Numerical Integration with an application to Sample size re-estimation

Adrian Mander and Jack Bowden

MRC Biostatistics Unit Hub for Trials Methodology Research

September 2012

Outline

- Give a brief introduction to quadrature
- Describe the Stata command and MATA function
    - how to use these for simple integrals
- Numerical difficulties
- Apply it to a harder problem of sample size re-estimation

Outline

- Give a brief introduction to quadrature
- Describe the Stata command and MATA function
    - how to use these for simple integrals
- Numerical difficulties
- Apply it to a harder problem of sample size re-estimation

Outline

- Give a brief introduction to quadrature
- Describe the Stata command and MATA function
    - how to use these for simple integrals
- Numerical difficulties
- Apply it to a harder problem of sample size re-estimation

Quadrature

Quadrature is another name for numerical integration, which is essentially transforming integration into a summation

$$\int_a^b W(x) f(x)\, \mathrm{dx} \approx \sum_{j=0}^{N-1} w_j f(x_j),$$

where w_j are weights and x_j are the abscissas.

- Functions $W(x)$ are chosen for the appropriate interval $[a, b]$
- the corresponding w_j and x_j values are found using orthogonal polynomials (defined by recurrence functions)

Quadrature

Quadrature is another name for numerical integration, which is essentially transforming integration into a summation

$$\int_a^b W(x) f(x)\ \mathrm{dx} \approx \sum_{j=0}^{N-1} w_j f(x_j),$$

where w_j are weights and x_j are the abscissas.

- Functions $W(x)$ are chosen for the appropriate interval $[a, b]$
- the corresponding w_j and x_j values are found using orthogonal polynomials (defined by recurrence functions)

Common forms of the weight function

Only considered three $W(x)$ functions over three ranges

1. [-1,1] — Gauss-Legendre quadrature, $W(x) = 1$
2. $[0,\infty]$ — Gauss-Lageurre quadrature, $W(x) = \exp(-x)$
3. $[-\infty,\infty]$ — Gauss-Hermite Quadrature , $W(x) = \exp(-x^2)$

All of these methods have been implemented in a Stata command `integrate` available on SSC.

Most of the calculation are written in MATA and uses the trick from Bill Gould to pass functions from Stata to Mata

Common forms of the weight function

Only considered three $W(x)$ functions over three ranges

1. [-1,1] — Gauss-Legendre quadrature, $W(x) = 1$
2. $[0,\infty]$ — Gauss-Lageurre quadrature, $W(x) = \exp(-x)$
3. $[-\infty,\infty]$ — Gauss-Hermite Quadrature , $W(x) = \exp(-x^2)$

All of these methods have been implemented in a Stata command `integrate` available on SSC.

Most of the calculation are written in MATA and uses the trick from Bill Gould to pass functions from Stata to Mata

How to find the weights/abscissas

The roots of the Legendre polynomial defined by

$$
\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
(n+1)P_{n+1}(x) &= (2n+1)xP_n(x) - nP_{n-1}(x)
\end{aligned}
$$

are the abscissas.

- Finding the roots say using `polyroots()` has limited precision of the machine.
- Golub and Welch solution was to construct a similarity matrix

How to find the weights/abscissas

The roots of the Legendre polynomial defined by

$$
\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
(n+1)P_{n+1}(x) &= (2n+1)xP_n(x) - nP_{n-1}(x)
\end{aligned}
$$

are the abscissas.

- Finding the roots say using `polyroots()` has limited precision of the machine.
- Golub and Welch solution was to construct a similarity matrix

Similarity Matrix corresponding to Legendre polynomial

$$\begin{pmatrix} 0 & \frac{1}{\sqrt{4*1^2-1}} & & & \\ \frac{1}{\sqrt{4*1^2-1}} & 0 & \frac{2}{\sqrt{4*2^2-1}} & & \\ & \frac{2}{\sqrt{4*2^2-1}} & \ddots & \ddots & \\ & & \ddots & & \\ & & & 0 & \frac{n-1}{\sqrt{4*(n-1)^2-1}} \\ & & & \frac{n-1}{\sqrt{4*(n-1)^2-1}} & 0 \end{pmatrix}$$

The eigenvalues are the abscissas and the eigenvectors are used to find the weights.

Hermite polynomial with $n > 60$ gives the wrong answers using eigensystem() function.

Similarity Matrix corresponding to Legendre polynomial

$$\begin{pmatrix} 0 & \frac{1}{\sqrt{4*1^2-1}} & & & \\ \frac{1}{\sqrt{4*1^2-1}} & 0 & \frac{2}{\sqrt{4*2^2-1}} & & \\ & \frac{2}{\sqrt{4*2^2-1}} & \ddots & \ddots & \\ & & \ddots & & \\ & & & 0 & \frac{n-1}{\sqrt{4*(n-1)^2-1}} \\ & & & \frac{n-1}{\sqrt{4*(n-1)^2-1}} & 0 \end{pmatrix}$$

The eigenvalues are the abscissas and the eigenvectors are used to find the weights.

Hermite polynomial with $n > 60$ gives the wrong answers using eigensystem() function.

Basic syntax

To calculate the following expression

$$\int_a^b f(x)\ \mathrm{dx}$$

In Stata

```
integrate, function( f(x) ) lower(a) upper(b)
```

In Mata if the function `f()` already exists then the function address is passed to integrate

```
integrate(&f(), a, b)
```

- $-\infty$ is specified by setting $a = .$
- similarly, if $b = .$ then the upper limit is ∞

Basic syntax

To calculate the following expression

$$\int_a^b f(x)\,\mathrm{dx}$$

In Stata

```
integrate, function( f(x) ) lower(a) upper(b)
```

In Mata if the function `f()` already exists then the function address is passed to integrate

```
integrate(&f(), a, b)
```

- $-\infty$ is specified by setting $a = .$
- similarly, if $b = .$ then the upper limit is ∞

Simple example - Stata

$$\int_0^3 x^2 \mathrm{d}x \tag{1}$$

Using the Stata command

```
integrate, f(x:^2) l(0) u(3)

 Note: The function to be integrated will be compiled using Mata and stored in your personal
 directory ~/ado/personal/ (make sure this is writeable)

The integral = 9
```

Could have done

```
integrate, f(x^2) l(0) u(3) vectorise
```

Simple example - Stata

$$\int_0^3 x^2 \mathrm{dx} \tag{1}$$

Using the Stata command

```
integrate, f(x:^2) l(0) u(3)

 Note: The function to be integrated will be compiled using Mata and stored in your personal
 directory ~/ado/personal/ (make sure this is writeable)

The integral = 9
```

Could have done

```
integrate, f(x^2) l(0) u(3) vectorise
```

Simple example - Mata

First define the integrand as a new function, the function must return a row vector and the variable of integration must be a rowvector.

```
real rowvector f(real rowvector x)
{
  return(x:^2)
}
```

Then to integrate this function type with Mata

```
: integrate(&f(), 0, 3)
  9
```

All the examples from now on will be based only on the Mata function. Which is available via SSC, `integrate.mata` contains a do file to compile the mata code

Mata syntax

The syntax of the Mata function

```
real scalar integrate(&function(), real scalar lower,
  real scalar upper |, real scalar quadpts,
  real rowvector xarg)
```

has optional arguments for number of quadrature points and a rowvector of additional arguments that are passed to the function()

- Note that integrate returns a **real scalar**

Mata syntax

The syntax of the Mata function

```
real scalar integrate(&function(), real scalar lower,
  real scalar upper |, real scalar quadpts,
  real rowvector xarg)
```

has optional arguments for number of quadrature points and a rowvector of additional arguments that are passed to the function()

- Note that integrate returns a **real scalar**

Double Integration

$$\int_0^1 \int_0^1 x + y \ \mathrm{dx}\ \mathrm{dy}$$

Want to just write

```
integrate( integrate(&f(),0,1) ,0,1)
```

- However `integrate()` does not return a rowvector so this syntax would **fail**

Double Integration

$$\int_0^1 \int_0^1 x + y \text{ dx dy}$$

Want to just write

```
integrate( integrate(&f(),0,1) ,0,1)
```

- However `integrate()` does not return a rowvector so this syntax would **fail**

Solution

First define

```
real rowvector fxy(real rowvector x, real rowvector y)
{
  return(x:+y)
}

real rowvector f_inner(real rowvector y)
{
  for(i=1; i<=cols(y);i++) {
    if (i==1) f=integrate(&fxy(), 0, 1, 40, y[i])
    else f = f, integrate(&fxy(), 0, 1, 40, y[i])
  }
  return(f)
}

: integrate(&f_inner(), 0, 1)
  1
```

Solution

First define

```
real rowvector fxy(real rowvector x, real rowvector y)
{
  return(x:+y)
}

real rowvector f_inner(real rowvector y)
{
  for(i=1; i<=cols(y);i++) {
    if (i==1) f=integrate(&fxy(), 0, 1, 40, y[i])
    else f = f, integrate(&fxy(), 0, 1, 40, y[i])
  }
  return(f)
}
```

```
: integrate(&f_inner(), 0, 1)
  1
```

Solution

First define

```
real rowvector fxy(real rowvector x, real rowvector y)
{
  return(x:+y)
}

real rowvector f_inner(real rowvector y)
{
  for(i=1; i<=cols(y);i++) {
    if (i==1) f=integrate(&fxy(), 0, 1, 40, y[i])
    else f = f, integrate(&fxy(), 0, 1, 40, y[i])
  }
  return(f)
}

: integrate(&f_inner(), 0, 1)
  1
```

Further Double Integration

$$\int_0^2 \int_0^{y^2} 6xy \text{ dx dy}$$

This is also a simple extension to the previous code

Solution

```
real rowvector fxy2(real rowvector x, real rowvector y)
{
  return(6:*x:*y)
}

real rowvector f_inner2(real rowvector y)
{
  for(i=1; i<=cols(y);i++) {
    if (i==1) f=integrate(&fxy2(), 0, y[i]^2, 40, y[i])
    else f = f, integrate(&fxy2(), 0, y[i]^2, 40, y[i])
  }
  return(f)
}

: integrate(&f_inner2(), 0, 2)
  32
```

Solution

```
real rowvector fxy2(real rowvector x, real rowvector y)
{
  return(6:*x:*y)
}

real rowvector f_inner2(real rowvector y)
{
  for(i=1; i<=cols(y);i++) {
    if (i==1) f=integrate(&fxy2(), 0, y[i]^2, 40, y[i])
    else f = f, integrate(&fxy2(), 0, y[i]^2, 40, y[i])
  }
  return(f)
}

: integrate(&f_inner2(), 0, 2)
  32
```

Sample size re-estimation

Usually when designing a clinical trial we pre-specify the value of a treatment effect (and all the nuisance parameters) to find the sample size.

- We plan to do a single interim analysis to re-evaluate this sample size
- Going to apply the methods to a real trial example

Trial details

- Currently limited treatment options for Osteoarthritis (OA) of the knee. Not suitable or ineffective for many people. Surgery often only remaining option
- Methotrexate used effectively for Rheumatoid arthritis but not OA
- Promising results from pilot study (n=30) showed significant pain reduction for methotrexate in OA
- Study team proposed to test the drug's performance in addition to standard care in a double blind, randomized, placebo controlled trial

The problem

- Initial grant application received positive feedback from funder
- Unfortunately it was rejected due to lack of evidence about the effect size likely to be seen in the RCT

Potential solution

Wanted to use a method that:

1. can be fully specified in advance of the trial;
2. can be implemented by an independent non-expert data monitoring committee;
3. is **not motivated** via a complex conditional error function; and
4. is motivated by clear decision framework linking interim effect size with future sample size via a simple and familiar formula

Potential solution

Wanted to use a method that:

1. can be fully specified in advance of the trial;
2. can be implemented by an independent non-expert data monitoring committee;
3. is **not motivated** via a complex conditional error function; and
4. is motivated by clear decision framework linking interim effect size with future sample size via a simple and familiar formula

Potential solution

Wanted to use a method that:

1. can be fully specified in advance of the trial;
2. can be implemented by an independent non-expert data monitoring committee;
3. is **not motivated** via a complex conditional error function; and
4. is motivated by clear decision framework linking interim effect size with future sample size via a simple and familiar formula

Notation

- Assume observations in experimental treatment group X and standard therapy group Y are normally distributed with means μ_x and μ_y and have a common **known variance** of σ^2
- **Parameter** of interest is $\delta = \frac{\mu_x - \mu_y}{\sigma}$. $H_0 : \delta \leq 0$
- Fixed design: n patients per arm
- Choose $n = \frac{2}{\delta^2}(Z_\alpha + Z_\beta)^2$, where $Z_u = \Phi^{-1}(1-u)$
- e.g. if $\delta = 0.35$, $\alpha = 0.025$ and $\beta = 0.2$ then $n = 128$ patients per arm

Estimation and inference for δ

- $\bar{x} \sim N(\mu_x, \sigma^2/n)$, $\bar{y} \sim N(\mu_y, \sigma^2/n)$ and $\hat{\delta} = \frac{\bar{x}-\bar{y}}{\sigma}$
- $z = \frac{\hat{\delta}}{\sqrt{2/n}} \sim N\left(\frac{\delta}{\sqrt{2/n}}, 1\right)$

Notation

- Assume observations in experimental treatment group X and standard therapy group Y are normally distributed with means μ_x and μ_y and have a common **known variance** of σ^2
- **Parameter** of interest is $\delta = \frac{\mu_x - \mu_y}{\sigma}$. $H_0 : \delta \leq 0$
- Fixed design: n patients per arm
- Choose $n = \frac{2}{\delta^2}(Z_\alpha + Z_\beta)^2$, where $Z_u = \Phi^{-1}(1-u)$
- e.g. if $\delta = 0.35$, $\alpha = 0.025$ and $\beta = 0.2$ then $n = 128$ patients per arm

Estimation and inference for δ

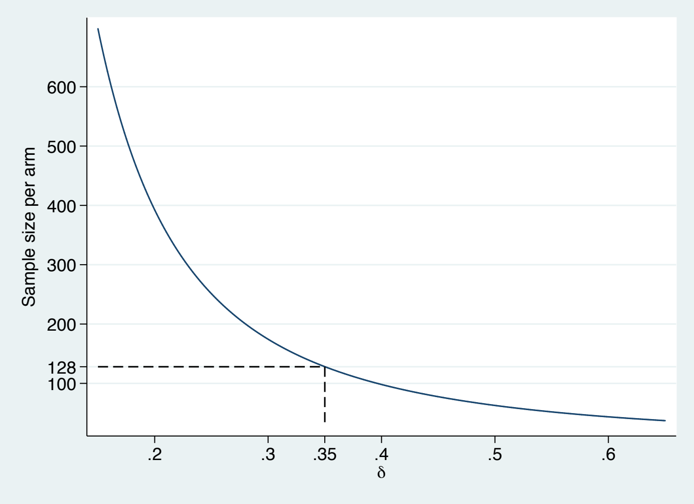
- $\bar{x} \sim N(\mu_x, \sigma^2/n)$, $\bar{y} \sim N(\mu_y, \sigma^2/n)$ and $\hat{\delta} = \frac{\bar{x} - \bar{y}}{\sigma}$
- $z = \frac{\hat{\delta}}{\sqrt{2/n}} \sim N\left(\frac{\delta}{\sqrt{2/n}}, 1\right)$

Notation

- Assume observations in experimental treatment group X and standard therapy group Y are normally distributed with means μ_x and μ_y and have a common **known variance** of σ^2
- **Parameter** of interest is $\delta = \frac{\mu_x - \mu_y}{\sigma}$. $H_0 : \delta \leq 0$
- Fixed design: n patients per arm
- Choose $n = \frac{2}{\delta^2}(Z_\alpha + Z_\beta)^2$, where $Z_u = \Phi^{-1}(1-u)$
- e.g. if $\delta = 0.35$, $\alpha = 0.025$ and $\beta = 0.2$ then $n = 128$ patients per arm

Estimation and inference for δ

- $\bar{x} \sim N(\mu_x, \sigma^2/n)$, $\bar{y} \sim N(\mu_y, \sigma^2/n)$ and $\hat{\delta} = \frac{\bar{x}-\bar{y}}{\sigma}$
- $z = \frac{\hat{\delta}}{\sqrt{2/n}} \sim N\left(\frac{\delta}{\sqrt{2/n}}, 1\right)$



- if $\delta << 0.35$ then substantially more than 128 people needed
- if $\delta >> 0.35$ then trial is a waste of resources

A general two stage strategy

- Suppose instead n_1 ($\ll n$) patients initially recruited giving: $\hat{\delta}_1 = \frac{\bar{x}-\bar{y}}{\sigma}$ and $z_1 = \frac{\hat{\delta}_1}{\sqrt{2/n_1}} \sim N\left(\frac{\hat{\delta}}{\sqrt{2/n_1}}, 1\right)$ at the **interim analysis**. Then if:

$$\begin{cases} z_1 > k & : \text{Stop the trial for efficacy} \\ z_1 < h & : \text{Stop the trial for futility} \\ h \le z_1 \le k & : \text{Recruit further } n_2 \text{ patients } (z_1 \uparrow \;\Rightarrow n_2 \downarrow) \end{cases}$$

Base inference at stage 2 on combined data via test statistic:

$$z = \frac{\sqrt{n_1}z_1 + \sqrt{n_2(z_1)}z_2}{\sqrt{n_1 + n_2(z_1)}} \text{ Reject } H_0 \text{ if } z \ge C$$

How to choose design parameters h, k, C and function $n_2(z_1)$?

A general two stage strategy

- Suppose instead n_1 ($\ll n$) patients initially recruited giving: $\hat{\delta}_1 = \frac{\bar{x}-\bar{y}}{\sigma}$ and $z_1 = \frac{\hat{\delta}_1}{\sqrt{2/n_1}} \sim N\left(\frac{\hat{\delta}}{\sqrt{2/n_1}}, 1\right)$ at the **interim analysis**. Then if:

$$\begin{cases} z_1 > k & : \text{Stop the trial for efficacy} \\ z_1 < h & : \text{Stop the trial for futility} \\ h \leq z_1 \leq k & : \text{Recruit further } n_2 \text{ patients } (z_1 \uparrow \;\Rightarrow n_2 \downarrow) \end{cases}$$

Base inference at stage 2 on combined data via test statistic:

$$z = \frac{\sqrt{n_1} z_1 + \sqrt{n_2(z_1)} z_2}{\sqrt{n_1 + n_2(z_1)}} \text{ Reject } H_0 \text{ if } z \geq C$$

How to choose design parameters h, k, C and function $n_2(z_1)$?

Chosing h,k,C via the Li et al. method

- Choose an overall type I error α and conditional power $1-\beta_1$
- Choose h and k almost freely (e.g based on p-value for z_1)
  - There are restrictions based on the **error probabilities**
- Find C such that:
  1. $P(z_1 > k|\delta = 0) + P(z > C|\delta = 0; h < z_1 < k) = \alpha$
  2. $P(z > C|\delta = \hat{\delta}_1, h < z_1 < k) \geq 1 - \beta_1$

Given $n_2(z_1) = \left(\frac{(C+Z_{\beta_1})^2}{z_1^2} - 1\right) n_1$, for $z_1 \in (h, k)$

- A very simple method
- No complex conditional error function (Proschan and Hunsberger, 1995)
- Critical value C **independent** of z_1
  - Whole design and analysis can be specified in advance

Finding C

From Li et al. (2002) they state that one can use numerical integration to solve

$$1 - \Phi(h) - \alpha = \int_h^k \Phi\left[\frac{C(C + Z_{\beta_1}) - z_1^2}{\sqrt{(C + Z_{\beta_1})^2 - z_1^2}}\right] \phi(z_1) \mathrm{d}z_1$$

this is solved for c (the other design parameters are selected previously)

Need to use optimize() and integrate() together!!

Programming up finding C

```
real rowvector findC(real rowvector x, real rowvector arg)
{
  c=arg[1]
  Zb = arg[2]
  return( normal((c:*(c:+Zb):-x:^2):/sqrt((c:+Zb):^2:-x:^2)):*normalden(x) )
}
void evalC(todo, c, h, k, alpha, Zb, y, g, H)
{
  y=(integrate(&findC(),h,k,60,(c, Zb))-(1-normal(h)-alpha))^2
}

void calculateC(h, k, alpha, power)
{
 Zb=invnormal(power)
 C = optimize_init()
 optimize_init_which(C, "min")
 optimize_init_evaluator(C, &evalC())
 optimize_init_tracelevel(C, "none")
 optimize_init_params(C, 1)
 optimize_init_argument(C,1,h)
 optimize_init_argument(C,2,k)
 optimize_init_argument(C,3,alpha)
 optimize_init_argument(C,4,Zb)
 c = optimize(C)
}
```

Programming up finding C

```
real rowvector findC(real rowvector x, real rowvector arg)
{
  c=arg[1]
  Zb = arg[2]
  return( normal((c:*(c:+Zb):-x:^2):/sqrt((c:+Zb):^2:-x:^2)):*normalden(x) )
}
void evalC(todo, c, h, k, alpha, Zb, y, g, H)
{
  y=(integrate(&findC(),h,k,60,(c, Zb))-(1-normal(h)-alpha))^2
}


void calculateC(h, k, alpha, power)
{
 Zb=invnormal(power)
 C = optimize_init()
 optimize_init_which(C, "min")
 optimize_init_evaluator(C, &evalC())
 optimize_init_tracelevel(C, "none")
 optimize_init_params(C, 1)
 optimize_init_argument(C,1,h)
 optimize_init_argument(C,2,k)
 optimize_init_argument(C,3,alpha)
 optimize_init_argument(C,4,Zb)
 c = optimize(C)
}
```

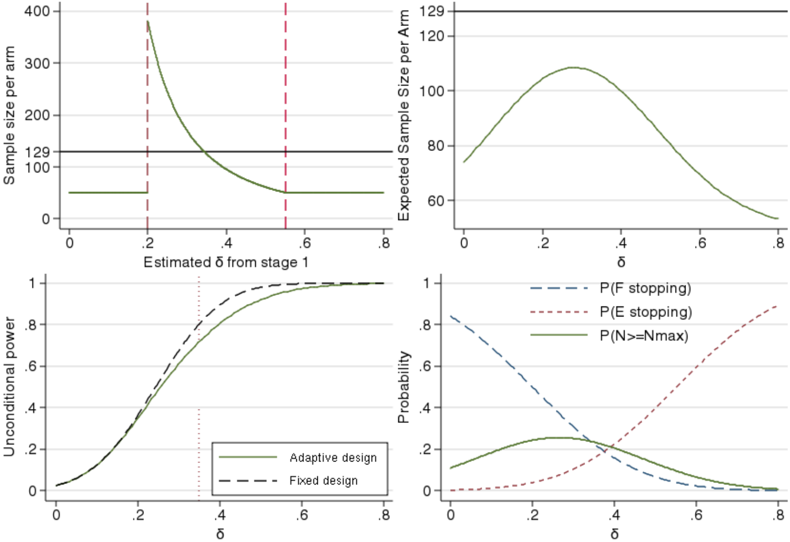
Stata code for Sample size re-estimation

```
. ssr
Sample Size Re-estimation
-------------------------
The following are set in the first stage
 The sample size per arm is  50
 The futility bound is 1
 The efficacy bound is 2.76
 The conditional power is .8
 The unconditional power is .8

The Li et al. critical value is 1.923

+------------------------------------------------+
|NOTE                                            |
| A fixed sample size requires   129  people     |
| for a treatment effect of .35,                 |
| unconditional power .8 and                     |
| one-sided significance of .025                 |
+------------------------------------------------+
```

ssr,graph



129 is the fixed design sample size

Conclusions

- **integrate** is a flexible function
  - Still need to get a better Gauss-Hermite solution
- **ssr**, the Stata command, is available to design sample size re-estimation
  - there are several methods that are available in a future publication Bowden and Mander