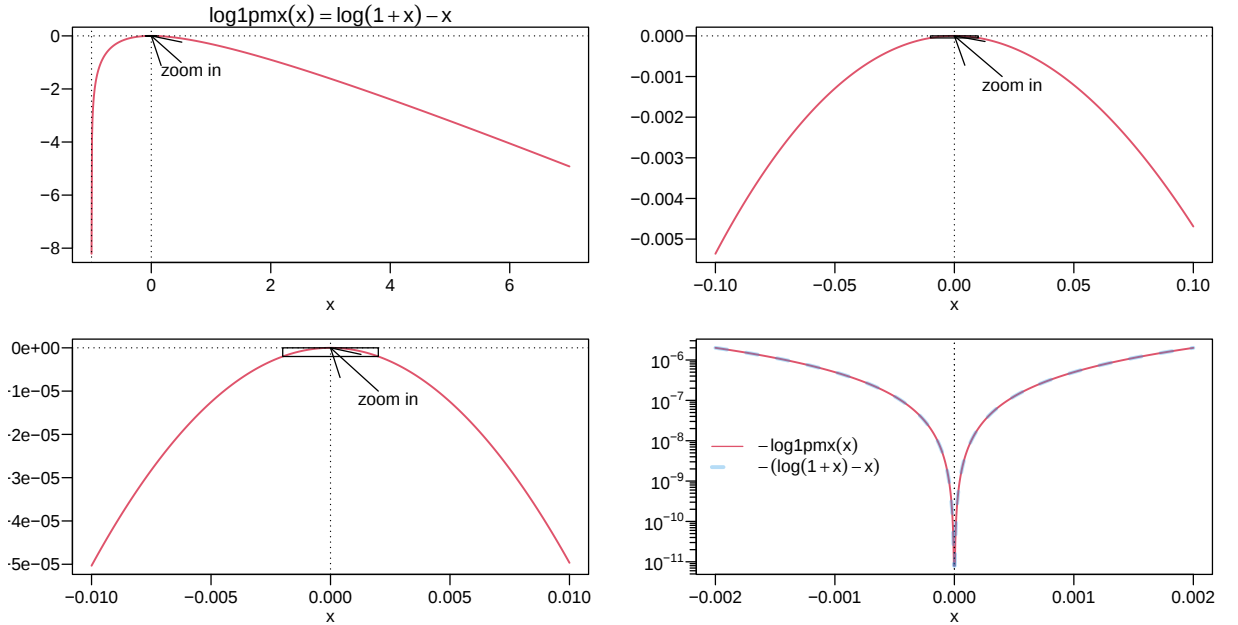
A Accuracy of $\log1pmx(x)$ Computations

As we've seen, the “binomial deviance” function $D_0(x, M) = \text{bd0}(\mathbf{x}, \mathbf{M})$ is crucial for accurate (saddlepoint) computations of binomial, Poisson, etc probabilities, and (at the end

of section 2), one stable way to compute $D_0(x, M)$ is via (24), i.e., with $t = (x - M)/M$, to compute the sum of two terms $D_0(x, M) = M \cdot (\log1pmx(t) + t \cdot \log1p(t))$.

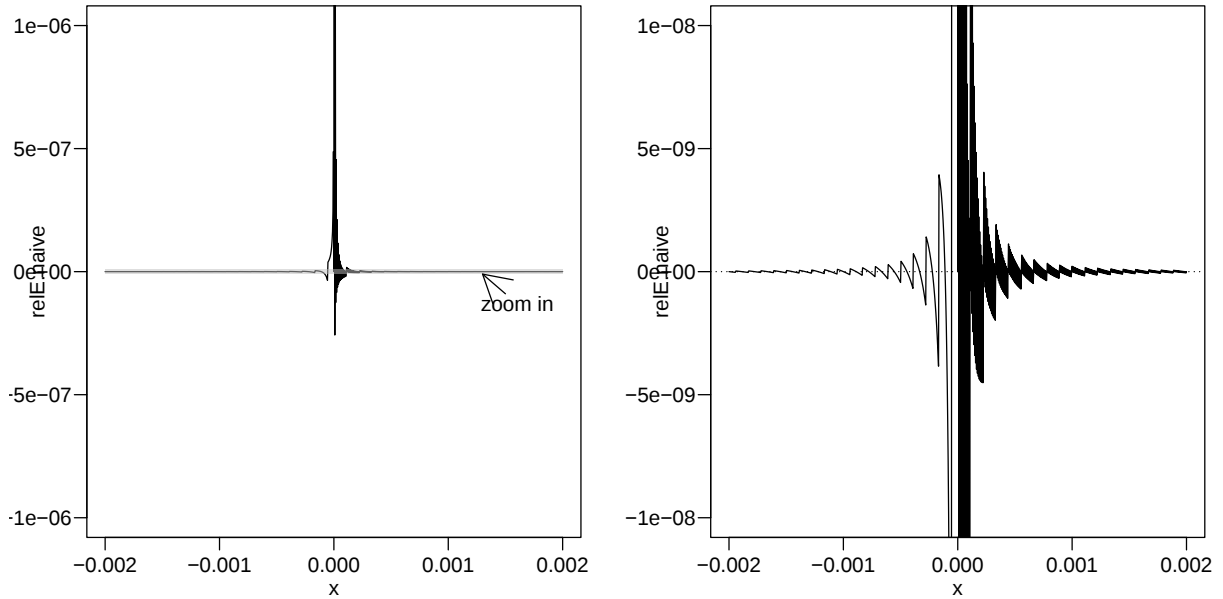
Here, we look more closely at the computation of $\log1pmx(x) := \log(1+x) - x$, at first visualizing the function, notably around (0,0) where numeric cancellations happen if no special care is taken.

```
> lcurve <- function(Fn, a,b, ylab = "", lwd = 1.5, ...)
+   plot(Fn, a,b, n=1001, col=2, ylab=ylab, lwd=lwd, ...,
+       panel.last = abline(h=0, v=-1:0, lty=3))
> par(mfrow=c(2,2), mar = 0.1 + c(2.5, 3, 1, 2), mgp = c(1.5, 0.6, 0), las=1)
> lcurve(log1pmx, -.9999, 7, main=quote(log1pmx(x) == log(1+x)-x))
>
> rect(-.1, log1pmx(-.1), .1, 0); zoomTo0(1/2, 1)
> lcurve(log1pmx, -.1, .1); rect(-.01, log1pmx(-.01), .01, 0); zoomTo0(.02, .001)
> lcurve(log1pmx, -.01, .01); rect(-.002, log1pmx(-.002), .002, 0); zoomTo0(2e-3, 1e-5)
> lcurve(function(x) -log1pmx(x), -.002, .002, log="y", yaxt="n") -> l1r
> sfsmisc::eaxis(2); abline(v=0, lty=3)
> d1r <- cbind(as.data.frame(l1r), y.naive = with(l1r, -(log(1+x)-x)))
> ## --> d1r is data frame w/ ("x", "y", "y.naive")
> c4 <- adjustcolor(4, 1/3)
> lines(y.naive ~ x, data=d1r, col=c4, lwd=3, lty=2)
> legend("left", legend=expression(- log1pmx(x), -(log(1+x)-x)),
+       col=c(palette()[2], c4), lwd=c(1,3), lty=1:2, bty="n")
```



Even if you can't see it in the above 4th plot, the accuracy of our $\log1pmx()$ is already vastly better than the naive $\log(1+x) - x$ computation:

```
> par(mfrow=1:2, mar = 0.1 + c(2.5, 3, 1, 2), mgp = c(1.5, 0.6, 0), las=1)
> d1r[, "relE.naive"] <- with(d1r, sfsmisc::relErrV(y, y.naive))
> plot(relE.naive ~ x, data=d1r, type="l", ylim = c(-1,1)*1e-6)
> y2 <- 1e-8
> rect(-.002, -y2, .002, y2, col=adjustcolor("gray",1/2), border="transparent")
> zoomTo(15e-4, 9*y2, 13e-4, -y2)
> plot(relE.naive ~ x, data=d1r, type="l", ylim = c(-1,1)*y2); abline(h=0,lty=3)
```



Now, we explore the accuracy achieved with R's, i.e., Welinder's algorithm, which uses relatively few terms of a continued-fraction representation of the Taylor series of $\log1pmx(x)$, using package **Rmpfr** and high precision arithmetic. see `'../tests/dnbinom-tst.R'`, 2b: `log1pmx()`. From there, it seems that the (hardcoded currently in R's `'pgamma.c'` as `double minLog1Value = -0.79149064` could or should (?) be changed to around -0.7 or e.g., -0.66.

In **DPQ**'s `log1pmx()` it is the argument `minL1 = -0.79149064`, there's a switch constant `eps2`, (hardwired in current R to `1e-2`, i.e., `eps2 = 0.02`) to switch from an explicit 5-term formula to the full `logcf()` based procedure. In **DPQ**, we already use `eps2 = 0.01` as default. Note that this does *not* influence the choice of `minL1` as long as `eps2` (order of 0.01) is far from the range in which we choose `minL1` (`[-0.85, -0.4]`).

((MM: Still: can we prove that 0.01 is "uniformly" better than 0.02 ?? `"../tests/dnbinom-tst.R"` rather suggests `eps2 = .00163` as optimal *for default tol = 1e-14* .))

A.1 Testing `dpois_raw()` / `dpois()` Poisson probabilities

Testing the Poisson probabilities ('density') with several versions of `bd0()`, `ebd0()` and ..., we found that using Welinder's proposed `ebd0()` was advantageous indeed to get full accuracy, and indeed better than all versions of `bd0()` computations we made available in package **DPQ**. However, the direct formula was more appropriate than `ebd0()` and all `bd0()` version for the cases where x/M or M/x were extremely small. Note: Since March 2022 MM has had another (private, uncommitted) tweak in `'src/nmath/dpois.c'` to *NOT* use `ebd0()` nor `bd0()` but a direct formula, when $2^{-1022} < \frac{x}{\lambda} \leq 2^{-52}$, i.e.,

Look at examples in file `"../man/dgamma-utils.Rd"` and then also `/u/maechler/R/MM/NUMERICS/dpq-functions/15628-dpois_raw_accuracy.R` .

B Accuracy of $p_1 l_1(t)$ Computations

Loader's "Binomial Deviance" $D_0(x, M) = \text{bd0}(x, M)$ function can also be re-expressed (mathematically) as $\text{bd0}(x, M) := D_0(x, M) := M \cdot p_1 l_1((x - M)/M)$ where we look into providing numerically stable formula for $p_1 l_1(t)$, our `p1l1(t)`, as its mathematical formula $p_1 l_1(t) = (t + 1) \log(1 + t) - t$ suffers from cancellation for small $|t|$ even when `log1p(t)`

is used instead of `log(1+t)`; see the derivations (18), (19), and (21) above, and the Taylor series expansion (22) which we provide in our R functions `p111`, and `p111ser`, respectively.

Using a hybrid implementation, `p111()` uses a direct formula, now the stable one in `p111p()`, for $|t| > c$ and a series approximation for $|t| \leq c$ for some cutoff c .

NB: The re-expression via `log1pmx()` is almost perfect; it fixes the cancellation problem entirely (and exposes the fact that `log1pmx()`'s internal cutoff seems sub optimal.

TODO — very unfinished. How much more here?

For now, look at the examples in `?p111`, or even run `example(p111)`.

C Accuracy of `stirlerr(x) = $\delta(x)$` Computations

Note that the “Stirling error”, $\delta(x) \equiv \text{stirlerr}(x)$, $\delta(x) := \log x! - \frac{1}{2} \log(2\pi x) - x \log(x) + x$ by Stirling’s formula is $\delta(x) = \frac{1}{12x} - \frac{1}{360x^3} + \frac{1}{1260x^5} - \frac{1}{1680x^7} + \frac{1}{1188x^9} + O(x^{-11})$, see (9).

A C code implementation had been provided by Loader and for years in R’s Mathlib, further improved by the author. Current version in the R sources at <https://svn.r-project.org/R/trunk/src/nmath/stirlerr.c>, mirrored, e.g., at <https://github.com/wch/r-source/blob/trunk/src/nmath/stirlerr.c>

TODO:

Look at examples in ‘`../tests/stirlerr-tst.R`’ to show the small accuracy loss with Loader’s defaults (for the cut offs of the number of terms used) and also how we explore improving these defaults to improve accuracy.

Consequently, I have have committed the results to the R sources, svn rev 86191, in March 2024, to be used from R version 4.4.0 on.

References

Loader, C. (2000). Fast and accurate computation of binomial probabilities. Technical report, Lucent; Murray Hill, NJ USA; available at <https://www.r-project.org/doc/reports/CLoader-dbinom-2002.pdf>.

Welinder, M. (2004). Bug 7307 - pgamma discontinuity. R’s [PR#7307](#), [comment #6](#).