University of BRISTOL

Value of information calculations in R

Howard Thom with

Wei Fang, Zhenru Wang, Mike Giles, Chris Jackson, Mike Giles, Christoph Andrieu and Nicky Welton

UCL, 11th July 2018

MRC Hubs for Trials Methodology Research

ConDuCT-II Hub

School of Social and Community Medicine

Overview

- Recall the formula for the expected value of partial perfect information (EVPPI)

$$EVPPI(X) = E_X\left[\max_{d\in D} E_{Y|X}[NB_d(X,Y)]\right] - \max_{d\in D} E_{X,Y}[NB_d(X,Y)]$$

- EVPPI can be estimated (in principle) by nested Monte Carlo simulation:

$$\frac{1}{N}\sum_{n=1}^{N}\max_{d\in D}\frac{1}{M}\sum_{m=1}^{M} NB_d\left(X^{(n)},Y^{(m,n)}\right) - \max_{d\in D}\frac{1}{MN}\sum_{n=1}^{N}\sum_{m=1}^{M} NB_d\left(X^{(n)},Y^{(m,n)}\right)$$

- This requires N samples of $X^{(n)}$ and M samples $Y^{(m,n)}$ conditional on each $X^{(n)}$
- This nested simulation is biased due to Jensen's inequality and computationally intensive.
- Near impossible to estimate using Excel; possible in R but may struggle for complex models
- R can parallelise computation to increase efficiency: run each sample from $X^{(n)}$ on separate CPU.
  - Package 'parallel' in R.
- Can also use meta-modelling techniques, some of which have been implemented in easy-to-use R packages or online servies.
- We propose using efficient Monte Carlo sampling schemes for efficient and accurate estimation

Meta-modelling approaches

- Consider the first term of the EVPPI

$$EVPPI(X) = E_X\left[\max_{d\in D} E_{Y|X}[NB_d(X,Y)]\right] - \max_{d\in D} E_{X,Y}[NB_d(X,Y)]$$

- We can remove the nested simulation by regressing $NB_d(X,Y)$ on X

$$NB_d(X,Y) = E_{Y|X}\{NB_d(X,Y)\} + \varepsilon$$
$$NB_d(X,Y) = g_d(X) + \varepsilon$$

- Where $g_d(X)$ is some unknown smooth function and $E(\varepsilon) = 0$
- The fitted values $\widehat{g_d}(X)$ are estimates of $E_{Y|X}\{NB_d(X,Y)\}$
- EVPPI now requires only a single loop

$$EVPPI(X) \approx \frac{1}{K}\sum_{k=1}^{K} \max_d \widehat{g_d}\left(X^{(k)}\right) - \max_d \frac{1}{K}\sum_{k=1}^{K} \widehat{g_d}\left(X^{(k)}\right)$$

- Need to choose a $g_d()$ that represents well how NB_d depends on X, Y, NB_d
- Estimating $g_d()$ itself may be computationally challenging

Generalised additive model

- The generalised additive model (GAM) is commonly used to estimate EVPPI for sets of up to 5 parameters.
- Assuming $s(X)$ is a smooth function, the GAM is

$$NB_d^{(k)} = s\left(X^{(k)}\right) + \varepsilon^{(k)}$$

- Where the function $s(X)$ is a sum of known 'basis' functions

$$s(X) = \sum_{n=1}^{N} \gamma_n b_n(X)$$

- The γ_n are parameters estimated from the samples of NB_d and X.
- In higher dimensions would need to specify a multivariate smooth function

$$NB_d = s\left(X_1, \dots, X_q\right) + \varepsilon$$

- Would need N^q basis functions where N is number of basis functions needed for each dimension.
- Computationally infeasible if number of independent variables greater than 5 or 6
- GAM are implemented in the BCEA for R package and the SAVI website.

Gaussian process regression

- A further option implemented in BCEA and SAVI is Gaussian process (GP) regression
- The regression model is

$$NB_d = g_d(\boldsymbol{X}_i) + \varepsilon$$
$$\text{with } \varepsilon \sim N(0, \sigma^2)$$

- The $\boldsymbol{X}_i$ are the set of q focal parameters
- Now the function $g_d(\boldsymbol{X}_i)$ is a GP which is equivalent to

$$NB(X) \sim Normal(\boldsymbol{H}\beta, \sigma^2 \boldsymbol{\Sigma}(\delta) + \nu \boldsymbol{I})$$

- Where

$$\boldsymbol{H} = \begin{pmatrix} 1 & X_1^{(1)} & \cdots & X_q^{(1)} \\ 1 & X_1^{(2)} & \cdots & X_q^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_1^{(N)} & \cdots & X_q^{(N)} \end{pmatrix}$$

- Hyperparameters are pre-specified by the SAVI software, which may adversely affect its performance.
- R code available that can better estimate the hyperparameters and conduct GP regression.
- GP involves inverting an $N \times N$ matrix at $O(N^3)$ computational cost.
- The dimension of the focal parameters $\boldsymbol{X}_i$ also influences the cost.

SPDE-INLA

- The BCEA package includes a highly efficient adaptation of the GP regression method.
- Adapts ideas from spatial statistics for approximation of high dimensional with low dimensional problems
- Approximates high dimensional GP with low dimensional GP that is solution to a form of SPDE.
  - This changes task from inverting dense $N \times N$ matrix at $O(N^3)$ cost to inverting sparse $N \times N$ matrix at $O(N^{3/2})$ cost
- SPDE is latent Gaussian model, efficiently solved by integrated nested Laplace approximation (INLA)
  - BCEA uses the 'R-INLA' package in R.
- SPDE approach doesn't work in high dimensional parameter space.
- BCEA therefore uses principal fitted components projection to reduce to 2-dimensions.
  - BCEA uses the 'ldr' package in R.
- Syntax in R is relatively simple:

evppi.inla<-evppi(param.indices,all.param.samples,bcea.model.object,method="INLA")

Multilevel Monte-Carlo

- Key idea is to choose an estimator $\widehat{EVPPI}_0$ with a larger bias but much lower cost than $\widehat{EVPPI}$.
- We can then construct the EVPPI estimator

$$\frac{1}{N_0}\sum_{n=1}^{N_0}\widehat{EVPPI}_0^{(n)} + \frac{1}{N_1}\sum_{n=1}^{N_1}\left(\widehat{EVPPI}^{(n)} - \widehat{EVPPI}_0^{(n)}\right)$$

- We then need only estimate the (cheap) $\widehat{EVPPI}_0$ and the difference $\widehat{EVPPI} - \widehat{EVPPI}_0$.
- So long as $\widehat{EVPPI}$ and $\widehat{EVPPI}_0$ are highly correlated the variance will be small and number of samples N_1 for accurate estimation will be small.
- The variance of $\widehat{EVPPI}_0$ is close to that of $\widehat{EVPPI}$ so N_0 is similar to number needed to estimate $\widehat{EVPPI}$
- This extends naturally via a sequence of estimators $\widehat{EVPPI}_0$, $\widehat{EVPPI}_1$, …, $\widehat{EVPPI}_L$ to the estimator

$$\frac{1}{N_0}\sum_{n=1}^{N_0}\widehat{EVPPI}_0^{(0,n)} + \sum_{l=1}^{L}\frac{1}{N_l}\left(\widehat{EVPPI}^{(l,n)} - \widehat{EVPPI}_0^{(l,n)}\right)$$

MLMC for EVPPI

- In practice it is more efficient o estimate the quantity

$$DIFF = EVPI - EVPPI$$

- The MLMC estimator using N_l outer samples for each level l is then

$$\widehat{DIFF} = \sum_{l=1}^{L} \frac{1}{N_l} \sum_{n=1}^{N_l} \left(\widehat{DIFF}_l^{(l,n)} - \widehat{DIFF}_{l-1}^{(l,n)} \right)$$

- Unlike NMC, MLMC provides an estimate of the estimation bias

$$E\left[\widehat{DIFF}_{L+1} - \widehat{DIFF}_L\right] \sim \sum_{l=L+1}^{\infty} E\left[\widehat{DIFF}_l - \widehat{DIFF}_{l-1}\right] = DIFF - E\left[\widehat{DIFF}_L\right]$$

- Number of levels L is chosen to achieve bias $\leq 0.5\varepsilon$ and N_l chosen to achieve variance $\leq 0.75\varepsilon^2$.
- This achieves Mean Square Error (MSE) ε^2 at a cost which is only $O(\varepsilon^{-2})$
- Nested Monte-Carlo would require $O(\varepsilon^{-3})$
- Package ‘mlmc’ exits in R and forthcoming publication will include example code.

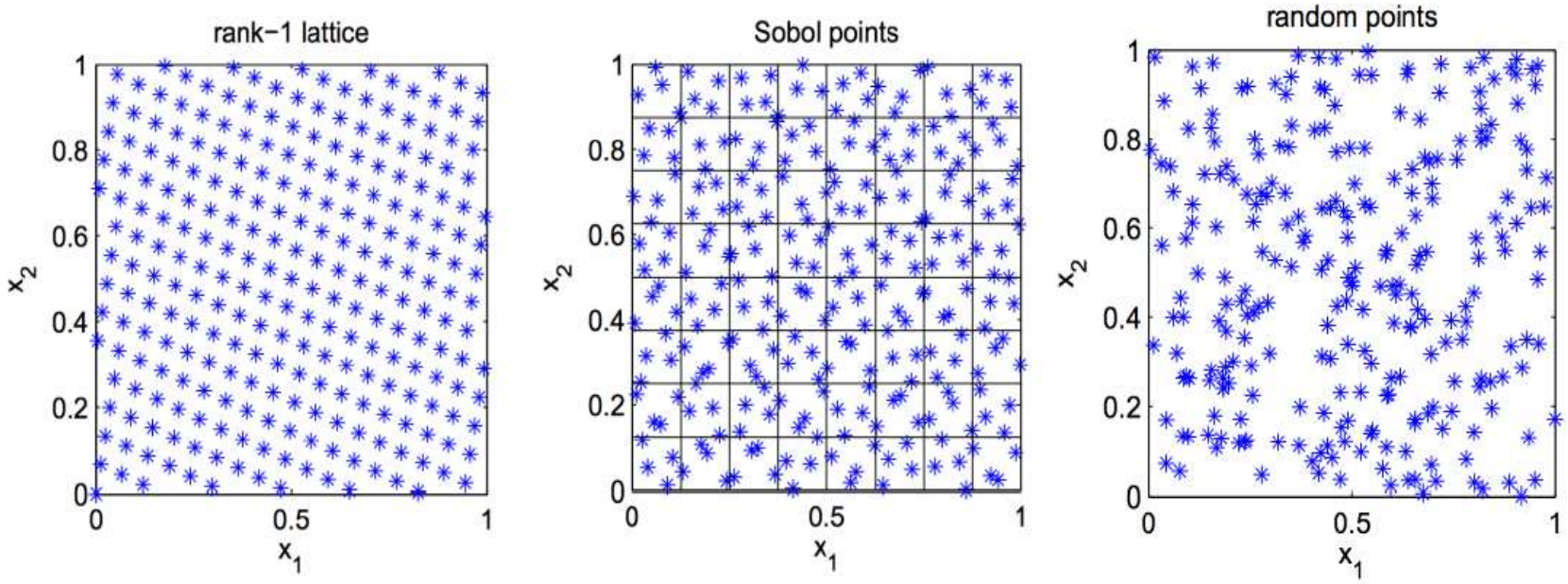
Quasi Monte Carlo

- Assuming g is a real-valued function and X is a random variable with cumulative distribution F, we estimate $E[g(X)]$ by

$$\frac{1}{N}\sum_{n=1}^{N} g(X_i)$$

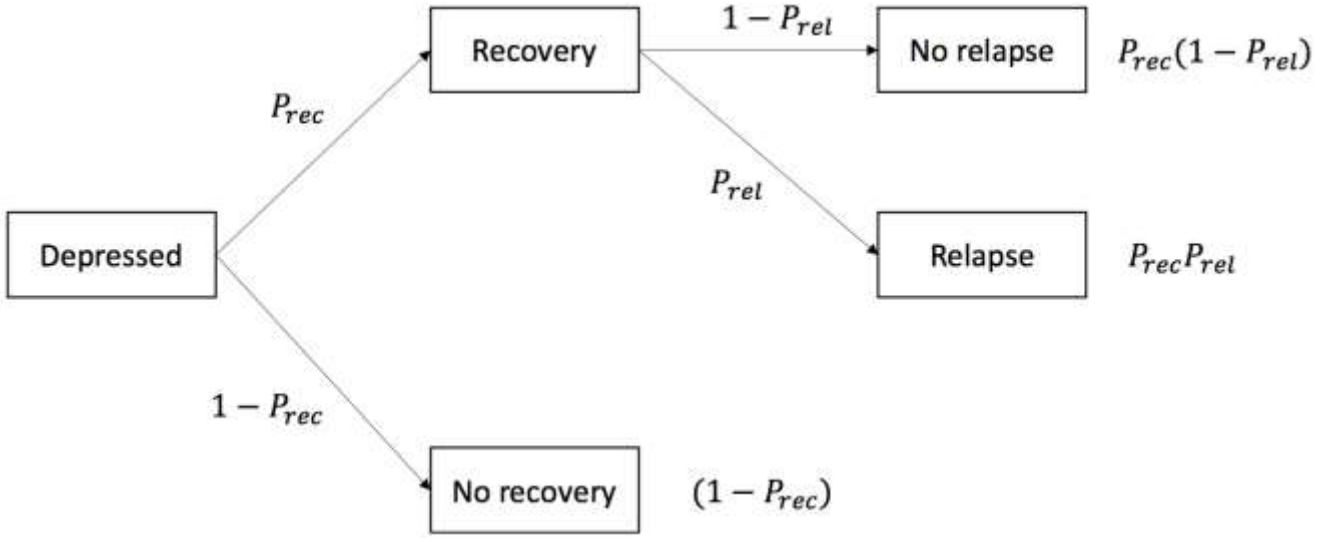
  where $X_i = F^{-1}(U_i)$ and $\{U_i\}$ are uniformly distributed in $[0,1]$.
- Standard MC chooses U_i for X_i randomly.
- QMC chooses U_i for X_i systematically.
- QMC is deterministic and achieves a better convergence rate of the error than MC
- For EVPPI, only apply QMC to outer samples N as not many inner samples M are needed for accuracy.
- A confidence interval for estimated EVPPI can be provided using randomized QMC.
- Randomized QMC is unbiased and has much lower variance than NMC.

Quasi Monte Carlo



- Example of QMC uniform random number generation in 2-dimensions for 256 points using using rank-1 lattice points or a Sobol sequence.

Depression toy example



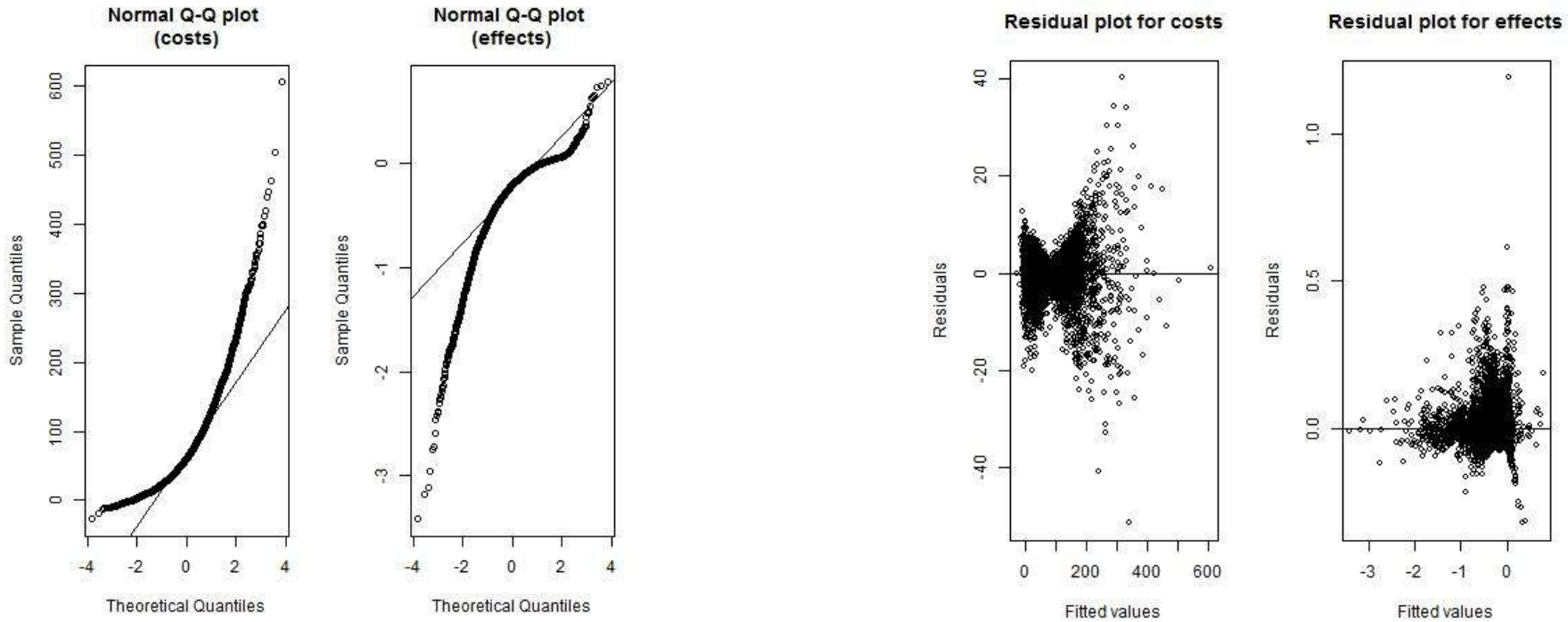
- Artificial example comparing no-treatment with CBT and antidepressants.
- Decision tree model follows structure used in NICE CG90 guidelines
- Analysed network meta-analysis data using Markov Chain Monte Carlo (MCMC)

Depression toy example - results

Parameters	MC reference (RMSE)	MC	QMC	MLMC	GAM (SE)	GP (SE)	INLA
Probabilities (6)	275.71 (0.5)	273.62 (5.19)	290.20 (3.03)	274.84 (12.00)	NA	332.84 (44.43)	293.44
Costs and QALYs (6)	287.29 (0.5)	286.86 (5.46)	286.93 (4.58)	285.13 (9.80)	NA	281.50 (29.92)	547.42
CBT lor (2)	7.35 (0.5)	29.53 (20.47)	44.65 (20.44)	22.03 (19.44)	11.01 (6.46)	NA	13.83
Antidepressant lor (2)	1.78 (0.25)	.28 (18.73)	5.12 (16.58)	4.10 (16.00)	2.26 (6.51)	NA	4.81

- MC, MLMC, and QMC based on 50,000 samples; GAM, GP, and INLA on 7500 to account for additional computation time for regression.
- MLMC did not offer reliable computational savings over MC.
- QMC offered substantial computational savings on larger EVPPI values.
- GAM and INLA perform well on small EVPPI values.

Diagnostic plots from INLA



- INLA gave unreliable estimates for the cost and utilities parameter set, but diagnostic plots alert us.
- These are generated using

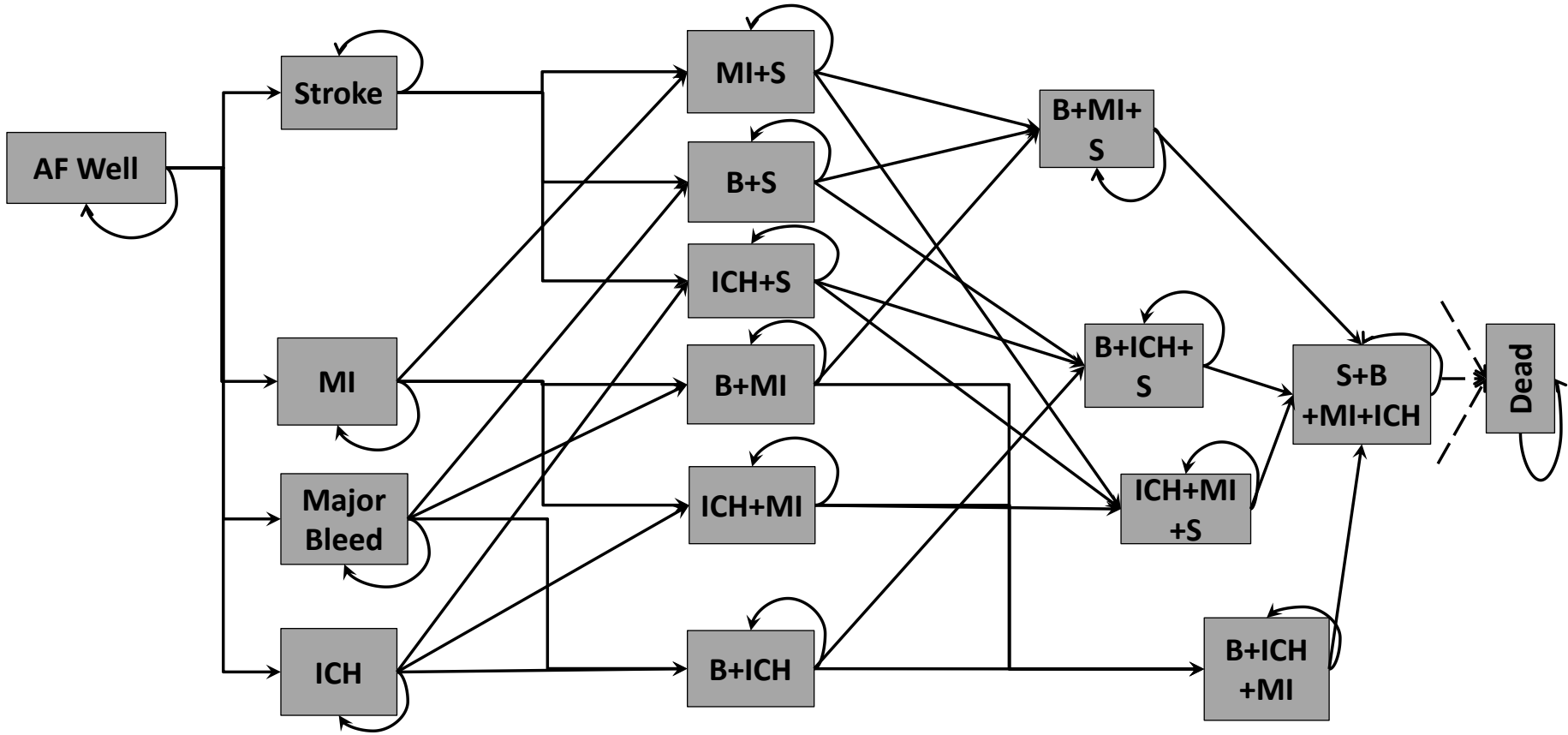
diag.evppi(x=evppi.inla.lor.anti,y=he.depression)

diag.evppi(x=evppi.inla.lor.anti,y=he.depression,diag="qqplot")

- Not clear what the problem is: the model structure is a simple decision tree and these (log) cost and utilities parameters follow independent normal distributions.

DOACs for atrial fibrillation

- 17 health states plus two transient events (TIA and SE)
- 5 treatment options: coumarin, apixaban, dabigatran, rivaroxaban, edoxaban.
- Competing risks NMA fit using MCMC (35-dim)
- Baseline hazards (7-dim) and hazard ratios relative to no treatment (7-dim) also used MCMC
- Impact of previous events on future risks, cost and utility parameters bring total to 99.



DOACs model results

Parameters (N)	Reference value (RMSE)	MC cost	MLMC cost	QMC cost	GP (SE)	INLA
Simple trial (14)	196.69 (1.23)	2144	68.63	-	561.26 (68.22)	741.09
Complex trial (79)	273.33 (1.23)	157	33.78	89.54	869.79 (69.87)	-
All MCMC (41)	348.23 (1.23)	139.9	24.47	31.72	782.38 (73.95)	256.56
Loghr MCMC (28)	286.48 (1.23)	317.1	31.85	21.21	978.03 (94.67)	551.98

- MC, MLMC, and QMC cost are 10^5 samples.
- INLA and GP based on only 7500 samples (max accepted by SAVI)
- GAM cannot be used as parameter sets too large.
- INLA and GP did not give accurate results with small number of simulations.
- Only (parallel computationally intensive) nested MC, MLMC and QMC provided reliable results…

Conclusions

- Wealth of options for estimating EVPPI in R
- For simple models (decision trees, low-state Markov) and low dimensional parameter sets, SAVI and BCEA can be efficient and reliable.
- In more complex situations, necessary to use parallel computing Monte Carlo or advanced sampling schemes.
- However, these are more challenging to implement and there are no black-box functions similar to BCEA or SAVI.
- Parallel computing and efficient sampling schemes are near impossible to implement in Excel.

Acknowledgements

- *Hubs for Trials Methodology Research (HTMR), Network Grant N79, Efficient sample schemes for estimation of value of information of future research.*
- *The HTMR Collaboration and innovation in Difficult and Complex randomised controlled Trials In Invasive procedures (ConDuCT-II) hub provided support for this research.*
- *Funding for the DOACs model was provided by NIHR HTA grant 11/92/17*
- *This study was supported by the NIHR Biomedical Research Centre at the University Hospitals Bristol NHS Foundation Trust and the University of Bristol. The views expressed in this presentation are those of the author(s) and not necessarily those of the NHS, the National Institute for Health Research or the Department of Health.*

MRC Hubs for Trials Methodology Research
ConDuCT-II Hub

NHS
University Hospitals Bristol
NHS Foundation Trust

Funded by
NIHR

University of BRISTOL

MLMC for EVPPI

- To apply MLMC to EVPPI we instead estimate the quantity

$$DIFF = EVPI - EVPPI = E_{X,Y}\left[\max_{d\in D} f(X,Y)\right] - E_X\left[\max_{d\in D} E_{Y|X}[f_d(X,Y)]\right]$$

- The NMC estimator with 2^l inner samples based on $X^{(n)}$ is

$$\widehat{DIFF}_l = \frac{1}{2^l}\sum_{m=1}^{2^l} \max_{d\in D} f_d\left(X^{(n)}, Y^{(n,m)}\right) - \max_{d\in D}\frac{1}{2^l}\sum_{m=1}^{2^l} f_d\left(X^{(n)}, Y^{(n,m)}\right)$$

- MLMC generates a sequence of differences $\widehat{DIFF}_l - \widehat{DIFF}_{l-1}$ and estimates the telescoping sum

$$E\left[\widehat{DIFF}_L\right] = E\left[\widehat{DIFF}_0\right] + \sum_{l=1}^{L} E\left[\widehat{DIFF}_l - \widehat{DIFF}_{l-1}\right]$$

- For low l, $\widehat{DIFF}_l - \widehat{DIFF}_{l-1}$ is cheap to evaluate, for high l it has a low variance so few samples are needed to estimate its expectation