

User Guide to Bayesian-Limit Software Package

Joel Heinrich for the CDF Statistics Committee

October 27, 2004

1 Introduction

Reference [1] examines a Bayesian approach to calculating an upper limit on a cross section in the presence of uncertainties on the acceptance and background level. The software written for that study is available for general use through the CDF Statistics Committee's web site[2]; this note provides additional documentation for users of that software.

Although the code was written in C, care was taken to ensure that the code also compiles and runs as C++. Because all necessary integrations are performed analytically in Ref. [1], the CPU time required to calculate a single limit using this software is of order 10^{-3} seconds. This speed makes it especially useful for Toy MC simulations, where, for example, limits might be needed for thousands of pseudo-experiments. Numerical precision is of order 12 digits.

2 The Problem

A complete description is given in [1], but it is useful to briefly summarize the situation here. We observe n events, including both signal and background, from which we will set an upper limit s_u , at confidence (or credibility) level β , on the cross section s . We have estimates of the expected background and efficiency; $b_0 \pm \sigma_b$ and $\epsilon_0 \pm \sigma_\epsilon$ respectively. The n observed events then represent a single deviate from a Poisson distribution having a mean of $s\epsilon + b$.

The prior p.d.f. for s is $s^{\alpha-1}ds$ where α is a constant chosen by the experimenter. Typical choices are $\alpha = 1$ and $\alpha = 0.5$.

The prior for ϵ is given by a gamma distribution[3]

$$\frac{\kappa(\kappa\epsilon)^{\mu-1}e^{-\kappa\epsilon}}{\Gamma(\mu)}d\epsilon$$

Here μ/κ is the mean of the ϵ -prior, $(\mu - 1)/\kappa$ is the mode, and μ/κ^2 is the variance. As ϵ includes factors involving luminosity as well as acceptance, it is not required to

be less than 1; it may have any positive value. The Bayesian-Limit routines use the mean and the variance to characterize the prior for ϵ , i.e.

$$\epsilon_0 \equiv \mu/\kappa \qquad \sigma_\epsilon \equiv \sqrt{\mu}/\kappa$$

One must be careful to match this convention. For example, should the user have an estimate of the efficiency based on the mode and the 2nd derivative at the mode, i.e. $\hat{\epsilon} \pm \hat{\sigma}_\epsilon$ where $\hat{\epsilon} = (\mu - 1)/\kappa$ and $\hat{\sigma}_\epsilon = \sqrt{\mu - 1}/\kappa$, we would then use

$$\epsilon_0 = \hat{\epsilon}[1 + (\hat{\sigma}_\epsilon/\hat{\epsilon})^2] \qquad \sigma_\epsilon = \hat{\sigma}_\epsilon\sqrt{1 + (\hat{\sigma}_\epsilon/\hat{\epsilon})^2}$$

when calling these routines.

The prior for b is also a gamma distribution,

$$\frac{\omega(\omega b)^{\rho-1}e^{-\omega b}}{\Gamma(\rho)}db$$

and our comments about the ϵ -prior apply here as well, after the substitutions $\mu \rightarrow \rho$, $\kappa \rightarrow \omega$, etc. For example, we have

$$b_0 \equiv \rho/\omega \qquad \sigma_b \equiv \sqrt{\rho}/\omega$$

3 Special Functions

Here we list the special functions that are involved in the solution of this problem. These brief summaries will help the user to understand the actual code and the equations in the next section. In some cases, we also mention the syntax for these functions in Maple and Mathematica, which can be useful for verification. Reference [4] is the ultimate source of information about special functions.

3.1 $\Gamma(x)$

The gamma function $\Gamma(x)$ is provided by the Unix/Linux standard math library as `double lgamma(double x)` which returns $\log(\Gamma(x))$.

3.2 $a^{\bar{k}}$ and $a^{\underline{k}}$

The “rising factorial” notation $a^{\bar{k}}$ is defined as

$$a^{\bar{k}} \equiv \frac{\Gamma(a+k)}{\Gamma(a)} = a(a+1)(a+2)\cdots(a+k-1)$$

and is read as “ a to the k rising”. The “falling factorial” notation [5] $a^{\underline{k}}$ is defined as

$$a^{\underline{k}} \equiv \frac{\Gamma(a+1)}{\Gamma(a-k+1)} = a(a-1)(a-2)\cdots(a-k+1)$$

and is read as “ a to the k falling”.

3.3 $M(a, b, x)$

The confluent hypergeometric function¹ is

$$M(a, b, x) = 1 + \frac{a x}{b 1!} + \frac{a(a+1) x^2}{b(b+1) 2!} + \frac{a(a+1)(a+2) x^3}{b(b+1)(b+2) 3!} + \dots$$

When a is an integer ≤ 0 this series terminates, so $M(-n, b, x)$ is a polynomial of order n in x .

3.4 $F(a, b; c; x)$

The hypergeometric function² is

$$F(a, b; c; x) = 1 + \frac{ab x}{c 1!} + \frac{a(a+1)b(b+1) x^2}{c(c+1) 2!} + \frac{a(a+1)(a+2)b(b+1)(b+2) x^3}{c(c+1)(c+2) 3!} + \dots$$

Once again, the series terminates when a or b are non positive integers.

3.5 $I_x(a, b)$

The incomplete beta function³ is

$$I_x(a, b) \equiv \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1} (1-t)^{b-1} dt = \frac{x^a (1-x)^b \Gamma(a+b)}{\Gamma(a+1)\Gamma(b)} F(a+b, 1; a+1; x)$$

The file `incompletebeta.c` defines the function

```
double incompletebeta(double a, double b, double x)
```

that evaluates the incomplete beta function via the hypergeometric series, using $I_x(a, b) = 1 - I_{1-x}(b, a)$ when $x > (a+1)/(a+b+2)$ to speed convergence.

The recursion

$$I_x(a, b) = \frac{\Gamma(a+b)}{\Gamma(a+1)\Gamma(b)} x^a (1-x)^b + I_x(a+1, b)$$

is used in several routines to replace a whole series of I_x calls.

3.6 $P(a, x)$

The incomplete gamma function⁴

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

¹`KummerM(a,b,x)` in Maple, `Hypergeometric1F1[a,b,x]` in Mathematica

²`hypergeom([a,b],[c],x)` in Maple, `Hypergeometric2F1[a,b,c,x]` in Mathematica

³`BetaRegularized[x,a,b]` in Mathematica

⁴`1-GAMMA(a,x)/GAMMA(a)` in Maple, `1-GammaRegularized[a,x]` in Mathematica

is defined in the file `incompletegamma.c` as `double incompletegamma(double a, double x)`. The function is evaluated using its power series expansion or its asymptotic expansion, depending on the relative sizes of x and a .

The recursion

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

is used in several routines to eliminate a whole series of calls to $P(a - k, x)$ for integer k .

4 The Software

The header file `bayesianlimit.h` declares all public functions. For a complete explanation of the Bayesian solution, see Ref. [1]. The complete list of files is:

- `bayesianlimit.h`
- `posterior.c`
- `postint.c`
- `blimit.c`
- `incompletebeta.c`
- `incompletegamma.c`

4.1 the marginalized posterior p.d.f. for s

The file `posterior.c` defines two public functions:

- `double posterior(double s, int n, double e0, double esig, double b0, double bsig, double alpha);`

returns the value of the posterior p.d.f. by directly evaluating⁵

$$p(s|n)ds = \frac{\Gamma(\mu + n)}{\Gamma(\mu - \alpha)\Gamma(\alpha + n)} \frac{s^{\alpha+n-1}\kappa^{\mu-\alpha} F(-n, \rho; 1 - n - \mu; (s + \kappa)/(s(\omega + 1)))}{(s + \kappa)^{\mu+n} F(-n, \rho; 1 - n - \alpha; 1/(\omega + 1))} ds$$

- The special case $\sigma_b = 0$ is evaluated by the function

`double posteriorb0(double s, int n, double e0, double esig, double b, double alpha);`

which returns the value of the posterior p.d.f. by directly evaluating

$$p(s|b, n)ds = \frac{\Gamma(\mu + n)}{\Gamma(\mu - \alpha)\Gamma(\alpha + n)} \frac{s^{\alpha+n-1}\kappa^{\mu-\alpha} M(-n, 1 - n - \mu, b(s + \kappa)/s)}{(s + \kappa)^{\mu+n} M(-n, 1 - n - \alpha, b)}$$

Note that `posterior(s, n, e0, esig, b0, 0, alpha)` is entirely equivalent to `posteriorb0(s, n, e0, esig, b0, alpha)`.

⁵Previous versions of this note incorrectly had ω instead of $\omega + 1$ in this equation.

4.2 the integral of the posterior p.d.f.

The file `postint.c` defines three public functions:

- `double postint(double su,int n,double e0,double esig,
double b0,double bsig,double alpha);`

returns the integral of the marginalized posterior p.d.f. for s from 0 to s_u by evaluating⁶

$$\int_0^{s_u} p(s|n)ds = \sum_{k=0}^n \frac{I_x(\alpha + n - k, \mu - \alpha) n^k \rho^{\bar{k}} (\omega + 1)^{-k}}{(\alpha + n - 1)^{\underline{k}} k!} \bigg/ \sum_{k=0}^n \frac{n^k \rho^{\bar{k}} (\omega + 1)^{-k}}{(\alpha + n - 1)^{\underline{k}} k!}$$

where $x = \frac{s_u}{s_u + \kappa}$.

- The special case $\sigma_b = 0$ is handled by

`double postintb0(double su,int n,double e0,double esig,
double b,double alpha);`

which uses

$$\int_0^{s_u} p(s|b,n)ds = \sum_{k=0}^n \frac{I_x(\alpha + n - k, \mu - \alpha) n^k b^k}{(\alpha + n - 1)^{\underline{k}} k!} \bigg/ \sum_{k=0}^n \frac{n^k b^k}{(\alpha + n - 1)^{\underline{k}} k!} \quad \left(x = \frac{s_u}{s_u + \kappa} \right)$$

- The special case $\sigma_\epsilon = 0$ and $\sigma_b = 0$ is handled by

`double postinte0b0(double su,int n,double e,double b,double alpha);`

which uses

$$\int_0^{s_u} p(s|\epsilon,b,n)ds = \sum_{k=0}^n \frac{P(\alpha + n - k, \epsilon s_u) n^k b^k}{(\alpha + n - 1)^{\underline{k}} k!} \bigg/ \sum_{k=0}^n \frac{n^k b^k}{(\alpha + n - 1)^{\underline{k}} k!}$$

4.3 the upper limit calculator

These functions, which calculate the upper limit s_u , are the items of greatest interest for the general user.

The file `blimit.c` defines three functions that invert the functions of the previous subsection:

- `double blimit(double beta,int n,double e0,double esig,
double b0,double bsig,double alpha);`

returns s_u such that $\int_0^{s_u} p(s|n)ds = \beta$

- `double blimitb0(double beta,int n,double e0,double esig,
double b,double alpha);`

returns s_u such that $\int_0^{s_u} p(s|b,n)ds = \beta$

- `double blimite0b0(double beta,int n,double e,double b,double alpha);`

returns s_u such that $\int_0^{s_u} p(s|\epsilon,b,n)ds = \beta$

⁶Previous versions of this note incorrectly had ω instead of $\omega + 1$ in this equation.

References

- [1] Joel Heinrich et al., “Interval estimation in the presence of nuisance parameters. 1. Bayesian approach.”, CDF Internal Note 7117, (2004), www-cdf.fnal.gov/publications/cdf7117_bayesianlimit.pdf .
- [2] www-cdf.fnal.gov/physics/statistics/statistics_software.html
- [3] S. Eidelman et al., Physics Letters **B592**, 1 (2004), §31.4.6, pdg.lbl.gov/2004/reviews/probrpp.ps ; William H. Press, et al., “Numerical Recipes”, 2nd edition, (Cambridge University Press, Cambridge, 1992), §7.3, lib-www.lanl.gov/numerical/bookcpdf/c7-3.pdf .
- [4] M. Abramowitz and I.A. Stegun, editors, “Handbook of Mathematical Functions”, (United States Department of Commerce, National Bureau of Standards, Washington, D.C. 1964; and Dover Publications, New York, 1968).
- [5] R. L. Graham, D. E. Knuth, and O. Patashnik, “Concrete Mathematics: A Foundation for Computer Science”, 2nd ed., (Addison-Wesley, Reading, MA, 1994); planetmath.org/encyclopedia/FallingFactorial.html .