Numerical integration

To be fancy, our topic is *quadrature*.

We seek $I = \int_a^b f(x)\, dx$.

Method 1: *The extended midpoint rule*

1. Pick N large.
2. Let $x_i = a + (i - 1/2)h$, $h = (b - a)/N$, for $i = 1, \ldots, N$.
3. Let $f_i = f(x_i)$
4. Then $I \approx h \sum_i f_i$

Sample code:

```
emr <-
function(f,a,b,n=1000)
{
  h <- (b-a)/n
  h*sum(f( seq(a+h/2, b, by=h) ))
}
```

Points:

- This is the simplest thing to do
- Error $O(1/N^2)$
- Don't use this in production

The basics

In the following, we consider $x_0, x_1, \ldots x_N$ such that $x_i = x_0 + ih$, and write $f_i = f(x_i)$.

Note: The common choice is $x_0 = a$ and $h = (b-a)/N$, so that $x_N = b$.

Closed formula: use x_0, x_N

Open formula: don't use x_0, x_N (e.g., when it's hard to calculate f at one of a or b)

Basic rules

Trapezoidal rule: $\int_{x_0}^{x_1} f(x)\,dx \approx h\{\frac{1}{2}f_0 + \frac{1}{2}f_1\}$

Simpson's rule: $\int_{x_0}^{x_2} f(x)\,dx \approx h\{\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{1}{3}f_2\}$

Extrapolation formulas: $\int_{x_0}^{x_1} f(x)\,dx \approx h\{f_1\}$

$\int_{x_0}^{x_1} f(x)\,dx \approx h\{\frac{3}{2}f_1 - \frac{1}{2}f_2\}$

$\int_{x_0}^{x_1} f(x)\,dx \approx h\{\frac{23}{12}f_1 - \frac{16}{12}f_2 + \frac{5}{12}f_3\}$

The basics (continued)

Extended trapezoidal rule (closed)

$$\int_a^b f(x)\,dx = h\{\tfrac{1}{2}f_0 + f_1 + f_2 + \ldots + f_{N-1} + \tfrac{1}{2}f_N\} + O(\tfrac{1}{N^2})$$

Extended Simpson's rule (closed)

$$\int_a^b f(x)\,dx = h\{\tfrac{1}{3}f_0 + \tfrac{4}{3}f_1 + \tfrac{2}{3}f_2 + \tfrac{4}{3}f_3 + \ldots + \tfrac{2}{3}f_{N-2} + \tfrac{4}{3}f_{N-1} + \tfrac{1}{3}f_N\} + O(\tfrac{1}{N^4})$$

Extended trapezoidal rule (open)

$$\int_a^b f(x)\,dx = h\{\tfrac{3}{2}f_1 + f_2 + \ldots + f_{N-2} + \tfrac{3}{2}f_{N-1}\} + O(\tfrac{1}{N^2})$$

Extended Simpson's rule (open)

$$\int_a^b f(x)\,dx = h\{\tfrac{27}{12}f_1 + 0 + \tfrac{13}{12}f_3 + \tfrac{4}{3}f_4 + \tfrac{2}{3}f_5 + \ldots + \tfrac{4}{3}f_{N-4} + \tfrac{13}{12}f_{N-3} + 0 + \tfrac{27}{12}f_{N-1}\} + O(\tfrac{1}{N^4})$$

Trapezoidal rule: Implementation

An important point regarding the trapezoidal rule: to go from N to $2N$, you can make use of your previous work.

We make use of the following basic engine for the trapezoidal rule. I assume that the function `f` is vectorized and that `n` is a power of 2.

```
subtrap <-
function(f,a,b,n=1)
{
  h <- (b-a)/n
  if(n==1) return( h*mean( f(c(a,b)) ) )
  else return(h*sum( f(seq(a+h,b,by=2*h)) ))
}
```

We use the above function by taking $I_{i+1} = \frac{1}{2}I_i + S_{i+1}$, where S_i is the output from the function `subtrap` and I_i is the estimated integral using $N = 2^{i-1}$ steps.

```
trap <-
function(f,a,b,tol=1e-8,maxit=1000)
{
  i.old <- subtrap(f,a,b,1); n <- 2
  for(i in 2:maxit) {
    s <- subtrap(f,a,b,n)
    i.new <- i.old/2 + s
    if(abs(i.new-i.old) < tol) break
    i.old <- i.new
    n <- n*2
  }
  i.new
}
```

Simpson's rule: Implementation

It is interesting to note that you can use the `subtrap` function to obtain Simpson's rule.

Let S_i be the output from `subtrap` with $N = 2^{i-1}$, and let I_i be our estimate of the integral at step i.

For Simpson's rule, we take $I_1 = 2S_1/3$ and then $I_{i+1} = I_i/2 + (4S_{i+1} - S_i)/3$.

```
simp <-
function(f,a,b,tol=1e-8,maxit=1000)
{
  i.old <- subtrap(f,a,b,1)*2/3
  n <- 2;  old.s <- 0
  for(i in 2:maxit) {
    s <- subtrap(f,a,b,n)
    i.new <- (i.old/2 + (4*s - old.s)/3)
    if(abs(i.new-i.old) < tol) break
    i.old <- i.new; old.s <- s
    n <- n*2
  }
  i.new
}
```

Consider the example $\int_0^2 x^2(1-x)sin(x^2)\,dx$.

With `tol=1e-8`, Simpson's rule uses $N = 512$ while the trapezoidal rule uses $N = 512 \times 128$.

Further explanation

Let $S(h)$ and $T(h)$ denote the approximations from Simpson's rule and the trapezoidal rule, respectively.

Then $S(h) = \frac{1}{3}\{4T(h) - T(2h)\}$

The `subtrap` function provides $X(h) = T(h) - \frac{1}{2}T(2h)$.

$$\begin{aligned}
\text{Thus } S(h) &= \tfrac{1}{3}\{4T(h) - T(2h)\} \\
&= \tfrac{1}{3}\{[2T(2h) - \tfrac{1}{2}T(4h)] + [4T(h) - \\
&\qquad 2T(2h)] - [T(2h) + \tfrac{1}{2}T(4h)]\} \\
&= \tfrac{1}{2} \cdot \tfrac{1}{3}\{4T(2h) - T(4h)\} + \tfrac{4}{3}\{T(h) - \\
&\qquad \tfrac{1}{2}T(2h)\} - \tfrac{1}{3}\{T(2h) - \tfrac{1}{2}T(4h)\} \\
&= \tfrac{1}{2}S(2h) + \tfrac{1}{3}\{4X(h) - X(2h)\}
\end{aligned}$$

Romberg's algorithm

Simpson's rule combines $T(h)$ and $T(2h)$ in a way that some error terms are cancelled. In Romberg's algorithm, this method is continued.

If the function being integrated is sufficiently smooth, impressive gains are obtained.

Let $T_{m0} = T[(b-a)2^{-m}]$ and define

$$\begin{aligned} T_{mn} &= \frac{4^n T_{m,n-1} - T_{m-1,n-1}}{4^n - 1} \\ &= T_{m,n-1} - \tfrac{1}{4^n-1}\{T_{m-1,n-1} - T_{m,n-1}\} \end{aligned}$$

The Romberg matrix:

$$\begin{pmatrix} T_{00} & & & \\ T_{10} & T_{11} & & \\ T_{20} & T_{21} & T_{22} & \\ T_{30} & T_{31} & T_{32} & T_{33} \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

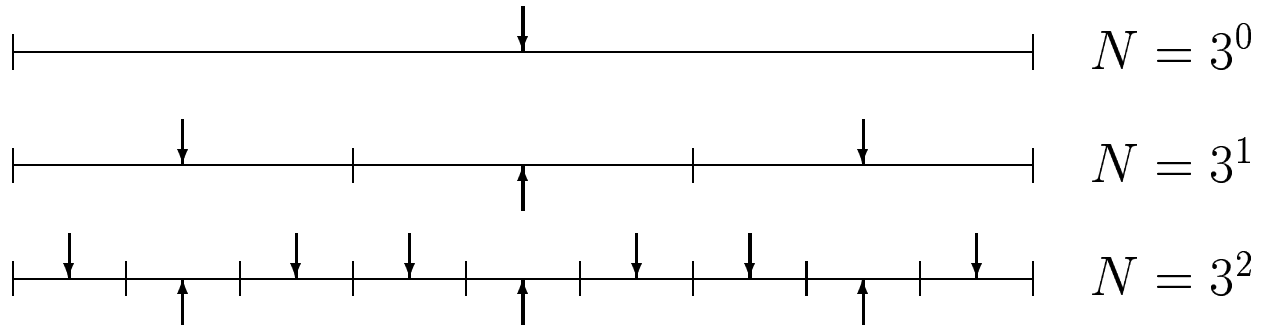
Recall that if X_i is the value from the ith call to `subtrap`, $T_{00} = X_1$ and $T_{i0} = X_{i+1} + \frac{1}{2}T_{i-1,0}$.

Improper integrals

The above methods need to be revised in the case that

1. The integrand goes to a finite limit at finite upper and lower limits, but cannot be calculated right on one of the limits (e.g., $\sin x/x$ at $x = 0$).
2. The upper limit of integration is ∞ or the lower limit is $-\infty$.
3. There is a singularity at either limit (e.g., $x^{-1/2}$ at $x = 0$.
4. There is a singularity at a known spot between the limits.

Solution: Replace the function `subtrap` with a version of the extended midpoint rule, tripling N at each call.



Use code like

```
x <- c(seq(a+h/2,b,by=3*h), seq(a+2.5*h,b,by=3*h))
```

We can easily revise `trap` and `simp` to use this. We thus solve issue 1.

Improper integrals (continued)

To solve the other issues, we use two techniques:

1. Change of variables.
2. Break up the integral into pieces

For example, if $a > 0$ and $f(t) \to 0$ faster than $1/t^2 \to 0$ as $t \to \infty$, then we can use $u = 1/t$ as follows:

$$\int_a^\infty f(t)\,dt = \int_0^{1/a} \tfrac{1}{u^2} f(\tfrac{1}{u})\,du$$

This also works if $b < 0$ and the lower limit is $-\infty$.

I think that's enough on that; you can see the general direction. See Lange §*.* for additional advice and examples.

Gaussian quadrature

The amazingly cool technique of *Gaussian quadrature* is appropriate when we wish to integrate $f(x)$ against a probability measure μ.

For fixed points x_i and positive weights w_i,

$$\int_{-\infty}^{\infty} f(x)\, d\mu(x) \approx \sum_{i=0}^{k} w_i f(x_i)$$

For the trapezoidal rule, $\mu = \text{uniform}(a, b)$ and the x_i are uniformly spaced. Generally, μ is non-uniform and the x_i cluster in regions of high probability.

If μ possesses an orthonormal polynomial sequence $\psi_n(x)$ for $n = 0, 1, 2, \ldots$, then $x_0, x_1, \ldots, x_k$ are the roots of $\psi_{k+1}(x)$.

Fact: If μ is not concentrated at a finite number of points, then the roots $\{x_i\}$ are real and distinct, and there exist positive weights w_i such that the quadrature formula above is exact when $f(x)$ is a polynomial of degree $\leq 2k + 1$.

Orthonomial polynomials (continued)

Consider the inner product

$$< f, g >= \int_{-\infty}^{\infty} f(x)g(x)\, d\mu(x)$$

The orthonomial polynomials mentioned above satisfy $< p_i, p_j >= \delta_{ij}$ (i.e., $= 0$ if $i \neq j$ and $= 1$ if $i = j$)

They can be constructed as follows:

Take $p_{-1}(x) = 0$ and $p_0(x) = 1$.

For $j \geq 0$, let $q_{j+1} = (x - a_j)p_j(x) - b_j p_{j-1}(x)$
and let $p_j = q_j / \sqrt{< q_j, q_j >}$

My recommendation: if you come across an integral of the form $\int f(x)\ d\mu(x)$, consider using Gaussian quadrature, and check out *Numerical Recipes in C*, §*.*.

Rather than calculating the abscissas x_i and weights w_i, you may wish to pull them from tables (in an old and dusty book).

Example 1

[Chakravarti et al., Genetics 128:175–182, 1991]

Suppose a genome consists of k chromosomes of common length L, and that we have typed m markers on a set of fully informative offspring. Let $G = kL$ be the genome length.

Let n_{ij} be the number of meioses typed on both markers i and j and r_{ij} be the number of recombinants observed.

We assume (falsely) no interference and (even more falsely) that the pairs are independent, and wish to estimate L (and hence G).

Log likelihood:

$$l(L) = \sum_{i \neq j} \log\{\int_0^{\theta'} \theta^{r_{ij}}(1-\theta)^{n_{ij}-r_{ij}} f(\theta; L)\, d\theta + (k-1)(1/2)^{n_{ij}}\}$$

where $\theta' = (1 - e^{-2L})/2$ (ie, the maximum recombination fraction for a chromosome of length L) and $f(\theta; L) = \{2L + \log(1-2\theta)\}/\{L^2(1-2\theta)\}$.

We need to calculate that integral (and then maximize l with respect to L, to get the MLE $\hat{L}$).

Major problem: The integral can be *really* small, hence we have a problem with *underflow*.

Example 1 (continued)

The trick:

Let $g_i = \log f_i$. Suppose we wish to calculate $\log(f_1 + f_2)$.

$$\begin{aligned} \text{Then } \log(f_1 + f_2) &= \log(e^{g_1} + e^{g_2}) \\ &= \log\{e^{g_1}(1 + e^{g_2 - g_1})\} \\ &= g_1 + \log(1 + e^{g_2 - g_1}) \end{aligned}$$

"Obviously," a problem occurs when $g_2 \gg g_1$, in which case $\log(f_1 + f_2) \approx g_2$, but the above formula will result in an overflow. Thus, I recommend the following:

```
addlog <-
function(a, b, thresh=200)
{
  if(b > a + thresh) return(b)
  else if(a > b + thresh) return(a)
  else return(a + log(1+exp(b-a)))
}
```

Note that one may also want a function `subtractlog()`.

We can modify any code to do numerical integration, to calculate $\log\{\int_a^b f(x)\,dx\}$ working only with $\log f(x)$. This will avoid the underflow problem.

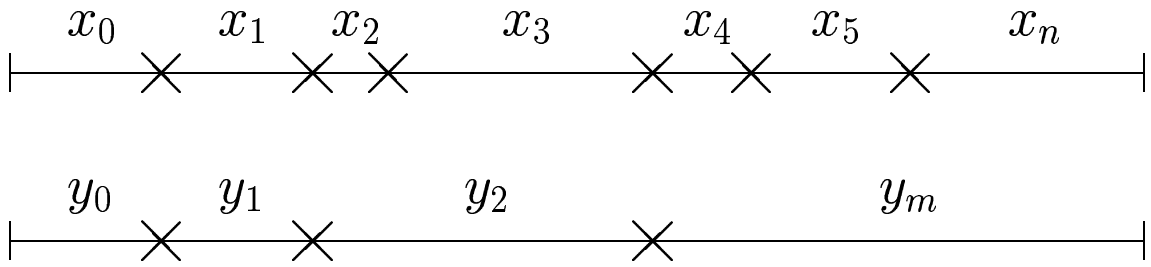
To calculate $S = \log \sum_{i=1}^n f_i$, use
$S_1 = g_1$, $S_{i+1} = \texttt{addlog}(S_i, g_{i+1})$

Example 2

[Broman and Weber, Am J Hum Genet 66:1911–1926, 2000]

Suppose we observe vectors $y_1, y_2, \ldots, y_N$, the realizations of a *thinned* stationary gamma renewal process on the interval $[0, L]$ (where L is known), with shape parameter ν. We seek to estimate ν.

Let's focus on a single realization $y = (y_1, \ldots, y_m)$.



Our model is that the points for the X process come from a stationary gamma renewal model with shape parameter ν and mean inter-point distance $1/2$, so that

$$x_1, x_2, \ldots \sim \text{ iid gamma}(\nu, 2\nu)$$

with density $f(x; \nu) = e^{-2\nu x}(2\nu)^\nu x^{\nu-1}/\Gamma(\nu)$.

The distribution of x_0 is that which is required to give stationarity: $g(x; \nu) = 2[1 - F(x; \nu)]$ where F is the cdf of f. The last point x_n is treated as censored.

Example 2 (continued)

The Y process (what we observe) is obtained by thinning the X process: we flip a coin at each point in the X process, independently; H $\rightarrow$ retain point; T $\rightarrow$ drop point.

The Y process is also a renewal process.

$$y_1, y_2, \ldots \sim \text{ iid } f^\star(x;\nu) = \sum_{k\geq 1}(1/2)^2 f_k(y;\nu)$$

where f_k is the density of a gamma distribution with shape and rate parameters $k\nu$ and 2ν—the convolution of $f(y;\nu)$ with itself k times.

The distribution of y_0 is

$$\begin{aligned} g^\star(y;\nu) &= (1/2)g(y;\nu) + \sum_{k\geq 1}(1/2)^{k+1}(g * f_k)(y;\nu) \\ &= 1 - F^\star(y;\nu) \end{aligned}$$

where $F^\star$ is the cdf of $f^\star$. Let $G^\star$ denote the cdf of $g^\star$.

Then y contributes the following to the log likelihood for ν:

$$l(\nu;y) = \begin{cases} \log[1 - G^\star(L;\nu)] & \text{if } m = 0 \\ \log g^\star(y_0;\nu) + \log g^\star(y_1;\nu) & \text{if } m = 1 \\ \log g^\star(y_0;\nu) + \sum_{j=1}^{m-1} \log f^\star(y_j;\nu) & \\ \qquad + \log g^\star(y_m;\nu) & \text{otherwise} \end{cases}$$

Example 2 (continued)

We seek to maximize $\sum_i l(\nu; y_i)$ where y_i is a vector like that we played with above.

To calculate $l(\nu; y_i)$, we need to be able to calculate

$$\begin{aligned} f^\star(y;\nu) &= \sum_{k=1}^{\infty} (1/2)^k f_k(y;\nu) \\ g^\star(y;\nu) &= 1 - F^\star(y;\nu) = \int_y^\infty f^\star(t;\nu)\, dt \\ 1 - G^\star(y;\nu) &= \int_y^\infty [1 - F^\star(t;\nu)]\, dt \end{aligned}$$

$f^\star(y;\nu)$ was calculated using ("by use of") its explicit formula, summing over k from 1 to 25.

$g^\star(y;\nu)$ was calculated by numerical integration of $f^\star$. (I used Simpson's rule, replacing the upper limit of the integral with some large value. Clearly I could have done better!)

$G^\star(y;\nu)$ was calculated by a nested numerical integral. I'm sure I could have done better!

To get the MLE $\hat{\nu}$, we are in the enviable situation of optimizing a function of one variable, $\sum_i l(\nu; y_i)$. I used "Brent's method" (pulled from *Numerical Recipes in C*).