

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

Martin Mächler  
Seminar für Statistik  
ETH Zurich

April 2021 ff (L<sup>A</sup>T<sub>E</sub>X<sup>ed</sup> December 1, 2022)

## Abstract

The auxiliary function `log1pmx()` (“log 1 plus minus x”), had been introduced when R’s `pgamma()` (incomplete  $\Gamma$  function) had been numerically improved by Morten Welinder’s contribution to R’s PR#7307, in Jan. 2005<sup>1</sup>, it is mathematically defined as  $\log1pmx(x) := \log(1+x) - x$  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. 2014<sup>2</sup>, 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.

## 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  $x = 0, \dots, n$ , and these are (in Rfunction `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()` 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,

---

<sup>1</sup>[https://bugs.R-project.org/show\\_bug.cgi?id=7307#c6](https://bugs.R-project.org/show_bug.cgi?id=7307#c6)

<sup>2</sup>[https://bugs.r-project.org/show\\_bug.cgi?id=15628](https://bugs.r-project.org/show_bug.cgi?id=15628)

even when accurate  $\log \Gamma(x) = \mathbf{lgamma}(x)$  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

$$p(x; n, p) = p(x; n, \frac{x}{n}) \cdot e^{-D(x; n, p)}, \quad (1)$$

where the ‘‘Deviance’’  $D(\cdot)$  is defined as

$$\begin{aligned} D(x; n, p) &= \log p(x; n, \frac{x}{n}) - \log p(x; n, p) \\ &= x \log \left( \frac{x}{np} \right) + (n - x) \log \left( \frac{n - x}{n(1 - p)} \right), \end{aligned} \quad (2)$$

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

$$\begin{aligned} D(x; n, p) &= np \tilde{D}_0\left(\frac{x}{np}\right) + nq \tilde{D}_0\left(\frac{n - x}{nq}\right) \\ &= D_0(x, np) + D_0(n - x, nq), \end{aligned} \quad (3)$$

where  $q := 1 - p$  and

$$\tilde{D}_0(r) := r \log(r) + 1 - r \quad \text{and} \quad (4)$$

$$D_0(x, M) := M \cdot \tilde{D}_0(x/M) \quad (5)$$

$$= M \cdot \left( \frac{x}{M} \log \left( \frac{x}{M} \right) + 1 - \frac{x}{M} \right) = x \log \left( \frac{x}{M} \right) + M - x \quad (6)$$

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

$$\tilde{D}_0(0) := 1 \quad \text{and} \quad (7)$$

$$D_0(0, M) := M \tilde{D}_0(0) = M \cdot 1 = M$$

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

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

Evaluating of  $p(x; n, p)$  in (1), in addition to  $D(x; n, p)$  in (3) also needs  $p(x; n, \frac{x}{n})$  where in turn, the Stirling De Moivre series is used:

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

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

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

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

<sup>4</sup>[https://bugs.r-project.org/show\\_bug.cgi?id=15628](https://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 (5), (6) for  $D_0(x, M)$  apply and are needed, e.g., for `pgamma()` computations for general non-negative  $(x, M > 0)$  where even  $x = 0$  is well defined, see (7) above.

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

$$p_\lambda(x) = e^{-\lambda} \frac{\lambda^x}{x!} \tag{11}$$

$$\begin{aligned} \log p_\lambda(x) &= -\lambda + x \log \lambda - \underbrace{\log(x!)}_{\log(1/\sqrt{2\pi x}) - (x \log x - x + \delta(x))} \\ &= \log \frac{1}{\sqrt{2\pi x}} - x \log \frac{x}{\lambda} + x - \lambda - \delta(x), \end{aligned} \tag{12}$$

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

$$p_\lambda(x) = \frac{1}{\sqrt{2\pi x}} e^{-\delta(x) - D_0(x, \lambda)} \tag{13}$$

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

Even for the  $t_\nu$  density, `dt()`, .....

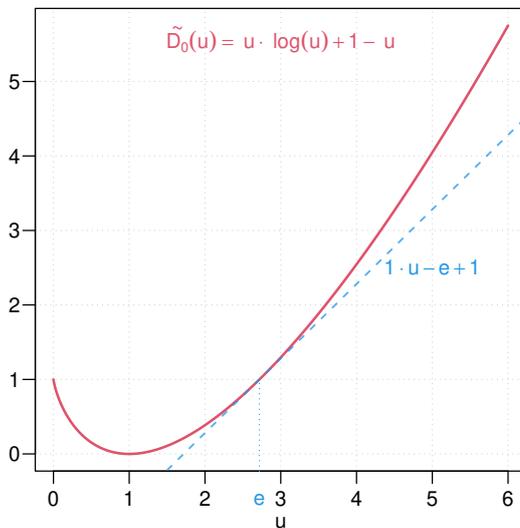
... but there have a direct approximations in package **DPQ**, currently functions `c_dt(nu)` and even more promisingly, `lb_chi(nu)`. ..... TODO .....

## 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) :

$$\begin{aligned} D_0(x, M) &:= M \cdot \tilde{D}_0\left(\frac{x}{M}\right), \quad \text{where} \\ \tilde{D}_0(u) &:= u \log(u) + 1 - u = u(\log(u) - 1) + 1. \end{aligned}$$

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



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  $D_0(x, M) = M \cdot \tilde{D}_0(\frac{x}{M})$  is numerically problematic. Further,

$$D_0(x, M) = M \cdot \left( \frac{x}{M} \left( \log\left(\frac{x}{M}\right) - 1 \right) + 1 \right) = x \log\left(\frac{x}{M}\right) - x + M. \quad (14)$$

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,  $\frac{x}{M} = 1 + t$ . Using  $t$ ,

$$t := \frac{x - M}{M} \quad (15)$$

$$\begin{aligned} D_0(x, M) &= \overbrace{M \cdot (1 + t)}^x \log(1 + t) - \overbrace{t \cdot M}^{x-M} = M \cdot ((t + 1) \log(1 + t) - t) = \\ &= M \cdot p_1 l_1(t), \end{aligned} \quad (16)$$

where

$$p_1 l_1(t) := (t + 1) \log(1 + t) - t = \frac{t^2}{2} - \frac{t^3}{6} \pm \dots, \quad (17)$$

$$\begin{aligned} &= (\log(1 + t) - t) + t \cdot \log(1 + t) \\ &= \log1pmx(t) + t \cdot \log1p(t) \end{aligned} \quad (18)$$

where

$$\log1pmx(x) := \log(1 + x) - x \approx -x^2/2 + x^3/3 - x^4/4 \pm \dots, \quad (19)$$

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

$$p_1 l_1(t) = \frac{t^2}{2} - \frac{t^3}{6} + \frac{t^4}{12} \pm \dots = \sum_{n=2}^{\infty} \frac{(-t)^n}{n(n-1)} = \frac{t^2}{2} \sum_{n=2}^{\infty} \frac{(-t)^{n-2}}{n(n-1)/2} = \frac{t^2}{2} \sum_{n=0}^{\infty} \frac{(-t)^n}{\binom{n+2}{2}} = \quad (20)$$

$$= \frac{t^2}{2} \left( 1 - t \left( \frac{1}{3} - t \left( \frac{1}{6} - t \left( \frac{1}{10} - t \left( \frac{1}{15} - \dots \right) \right) \right) \right) \right), \quad (21)$$

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

$$D_0(x, M) = \lim_{k \rightarrow \infty} \text{p1l1ser}\left(\frac{x - M}{M}, k, F = \frac{(x - M)^2}{M}\right), \quad (22)$$

where the approximation of course uses a finite  $k$  instead of the limit  $k \rightarrow \infty$ .

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 (18), indeed, with  $t = (x - M)/M$  the evaluation of

$$D_0(x, M) = M \cdot p_1 l_1(t) = M \cdot (\log1pmx(t) + t \cdot \log1p(t)), \quad (23)$$

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

Note that  $x * \log1p(x)$  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.

```

> par(mfcol=1:2, mar = 0.1 + c(2.5, 3, 1, 2), mgp = c(1.5, 0.6, 0), las=1)
> p.p111(-7/8, 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.p111(-1e-4, 1.5e-4, ylim=1e-8*c(-.6, 1), do.legend=FALSE)

```

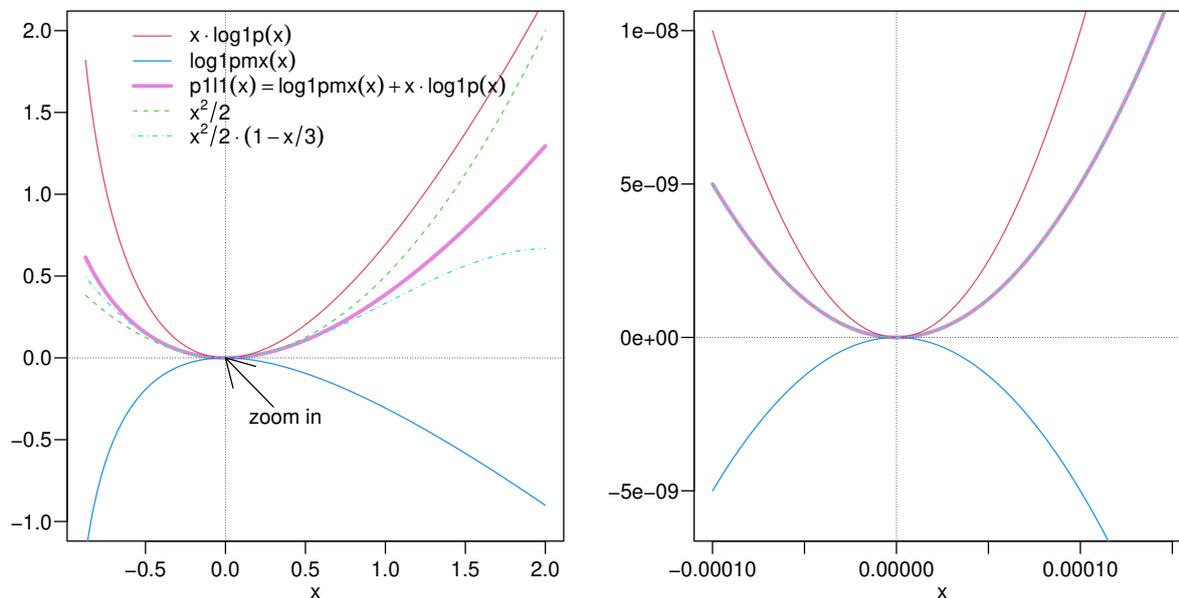


Figure 1:  $p_{11}(t) = p_{11}()$  and its constituents,  $x \cdot \log_1 p(x)$  and  $\log_1 p_{mx}() = \log_1 p_{mx}()$ , with Rfunctions from our **DPQ** package. On the right, zoomed in 4 and 8 orders of magnitude, where the Taylor approximations  $x^2/2$  and  $x^2/2 - x^3/6$  are visually already perfect.

## A Accuracy of $\log_1 p_{mx}(x)$ Computations

As we've seen, the ‘‘binomial deviance’’ function  $D_0(x, M) = \text{bd0}(x, M)$  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 (23), i.e., with  $t = (x - M)/M$ , to compute the sum of two terms  $D_0(x, M) = M \cdot (\log_1 p_{mx}(t) + t \cdot \log_1 p(t))$ .

Here, we look more closely at the computation of  $\log_1 p_{mx}(x) := \log(1+x) - x$ , at first visualizing the function, notably around  $(0,0)$  where numeric cancellations happen if no special care is taken.

```

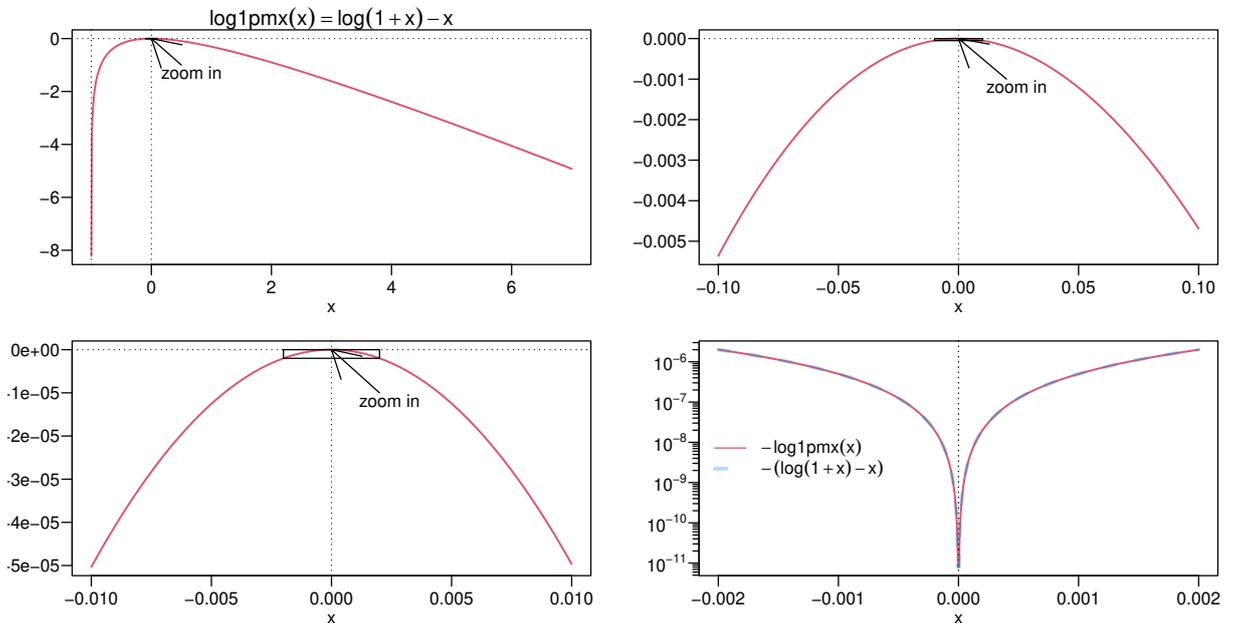
> lcurve <- function(Fn, a,b, ylab = "", lwd = 1.5, ...)
+   plot(Fn, a,b, n=1001, col=2, ylab=ylab, lwd=lwd, ...,
+       panel.last = abline(h=0, v=-1:0, lty=3))
> par(mfrow=c(2,2), mar = 0.1 + c(2.5, 3, 1, 2), mgp = c(1.5, 0.6, 0), las=1)
> lcurve(log1pmx, -.9999, 7, main=quote(log1pmx(x) == log(1+x)-x))
>   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)))
> 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")

```

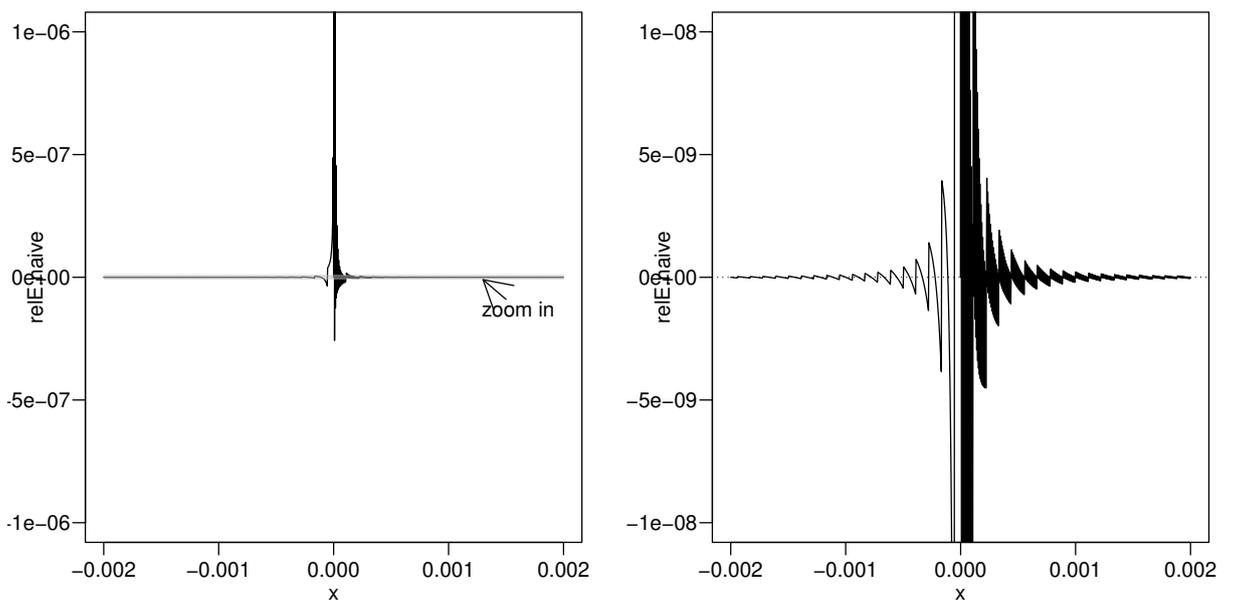


The accuracy of our `log1pmx()` is already vastly better than the naive  $\log(1+x) - x$  computation:

```

> 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 as continued-fraction representation of the Taylor series of  $\log_1\text{pmx}(x)$ , 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 ?? ))

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

Testing the Poisson probabilities ('density') with several versions of `bd0()`, `ebd0()` and the direct formula where more appropriate (non-log case, Look at examples in `"./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  $\text{bd0}(x, M) = M * p1l1((x - M)/M)$  where we look into providing numerically stable formula for  $p1l1(t)$  as its mathematical formula  $p1l1(t) = (t + 1) \log(1 + t) - t$  suffers from cancellation for small  $|t|$  even when `log1p(t)` is used instead of `log(1+t)`; see the derivations (16), (17), and (19) above, and the Taylor series expansion (20) which we provide in our Rfunctions `p1l1()`, and `p1l1ser`, respectively.

Using a hybrid implementation, `p1l1()` uses a direct formula, now the stable one in `p1l1p()`, 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 `?p1l1`, or even run `example(p1l1)`.

## C Accuracy of $\text{stirlerr}(x) = \delta(x)$ 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 now in R's Mathlib 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.

## References

Loader, C. (2000). Fast and accurate computation of binomial probabilities. Technical report, Lucent; Murray Hill, NJ USA.