Efficient multivariate normal distribution calculations in Stata

Michael Grayling

Supervisor: Adrian Mander

MRC Biostatistics Unit

2015 UK Stata Users Group Meeting
10/09/15

Why and what?

Why do we need to be able to work with the Multivariate Normal Distribution?

- The normal distribution has significant importance in statistics.
- Much real world data either is, or is assumed to be, normally distributed.
- Whilst the central limit theorem tells us the mean of many random variables drawn independently from the same distribution will be approximately normally distributed.
- Today however a considerable amount of statistical analysis performed is not univariate, but multivariate in nature.
- Consequently the generalisation of the normal distribution to higher dimensions; the multivariate normal distribution, is of increasing importance.

Why and what?

Definition

- Consider a m-dimensional random variable X. If X has a (non-degenerate) MVN distribution with location parameter (mean vector) $\boldsymbol{\mu} \in \mathbb{R}^m$ and positive definite covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$, denoted $X \sim N_m(\boldsymbol{\mu}, \Sigma)$, then its distribution has density $f_X(\mathbf{x})$ for $\mathbf{x} = (x_1, \ldots, x_m) \in \mathbb{R}^m$ given by:

$$f_X(\mathbf{x}) = \phi_m(\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^m}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathsf{T}} \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right] \in \mathbb{R},$$

where $|\Sigma| = \det(\Sigma)$.

- In this instance we have:

$$\mathbb{E}(X) = \boldsymbol{\mu},$$
$$\mathrm{Var}(X) = \Sigma,$$
$$\mathbb{P}(a_i \leq x_i \leq b_i : i = 1, \ldots, m) = P(\mathbf{a}, \mathbf{b}, \boldsymbol{\mu}, \Sigma) = \int_{a_1}^{b_1} \ldots \int_{a_m}^{b_m} \phi_m(\boldsymbol{\mu}, \Sigma) d\mathbf{x}.$$

The multivariate normal distribution in Stata

What's available?

- `drawnorm` allows random samples to be drawn from the multivariate normal distribution.
- `binormal` allows the computation of cumulative bivariate normal probabilities.
- `mvnp` allows the computation of cumulative multivariate normal probabilities through simulation using the GHK simulator.

```
. set obs 1000
. matrix R = (1, .25 \ .25, 1)
. drawnorm v1 v2, corr(R) seed(13131313)
. matrix C = cholesky(R)
. ge x_b = binormal(v1,v2,.25)
. mdraws, neq(2) dr(500) prefix(p)
. egen x_s = mvnp(v1 v2), dr(500) chol(C) prefix(p) adoonly
. su x_b x_s

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         x_b |      1000    .2911515     .238888   6.76e-06   .9953722
         x_s |      1000    .2911539    .2388902   6.76e-06   .9953699
```

The new commands

- Utilise Mata and one of the new efficient algorithms that has been developed to quickly compute probabilities over any range of integration.
- Additionally, there's currently no easy means to compute equi-coordinate quantiles which have a range of applications:

$$p = \int_{-\infty}^{q} \dots \int_{-\infty}^{q} \phi_m(\boldsymbol{\mu}, \Sigma) d\boldsymbol{\theta},$$

  so use interval bisection to search for q, employing the former algorithm for probabilities to evaluate the RHS.
- Final commands named `mvnormalden`, `mvnormal`, `invmvnormal` and `rmvnormal`, with all four using Mata.
- `mvnormal` in particular makes use of a recently developed Quasi-Monte Carlo Randomised Lattice algorithm for performing the required integration.
- All four are easy to use with little user input required.

Talk outline

- Discuss the transformations and algorithm that allows the distribution function to be worked with efficiently.
- Detail how this code can then be used to compute equi-coordinate quantiles.
- Compare the performance of `mvnormal` to `mvnp`.
- Demonstrate how `mvnormal` can be used to determine the operating characteristics of a group sequential clinical trial.

Working with the distribution function

Transforming the integral

- First we use a Cholesky decomposition transformation: $\boldsymbol{\theta} = C\mathbf{y}$, where $CC^{\top} = \Sigma$:

$$P(\mathbf{a}, \mathbf{b}, \mathbf{0}, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^m}} \int_{a_1}^{b_1} \dots \int_{a_m}^{b_m} e^{-\frac{1}{2}\boldsymbol{\theta}^{\top}\Sigma^{-1}\boldsymbol{\theta}} d\boldsymbol{\theta},$$
$$= \frac{1}{\sqrt{(2\pi)^m}} \int_{a_1'}^{b_1'} e^{-y_1^2/2} \int_{a_m'}^{b_m'} e^{-y_m^2/2} d\mathbf{y}.$$

- Next transform each of the y_i's separately using $y_i = \Phi^{-1}(z_i)$:

$$P(\mathbf{a}, \mathbf{b}, \mathbf{0}, \Sigma) = \int_{d_1}^{e_1} \dots \int_{d_m(z_1,\dots,z_{m-1})}^{e_m(z_1,\dots,z_{m-1})} d\mathbf{z}.$$

- Turn the problem in to a constant limit form using $z_i = d_i + w_i(e_i - d_i)$:

$$P(\mathbf{a}, \mathbf{b}, \mathbf{0}, \Sigma) = (e_1 - d_1) \int_0^1 (e_2 - d_2) \dots \int_0^1 (e_m - d_m) \int_0^1 d\mathbf{w}.$$

Quasi Monte Carlo Randomised Lattice Algorithm

- Specify a number of shifts of the Monte Carlo algorithm M, a number of samples for each shift N, and a Monte Carlo confidence factor α. Set $I = V = 0$, $\mathbf{d} = (d_1, \dots, d_m) = \mathbf{e} = (e_1, \dots, e_m) = (0, \dots, 0)$ and $\mathbf{y} = (y_1, \dots, y_{m-1}) = (0, \dots, 0)$. Compute the Cholesky factor $C = \{c_{ij}\}$.
- For $i = 1, \dots, M$:
  - Set $I_i = 0$ and generate uniform random $\mathbf{\Delta} = (\Delta_1, \dots, \Delta_{m-1}) \in [0,1]^{m-1}$.
  - For $j = 1, \dots, N$:
    - Set $\mathbf{w} = \left|2 \times \mathrm{mod}(j\sqrt{\mathbf{p}} + \mathbf{\Delta}, 1) - 1\right|$, where $\mathbf{p}$ is a vector of the first $m-1$ prime numbers.
    - Set $d_1 = \Phi(a_1/c_{11})$, $e_1 = \Phi(b_1/c_{11})$ and $f_1 = e_1 - d_1$.
    - For $k = 2, \dots, m$:
      - Set $y_{k-1} = \Phi^{-1}\left(d_{k-1} + w_{k-1}(e_{k-1} - d_{k-1})\right)$, $d_k = \Phi\left(\left(a_i - \sum_{j=1}^{i-1} c_{ij}y_j\right)/c_{ii}\right)$, $e_k = \Phi\left(\left(b_i -\right.\right.$

Computing equi-coordinate quantiles

- Recall the definition of an equi-coordinate quantile:

$$p = f(q) = \int_{-\infty}^{q} \dots \int_{-\infty}^{q} \Phi_m(\boldsymbol{\mu}, \Sigma) d\boldsymbol{\theta}.$$

- We can compute q for any p efficiently using the algorithm discussed previously to evaluate the RHS for any q, and modified interval bisection to search for the correct q.

- Optimize does not work well because of the small errors present when you evaluate the RHS.

- Choose a maximum number of interactions $i_{\max}$, and a tolerance ϵ.
- Initialise $a = -10^6$, $b = 10^6$ and $i = 1$. Compute $f(a)$ and $f(b)$.
- While $i \leq i_{\max}$:
  - Set $c = a - \left[(b-a)/\big(f(b) - f(a)\big)\right] f(a)$ and compute $f(c)$.
  - If $f(c) = 0$ or $(b-a)/2 < \epsilon$ break. Else:
    - If $f(a), f(c) < 0$ or $f(a), f(c) > 0$ set $a = c$ and $f(a) = f(c)$. Else set $b = c$ and $f(b) = f(c)$.
  - Set $i = i + 1$.
- Return $q = c$.

Syntax

mvnormal

$\mathbf{a}$	$\mathbf{b}$	$\boldsymbol{\mu}$	Σ	M

```
mvnormal, LOWer(numlist miss) UPPer(numlist miss) MEan(numlist) Sigma(string) [SHIfts(integer 12) ///
          SAMples(integer 1000) ALPha(real 3)]
```

N	α

```
invmvnormal, p(real) MEan(numlist) Sigma(string) [Tail(string) SHIfts(integer 12) SAMples(integer 1000) ///
             ALPha(real 3) Itermax(integer 1000000) TOLerance(real 0.000001)]
```

Syntax

invmvnormal

```
mvnormal, LOWer(numlist miss) UPPer(numlist miss) MEan(numlist) Sigma(string) [SHIfts(integer 12) ///
          SAMples(integer 1000) ALPha(real 3)]
```

p $\boldsymbol{\mu}$ Σ M N

```
invmvnormal, p(real) MEan(numlist) Sigma(string) [Tail(string) SHIfts(integer 12) SAMples(integer 1000) ///
             ALPha(real 3) Itermax(integer 1000000) TOLerance(real 0.000001)]
```

α $i_{\max}$ ϵ

Syntax

invmvnormal

```
mvnormal, LOWer(numlist miss) UPPer(numlist miss) MEan(numlist) Sigma(string) [SHIfts(integer 12) ///
          SAMples(integer 1000) ALPha(real 3)]
```

lower, upper, or both

p $\boldsymbol{\mu}$ Σ M N

```
invmvnormal, p(real) MEan(numlist) Sigma(string) [Tail(string) SHIfts(integer 12) SAMples(integer 1000) ///
             ALPha(real 3) Itermax(integer 1000000) TOLerance(real 0.000001)]
```

α $i_{\max}$ ϵ

Performance Comparison

Set-up

- Compare the average time required to compute a single particular integral, and the associated average absolute error by `mvnp` for different numbers of draws and across different dimensions, in comparison to `mvnormal`.
- Take the case $\Sigma_{ii} = 1, \Sigma_{ij} = 0.5$ for $i \neq j$, with $\mu_i = 0$ for all i.
- First determine the 95% both tailed quantile about $\mathbf{0}$ using `invmvnormal`, then assess how close the value returned by `mvnp` and `mvnormal` is to 0.95 on average, across 100 replicates.
- Do this for the 3, 5, 7 and 10 dimensional problems, with draws set to 5 (default), 10, 25, 50, 75, 100 and 200.
- Caveats:
  - This is the case when you desire the value to only one integral.
  - `mvnormal` will soon be changed to become more efficient through variable re-ordering methods and parallelisation.

Performance Comparison

Using `invmvnormal` and `mvnormal`

- First initialise the covariance matrix Sigma, then pass this and the other required characteristics to `invmvnormal`:

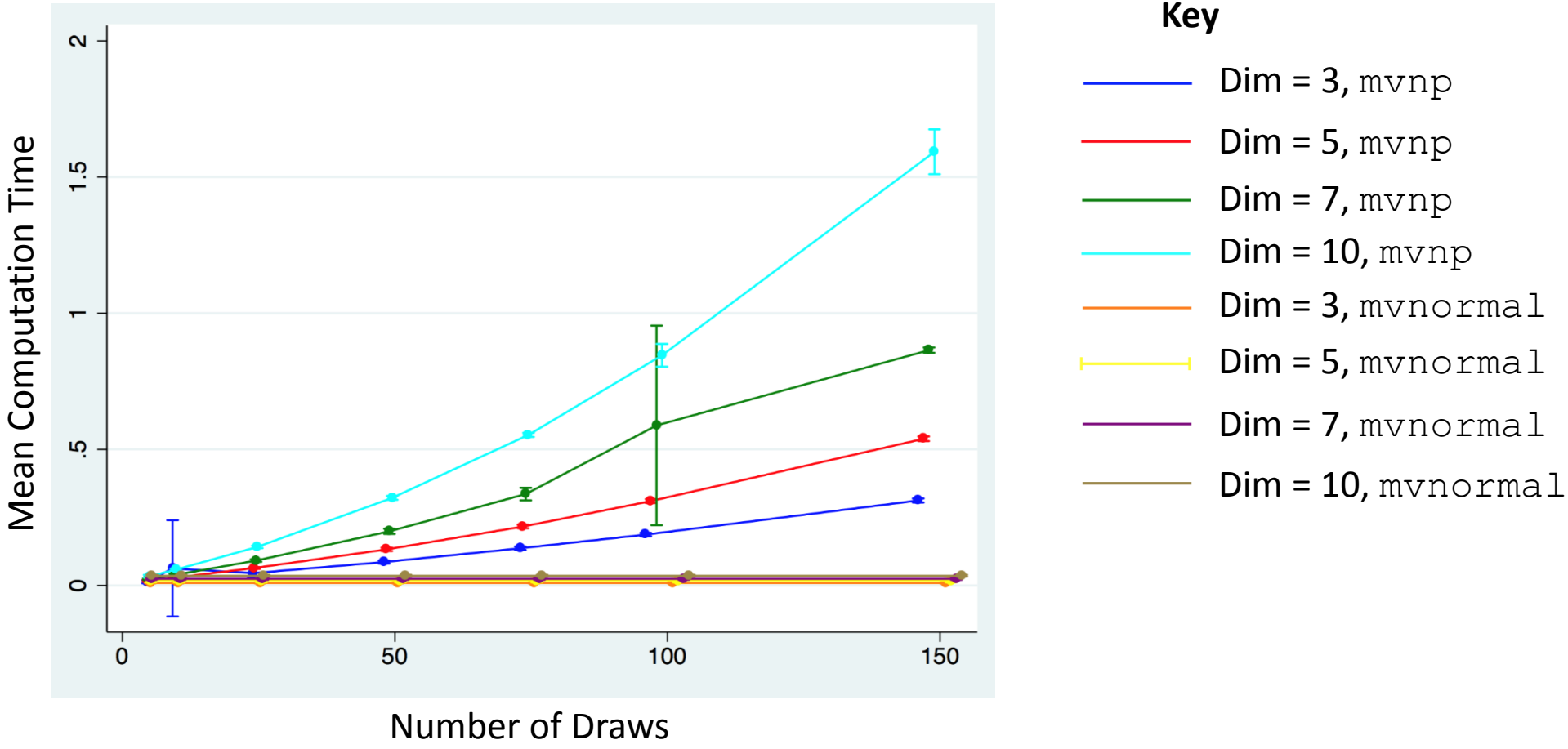
```
. mat Sigma = 0.5*I(3) + J(3, 3, 0.5)

. invmvnormal, p(0.95) mean(0, 0, 0) sigma(Sigma) tail(both)
Quantile = 2.3487841
Error = 1.257e-08
Flag = 0
fQuantile = 9.794e-06
Iterations = 185
```

- We can verify further the accuracy of this quantile value using `mvnormal`:

```
. mvnormal, lower(-2.3487841, -2.3487841, -2.3487841) upper(2.3487841, 2.3487841, 2.3487841) sigma(Sigma) mean(0, 0, 0)
Integral = .94999214
Error = .00006841
```

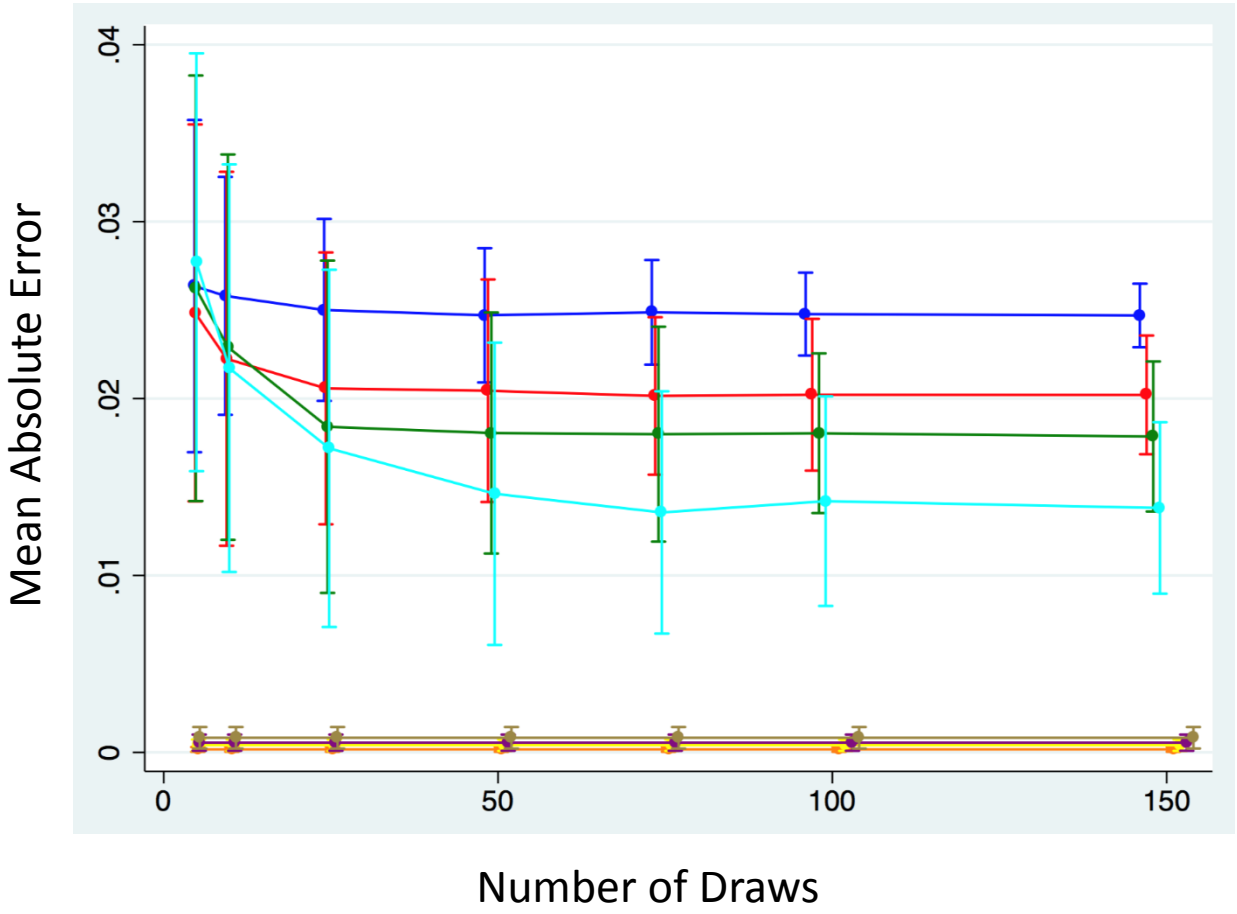
Performance Comparison

Mean Computation Time



Performance Comparison

Mean Absolute Error



Performance Comparison

Relative Performance

Dimension	Draws = 5		Draws = 50		Draws = 75		Draws = 150	
	Rel. Mean Error	Rel. Time Req.	Rel. Mean Error	Rel. Time Req.	Rel. Mean Error	Rel. Time Req.	Rel. Mean Error	Rel. Time Req.
3	156.5	1.55	147.0	9.53	147.6	15.11	147.0	34.52
5	62.0	0.94	51.0	7.18	50.3	11.71	50.5	29.15
7	47.8	0.97	32.8	8.18	32.7	13.81	32.5	35.53
10	33.5	0.95	17.6	8.97	16.3	15.40	16.7	44.37

Triangular Test

- Suppose we wish to design a group sequential clinical trial to compare the performance of two drugs, A and B, and ultimately to test the following hypotheses:

$$H_0 : \mu_B - \mu_A \leq 0, \qquad H_1 : \mu_B - \mu_A > 0.$$

- We plan to recruit n patients to each drug in each of a maximum of L stages, and desire a type-I error of α when $\mu_B - \mu_A = 0$ and a type-II error of β when $\mu_B - \mu_A = \delta$.

- We utilise the following standardised test statistics at each analysis:

$$Z_l = (\hat{\mu}_B - \hat{\mu}_A) I_l^{1/2},$$

and wish to determine early stopping efficacy and futility boundaries; e_l and f_l, $l = 1, \ldots, L$ in order to give the required operating characteristics.

- Additionally, information is linked to sample size by $n = 2\sigma^2 I_1$ where σ^2 is the variance of the patient responses on treatment A or B.

Triangular Test

- Whitehead and Stratton (1983) demonstrated this could be approximately achieved by taking:

$$f_l = I_l^{-1/2}\left[-\frac{2}{\tilde{\delta}}\log\left(\frac{1}{2\alpha}\right) + 0.583\left(\frac{I_L}{L}\right) + \frac{3\tilde{\delta}}{4}\frac{l}{L}I_L\right],$$
$$e_l = I_l^{-1/2}\left[\frac{2}{\tilde{\delta}}\log\left(\frac{1}{2\alpha}\right) - 0.583\left(\frac{I_L}{L}\right) + \frac{\tilde{\delta}}{4}\frac{l}{L}I_L\right],$$
$$\tilde{\delta} = \frac{2\Phi^{-1}(1-\alpha)}{\Phi^{-1}(1-\alpha)+\Phi^{-1}(1-\beta)}\delta.$$

- Desiring $f_L = e_L$ to ensure a decision is made at the final analysis, we have:

$$I_L = \left[\left(\frac{4\times 0.583^2}{L} + 8\log\left(\frac{1}{2\alpha}\right)\right)^{1/2} - \frac{2\times 0.583}{L^{1/2}}\right]\frac{1}{\tilde{\delta}^2}.$$

Computing the Designs Performance

- We can compute the expected sample size or power at any true treatment effect $\theta = \mu_B - \mu_A$ using multivariate integration and the following facts:

$$\mathbb{E}(Z_l) = \theta I_l^{1/2}, \qquad l = 1, .., L,$$
$$\mathrm{Cov}\left(Z_{l_1}, Z_{l_2}\right) = \left(I_{l_1}/I_{l_2}\right)^{1/2}, \quad 1 \le l_1 \le l_2 \le L.$$

- For example, define $P_{fl}(\theta)$ and $P_{el}(\theta)$ to be the probabilities we stop for futility or efficacy at stage l respectively. Then for example:

$$P_{f3}(\theta) = \int_{f_1}^{e_1} \int_{f_2}^{e_2} \int_{-\infty}^{f_3} \Phi\left(\boldsymbol{\theta}, \mathrm{Cov}(\mathbf{Z})\right) d\boldsymbol{\Phi}, \quad \text{for } \boldsymbol{\theta} = (\theta, \ldots, \theta)^\top, \mathbf{Z} = (Z_1, \ldots, Z_3)^\top.$$

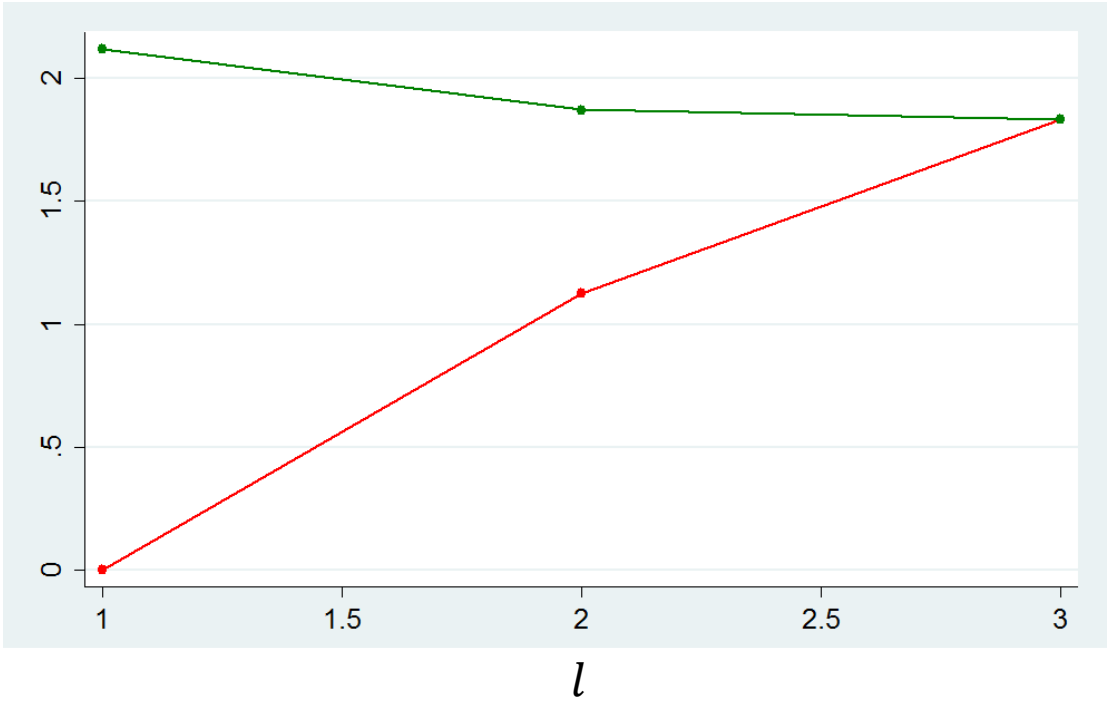
- Then we have:

$$\mathbb{E}(N|\theta) = \sum_{l=1}^{L} 2n\left[P_{fl}(\theta) + P_{el}(\theta)\right] \text{ and } \mathrm{Power}(\theta) = \sum_{l=1}^{L} P_{el}(\theta).$$

Group sequential clinical trial design

Power and expected sample size

- As an example, determine the design for $L = 3, \delta = 0.2, \alpha = 0.05, \beta = 0.2, \sigma = 1.$



True treatment effect	Fixed Sample Design		Triangular Test	
	$\mathbb{E}(N\|\theta)$	$\text{Power}(\theta)$	$\mathbb{E}(N\|\theta)$	$\text{Power}(\theta)$
$\theta = 0$	620	0.050	401.6	0.051
$\theta = \delta$	620	0.808	469.3	0.801

Conclusion

Complete and simple to use

- Created four easy to use commands that allow you to work with the multivariate normal distribution.
- Performance of these commands is seen to be very good.
- Complementary to `mvnp` with the relative efficiency dependent on the number of required integrals.
- Similarly, we have created commands for working with the multivariate t distribution.
- Moving forward, we would like to add functionality to allow alternate specialist algorithms to be used.

Conclusion

Questions?

MRC | Biostatistics Unit

wellcome trust