EVPPI_Bristol

Zhenru Wang

11/03/2017

This is an R Markdown document which contains all the codes and results of all the EVPI and EVPPI for the EVPPI project in University of Bristol. Due to some update issues in the MLMC package, we currently use the functions in the package but with some modifications on codes.

The following four R codes are developed by Zhenru to plot CEAC, calculate different EVPI and EVPPI with N outer samples and n inner samples using standard nested MC, and calculate EVPPI using QMC.

```
source("ceac.R")
source("mc_EVPI.R")
source("mc_EVPPI_P.R")
source("mc_EVPPI_CQ.R")
source("mc_EVPPI_lor2.R")
source("qmc_EVPPI_P.R")
source("qmc_EVPPI_CQ.R")
source("qmc_EVPPI_lor2.R")
```

EVPI

Suppose that the perfect information is available. That means by density ρ_Z we can generate samples of Z and for each sample $Z^{(n)}$ we can find the optimal decision d for that specific decision. Therefore the expected value of perfect information (EVPI) is

$$\text{EVPI} = \mathbb{E}_Z\left[\max_{d\in D} f_d(Z)\right] - \max_{d\in D} \mathbb{E}_Z\left[f_d(Z)\right].$$

CEAC

Following is the code for CEAC plot, where N is the number of samples for each λ, and m is the number of different λ equally spread between 0 and 5×10^4.

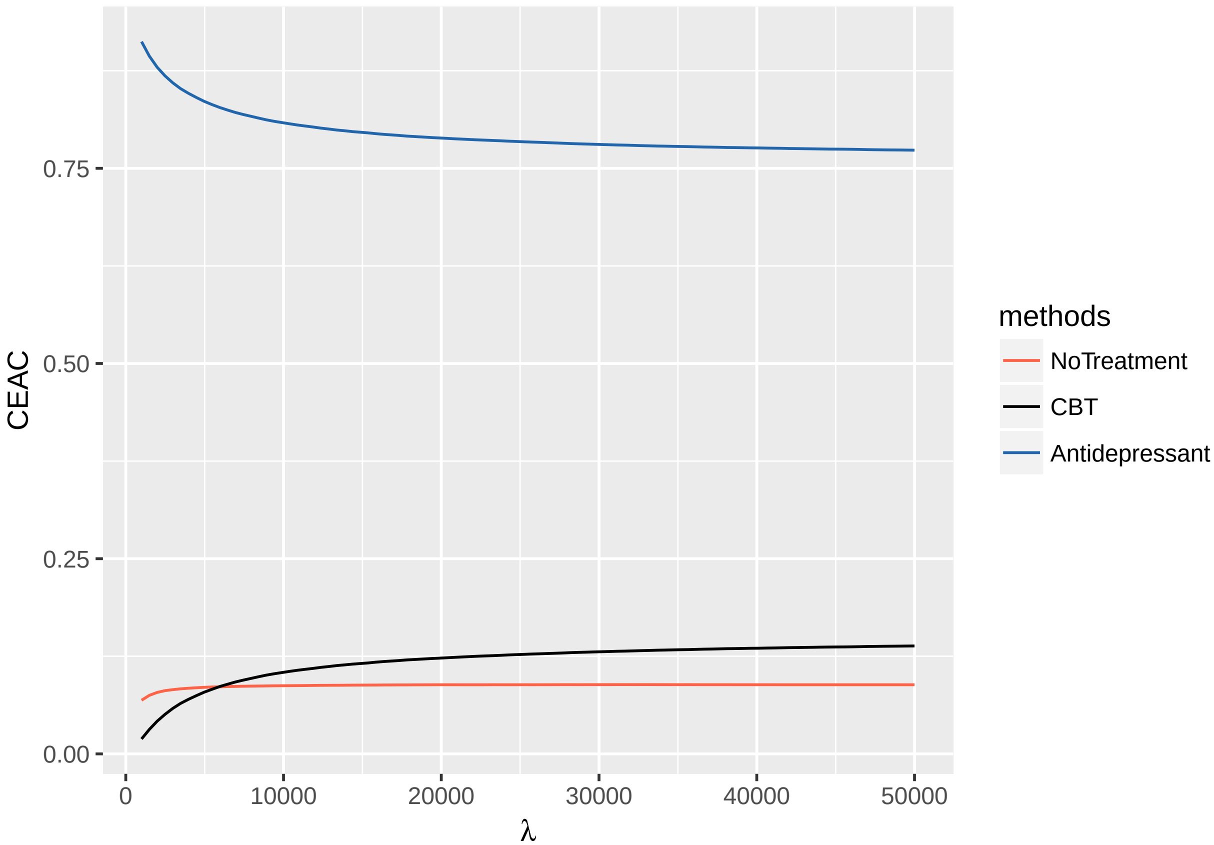
```
ceac <- function(N,m){
    pkg <- c("Matrix","MASS","boot","tictoc","sna","ggplot2","RColorBrewer","foreach",
             "reshape2")
    lapply(pkg, library, character.only = TRUE)

    tic()
    set.seed(666)

    mean_rec <- c(0.99,1.33)
    var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
    mean_rel <- c(-1.48,-0.4)
    var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)

    lor_rec <- mvrnorm(N,mean_rec,var_rec)
    lor_rel <- mvrnorm(N,mean_rel,var_rel)
```

```
    P_rec <- matrix(0,N,3)
    P_rel <- matrix(0,N,3)
    P_rec[,1] <- rbeta(N,6,200)
    P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec[,1])
    P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec[,2])

    P_rel[,1] <- rbeta(N,2,100)
    P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel[,1])
    P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel[,2])

    C_rec <- rnorm(N,1000,50)
    C_rel <- rnorm(N,2000,100)
    C_norec <- rnorm(N,2500,125)

    Q_rec <- rnorm(N,26,2)
    Q_rel  <- rnorm(N,23,3)
    Q_norec <- rnorm(N,20,4)

    C_t <- t(replicate(N,c(0,300,30)))
    lambda <- seq(1e3, 5e4, length.out = m)
    CEAC <- matrix(0,m,3)

    for(i in seq(1, m)){
        QALY <- P_rec*(1-P_rel)*Q_rec + P_rec*P_rel*Q_rel + (1-P_rec)*Q_norec
        Cost <- P_rec*(1-P_rel)*C_rec + P_rec*P_rel*C_rel + (1-P_rec)*C_norec + C_t

        NB <- lambda[i]*QALY - Cost
        NBmax <- replicate(3,apply(NB,1,max))
        CEAC[i,] <- colSums(NB == NBmax)/N
    }

    toc()

    dt <- data.frame("lambda" = lambda, "NoTreatment" = CEAC[,1], "CBT" = CEAC[,2],
                     "Antidepressant" = CEAC[,3])
    dt <- melt(dt, id="lambda")
    names(dt) <- c("lambda",'methods','yValue')

    my_palette <- c("NoTreatment" = "tomato", "CBT" = "black",
                    "Antidepressant" = "#2166ac")

    p <- ggplot(data = dt) + scale_color_manual(values = my_palette)
    + geom_line(aes(x = lambda, y = yValue, color = methods))
    + ylab("CEAC")+ xlab(expression(lambda))
    p
}
```

```
p <- ceac(1e5,1e2)
```

```
## 58.843 sec elapsed
```

```
p
```



From the plot, we can see that after $\lambda > 5000$, Antidepressant is the best treatment of the three.

EVPPI

Here is the code for **mc_EVPI.R** using standard MC, which requre `Matrix`, `MASS` and `boot` packages:

```
# EVPI (P) = 573.6317 +/- 5.7958, var = 3.7324, N=1.0e+06. 6.719 sec

mc_EVPI <- function(N){

    pkg <- c("Matrix","MASS","boot","tictoc")
    lapply(pkg, library, character.only = TRUE)

    tic()
    set.seed(666)

    EVPI_sum <- matrix(0,3,2)
    QALY_mean <- rep(0,3)
    COST_mean <- rep(0,3)

    mean_rec <- c(0.99,1.33)
    var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
    mean_rel <- c(-1.48,-0.4)
    var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)

    for(N1 in seq(1,N,by=1e4)){
```

```r
        N2 <- min(1e4, N-N1+1)

        P_rec <- matrix(0,N2,3)
        P_rel <- matrix(0,N2,3)
        lor_rec <- mvrnorm(N2,mean_rec,var_rec)
        lor_rel <- mvrnorm(N2,mean_rel,var_rel)

        P_rec[,1] <- rbeta(N2,6,200)
        P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec[,1])
        P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec[,2])

        P_rel[,1] <- rbeta(N2,2,100)
        P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel[,1])
        P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel[,2])

        C_rec <- rnorm(N2,1000,50)
        C_rel <- rnorm(N2,2000,100)
        C_norec <- rnorm(N2,2500,125)

        Q_rec <- rnorm(N2,26,2)
        Q_rel  <- rnorm(N2,23,3)
        Q_norec <- rnorm(N2,20,4)

        C_t <- c(0,300,30)
        lambda <- 2e4

        QALY <- P_rec*(1-P_rel)*Q_rec + P_rec*P_rel*Q_rel + (1-P_rec)*Q_norec
        Cost <- P_rec*(1-P_rel)*C_rec + P_rec*P_rel*C_rel + (1-P_rec)*C_norec

        Cost[,2] <- Cost[,2] +C_t[2]
        Cost[,3] <- Cost[,3] +C_t[3]

        NB <- lambda*QALY - Cost
        QALY_mean <- QALY_mean + colSums(QALY)
        COST_mean <- COST_mean + colSums(Cost)

        NB <- replicate(3,apply(NB,1,max)) -NB
        EVPI_sum[,1] <- EVPI_sum[,1] + colSums(NB)
        EVPI_sum[,2] <- EVPI_sum[,2] + colSums(NB^2)
    }

    EVPI <- min(EVPI_sum[,1]/N)
    ind <- which.min(EVPI_sum[,1])
    EVPI_var <- EVPI_sum[ind,2]/N-EVPI^2
    var <- EVPI_var/N
    QALY_mean <- QALY_mean/N
    COST_mean <- COST_mean/N
    toc()

    cat(sprintf("EVPI (P) = %.4f +/- %.4f, var = %.4f, N=%.1e. \n\n ",
                EVPI,3*sqrt(var),var,N))
    return(list(EVPI=EVPI,var=var,QALY_mean=QALY_mean,COST_mean=COST_mean))
}
```

We can get EVPI by

```
EVPI <- mc_EVPI(1e6)
```

```
## 6.458 sec elapsed
## EVPI (P) = 573.6317 +/- 5.7958, var = 3.7324, N=1.0e+06.
##
##
```

EVPPI

Assume that the unknown parameters can be decomposed into two random variables as $Z = (X, Y)$ with $\Omega = \Omega_X \times \Omega_Y$ and only information of X is available. That means we can generate samples of X first and for each sample $X^{(n)}$, we can calculate the maximum of the conditional expectation of Y based on $X^{(n)}$. Therefore the expected value of partial perfect information (the value of X) is

$$\text{EVPPI} = \mathbb{E}_X \left[\max_{d \in D} \mathbb{E}_{Y|X} \left[f_d(X, Y) \right] \right] - \max_{d \in D} \mathbb{E}_Z \left[f_d(Z) \right]$$

We use both standard MC and QMC for the estimates.

To implement QMC estimate, we start from generating Sobal sequences for uniform distributed random variables U. For a r.v. X with CDF F, we then get X by

$$X = F^{-1}(U).$$

For our specific problem, we need the inverse CDF for β distrubution, which is implemented as the following **sfunc.R**. This code is written by Thomas Lumley, see https://stat.ethz.ch/pipermail/r-help/2002-January/017624.html.

```
##
##  UCS / The R ToolBox
##

## Filename: sfunc.R
## Modified: Sat Feb 21 18:24:49 2004 (evert)
##   Author: Stefan Evert
##  Purpose: Special functions (beta, gamma, confidence intervals)


## (complete) gamma function and its logarithm (all logarithms are base 10)
Cgamma <- function (a, log=FALSE) {
  if (log) {
    lgamma(a) / log(10)
  } else {
    gamma(a)
  }
}

## regularised gamma function (lower P(a,x) and upper Q(a,x)) and its inverse
Rgamma <- function (a, x, lower=TRUE, log=FALSE) {
  if (log) {
    pgamma(x, shape=a, scale=1, lower.tail=lower, log=TRUE) / log(10)
  } else {
    pgamma(x, shape=a, scale=1, lower.tail=lower, log=FALSE)
```

```r
  }
}
Rgamma.inv <- function(a, y, lower=TRUE, log=FALSE) {
  if (log) {
    qgamma(y * log(10), shape=a, scale=1, lower.tail=lower, log=TRUE)
  } else {
    qgamma(y, shape=a, scale=1, lower.tail=lower, log=FALSE)
  }
}

## incomplete gamma function (lower gamma(a,x) and upper Gamma(a,x)) and its inverse
Igamma <- function (a, x, lower=TRUE, log=FALSE) {
  if (log) {
    Cgamma(a, log=TRUE) + Rgamma(a, x, lower, log=TRUE)
  } else {
    Cgamma(a, log=FALSE) * Rgamma(a, x, lower, log=FALSE)
  }
}
Igamma.inv <- function (a, y, lower=TRUE, log=FALSE) {
  if (log) {
    Rgamma.inv(a, y - Cgamma(a, log=TRUE), lower, log=TRUE)
  } else {
    Rgamma.inv(a, y / Cgamma(a, log=FALSE), lower, log=FALSE)
  }
}

## beta function and its logarithm
Cbeta <- function(a, b, log=FALSE) {
  if (log) {
    lbeta(a, b) / log(10)
  } else {
    beta(a, b)
  }
}

## regularised beta function I(x; a, b) and its inverse
Rbeta <- function (x, a, b, log=FALSE) {
  if (log) {
    pbeta(x, shape1=a, shape2=b, log=TRUE) / log(10)
  } else {
    pbeta(x, shape1=a, shape2=b, log=FALSE)
  }
}
Rbeta.inv <- function (y, a, b, log=FALSE) {
  if (log) {
    qbeta(y * log(10), shape1=a, shape2=b, log=TRUE)
  } else {
    qbeta(y, shape1=a, shape2=b, log=FALSE)
  }
}

## incomplete beta function B(x; a, b) and its inverse
Ibeta <- function (x, a, b, log=FALSE) {
```

```
  if (log) {
    Cbeta(a, b, log=TRUE) + Rbeta(x, a, b, log=TRUE)
  } else {
    Cbeta(a, b, log=FALSE) * Rbeta(x, a, b, log=FALSE)
  }
}
Ibeta.inv <- function (y, a, b, log=FALSE) {
  if (log) {
    Rbeta.inv(y - Cbeta(a, b, log=TRUE), a, b, log=TRUE)
  } else {
    Rbeta.inv(y / Cbeta(a, b, log=FALSE), a, b, log=FALSE)
  }
}

## two-sided confidence interval for success probability p of binomial distribution,
## given that k successes out of size trials have been observed
##   p.limit <- binom.conf.interval(k, size, conf.level=0.05, limit="lower", one.sided=FALSE)
## conf.level = confidence level, e.g. 0.01 corresponds to 99% confidence
## one.sided  = if TRUE, one-sided confidence level, otherwise two-sided
binom.conf.interval <- function(k, size, limit=c("lower","upper"),
                                conf.level=0.05, one.sided=FALSE) {
  l <- length(k)
  if (l != length(size)) {
    if (length(size)==1) {
      size <- rep(size, times=l)
    } else {
      stop("Parameters k and size must be vectors of identical length.")
    }
  }
  limit <- match.arg(limit)
  # use regularised incomplete Beta function (= distribution function of Beta distribution)
  # to compute two-sided confidence intervals for parameter of binomial distribution
  if (one.sided) alpha <- conf.level else alpha <- conf.level / 2
  if (limit == "lower") {
    return(qbeta(alpha, k, size - k + 1))
  }
  else {
    return(qbeta(1 - alpha, k + 1, size - k))
  }
}
```

EVPPI for P

Here is the code for **mc_EVPPI_P.R** using standard MC, where N is the number of outer samples, and n is the number of inner samples.

```
# EVPPI (P) = 275.4283 +/- 3.0524, var = 1.0352, N=1.0e+06, n=100. 193.309 sec

mc_EVPPI_P <- function(N,n){

    pkg <- c("Matrix","MASS","boot","tictoc","randtoolbox")
    lapply(pkg, library, character.only = TRUE)
```

```r
tic()
set.seed(666)

EVPPI_sum <- matrix(0,3,2)

mean_rec <- c(0.99,1.33)
var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
mean_rel <- c(-1.48,-0.4)
var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)

C_t <- c(0,300,30)
lambda <- 2e4

for(N1 in seq(1,N,by=1e4)){
    N2 <- min(1e4, N-N1+1)

    lor_rec <- mvrnorm(N2,mean_rec,var_rec)
    lor_rel <- mvrnorm(N2,mean_rel,var_rel)
    P_rec <- matrix(0,N2,3)
    P_rel <- matrix(0,N2,3)
    P_rec[,1] <- rbeta(N2,6,200)
    P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec[,1])
    P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec[,2])

    P_rel[,1] <- rbeta(N2,2,100)
    P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel[,1])
    P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel[,2])

    C_rec <- matrix(rnorm(n*N2,1000,50),n,N2)
    C_rel <- matrix(rnorm(n*N2,2000,100),n,N2)
    C_norec <- matrix(rnorm(n*N2,2500,125),n,N2)

    Q_rec <- matrix(rnorm(n*N2,26,2),n,N2)
    Q_rel  <- matrix(rnorm(n*N2,23,3),n,N2)
    Q_norec <- matrix(rnorm(n*N2,20,4),n,N2)

    QALY = matrix(0,n,3)
    Cost = matrix(0,n,3)

    for(i in seq(1,N2)){
        QALY[,1] <- P_rec[i,1]*(1-P_rel[i,1])*Q_rec[,i] +
            P_rec[i,1]*P_rel[i,1]*Q_rel[,i] + (1-P_rec[i,1])*Q_norec[,i]
        QALY[,2] <- P_rec[i,2]*(1-P_rel[i,2])*Q_rec[,i] +
            P_rec[i,2]*P_rel[i,2]*Q_rel[,i] + (1-P_rec[i,2])*Q_norec[,i]
        QALY[,3] <- P_rec[i,3]*(1-P_rel[i,3])*Q_rec[,i] +
            P_rec[i,3]*P_rel[i,3]*Q_rel[,i] + (1-P_rec[i,3])*Q_norec[,i]

        Cost[,1] <- P_rec[i,1]*(1-P_rel[i,1])*C_rec[,i] +
            P_rec[i,1]*P_rel[i,1]*C_rel[,i] + (1-P_rec[i,1])*C_norec[,i]
        Cost[,2] <- P_rec[i,2]*(1-P_rel[i,2])*C_rec[,i] +
            P_rec[i,2]*P_rel[i,2]*C_rel[,i] + (1-P_rec[i,2])*C_norec[,i] + C_t[2]
        Cost[,3] <- P_rec[i,3]*(1-P_rel[i,3])*C_rec[,i] +
            P_rec[i,3]*P_rel[i,3]*C_rel[,i] + (1-P_rec[i,3])*C_norec[,i] + C_t[3]
```

```
            NB <- colMeans(lambda*QALY - Cost)
            NB <- max(NB) - NB;
            EVPPI_sum[,1] <- EVPPI_sum[,1] + NB
            EVPPI_sum[,2] <- EVPPI_sum[,2] + NB^2
        }
    }

    EVPPI <- min(EVPPI_sum[,1]/N)
    ind <- which.min(EVPPI_sum[,1])
    var <- (EVPPI_sum[ind,2]/N-EVPPI^2)/N
    toc()
    cat(sprintf("EVPPI (P) = %.4f +/- %.4f, var = %.4f, N=%.1e, n=%d. \n\n ",
                EVPPI,3*sqrt(var),var,N,n))
    return(list(EVPPI=EVPPI,var=var))
}
```

We also calculate by QMC **qmc_EVPPI_P.R**, where we generate Sobal sequences for N outer samples, and standard simulations for n inner samples. We randomise M times to get a confidence interval.

```
qmc_EVPPI_P <- function(N,n,M){

    pkg <- c("Matrix","MASS","boot","tictoc","randtoolbox")
    lapply(pkg, library, character.only = TRUE)
    source("sfunc.R")

    tic()
    set.seed(6666)

    mean_rec <- c(0.99,1.33)
    var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
    L_rec = chol(var_rec)
    mean_rel <- c(-1.48,-0.4)
    var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)
    L_rel = chol(var_rel)

    C_t <- c(0,300,30)
    lambda <- 2e4

    EVPPI <- matrix(0,M,2)

    for(m in seq(1,M)){
        EVPPI_sum <- rep(0,3)

        for(N1 in seq(1,N,by=1e4)){
            N2 <- min(1e4, N-N1+1)

            U <- sobol(N2, dim = 6, init = TRUE, scrambling = 1, seed = 666,
                       normal = FALSE)

            P_rec <- matrix(0,N2,3)
            P_rel <- matrix(0,N2,3)

            X_rec = qnorm(U[,1:2])
            lor_rec <- t(replicate(N2,mean_rec)) + X_rec%*%L_rec
```

```
        X_rel = qnorm(U[,3:4])
        lor_rel <- t(replicate(N2,mean_rel)) + X_rel%*%L_rel

        P_rec[,1] <- Rbeta.inv(U[,5], 6, 200, log=FALSE)
        P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec[,1])
        P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec[,2])
        P_rel[,1] <- Rbeta.inv(U[,6], 2, 100, log=FALSE)
        P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel[,1])
        P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel[,2])

        C_rec <- matrix(rnorm(n*N2,1000,50),n,N2)
        C_rel <- matrix(rnorm(n*N2,2000,100),n,N2)
        C_norec <- matrix(rnorm(n*N2,2500,125),n,N2)

        Q_rec <- matrix(rnorm(n*N2,26,2),n,N2)
        Q_rel  <- matrix(rnorm(n*N2,23,3),n,N2)
        Q_norec <- matrix(rnorm(n*N2,20,4),n,N2)

        QALY = matrix(0,n,3)
        Cost = matrix(0,n,3)

        for(i in seq(1,N2)){
            QALY[,1] <- P_rec[i,1]*(1-P_rel[i,1])*Q_rec[,i] +
                P_rec[i,1]*P_rel[i,1]*Q_rel[,i] + (1-P_rec[i,1])*Q_norec[,i]
            QALY[,2] <- P_rec[i,2]*(1-P_rel[i,2])*Q_rec[,i] +
                P_rec[i,2]*P_rel[i,2]*Q_rel[,i] + (1-P_rec[i,2])*Q_norec[,i]
            QALY[,3] <- P_rec[i,3]*(1-P_rel[i,3])*Q_rec[,i] +
                P_rec[i,3]*P_rel[i,3]*Q_rel[,i] + (1-P_rec[i,3])*Q_norec[,i]

            Cost[,1] <- P_rec[i,1]*(1-P_rel[i,1])*C_rec[,i] +
                P_rec[i,1]*P_rel[i,1]*C_rel[,i] + (1-P_rec[i,1])*C_norec[,i]
            Cost[,2] <- P_rec[i,2]*(1-P_rel[i,2])*C_rec[,i] +
                P_rec[i,2]*P_rel[i,2]*C_rel[,i] + (1-P_rec[i,2])*C_norec[,i] + C_t[2]
            Cost[,3] <- P_rec[i,3]*(1-P_rel[i,3])*C_rec[,i] +
                P_rec[i,3]*P_rel[i,3]*C_rel[,i] + (1-P_rec[i,3])*C_norec[,i] + C_t[3]

            NB <- colMeans(lambda*QALY - Cost)
            NB <- max(NB) - NB;
            EVPPI_sum <- EVPPI_sum + NB
        }
    }
    EVPPI[m,1] <- min(EVPPI_sum/N)
    EVPPI[m,2] <- EVPPI[m,1]^2
}


EVPPI_QMC <- sum(EVPPI[,1])/M
EVPPI_var_QMC <- sum(EVPPI[,2])/M - EVPPI_QMC^2
var <- EVPPI_var_QMC/M
toc()
cat(sprintf("QMC: EVPPI (P) = %.4f +/- %.4f, var = %.4f, N=%.1e, n=%d, M=%d. \n\n ",
            EVPPI_QMC,3*sqrt(var),var,N,n,M))
return(list(EVPPI_QMC=EVPPI_QMC,EVPPI_var_QMC=EVPPI_var_QMC,var=var))
```

```
}
```

We can also calculate EVPPI for P using QMC, see **qmc_EVPPI_P.R**:

```
# QMC: EVPPI (P) = 274.1781 +/- 0.5093, var = 0.0288, N=1.0e+04, n=100, M=16. 28.719 sec

qmc_EVPPI_P <- function(N,n,M){

    source("sfunc.R")
    pkg <- c("Matrix","MASS","boot","tictoc","randtoolbox")
    lapply(pkg, library, character.only = TRUE)

    tic()
    set.seed(6666)

    mean_rec <- c(0.99,1.33)
    var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
    L_rec = chol(var_rec)
    mean_rel <- c(-1.48,-0.4)
    var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)
    L_rel = chol(var_rel)

    C_t <- c(0,300,30)
    lambda <- 2e4

    EVPPI <- matrix(0,M,2)

    for(m in seq(1,M)){
        EVPPI_sum <- rep(0,3)

        for(N1 in seq(1,N,by=1e4)){
            N2 <- min(1e4, N-N1+1)

            U <- sobol(N2, dim = 6, init = TRUE, scrambling = 1, seed = 666,
                       normal = FALSE)

            P_rec <- matrix(0,N2,3)
            P_rel <- matrix(0,N2,3)

            X_rec = qnorm(U[,1:2])
            lor_rec <- t(replicate(N2,mean_rec)) + X_rec%*%L_rec
            X_rel = qnorm(U[,3:4])
            lor_rel <- t(replicate(N2,mean_rel)) + X_rel%*%L_rel

            P_rec[,1] <- Rbeta.inv(U[,5], 6, 200, log=FALSE)
            P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec[,1])
            P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec[,2])
            P_rel[,1] <- Rbeta.inv(U[,6], 2, 100, log=FALSE)
            P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel[,1])
            P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel[,2])

            C_rec <- matrix(rnorm(n*N2,1000,50),n,N2)
            C_rel <- matrix(rnorm(n*N2,2000,100),n,N2)
            C_norec <- matrix(rnorm(n*N2,2500,125),n,N2)
```

```
            Q_rec <- matrix(rnorm(n*N2,26,2),n,N2)
            Q_rel  <- matrix(rnorm(n*N2,23,3),n,N2)
            Q_norec <- matrix(rnorm(n*N2,20,4),n,N2)

            QALY = matrix(0,n,3)
            Cost = matrix(0,n,3)

            for(i in seq(1,N2)){
                QALY[,1] <- P_rec[i,1]*(1-P_rel[i,1])*Q_rec[,i] +
                    P_rec[i,1]*P_rel[i,1]*Q_rel[,i] + (1-P_rec[i,1])*Q_norec[,i]
                QALY[,2] <- P_rec[i,2]*(1-P_rel[i,2])*Q_rec[,i] +
                    P_rec[i,2]*P_rel[i,2]*Q_rel[,i] + (1-P_rec[i,2])*Q_norec[,i]
                QALY[,3] <- P_rec[i,3]*(1-P_rel[i,3])*Q_rec[,i] +
                    P_rec[i,3]*P_rel[i,3]*Q_rel[,i] + (1-P_rec[i,3])*Q_norec[,i]

                Cost[,1] <- P_rec[i,1]*(1-P_rel[i,1])*C_rec[,i] +
                    P_rec[i,1]*P_rel[i,1]*C_rel[,i] + (1-P_rec[i,1])*C_norec[,i]
                Cost[,2] <- P_rec[i,2]*(1-P_rel[i,2])*C_rec[,i] +
                    P_rec[i,2]*P_rel[i,2]*C_rel[,i] + (1-P_rec[i,2])*C_norec[,i] + C_t[2]
                Cost[,3] <- P_rec[i,3]*(1-P_rel[i,3])*C_rec[,i] +
                    P_rec[i,3]*P_rel[i,3]*C_rel[,i] + (1-P_rec[i,3])*C_norec[,i] + C_t[3]

                NB <- colMeans(lambda*QALY - Cost)
                NB <- max(NB) - NB;
                EVPPI_sum <- EVPPI_sum + NB
            }
        }
        EVPPI[m,1] <- min(EVPPI_sum/N)
        EVPPI[m,2] <- EVPPI[m,1]^2
    }


    EVPPI_QMC <- sum(EVPPI[,1])/M
    EVPPI_var_QMC <- sum(EVPPI[,2])/M - EVPPI_QMC^2
    var <- EVPPI_var_QMC/M
    toc()
    cat(sprintf("QMC: EVPPI (P) = %.4f +/- %.4f, var = %.4f, N=%.1e, n=%d, M=%d. \n\n ",
                EVPPI_QMC,3*sqrt(var),var,N,n,M))
    return(list(EVPPI_QMC=EVPPI_QMC,var=var))
}
```

Then we compare with two methods:

```
mc_EVPPI_P(1e5,100)
```

```
## 17.525 sec elapsed
## EVPPI (P) = 276.2663 +/- 9.7343, var = 10.5284, N=1.0e+05, n=100.
##
##

## $EVPPI
## [1] 276.2663
##
## $var
## [1] 10.52843
```

```
qmc_EVPPI_P(1e4,100,16)
```

```
## 28.654 sec elapsed
## QMC: EVPPI (P) = 273.8743 +/- 0.7407, var = 0.0610, N=1.0e+04, n=100, M=16.
##
##

## $EVPPI_QMC
## [1] 273.8743
##
## $var
## [1] 0.06095495
```

EVPPI for CQ

Here is the code for **mc_EVPPI_CQ.R** using standard MC:

```
mc_EVPPI_CQ <- function(N,n){
    pkg <- c("Matrix","MASS","boot","tictoc","randtoolbox")
    lapply(pkg, library, character.only = TRUE)

    tic()
    set.seed(666)

    EVPPI_sum <- matrix(0,3,2)

    mean_rec <- c(0.99,1.33)
    var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
    mean_rel <- c(-1.48,-0.4)
    var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)

    C_t <- c(0,300,30)
    lambda <- 2e4

    for(N1 in seq(1,N,by=1e4)){
        N2 <- min(1e4, N-N1+1)

        C_rec <- rnorm(N2,1000,50)
        C_rel <- rnorm(N2,2000,100)
        C_norec <- rnorm(N2,2500,125)
        Q_rec <- rnorm(N2,26,2)
        Q_rel  <- rnorm(N2,23,3)
        Q_norec <- rnorm(N2,20,4)

        P_rec <- matrix(0,N2*n,3)
        P_rel <- matrix(0,N2*n,3)

        lor_rec <- mvrnorm(N2*n,mean_rec,var_rec)
        lor_rel <- mvrnorm(N2*n,mean_rel,var_rel)

        P_rec[,1] <- rbeta(N2*n,6,200)
        P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec[,1])
        P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec[,2])
```

```
        P_rel[,1] <- rbeta(N2*n,2,100)
        P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel[,1])
        P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel[,2])

        for(i in seq(1,N2)){
            ind <- seq((i-1)*n+1,i*n)

            QALY <- P_rec[ind,]*(1-P_rel[ind,])*Q_rec[i] + P_rec[ind,]*P_rel[ind,]*Q_rel[i] + (1-P_rec[i
            Cost <- P_rec[ind,]*(1-P_rel[ind,])*C_rec[i] + P_rec[ind,]*P_rel[ind,]*C_rel[i] + (1-P_rec[i
            Cost[,2] <- Cost[,2] + C_t[2]
            Cost[,3] <- Cost[,3] + C_t[3]

            NB <- colMeans(lambda*QALY - Cost)
            NB <- max(NB) - NB;
            EVPPI_sum[,1] <- EVPPI_sum[,1] + NB
            EVPPI_sum[,2] <- EVPPI_sum[,2] + NB^2
        }
    }

    EVPPI <- min(EVPPI_sum[,1]/N)
    ind <- which.min(EVPPI_sum[,1])
    var <- (EVPPI_sum[ind,2]/N-EVPPI^2)/N
    toc()
    cat(sprintf("EVPPI (CQ) = %.4f +/- %.4f, var = %.4f, N=%.1e. \n\n ",
                EVPPI,3*sqrt(var),var,N))
    return(list(EVPPI=EVPPI,var=var))
}
```

Here is the code for **qmc_EVPPI_CQ.R** using QMC:

```
# QMC: EVPPI (CQ) = 286.7743 +/- 0.9500, var = 0.1003, N=1.0e+04, n=100, M=16. 35.189 sec

qmc_EVPPI_CQ <- function(N,n,M){

    pkg <- c("Matrix","MASS","boot","tictoc","randtoolbox")
    lapply(pkg, library, character.only = TRUE)
    source("sfunc.R")

    tic()
    set.seed(666)

    mean_rec <- c(0.99,1.33)
    var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
    mean_rel <- c(-1.48,-0.4)
    var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)

    C_t <- c(0,300,30)
    lambda <- 2e4

    EVPPI <- matrix(0,M,2)

    for(m in seq(1,M)){
        EVPPI_sum <- rep(0,3)
        for(N1 in seq(1,N,by=1e4)){
```

```
            N2 <- min(1e4, N-N1+1)

            U <- sobol(N, dim = 6, init = TRUE, scrambling = 1, seed = 666,
                       normal = FALSE)
            C_rec <- qnorm(U[,1],1000,50)
            C_rel <- qnorm(U[,2],2000,100)
            C_norec <- qnorm(U[,3],2500,125)
            Q_rec <- qnorm(U[,4],26,2)
            Q_rel  <- qnorm(U[,5],23,3)
            Q_norec <- qnorm(U[,6],20,4)

            P_rec <- matrix(0,N2*n,3)
            P_rel <- matrix(0,N2*n,3)

            lor_rec <- mvrnorm(N2*n,mean_rec,var_rec)
            lor_rel <- mvrnorm(N2*n,mean_rel,var_rel)

            P_rec[,1] <- rbeta(N2*n,6,200)
            P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec[,1])
            P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec[,2])

            P_rel[,1] <- rbeta(N2*n,2,100)
            P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel[,1])
            P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel[,2])

            for(i in seq(1,N2)){
                ind <- seq((i-1)*n+1,i*n)

                QALY <- P_rec[ind,]*(1-P_rel[ind,])*Q_rec[i] +
                    P_rec[ind,]*P_rel[ind,]*Q_rel[i] + (1-P_rec[ind,])*Q_norec[i]
                Cost <- P_rec[ind,]*(1-P_rel[ind,])*C_rec[i] +
                    P_rec[ind,]*P_rel[ind,]*C_rel[i] + (1-P_rec[ind,])*C_norec[i]
                Cost[,2] <- Cost[,2] + C_t[2]
                Cost[,3] <- Cost[,3] + C_t[3]

                NB <- colMeans(lambda*QALY - Cost)
                NB <- max(NB) - NB;
                EVPPI_sum <- EVPPI_sum + NB
            }
        }
        EVPPI[m,1] <- min(EVPPI_sum/N)
        EVPPI[m,2] <- EVPPI[m,1]^2
    }

    EVPPI_QMC <- sum(EVPPI[,1])/M
    EVPPI_var_QMC <- sum(EVPPI[,2])/M - EVPPI_QMC^2
    var <- EVPPI_var_QMC/M
    toc()
    cat(sprintf("QMC: EVPPI (CQ) = %.4f +/- %.4f, var = %.4f, N=%.1e, n=%d, M=%d. \n\n ",
                EVPPI_QMC,3*sqrt(var),var,N,n,M))
    return(list(EVPPI_QMC=EVPPI_QMC,var=var))
}
```

Then we compare with two methods:

```r
mc_EVPPI_CQ(1e5,100)
```

```
## 24.835 sec elapsed
## EVPPI (CQ) = 286.8287 +/- 11.8339, var = 15.5602, N=1.0e+05.
##
##

## $EVPPI
## [1] 286.8287
##
## $var
## [1] 15.56023
```

```r
qmc_EVPPI_CQ(1e4,100,16)
```

```
## 37.466 sec elapsed
## QMC: EVPPI (CQ) = 285.8893 +/- 0.8374, var = 0.0779, N=1.0e+04, n=100, M=16.
##
##

## $EVPPI_QMC
## [1] 285.8893
##
## $var
## [1] 0.07790653
```

EVPPI for lor of CBT

Here is the code for **mc_EVPPI_lor2.R** using standard MC:

```r
# EVPPI (lor2) = 35.7332 +/- 0.4270, var = 0.0203, N=1.0e+06. 341.111 sec

mc_EVPPI_lor2 <- function(N,n){
    source("net_benefit.R")
    pkg <- c("Matrix","MASS","boot","tictoc","randtoolbox")
    lapply(pkg, library, character.only = TRUE)

    tic()
    set.seed(666)

    EVPPI_sum <- matrix(0,3,2)

    mean_rec <- c(0.99,1.33)
    var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
    mean_rel <- c(-1.48,-0.4)
    var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)

    for(N1 in seq(1,N,by=1e4)){
        N2 <- min(1e4, N-N1+1)

        lor_rec2 <- rnorm(N2,mean_rec[1],sqrt(var_rec[1,1]))
        lor_rel2 <- rnorm(N2,mean_rel[1],sqrt(var_rel[1,1]))

        lor_rec2 <- c(t(replicate(n,lor_rec2)))
```

```r
        lor_rel2 <- c(t(replicate(n,lor_rel2)))

        mean_rec3 <- mean_rec[2] + var_rec[1,2]/var_rec[1,1]*(lor_rec2 - mean_rec[1])
        var_rec3 <- var_rec[2,2] - var_rec[1,2]*var_rec[2,1]/var_rec[1,1]
        mean_rel3 <- mean_rel[2] + var_rel[1,2]/var_rel[1,1]*(lor_rel2 - mean_rel[1])
        var_rel3 <- var_rel[2,2] - var_rel[1,2]*var_rel[2,1]/var_rel[1,1]

        C_t <- c(0,300,30)
        lambda <- 2e4

        C_rec <- matrix(rnorm(n*N2,1000,50),n,N2)
        C_rel <- matrix(rnorm(n*N2,2000,100),n,N2)
        C_norec <- matrix(rnorm(n*N2,2500,125),n,N2)
        Q_rec <- matrix(rnorm(n*N2,26,2),n,N2)
        Q_rel  <- matrix(rnorm(n*N2,23,3),n,N2)
        Q_norec <- matrix(rnorm(n*N2,20,4),n,N2)

        lor_rec3 <-  rnorm(n*N2,0,sqrt(var_rec3))+mean_rec3
        lor_rel3 <-  rnorm(n*N2,0,sqrt(var_rel3))+mean_rel3

        P_rec <- matrix(0,N2*n,3)
        P_rel <- matrix(0,N2*n,3)
        P_rec[,1] <- rbeta(N2*n,6,200)
        P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec2)
        P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec3)

        P_rel[,1] <- rbeta(N2*n,2,100)
        P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel2)
        P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel3)

        for(i in seq(1,N2)){
            ind <- seq((i-1)*n+1,i*n)

            Result <- net_benefit(lambda,P_rec[ind,],P_rel[ind,],C_rec[,i],C_rel[,i],
                                  C_norec[,i],Q_rec[,i],Q_rel[,i],Q_norec[,i],C_t)
            NB <- colMeans(Result$NB)
            NB <- max(NB) - NB;
            EVPPI_sum[,1] <- EVPPI_sum[,1] + NB
            EVPPI_sum[,2] <- EVPPI_sum[,2] + NB^2
        }
    }

    EVPPI <- min(EVPPI_sum[,1]/N)
    ind <- which.min(EVPPI_sum[,1])
    var <- (EVPPI_sum[ind,2]/N-EVPPI^2)/N
    toc()
    cat(sprintf("EVPPI (lor2) = %.4f +/- %.4f, var = %.4f, N=%.1e. \n\n ",
                EVPPI,3*sqrt(var),var,N))
    return(list(EVPPI=EVPPI,var=var))
}
```

EVPPI for lor2 using QMC is implemented in **qmc_EVPPI_lor2.R**:

```
qmc_EVPPI_lor2 <- function(N,n,M){

    pkg <- c("Matrix","MASS","boot","tictoc","randtoolbox")
    lapply(pkg, library, character.only = TRUE)
    source("sfunc.R")
    source("net_benefit.R")

    tic()
    set.seed(666)

    mean_rec <- c(0.99,1.33)
    var_rec <- matrix(c(0.22,0.15,0.15,0.2),2,2)
    mean_rel <- c(-1.48,-0.4)
    var_rel <- matrix(c(0.14,0.05,0.05,0.11),2,2)

    EVPPI <- matrix(0,M,2)

    for(m in seq(1,M)){
        EVPPI_sum <- rep(0,3)
        for(N1 in seq(1,N,by=1e4)){
            N2 <- min(1e4, N-N1+1)

            U <- sobol(N2, dim = 2, init = TRUE, scrambling = 1, seed = 666,
                       normal = FALSE)
            lor_rec2 <- qnorm(U[,1],mean_rec[1],sqrt(var_rec[1,1]))
            lor_rel2 <- qnorm(U[,2],mean_rel[1],sqrt(var_rel[1,1]))
            lor_rec2 <- c(t(replicate(n,lor_rec2)))
            lor_rel2 <- c(t(replicate(n,lor_rel2)))

            mean_rec3 <- mean_rec[2] + var_rec[1,2]/var_rec[1,1]*(lor_rec2 - mean_rec[1])
            var_rec3 <- var_rec[2,2] - var_rec[1,2]*var_rec[2,1]/var_rec[1,1]
            mean_rel3 <- mean_rel[2] + var_rel[1,2]/var_rel[1,1]*(lor_rel2 - mean_rel[1])
            var_rel3 <- var_rel[2,2] - var_rel[1,2]*var_rel[2,1]/var_rel[1,1]

            C_t <- c(0,300,30)
            lambda <- 2e4

            C_rec <- matrix(rnorm(n*N2,1000,50),n,N2)
            C_rel <- matrix(rnorm(n*N2,2000,100),n,N2)
            C_norec <- matrix(rnorm(n*N2,2500,125),n,N2)
            Q_rec <- matrix(rnorm(n*N2,26,2),n,N2)
            Q_rel  <- matrix(rnorm(n*N2,23,3),n,N2)
            Q_norec <- matrix(rnorm(n*N2,20,4),n,N2)

            lor_rec3 <-  rnorm(n*N2,0,sqrt(var_rec3))+mean_rec3
            lor_rel3 <-  rnorm(n*N2,0,sqrt(var_rel3))+mean_rel3

            P_rec <- matrix(0,N2*n,3)
            P_rel <- matrix(0,N2*n,3)
            P_rec[,1] <- rbeta(N2*n,6,200)
            P_rec[,2] <- inv.logit(logit(P_rec[,1])+lor_rec2)
            P_rec[,3] <- inv.logit(logit(P_rec[,1])+lor_rec3)
```

```
            P_rel[,1] <- rbeta(N2*n,2,100)
            P_rel[,2] <- inv.logit(logit(P_rel[,1])+lor_rel2)
            P_rel[,3] <- inv.logit(logit(P_rel[,1])+lor_rel3)

            for(i in seq(1,N2)){
                ind <- seq((i-1)*n+1,i*n)

                Result <-net_benefit(lambda,P_rec[ind,],P_rel[ind,],C_rec[,i],C_rel[,i],
                                     C_norec[,i],Q_rec[,i],Q_rel[,i],Q_norec[,i],C_t)
                NB <- colMeans(Result$NB)
                NB <- max(NB) - NB
                EVPPI_sum <- EVPPI_sum + NB
            }
        }
        EVPPI[m,1] <- min(EVPPI_sum/N)
        EVPPI[m,2] <- EVPPI[m,1]^2
    }

    EVPPI_QMC <- sum(EVPPI[,1])/M
    EVPPI_var_QMC <- sum(EVPPI[,2])/M - EVPPI_QMC^2
    var <- EVPPI_var_QMC/M
    toc()
    cat(sprintf("QMC: EVPPI (lor2) = %.4f +/- %.4f, var = %.4f, N=%.1e, n=%d, M=%d.\n\n ",
                EVPPI_QMC,3*sqrt(var),var,N,n,M))
    return(list(EVPPI_QMC=EVPPI_QMC,var=var))
}
```

Here **net_benefit.R** calculates the net benefit for each sample:

```
net_benefit <- function(lambda,P_rec,P_rel,C_rec,C_rel,C_norec,Q_rec,Q_rel,Q_norec,C_t){
  QALY <- P_rec*(1-P_rel)*Q_rec + P_rec*P_rel*Q_rel + (1-P_rec)*Q_norec
  Cost <- P_rec*(1-P_rel)*C_rec + P_rec*P_rel*C_rel + (1-P_rec)*C_norec

  Cost[,2] <- Cost[,2] +C_t[2]
  Cost[,3] <- Cost[,3] +C_t[3]

  NB <- lambda*QALY - Cost

  return( list(NB=NB,QALY=QALY,COST=Cost))
}
```

Then we compare with two methods:

```
mc_EVPPI_lor2(1e5,1000)
```

```
## 219.324 sec elapsed
## EVPPI (lor2) = 6.4652 +/- 1.1050, var = 0.1357, N=1.0e+05.
##
##

## $EVPPI
## [1] 6.465223
##
## $var
## [1] 0.1356759
```

```r
qmc_EVPPI_lor2(1e4,1000,16)
```

```
## 330.004 sec elapsed
## QMC: EVPPI (lor2) = 6.6383 +/- 0.1287, var = 0.0018, N=1.0e+04, n=1000, M=16.
##
##

## $EVPPI_QMC
## [1] 6.638267
##
## $var
## [1] 0.001841277
```

Summary

QMC works very well for all the calculations in this toy model.