Mathematical Model for the Cost-effectiveness of Noval Oral AntiCoagulants in Atrial Fibrillation

Wei Fang

August 27, 2018

1 Model Input

Six treatments are **Warfarin, no treatment, Apixaban, Dabigatran, Edoxaban, Rivaroxaban**, with possibilities to switch to other treatments.

Seventeen health states in the Markov model:

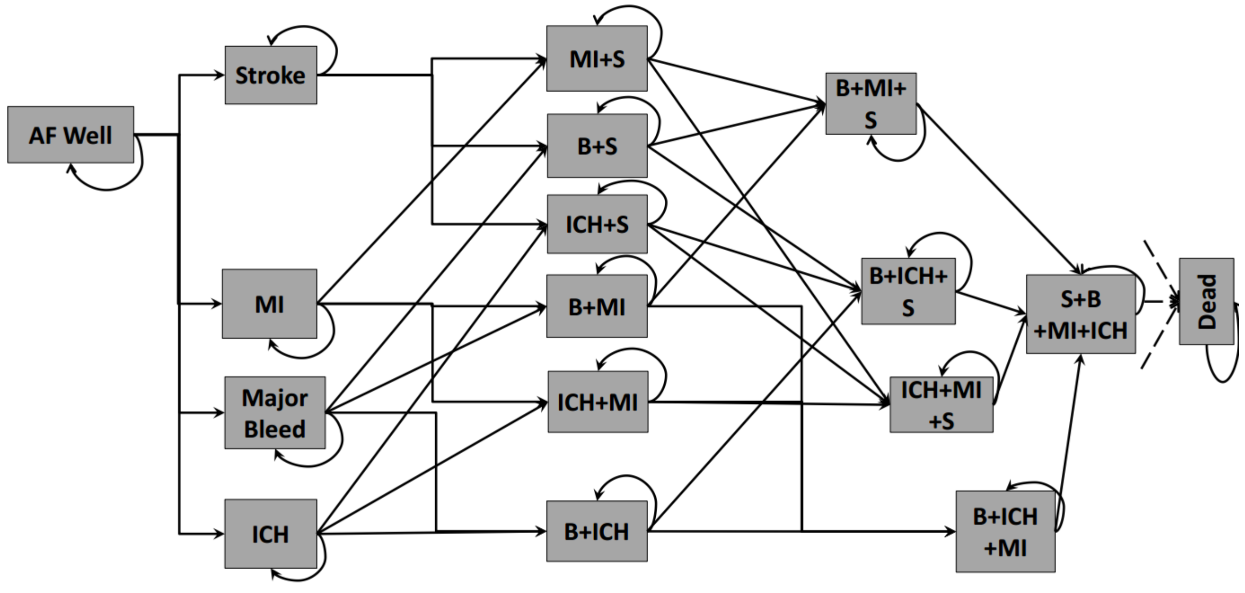

Figure 1: Health states in the Markov model for AF

Dead state is same for all treatments and other sates are different for different treatments. Therefore there are $16 * 6 + 1 = 97$ states in the transition matrix.

```
> state.names
 [1] "Coumarin (INR 2-3) Well"         "Coumarin (INR 2-3)  B "
 [3] "Coumarin (INR 2-3)  I"           "Coumarin (INR 2-3)  M "
 [5] "Coumarin (INR 2-3)  S"           "Coumarin (INR 2-3)  B+I"
 [7] "Coumarin (INR 2-3)  B+M"         "Coumarin (INR 2-3)  B+S"
 [9] "Coumarin (INR 2-3)  I+M"         "Coumarin (INR 2-3)  I+S"
[11] "Coumarin (INR 2-3)  M+S"         "Coumarin (INR 2-3)  B+I+M"
[13] "Coumarin (INR 2-3)  B+I+S"       "Coumarin (INR 2-3)  B+M+S"
[15] "Coumarin (INR 2-3)  I+M+S"       "Coumarin (INR 2-3)  B+I+M+S"
[17] "Apixaban (5mg bd) Well"          "Apixaban (5mg bd)  B "
[19] "Apixaban (5mg bd)  I "           "Apixaban (5mg bd)  M "
[21] "Apixaban (5mg bd)  S "           "Apixaban (5mg bd)  B + I "
[23] "Apixaban (5mg bd)  B + M "       "Apixaban (5mg bd)  B + S "
[25] "Apixaban (5mg bd)  I + M "       "Apixaban (5mg bd)  I + S "
[27] "Apixaban (5mg bd)  M + S "       "Apixaban (5mg bd)  B + I + M "
[29] "Apixaban (5mg bd)  B + I + S "   "Apixaban (5mg bd)  B + M + S "
[31] "Apixaban (5mg bd)  I + M + S "   "Apixaban (5mg bd)  B + I + M + S "
[33] "Dabigatran (150mg bd) Well"      "Dabigatran (150mg bd)  B "
[35] "Dabigatran (150mg bd)  I "       "Dabigatran (150mg bd)  M "
[37] "Dabigatran (150mg bd)  S "       "Dabigatran (150mg bd)  B + I "
[39] "Dabigatran (150mg bd)  B + M "   "Dabigatran (150mg bd)  B + S "
[41] "Dabigatran (150mg bd)  I + M "   "Dabigatran (150mg bd)  I + S "
[43] "Dabigatran (150mg bd)  M + S "   "Dabigatran (150mg bd)  B + I + M "
[45] "Dabigatran (150mg bd)  B + I + S""Dabigatran (150mg bd)  B + M + S "
[47] "Dabigatran (150mg bd)  I + M + S""Dabigatran (150mg bd)  B+I+M+S"
[49] "Edoxaban (60mg od) Well"         "Edoxaban (60mg od)  B "
[51] "Edoxaban (60mg od)  I "          "Edoxaban (60mg od)  M "
[53] "Edoxaban (60mg od)  S "          "Edoxaban (60mg od)  B + I "
[55] "Edoxaban (60mg od)  B + M "      "Edoxaban (60mg od)  B + S "
[57] "Edoxaban (60mg od)  I + M "      "Edoxaban (60mg od)  I + S "
[59] "Edoxaban (60mg od)  M + S "      "Edoxaban (60mg od)  B + I + M "
[61] "Edoxaban (60mg od)  B + I + S "  "Edoxaban (60mg od)  B + M + S "
[63] "Edoxaban (60mg od)  I + M + S "  "Edoxaban (60mg od)  B+I+M+S"
[65] "Rivaroxaban (20mg od) Well"      "Rivaroxaban (20mg od)  B "
[67] "Rivaroxaban (20mg od)  I "       "Rivaroxaban (20mg od)  M "
[69] "Rivaroxaban (20mg od)  S "       "Rivaroxaban (20mg od)  B + I "
[71] "Rivaroxaban (20mg od)  B + M "   "Rivaroxaban (20mg od)  B + S "
[73] "Rivaroxaban (20mg od)  I + M "   "Rivaroxaban (20mg od)  I + S "
[75] "Rivaroxaban (20mg od)  M + S "   "Rivaroxaban (20mg od)  B + I + M "
[77] "Rivaroxaban (20mg od)  B + I + S""Rivaroxaban (20mg od)  B + M + S "
[79] "Rivaroxaban (20mg od)  I + M + S""Rivaroxaban (20mg od)  B+I+M+S"
[81] "No treatment Well"               "No treatment  B "
[83] "No treatment  I "                "No treatment  M "
[85] "No treatment  S "                "No treatment  B + I "
[87] "No treatment  B + M "            "No treatment  B + S "
[89] "No treatment  I + M "            "No treatment  I + S "
[91] "No treatment  M + S "            "No treatment  B + I + M "
[93] "No treatment  B + I + S "        "No treatment  B + M + S "
[95] "No treatment  I + M + S "        "No treatment  B + I + M + S "
[97] "Dead"
```

1.1 Costs

1.1.1 Event Costs

Eight events including four acute events, two transient events, death and stay (nothing happens).

```
> event.names
[1] "MI"                  "Ischemic stroke"      "Death (all causes)"
[4] "Transient ischemic attack (TIA)" "Clinically relevant bleeding" "SE"
[7] "ICH"                  "Stay"
```

Costs of different events follow different distribution:

$$\begin{aligned}
\text{Cost}_{MI} &\sim Uniform\,(2415.24, 7245.72) \\
\text{Cost}_{S} &\sim Normal\,(11626, 1325) \\
\text{Cost}_{D} &= 0 \\
\text{Cost}_{TIA} &\sim Uniform\,(532, 1596) \\
\text{Cost}_{B} &\sim Uniform\,(875.75, 2627.25) \\
\text{Cost}_{SE} &\sim Uniform\,(1186.5, 3559.5) \\
\text{Cost}_{ICH} &\sim Normal\,(11453, 3350) \\
\text{Cost}_{Stay} &= 0.
\end{aligned}$$

1.1.2 Treatment cost

Costs of different treatments (per 3 month cycle) are

$$\begin{aligned}
\text{Wafarin, Coumarin (INR 2-3)} &\sim Uniform(52.57, 157.70) \\
\text{Apixaban} &= 200.42 \\
\text{Dabigatran} &= 200.44 \\
\text{Edoxaban} &= 200.44 \\
\text{Rivaroxaban} &= 191.63 \\
\text{No treatment} &= 0.
\end{aligned}$$

1.1.3 Health State cost

Post-stroke and Post-ICH management annual cost follow $Normal(3613, 363.1483)$. Divided by 4 to get costs per 3 months. The cost per cycle for patients with a history of both stroke and ICH was assumed the same as the cost for a history of only higher one of these states.

1.2 Transition Matrix

1.2.1 Switch Probability

Due to the higher risk of MI on dabigatran, patients on dabigatran who experience an MI are assumed to always switch to warfarin. Patients who experience an ICH, whether on warfarin or a NOAC, will always switch to no treatment. Following a stroke or bleed, patients switch may switch from a NOAC to warfarin or from warfarin to no treatment with a probability sampled from a beta(0.3, 0.7) distribution (mean 0.30, 95% CrI 0.00-1.00). Patients experiencing SE and TIA make the same transition with a lower probability sampled from beta(0.1, 0.9) distribution (mean 0.10, 95% CrI 0.00-1.00).

$$\begin{aligned}
\text{Switch}_{MI}(Dabigatran \rightarrow Warfarin) &= 1 \\
\text{Switch}_{ICH}(NOAC \rightarrow NoT) &= 1 \\
\text{Switch}_{ICH}(Warfarin \rightarrow NoT) &= 1 \\
\text{Switch}_{S,B}(NOAC \rightarrow Warfarin)\ , \text{Switch}_{S,B}(Warfarin \rightarrow NoT) &\sim Beta(0.3, 0.7) \qquad (2) \\
\text{Switch}_{SE,TIA}(NOAC \rightarrow Warfarin)\ , \text{Switch}_{SE,TIA}(Warfarin \rightarrow NoT) &\sim Beta(0.1, 0.9) \qquad (2)
\end{aligned}$$

1.2.2 Effects of previous events

The effect of prior events on future risks was assumed to be multiplicative so a history of, say, the log hazard ratio of future stroke conditional on history of stroke and ICH is given by adding the log hazard ratios for each of these events. All the log hazard ratios follow normal distributions. Except MI on ICH, all the events will increase the hazard ratios of the events.

Previous on	**MI**	**Stroke**	**Death**	**TIA**	**Bleed**	**SE**	**ICH**
Bleed	1	0.2776317	0.2770719	0.3074847	1.199965	0.3074847	1.264127
		0.04439356	0.1499456	0.03758284	0.04145891	0.03758284	0.08231102
ICH	1	0.5766134	0.2770719	0.5988365	1.081805	0.5988365	2.322388
		0.06718111	0.1499456	0.05880706	0.07064388	0.05880706	0.08949929
MI	1	0.2151114	0.02839948	0.2546422	0.2151114	0.2546422	-0.0618754
		0.03269775	0.1775648	0.02771271	0.04090374	0.02771271	0.09229338
Stroke	1	1.386294	0.2770719	1.283708	0.3293037	1.283708	0.4946962
		0.02808957	0.1499456	0.02404401	0.04584016	0.02404401	0.085047

Table 1: Log hazard ratios of effects of previous events

1.2.3 Hazard ratio from MCMC

The hazard ratios of seven events (except for "Stay" which is determined by others) for Wafarin is stored in **bugs.baseline** file. The hazard ratios relative to Wafarin of the seven events are stored in **bugs.loghr** file. All are generated by MCMC and determine the probabilities of individual event for AF well states. The effects of previous events simulated above then determine the probability of joint events. The transition probability can be determined by conditional probability.

1.3 Utility

1.3.1 Utility factor for different age

Proportional utility decrements from Kind et al. 1999. Beta distributions for each age range estimated by Pete Bryden. Use weighted average of males (60%) and females (40%) to represent AF population. Ratio of each category to 70 year olds is the proportional decrement/increment.

Age	Male	Female
35	Beta(656.7,64.95)	Beta(1006.6,99.5)
45	Beta(341.41,65.03)	Beta(544.1,96.02)
55	Beta(330.43,93.2)	Beta(526.59,123.52)
65	Beta(388.47,109.57)	Beta(551.74,155.62)
75	Beta(191.17,63.72)	Beta(406.37,165.98)
85	Beta(191.17,63.72)	Beta(406.37,165.98)
95	Beta(191.17,63.72)	Beta(406.37,165.98)
105	Beta(191.17,63.72)	Beta(406.37,165.98)

Table 2: Utility for different age and different sex

1.3.2 Health state utilities

From sources identified in Bayer Table 49), these are combined proportionally. All utilities are later divided by 4 to make them 3-monthly. Proportion to AF well will be used. The utility for post bleed was assumed to the same as for post stroke. Utilities for chronic health states are assumed to be multiplicative. For example, the utility of a patient who has experienced both an ischaemic stroke and a myocardial infarction will be the product of the two utility scores. The state utilities were assumed to reduce with age by factors estimated relative to a reference age (65-75) generated above.

$$\begin{aligned}
\text{AF well} &\sim Normal(0.779, 0.0045)\\
\text{Post Stroke} &\sim Normal(0.69, 0.0205)\\
\text{Post ICH} &\sim Beta(3.941, 1.385)\\
\text{Post MI} &\sim Normal(0.718, 0.0163)\\
\text{Post Bleed} &= \text{Post Stroke}
\end{aligned}$$

1.3.3 Event disutilities

The disutilities for different events are

$$\begin{aligned}
\text{DisU}_{TIA} &\sim Uniform(-0.1965, -0.0655)\\
\text{DisU}_{SE} &\sim Uniform(-0.1965, -0.0655)\\
\text{DisU}_{S} &\sim Uniform(-0.885, -0.295)\\
\text{DisU}_{ICH} &\sim Normal(0.6, 0.064) - \text{AF Well}\\
\text{DisU}_{MI} &\sim Normal(0.683, 0.0156) - \text{AF Well}\\
\text{DisU}_{B} &\sim Normal(-0.03, 0.001531).
\end{aligned}$$

2 EVPI

In this section, we calculate the EVPI value using nested simulation. For MCMC samples, we use random selection with replacement. Here is the function in **EVPI_std.R**:

```
EVPI_std<-function(N)
{
  # this function generate all the random samples for the EVPI calculation
  # 34 normal, 8 uniform, 15 beta, MCMC 7 (baseline), MCMC 28 (loghr),
     MCMC 7 (no treatment)
  # then pass all the samples to new age.independent.generate.
     probabilities_2 function to get
  # the samples of parameters and then calculate the net benefit function
  # N is the number of samples
  sum1 <- rep(0, 2)

  for(N1 in seq(1, N, by=1000)) {
    NN <- min(1000, N-N1+1) # we calculate 1000 samples each time

  # Event cost
  # Dimension 6, first 4 are normal and last 2 are uniform, for the
     facility of QMC version
```

```r
Event.cost.samples = matrix(NA,NN,6)
for(m in 1:4){
Event.cost.samples[,m] = runif(NN)
}
Event.cost.samples[,5] = rnorm(NN)
Event.cost.samples[,6] = rnorm(NN)

# Treatment cost
# Dimension 1, uniform
Treatment.cost.samples = matrix(NA,NN,1)
Treatment.cost.samples[,1] = runif(NN)

# Healthstate cost
# Dimension 2, normal
Health.cost.samples = matrix(NA,NN,2)
Health.cost.samples[,1] = rnorm(NN)
Health.cost.samples[,2] = rnorm(NN)


# Switch Probability
# Dimension 4, first 3 beta (0.1,0.9), last 1 beta (0.3,0.7)
Switch.probability.samples = matrix(NA,NN,4)
Switch.probability.samples[,1] = rbeta(NN,0.1,0.9)
Switch.probability.samples[,2] = rbeta(NN,0.1,0.9)
Switch.probability.samples[,3] = rbeta(NN,0.1,0.9)
Switch.probability.samples[,4] = rbeta(NN,0.3,0.7)


# Effects of previous events
# Dimension 24, all normal
Effect.history.samples = matrix(NA,NN,24)
for(m in 1:24){
  Effect.history.samples[,m] = rnorm(NN)
}

# Random selection of MCMC samples
# Dimension 3, selcet 3 set of MCMC samples
MCMC.selection = sample(1:29999, NN, replace = TRUE)
MCMC.baseline.samples = bugs.baseline[MCMC.selection,]
MCMC.selection = sample(1:29999, NN, replace = TRUE)
MCMC.loghr.samples = bugs.loghr[MCMC.selection,]
MCMC.selection = sample(1:59999, NN, replace = TRUE)
MCMC.noTreatment.samples = hr.no.treatment[MCMC.selection,]

# Utility factor for different age
# Dimension 4, only for 65 and 75, all beta
Utility.age.samples = matrix(NA,NN,4)
Utility.age.samples[,1] = rbeta(NN,388.47,109.57)
Utility.age.samples[,2] = rbeta(NN,551.74,155.62)
Utility.age.samples[,3] = rbeta(NN,191.17,63.72)
Utility.age.samples[,4] = rbeta(NN,406.37,165.98)
```

```
  # Health state utilities
  # Dimension 4, first 3 normal, last 1 beta
  Utility.state.samples = matrix(NA,NN,4)
  Utility.state.samples[,1] = rnorm(NN)
  Utility.state.samples[,2] = rnorm(NN)
  Utility.state.samples[,3] = rnorm(NN)
  Utility.state.samples[,4] = rbeta(NN,3.941,1.385)

  # Events utilities
  # Dimension 6, first 3 uniform, last 3 normal
  Utility.event.samples = matrix(NA,NN,6)
  Utility.event.samples[,1] = runif(NN)
  Utility.event.samples[,2] = runif(NN)
  Utility.event.samples[,3] = runif(NN)
  Utility.event.samples[,4] = rnorm(NN)
  Utility.event.samples[,5] = rnorm(NN)
  Utility.event.samples[,6] = rnorm(NN)

  # New function to generate the probabilities in the Markov model
  age.independent.samples<-age.independent.generate.probabilities_2(NN,
                      Event.cost.samples,Treatment.cost.samples,Health.
                         cost.samples,
                      Switch.probability.samples,Effect.history.samples,
                      MCMC.baseline.samples,MCMC.loghr.samples,MCMC.
                         noTreatment.samples,
                      Utility.age.samples,Utility.state.samples,Utility.
                         event.samples
    )

  # Calculate the net Benefits of each treatment for each sample
  model.outputs<-noac.net.benefit(n.samples=NN,n.cycles=n.cycles,initial.
     age=initial.age,lambdas=lambdas,age.independent.samples=age.
     independent.samples)

  NetB = model.outputs$NB
  # find the optimal treatment without any information: 2
  Ind = which.is.max(colMeans(NetB))
  EVPI_sample = apply(NetB,1,max)-NetB[,Ind,]

  sum1[1] = sum1[1]+sum(EVPI_sample)
  sum1[2] = sum1[2]+sum(EVPI_sample^2)

  }
  EVPI = sum1[1]/N
  std_EVPI = sqrt(sum1[2]/N-EVPI^2)/sqrt(N)

  return(c(EVPI,std_EVPI))
}
```

The optimal treatment without any information is 2 **Apixaban**.

In order to accelerate the calculation, we use parallel package to calculate more samples in the first part of **Parellel_Core.R**:

```
library(foreach)
library(doParallel)
library(parallel)
numCores <- detectCores()
cl <- makeCluster(numCores)
registerDoParallel(cl)
# the above code prepare the parallel cores and clusters

N = 128000*8

# for each calculation of each core, we only calculate 1000 samples for
    efficiency
NN = 1000
inputs <- 1:(N/NN)

# This loop does the parallel calculation
results <- foreach(i=inputs) %dopar% {
  EVPI_std_p(NN)
}

# Due to some function unavailable in the parallel loop, we do statitics
    outside
sum1 = rep(0,2)
for(i in inputs){
  NetB = matrix(unlist(results[i]),NN,5)
  Ind = which.is.max(colMeans(NetB))
  EVPI_sample = apply(NetB,1,max)-NetB[,Ind]

  sum1[1] = sum1[1]+sum(EVPI_sample)
  sum1[2] = sum1[2]+sum(EVPI_sample^2)
}

EVPI = sum1[1]/N
EVPI_std = sqrt(sum1[2]/N-EVPI^2)/sqrt(N)
```

Each 1000 samples calculation in one core takes approximately 12 min to accomplish all the calculation. The final EVPI value we calculate using 2880000 samples is 416.7916, with standard deviation 0.711 and the confidence interval $[414.657, 418.926]$. (105000s)

3 EVPPI

Here is the code in the second part of **Parellel_Core.R**

```
N = 1024 # number of out samples
M = 32 # number of inner samples
```

```r
NN = 1024 # number of samples calculated in one core each time
inputs <- 1:(N*M/NN)
# This loop does the parallel calculation
results1 <- foreach(i=inputs) %dopar% {
  EVPPI_Cost_EventCost_std_p(M,NN/M)
}

# Use multivariate normal distribution to approximate the covariance
    structure between MCMC samples
# firts calculate the mean and variance of the Multivariate normal
test = bugs.loghr[,8:35]
sigma_loghr = cov(test)
mu_loghr = colMeans(test)
results15 <- foreach(i=inputs,.packages='MASS') %dopar%{
  EVPPI_NOAC_std_p(M,NN/M)
}

# The following code can calculate the the EVPPI value for any parameters
# First summerize the parallel results to a new matrix
NetB = matrix(NA,M*N,5)
for(i in inputs){
  NetB[((i-1)*NN+1):(i*NN),] = matrix(unlist(results[i]),NN,5)
}

# Calculate net benefit of the optimal decision for each sample
NetB_max = apply(NetB,1,max)
NetB_max_sample = apply(matrix(NetB_max,N,M,byrow=TRUE),1,mean)

# find the optimal decision without any information and calculate the Net
    benefit
Ind = which.is.max(colMeans(NetB))
NetB_low_sample = apply(matrix(NetB[,Ind],N,M,byrow=TRUE),1,mean)

# find the optimal decision with partial information and calculate the net
     benefit
NetB_mid = matrix(NA,N,5)
for(i in 1:5){
  NetB_mid[,i] = apply(matrix(NetB[,i],N,M,byrow=TRUE),1,mean)
}
NetB_mid_sample = apply(NetB_mid,1,max)

# Construct the samples for both DIFF and EVPPI itself
DIFF_sample = NetB_max_sample- NetB_mid_sample
EVPPI_sample = NetB_mid_sample- NetB_low_sample

# Calculate the estimated value and Monte Carlo standard deviation
DIFF = mean(DIFF_sample)
DIFF_std = sqrt(var(DIFF_sample)/N)
EVPPI = mean(EVPPI_sample)
EVPPI_std = sqrt(var(EVPPI_sample)/N)
```

The EVPPI values we calculated using standard Monte Carlo method are summerized in table 8 and 9 with different number of inner samples $M = 32$ and $M = 128$ with $N = 1024$.

	DIFF	DIFF sd	EVPPI	EVPPI sd	Bias
All Cost	426.74	6.92	0	0	9.91
Event Cost	407.98	6.59	0	0	8.12
Treatment Cost	422.45	7.05	0.72 (2)	0.51	1.31
State Cost	414.99	6.56	0.28 (3)	0.19	9.44
Switch Probability	421.11	13.08	0	0	0.26
Previous Effect	418.78	7.16	0.47 (3)	0.46	5.34
All MCMC	**67.94**	6.82	354.22	23.01	4.22
Baseline MCMC	420.44	8.87	4.64	1.59	6.66
Loghr MCMC	**125.64**	5.96	264.49	18.33	1.83
No Treatment MCMC	413.82	6.31	0.58 (1)	0.58	7.29
Loghr 8:21 MCMC	**215.01**	8.01	196.93	15.76	11.36
Loghr Complex	**156.44**	9.49	248.37	25.49	5.98
All Utility	426.36	7.21	0.10 (1)	0.10	11.26
Age Utility	423.54	6.76	0.51 (2)	0.44	3.40
State Utility	430.52	6.90	0.32 (3)	0.18	2.34
Event Utility	416.18	6.78	0.23 (1)	0.23	3.45

Table 3: EVPPI value 1024*32

	DIFF	DIFF sd	EVPPI	EVPPI sd	Bias
All Cost	415.41	3.42	0	0	2.02
Event Cost	412.44	3.48	0	0	1.08
Treatment Cost	420.13	3.39	0	0	0.03
State Cost	417.69	3.41	0	0	1.12
Switch Probability	421.60	11.74	0	0	0.01
Previous Effect	416.23	3.34	0	0	0.34
All MCMC	**67.88**	5.99	360.96	22.76	0.89
Baseline MCMC	410.97	6.46	0.87	0.54	0.57
Loghr MCMC	146.61	6.84	307.00	21.25	0.17
No Treatment MCMC	416.83	4.30	0	0	1.05
Loghr 8:21 MCMC	**218.01**	7.25	184.51	16.01	3.85
Loghr Complex	**124.56**	7.23	256.34	27.13	1.74
All Utility	413.46	3.87	0	0	1.76
Age Utility	416.73	3.38	0	0	0.20
State Utility	411.97	3.49	0	0	0.08
Event Utility	411.56	4.02	0	0	0.17

Table 4: EVPPI value 1024*128

4 MLMC

The MLMC calculation are mainly done in the **MLMC_Core.R** through the main driver function together with the level functions for different parameters.

```
require(ggplot2)
require(grid)
require(Rcpp)
require(tictoc)
require(doRNG)

source("/home/fangw/Dropbox/EVPPI/Bristol/Wei/R_MLMC/mlmc.R") #MLMC code
    to calculate the value to some accuracy
source("/home/fangw/Dropbox/EVPPI/Bristol/Wei/R_MLMC/mlmc.test.R") #test
    function of the convergence rates and output the MLMC results
source("/home/fangw/Dropbox/EVPPI/Bristol/Wei/R_MLMC/plot.mlmc.test.R") #
    function to plot the result
source("/home/fangw/Dropbox/EVPPI/Bristol/Wei/R_MLMC/multiplot.R") #this
    is the function from ggplot

source("EVPI_l_p.R")
source("EVPI_std_p.R")
source("EVPPI_l_p.R")
source("EVPPI_l_p2.R")
set.seed(666) # Set random seed to ensure the same output
# without this MLMC will give different output each time
tic()
tst <- mlmc.test(EVPPI_l_p, M=2, N=128,
                 L=5, N0=128,
                 eps.v=c(60,30,15,7),
                 Lmin=2, Lmax=10)
toc()
set.seed(666)
tic()
tst2 <- mlmc.test(EVPPI_l_p2, M=2, N=128,
                 L=3, N0=128,
                 eps.v=c(20,10,5,2,1),
                 Lmin=2, Lmax=10,option=1,l0=0)
#The option provides the two alternative estimators: 0 for EVPPI, 1 for
    DIFF (default)
# The l0 define initial level for EVPPI estimator
toc()

# plot the MLMC results
plot(tst2,which=c("var", "mean", "Nl", "cost"),cols=2)
```

4.1 EVPPI for NOAC simple trial

Here we need to use level function **EVPPI_NOAC_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
-------------------------------------------------------------------------


 0   8.6760e+01  8.6760e+01  8.0402e+04  8.0402e+04  0.0000e+00  0.00e+00
 1   5.2474e+01  1.3241e+02  1.5605e+04  5.5679e+04  1.5059e+01  3.99e-02
 2   3.3453e+01  1.9348e+02  8.0020e+03  5.8800e+04  1.7461e+01  1.83e-01
 3   1.9507e+01  2.2443e+02  4.1489e+03  7.8523e+04  1.8711e+01  7.34e-02

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 0.778159  (exponent for (MLMC weak convergence)
 beta  in 0.947642  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)

*****************************
*** MLMC complexity tests ***
*****************************

  eps       value   mlmc_cost   std_cost  savings     N_l
-----------------------------------------------------------
20.0000  2.0103e+02  8.966e+03  1.675e+04     1.87       765       357
         229        105        44        17
10.0000  2.0814e+02  3.872e+04  1.340e+05     3.46      3909      1733
         862        335        134       58        29
5.0000  2.1147e+02  1.843e+05  1.072e+06     5.82     16794      8228
        3719       1489       606       266       107       55
2.0000  2.1875e+02  1.612e+06  2.680e+07    16.63    126027     62525
   27875     12290      4682      1855       724       270       253
         127
1.0000  2.2010e+02  6.863e+06  2.144e+08    31.24    520757    253928
   116046     48289     19047      8511      3574      1499       558
         420        192
```

The DIFF value we calculate is 220.10 with confidence interval [218.14, 222.06]. Therefore, the EVPPI value we calculate is 196.6916 with confidence interval [194.2867, 199.0965] (215544s)

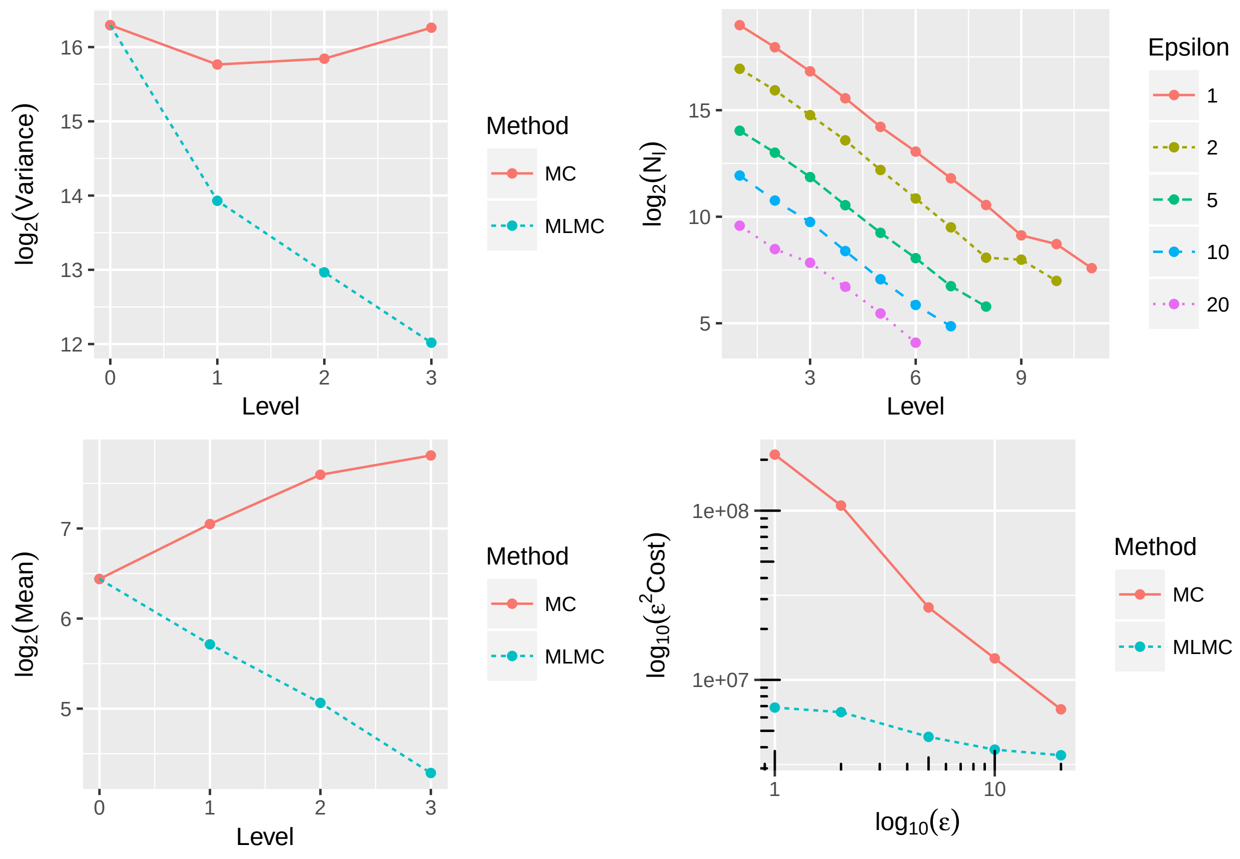

Figure 2: EVPPI for NOAC simple

4.2 EVPPI for NOAC complex trial

Here we need to use level function **EVPPI_NOAC_Complex_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)    var(Pf-Pc)    var(Pf)     kurtosis     check
---------------------------------------------------------------------------

 0   6.1126e+01  6.1126e+01  4.1669e+04  4.1669e+04  0.0000e+00  0.00e+00
 1   2.7592e+01  1.0054e+02  1.4422e+04  6.0900e+04  2.7312e+01  7.80e-02
 2   2.0609e+01  1.3549e+02  1.4197e+04  9.2400e+04  4.7710e+01  8.06e-02
 3   1.1090e+01  1.5998e+02  4.9590e+03  1.1844e+05  5.7353e+01  7.03e-02

******************************************************
*** Linear regression estimates of MLMC parameters ***
```

```
*****************************************************

 alpha in 0.894036  (exponent for (MLMC weak convergence)
 beta  in 1.517417  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)

****************************
*** MLMC complexity tests ***
****************************

  eps        value    mlmc_cost   std_cost  savings     N_l
---------------------------------------------------------
20.0000  1.1219e+02  2.822e+03  6.317e+03     2.24       433       157
         128        19
10.0000  1.4409e+02  2.266e+04  5.053e+04     2.23      3739      1412
         418       227        80
5.0000  1.4464e+02  1.388e+05  4.043e+05     2.91     17604      6735
        3371      1302       509       196
```

```
*********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
*********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
--------------------------------------------------------------------------


 0   1.1194e+02  1.1194e+02  1.3949e+05  1.3949e+05  0.0000e+00  0.0000e
    +00
 1   3.2710e+01  9.7293e+01  3.1392e+04  1.2657e+05  7.9995e+01  1.9703e
    -01
 2   1.1260e+01  1.4634e+02  2.4706e+03  1.1253e+05  2.3498e+01  1.9233e
    -01
 3   2.5298e+00  1.1012e+02  7.0752e+02  4.5988e+04  1.2464e+02  2.5350e
    -01

 WARNING: kurtosis on finest level = 124.635394
 indicates MLMC correction dominated by a few rare paths;
 for (information on the connection to variance of sample variances,
 see http://mathworld.wolfram.com/SampleVarianceDistribution.html


*****************************************************
*** Linear regression estimates of MLMC parameters ***
*****************************************************

 alpha in 2.154073  (exponent for (MLMC weak convergence)
 beta  in 1.804028  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)

****************************
*** MLMC complexity tests ***
```

```
*****************************

  eps        value    mlmc_cost    std_cost   savings     N_l
--------------------------------------------------------------
20.0000   9.7249e+01   2.222e+03   3.001e+03      1.35        303        148
          128
10.0000   1.3635e+02   1.501e+04   9.811e+03      0.65       2546        958
          433        164
5.0000   1.3962e+02   9.892e+04   1.570e+05      1.59      12467       5067
         1805        939        438        160
2.0000   1.4248e+02   8.616e+05   1.962e+06      2.28      92840      37793
    16485       5855       3957       1254       721
1.0000   1.4346e+02   3.378e+06   1.570e+07      4.65     372582     153278
    64365      27523      10405       5147      1801       669
```

The DIFF value we calculate is 143.46 with confidence interval [141.50, 145.42]. Therefore, the EVPPI value we calculate is 273.3316 with confidence interval [270.9267, 275.7365]. (185966s)

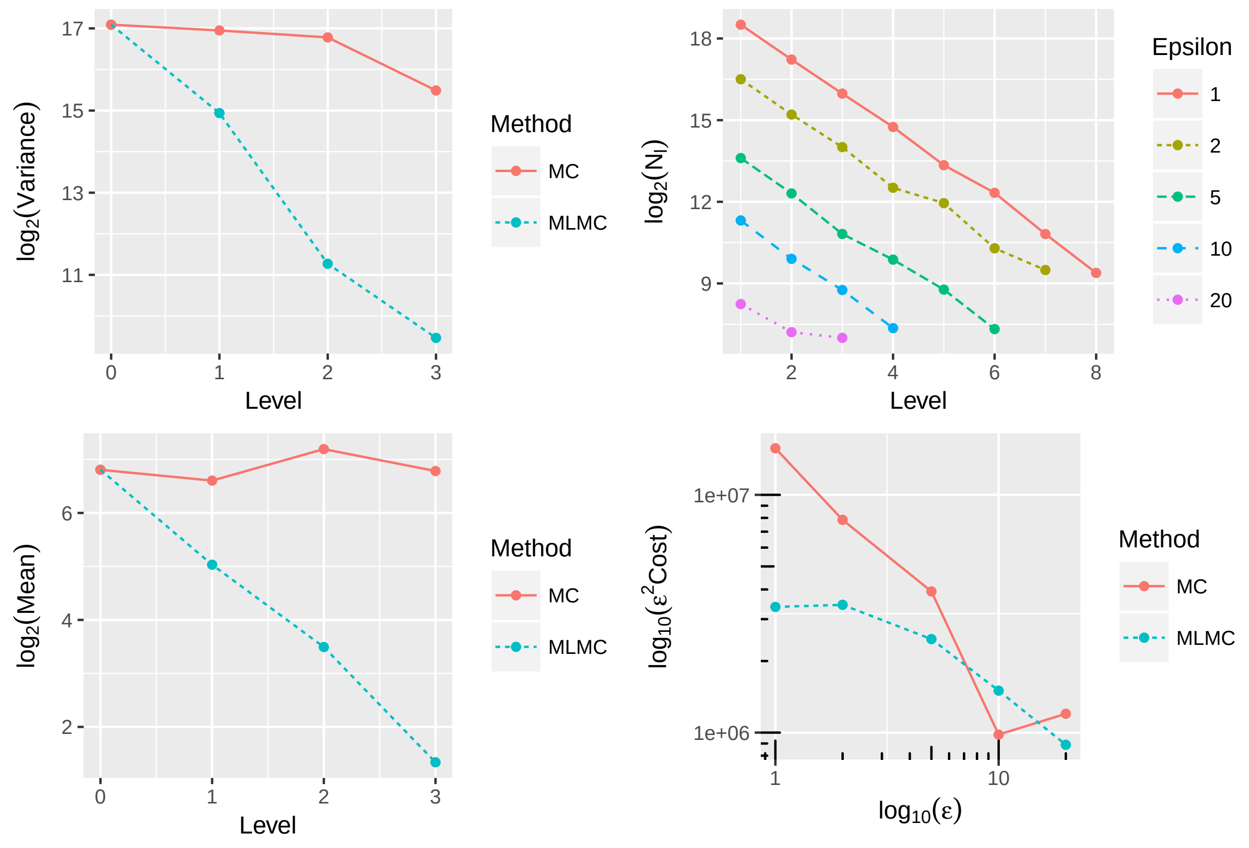

Figure 3: EVPPI for NOAC complex

4.3 EVPPI for All cost

Here we need to use level function **EVPPI_All_Cost_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
---------------------------------------------------------------------------

 0    2.7993e+02  2.7993e+02  6.9418e+05  6.9418e+05  0.0000e+00  0.00e+00
 1   -1.3631e+02  6.0805e+01  7.3320e+04  3.7684e+04  8.5284e+00  2.40e-01
 2   -4.8568e+01  2.5331e+01  1.7550e+04  4.2014e+04  1.6180e+01  9.28e-02
 3   -2.1944e+01  0.0000e+00  1.1263e+04  1.0000e-10  3.7258e+01  4.10e-02

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 1.146185  (exponent for (MLMC weak convergence)
 beta  in 0.639862  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be 256. Therefore, we use standard MC with 256 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 0.913. Then the confidence interval is $[0.000, 1.789]$.

4.4 EVPPI for Event Cost

Here we need to use level function **EVPPI_Cost_EventCost_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
---------------------------------------------------------------------------

 0    2.7172e+02  2.7172e+02  7.7953e+05  7.7953e+05  0.0000e+00  0.00e+00
 1   -1.4814e+02  7.4038e+01  8.5574e+04  5.1336e+04  1.4358e+01  1.33e-01
 2   -6.1053e+01  1.9482e+01  2.4539e+04  2.2669e+04  1.4521e+01  4.59e-02
 3   -2.2265e+01  3.3298e+00  1.4445e+04  1.4081e+03  8.3124e+01  7.47e-02

******************************************************
*** Linear regression estimates of MLMC parameters ***
```

```
*****************************************************

 alpha in 1.455305  (exponent for (MLMC weak convergence)
 beta  in 0.764503  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 256. Therefore, we use standard MC with 256 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 0.393. Then the confidence interval is $[0.000, 0.770]$.

4.5 EVPPI for Treatment Cost

Here we need to use level function **EVPPI_Cost_TreatmentCost_std_p.R**:

```
*********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
*********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
---------------------------------------------------------------------------

 0    2.0412e+02  2.0412e+02  3.7264e+05  3.7264e+05  0.0000e+00  0.00e+00
 1   -1.7125e+02  5.7550e+01  1.0379e+05  5.1527e+04  1.0801e+01  8.02e-02
 2   -7.2813e+01  1.8055e+01  3.7120e+04  5.1660e+03  3.7265e+01  2.55e-01
 3   -9.7613e+00  2.2268e+00  2.3482e+03  6.0921e+02  5.2293e+01  1.57e-01

*****************************************************
*** Linear regression estimates of MLMC parameters ***
*****************************************************

 alpha in 2.899053  (exponent for (MLMC weak convergence)
 beta  in 3.982562  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 256. Therefore, we use standard MC with 256 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 0.004. Then the confidence interval is $[0.000, 0.0078]$.

4.6 EVPPI for State Cost

Here we need to use level function **EVPPI_Cost_StateCost_std_p.R**:

```
*******************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
*******************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
--------------------------------------------------------------------------


 0    2.4231e+02  2.4231e+02  3.7153e+05  3.7153e+05  0.0000e+00  0.00e+00
 1   -1.0356e+02  9.2781e+01  5.4053e+04  1.2632e+05  1.2303e+01  1.44e-01
 2   -7.9324e+01  1.6767e+01  4.6819e+04  8.0165e+03  1.8418e+01  1.88e-02
 3   -2.7354e+01  9.4154e-01  9.7442e+03  1.1259e+02  2.3553e+01  2.18e-01

****************************************************
*** Linear regression estimates of MLMC parameters ***
****************************************************

 alpha in 1.535978  (exponent for (MLMC weak convergence)
 beta  in 2.264494  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 256. Therefore, we use standard MC with 256 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 0.386. Then the confidence interval is $[0.000, 0.757]$.

4.7 EVPPI for Switch probability

Here we need to use level function **EVPPI_Prob_Switch_std_p.R**:

```
*******************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
*******************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
--------------------------------------------------------------------------


 0    2.2445e+02  2.2445e+02  4.1555e+05  4.1555e+05  0.0000e+00  0.00e+00
 1   -1.1808e+02  2.5597e+01  1.0356e+05  1.9642e+04  1.6402e+01  2.75e-01
 2   -3.4387e+01  0.0000e+00  2.0737e+04  1.0000e-10  3.1868e+01  1.16e-01
 3   -2.9826e+00  1.8597e-01  5.1082e+02  3.5473e+00  9.0591e+01  4.88e-01

****************************************************
*** Linear regression estimates of MLMC parameters ***
****************************************************

 alpha in 3.527209  (exponent for (MLMC weak convergence)
 beta  in 5.343242  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 128. Therefore, we use standard MC with 128 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 0.01. Then the confidence interval is $[0.000, 0.0196]$.

4.8 EVPPI for Previous effect

Here we need to use level function **EVPPI_Prob_Effect_std_p.R**:

```
*****************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
*****************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
----------------------------------------------------------------------------

 0    2.0114e+02  2.0114e+02  2.7529e+05  2.7529e+05  0.0000e+00  0.00e+00
 1   -1.5803e+02  6.9698e+01  1.2253e+05  4.6829e+04  1.5797e+01  9.18e-02
 2   -8.5088e+01  4.0807e+01  4.6034e+04  5.6138e+04  1.4975e+01  3.17e-01
 3   -2.1382e+01  3.8988e+00  5.9737e+03  8.0218e+02  2.8962e+01  1.70e-01

*************************************************
*** Linear regression estimates of MLMC parameters ***
*************************************************

 alpha in 1.992550  (exponent for (MLMC weak convergence)
 beta  in 2.946002  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 128. Therefore, we use standard MC with 128 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros.) The estimated bias is 0.34. Then the confidence interval is $[0.000, 0.666]$.

4.9 EVPPI for All MCMC

Here we need to use level function **EVPPI_HR_all_std_p.R**:

EVPPI: option 0

```
*****************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
```

```
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
--------------------------------------------------------------------------


 0    3.8457e+02  3.8457e+02  8.0224e+05  8.0224e+05  0.0000e+00  0.00e+00
 1   -1.1452e+01  3.6307e+02  3.2293e+03  8.5760e+05  4.2433e+01  2.01e-02
 2   -1.9985e+01  3.2853e+02  1.1079e+04  4.0575e+05  6.6371e+01  3.29e-02
 3   -9.1749e+00  1.8893e+02  2.4910e+03  1.9302e+05  6.5502e+01  4.36e-01

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 1.123177  (exponent for (MLMC weak convergence)
 beta  in 2.152993  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

DIFF: option 1

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
--------------------------------------------------------------------------


 0   2.8970e+01  2.8970e+01  3.5976e+04  3.5976e+04  0.0000e+00  0.00e+00
 1   1.1452e+01  6.0101e+01  3.2293e+03  2.0839e+05  4.2433e+01  1.05e-01
 2   1.9985e+01  1.2140e+02  1.1079e+04  2.3886e+05  6.6371e+01  1.48e-01
 3   9.1749e+00  7.2088e+01  2.4910e+03  4.0982e+04  6.5502e+01  2.97e-01

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 1.123177  (exponent for (MLMC weak convergence)
 beta  in 2.152993  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)

*****************************
*** MLMC complexity tests ***
*****************************

  eps       value   mlmc_cost   std_cost  savings     N_l
-----------------------------------------------------------
20.0000  5.5324e+01  4.054e+03  2.186e+03     0.54       331       128
         228        66
10.0000  6.3087e+01  1.031e+04  1.749e+04     1.70       791       602
         156       189        64
5.0000  5.4228e+01  5.489e+04  1.399e+05     2.55      2124      7374
```

```
         1749         213          71          23
2.0000   6.8298e+01   5.255e+05   1.749e+06      3.33      85231      35962
         9206        2525        1661         431        128
1.0000   6.8559e+01   2.447e+06   1.399e+07      5.72     269425     136067
    45267      20159        6297        4560        940        256
```

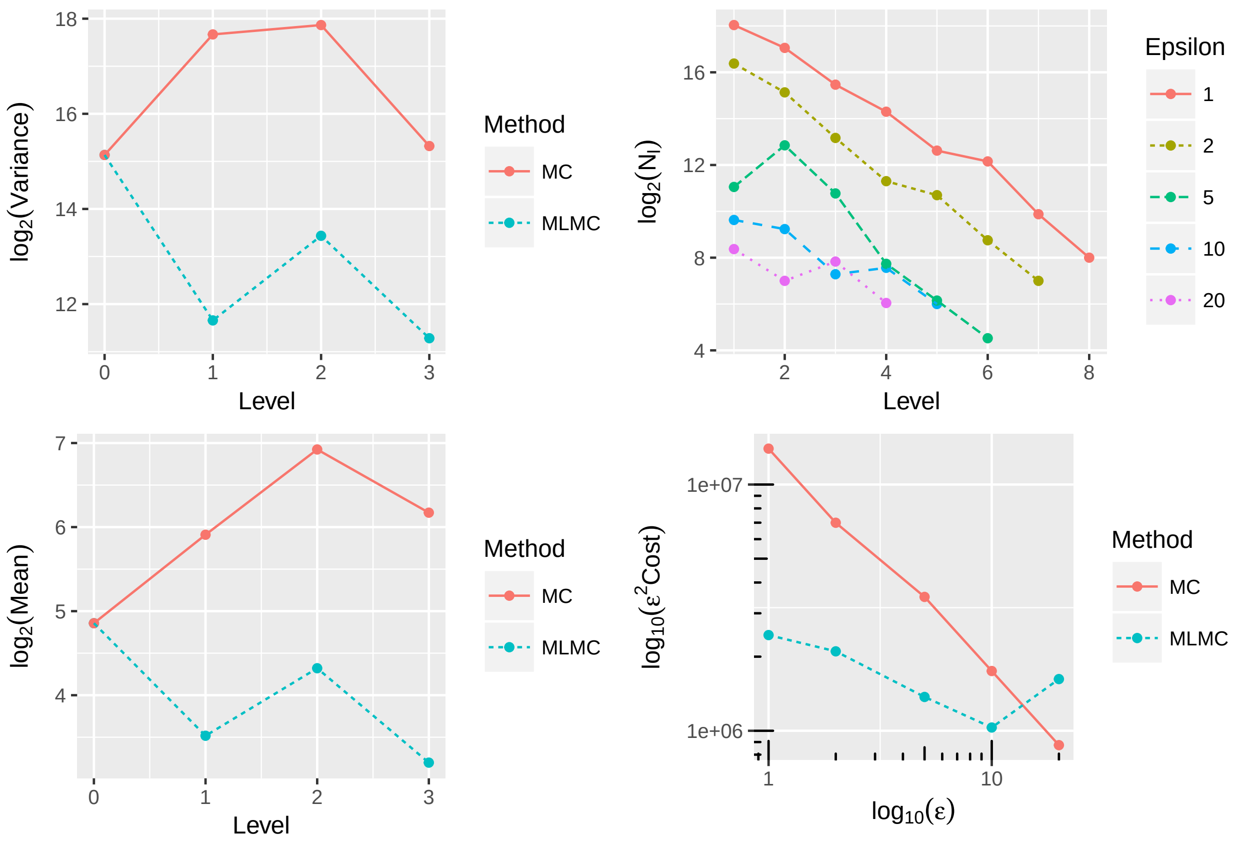

Figure 4: EVPPI for All MCMC

The DIFF value we calculate is 68.56 with confidence interval [65.56, 71.56]. Therefore, the EVPPI value we calculate is 348.233 with confidence interval [345.8281, 350.6379]. (139217s=38.67h)

4.10 EVPPI for Baseline HR

Here we need to use level function **EVPPI_HR_baseline_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************
```

```
 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
-------------------------------------------------------------------------


 0    1.3686e+02  1.3686e+02  2.5214e+05  2.5214e+05  0.0000e+00  0.00e+00
 1   -1.0349e+02  9.6359e+01  4.3247e+04  8.4533e+04  9.1546e+00  2.37e-01
 2   -7.7687e+01  5.1619e+01  6.5472e+04  3.9662e+04  6.7580e+01  1.66e-01
 3   -2.2721e+01  1.0865e+01  7.3008e+03  4.9581e+03  4.0040e+01  1.91e-01

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 1.773639  (exponent for (MLMC weak convergence)
 beta  in 3.164743  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 128. Therefore, we use standard MC with 128 inner samples and 1024 outer samples to get the EVPPI value 0.87 with standard deviation 0.54. (1021 out of 1024 samples are zeros) The estimated bias is 0.57. Then the confidence interval is $[0, 1.539]$.

4.11 EVPPI for all Loghr

Here we need to use level function **EVPPI_HR_loghr_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
-------------------------------------------------------------------------


 0   3.6351e+01  3.6351e+01  1.3441e+04  1.3441e+04  0.0000e+00  0.00e+00
 1   3.5991e+01  8.8299e+01  2.5870e+04  6.3344e+04  6.5288e+01  1.13e-01
 2   2.0207e+01  1.0382e+02  4.4732e+03  2.5360e+04  2.4222e+01  3.69e-02
 3   6.0725e+00  1.4635e+02  1.1244e+03  9.2901e+04  4.7688e+01  2.76e-01

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 1.734459  (exponent for (MLMC weak convergence)
 beta  in 1.992120  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)

*****************************
```

```
*** MLMC complexity tests ***
*****************************

  eps         value    mlmc_cost    std_cost  savings      N_l
--------------------------------------------------------
20.0000  1.1135e+02  2.728e+03  4.955e+03     1.82        274        153
         128         34
10.0000  1.1358e+02  8.470e+03  1.982e+04     2.34       1365        781
         199         64
5.0000  1.1442e+02  6.353e+04  3.171e+05     4.99       7882       3361
         1540        525        255          85
2.0000  1.3090e+02  7.794e+05  7.928e+06    10.17      69648      37238
    16782       5961       2558       1014       525        186
1.0000  1.3031e+02  3.185e+06  3.171e+07     9.96     301320     158643
    69036      28060      10127       4438      1547        549
```

The DIFF value we calculate is 130.31 with confidence interval [127.31, 133.31]. Therefore, the EVPPI value we calculate is 286.482 with confidence interval [284.0771, 288.8869].(266317s)

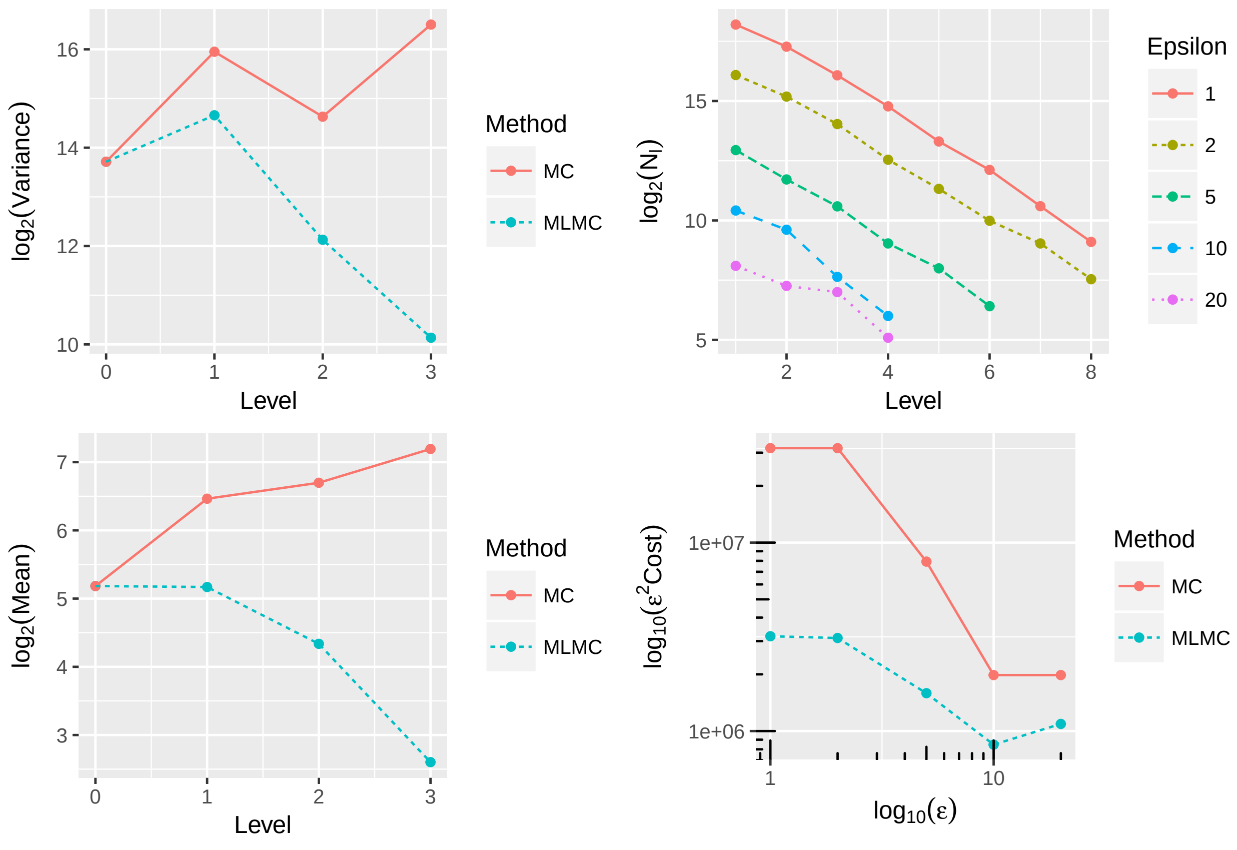

Figure 5: EVPPI for All Loghr

4.12 EVPPI for No Treatment MCMC

Here we need to use level function **EVPPI_HR_NT_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
---------------------------------------------------------------------------

 0    2.0636e+02  2.0636e+02  3.7300e+05  3.7300e+05  0.0000e+00  0.00e+00
 1   -1.3511e+02  1.1123e+02  8.3966e+04  1.1336e+05  1.8679e+01  1.21e-01
 2   -4.9496e+01  1.3164e+01  1.5145e+04  5.1743e+03  1.6374e+01  3.44e-01
 3   -1.8895e+01  8.3564e+00  6.3101e+03  4.3894e+03  5.3398e+01  2.44e-01

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 1.389316  (exponent for (MLMC weak convergence)
 beta  in 1.263155  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 128. Therefore, we use standard MC with 128 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 1.05. Then the confidence interval is $[0.00, 2.058]$.

4.13 EVPPI for All utility

Here we need to use level function **EVPPI_All_Utility_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
---------------------------------------------------------------------------

 0    2.6342e+02  2.6342e+02  6.0452e+05  6.0452e+05  0.0000e+00  0.00e+00
 1   -1.4821e+02  1.2645e+02  9.5084e+04  1.5098e+05  1.3058e+01  2.87e-02
 2   -7.2133e+01  1.5102e+01  3.4588e+04  1.0380e+04  1.6418e+01  2.18e-01
 3   -2.8512e+01  2.7845e+00  1.2608e+04  7.0456e+02  3.0304e+01  2.53e-01

******************************************************
*** Linear regression estimates of MLMC parameters ***
```

```
*****************************************************

 alpha in 1.339103  (exponent for (MLMC weak convergence)
 beta  in 1.455979  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 256. Therefore, we use standard MC with 256 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 are zeros) The estimated bias is 0.696. Then the confidence interval is $[0, 1.363]$.

4.14 EVPPI for Age Utility

Here we need to use level function **EVPPI_Utility_Age_std_p.R**:

```
*********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
*********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
--------------------------------------------------------------------------

 0    2.1015e+02  2.1015e+02  4.1165e+05  4.1165e+05  0.0000e+00  0.00e+00
 1   -1.4175e+02  1.2138e+02  9.3895e+04  1.1524e+05  1.7413e+01  1.55e-01
 2   -5.8417e+01  2.0024e+01  2.6369e+04  8.2395e+03  3.4866e+01  2.73e-01
 3   -1.4088e+01  3.0714e+00  2.9083e+03  1.1637e+03  2.1248e+01  6.04e-02

*****************************************************
*** Linear regression estimates of MLMC parameters ***
*****************************************************

 alpha in 2.051910  (exponent for (MLMC weak convergence)
 beta  in 3.180579  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 128. Therefore, we use standard MC with 128 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 0.20. Then the confidence interval is $[0.00, 0.392]$.

4.15 EVPPI for Health Utility

Here we need to use level function **EVPPI_Utility_state_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
---------------------------------------------------------------------------

 0    2.6220e+02  2.6220e+02  5.6145e+05  5.6145e+05  0.0000e+00  0.00e+00
 1   -1.5104e+02  1.1931e+02  1.0917e+05  1.2418e+05  1.3221e+01  2.14e-02
 2   -6.8958e+01  1.9986e+01  2.7646e+04  1.6142e+04  1.9734e+01  1.77e-01
 3   -1.2682e+01  2.5258e+00  2.9814e+03  8.1020e+02  3.5353e+01  8.57e-02

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 2.442887  (exponent for (MLMC weak convergence)
 beta  in 3.213025  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 128. Therefore, we use standard MC with 128 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 0.08. Then the confidence interval is $[0.00, 0.159]$.

4.16 EVPPI for Event Utility

Here we need to use level function **EVPPI_Utility_event_std_p.R**:

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
---------------------------------------------------------------------------

 0    2.6180e+02  2.6180e+02  5.6242e+05  5.6242e+05  0.0000e+00  0.00e+00
 1   -1.4777e+02  1.0699e+02  9.3080e+04  9.3625e+04  1.4503e+01  1.95e-02
 2   -6.9250e+01  1.1174e+01  3.0268e+04  2.8036e+03  3.1192e+01  1.87e-01
 3   -1.5453e+01  1.5139e+00  3.7475e+03  2.9107e+02  3.1691e+01  1.66e-01

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 2.163937  (exponent for (MLMC weak convergence)
 beta  in 3.013764  (exponent for (MLMC variance)
gamma in 1.000000  (exponent for (MLMC cost)
```

Together with Table 8 and 9, the EVPPI value drops a lot and to bound the bias, we need to increase the number of inner samples M to be at least 128. Therefore, we use standard MC with 128 inner samples and 1024 outer samples to get the EVPPI value 0 with standard deviation 0. (All 1024 samples are zeros) The estimated bias is 0.17. Then the confidence interval is $[0, 0.333]$.

5 Check

5.1 check4: 32000 samples

```
           "Coum"    "Apix"    "Dabi"    "Edo"     "Riva"
--------------------------------------------------------------
Total Cost 24283.5   23149.6   22905.8   23811.3   24667.1
Max       323388.8  289015.9  289089.7  288727.1  290651.8
Min          406.5     784.0     768.2     811.0     811.1
sd         12015.6   10289.8   10452.3   10333.0   10580.8
CreHigh    24415.1   23262.4   23020.3   23924.5   24783.0
CreLow     24151.8   23036.9   22791.3   23698.1   24551.1
--------------------------------------------------------------
IncreCost    0       -1133.8   -1377.6    -472.1     383.5
Max          0        8943.2   48985.2   51552.1   43808.5
Min          0     -107912.0  -94302.1 -113923.1  -94608.9
sd           0        3026.6    3019.4    3063.5    2881.6
CreHigh      0       -1100.6   -1344.6    -438.6     415.1
CreLow       0       -1167.0   -1410.7    -505.7     351.9
--------------------------------------------------------------
QALYs      4.901       5.195     5.106     5.125     5.153
Max       13.365      12.416    12.393    12.434    12.449
Min       -0.018       0.204     0.171     0.192     0.186
Negative     1           0         0         0         0
sd         0.698       0.724     0.701     0.700     0.726
CreHigh    4.908       5.203     5.114     5.133     5.161
CreLow     4.893       5.187     5.099     5.118     5.145
---------------------------------------------------------------
IncreQALY    0         0.294     0.205     0.224     0.252
Max          0         1.576     2.308     4.090     4.092
Min          0        -2.023    -2.052    -2.569    -1.936
sd           0         0.230     0.199     0.202     0.232
CreHigh      0         0.296     0.208     0.227     0.255
CreLow       0         0.291     0.203     0.222     0.249
--------------------------------------------------------------
NetB2       73737.1   80752.5   79231.6   78706.9   78402.9
Max        146029.9  144205.3  143842.1  143611.7  143513.6
Min       -231459.0 -191770.6 -191899.5 -191681.9 -193217.0
sd          14665.6   15340.4   15032.3   14557.9   14907.5
```

```
CreHigh    73897.8     80920.6     79396.3    78866.4    78566.2
CreLow     73576.4     80584.5     79066.9    78547.4    78239.5
---------------------------------------------------------------
IncreNetB2    0         7015.4      5494.5     4969.7     4665.7
Max           0       118727.2    107111.8   120936.6   108042.5
Min           0       -11266.6    -54047.5   -80780.5   -49398.5
sd            0         5162.7      4701.7     4386.3     4356.8
CreHigh       0         7071.9      5546.0     5017.8     4713.4
CreLow        0         6958.8      5443.0     4921.6     4618.0
---------------------------------------------------------------
NetB3     122747.5    132703.7    130300.4   129966.1   129937.9
Max       259690.9    256325.3    255780.6   255546.5   255520.2
Min      -185494.1   -168940.8   -170475.4  -169636.7  -170174.8
sd         19922.7     21392.0     20779.4    20291.2    20869.3
CreHigh   122965.8    132938.1    130528.1   130188.4   130166.6
CreLow    122529.2    132469.3    130072.8   129743.7   129709.3
---------------------------------------------------------------
IncreNetB3    0         9956.2      7552.9     7218.5     7190.4
Max           0       124134.7    113516.6   124443.4   114759.2
Min           0       -22417.7    -70342.0  -106473.4   -66166.4
sd            0         7165.1      6368.7     6039.4     6368.5
CreHigh       0        10034.7      7622.7     7284.7     7260.2
CreLow        0         9877.6      7483.1     7152.3     7120.6
```

6 Comparison of different schemes

EVPPI	Value	Inner Num	RMSE	Bias	Sd	Std MC	MLMC	QMC
Loghr Simple	196.692	2048	1.227	0.25	1.201	2144	68.63	
Loghr Complex	273.3316	256	1.227	0.25	1.201	157	33.78	
All Cost	0	256	0.913	0.913	0.00	2.62	NA	
Event Cost	0	256	0.393	0.393	0.00	2.62	NA	
Treatment Cost	0	256	0.004	0.004	0.00	2.62	NA	
State Cost	0	256	0.386	0.386	0.00	2.62	NA	
Switch Probability	0	128	0.01	0.01	0.00	1.31	NA	
Previous Effect	0	128	0.34	0.34	0.00	1.31	NA	
All MCMC	348.233	256	1.227	0.25	1.201	139.9	24.47	
Baseline MCMC	0.87	128	0.785	0.57	0.54	1.31	NA	
Loghr MCMC	286.482	256	1.227	0.25	1.201	317.1	31.85	
No Treatment MCMC	0	128	1.05	1.05	0.00	1.31	NA	
All Utility	0	256	0.696	0.696	0.00	2.62	NA	
Age Utility	0	128	0.20	0.20	0.00	1.31	NA	
Health Utility	0	128	0.08	0.08	0.00	1.31	NA	
Event Utility	0	128	0.17	0.17	0.00	1.31	NA	

Table 5: EVPPI: comparison of computational cost (10^5)

7 Population EVPI and EVPPI values

Assume 5000 patients per year, discounting at 1.035, and summing over technology lifetime of 10 years gives a total of 43038.43 patients. Multiply this by individual EVPI/EVPPI values.

EVPI	Value	Confidence interval
Loghr Simple	17 938 056	[17 846 186, 18 029 917]

Table 6: Population EVPI

EVPPI	Value	Confidence interval
Loghr Simple	8 465 315	[8 361 795, 8 568 801]
Loghr Complex	11 763 763	[11 660 260, 11 867 266]
All Cost	0	[0, 76 995]
Event Cost	0	[0, 33 139]
Treatment Cost	0	[0, 335]
State Cost	0	[0, 32 580]
Switch Probability	0	[0, 843]
Previous Effect	0	[0, 28 663]
All MCMC	14 987 402	[14 883 898, 15 090 982]
Baseline MCMC	37 443	[0, 66 236]
Loghr MCMC	12 329 736	[12 226 232, 12 433 239]
No Treatment MCMC	0	[0, 88 573]
All Utility	0	[0, 58 661]
Age Utility	0	[0, 16 871]
Health Utility	0	[0, 6 843]
Event Utility	0	[0, 14 331]

Table 7: Population EVPPI

8 Multivariate t distribution

The conventional version of the p-dimensional multivariate t (MVT) distribution $X \sim t_p(\mu, \Sigma, \nu)$, with mean μ, scale matrix Σ (generally not the covariance of X), and degrees of freedom ν (which determine the thickness of the tail), has the probability density function

$$f(x) = \frac{\Gamma\{(\nu+p)/2\}}{\Gamma(\nu/2)(\nu\pi)^{p/2}|\Sigma|^{1/2}}\{1+\nu^{-1}(x-\mu)^T\Sigma^{-1}(x-\mu)\}^{-(\nu+p)/2}. \tag{8.1}$$

The conditional distribution of the multivariate t distribution also follows the multivariate t distribution. Assuming $X = (X_1, X_2)_{p_1,p_2}$, we can define

$$\begin{aligned}
\mu_{2|1} &= \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(X_1 - \mu_1) \\
\Sigma_{22|1} &= \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12} \\
d_1 &= (X_1 - \mu_1)^T\Sigma_{11}^{-1}(X_1 - \mu_1)
\end{aligned}$$

and then the conditional distribution of X_2 given X_1 is

$$X_2 \mid X_1 \sim t_{p_2}\left(\mu_{2|1},\ \frac{\nu + d_1}{\nu + p_1}\Sigma_{22|1},\ \nu + p_1\right)$$

```
# The estimation of the mean and covariance matrix of the multivariate
   normal distribution for the MCMC samples
test = bugs.loghr[,8:35]
sigma_loghr = cov(test)
mu_loghr = colMeans(test)

# The estimation of the mean and scale matrix of the multivariate t
   distribution for the MCMC samples
test_t = cov.trob(test)
sigma_loghr_t = test_t$cov
mu_loghr_t = test_t$center

# Generate conditional samples of multivariate normal distribution
  temp = sample(1:29999, N, replace=TRUE)
  MCMC.selection = rep(temp, times = 1, each = M)
  MCMC.loghr.samples = bugs.loghr[MCMC.selection,]

  index <- (1:14)
  nindex <- setdiff(1:28,index)
  sigma_coef <- sigma_loghr[nindex,index]%*%solve(sigma_loghr[index,index
     ])
  sigma_cond <- sigma_loghr[nindex,nindex] - sigma_coef%*%sigma_loghr[
     index,nindex]

  mu_con <- matrix(mu_loghr[nindex],length(nindex),NN) + sigma_coef%*%(t(
     MCMC.loghr.samples[,index+7])-matrix(mu_loghr[index],length(index),NN
     ))

  # X1 only affects the mean of the conditional distribution
  MCMC.loghr.samples[,nindex+7] <-  t(mu_con)+ mvrnorm(NN, rep(0,length(
     nindex)), sigma_cond)

#  Generate conditional samples of multivariate t distribution (MVT)
  temp = sample(1:29999, N, replace=TRUE)
  MCMC.selection = rep(temp, times = 1, each = M)
  MCMC.loghr.samples = bugs.loghr[MCMC.selection,]
  # conditional MVT is still MVT
  index <- (1:14)
  nindex <- setdiff(1:28,index)
  sigma_coef <- sigma_loghr_t[nindex,index]%*%solve(sigma_loghr_t[index,
     index])
  sigma_cond <- sigma_loghr_t[nindex,nindex] - sigma_coef%*%sigma_loghr_t[
     index,nindex]

  mu_con <- matrix(mu_loghr_t[nindex],length(nindex),NN) + sigma_coef%*%(t
     (MCMC.loghr.samples[,index+7])-matrix(mu_loghr_t[index],length(index)
     ,NN))
 # note that different X1 will result in the mean and scale matrix of the
     MVT, so we need to generate all the conditional samples one group by
     one group
  for(i in 1:N){
    d1_t <- t(t(MCMC.loghr.samples[i,index+7])-mu_loghr_t[index])%*%solve(
       sigma_loghr_t[index,index])%*%(t(MCMC.loghr.samples[i,index+7])-mu_
       loghr_t[index])
    temp = as.numeric(d1_t)
    MCMC.loghr.samples[((i-1)*NN+1):(i*NN),nindex+7] <-   rmvt(n=NN, sigma
        = sigma_cond*(5+temp)/(5+14), delta=t(mu_con)[1,], df = 19)
  }
```

Compared with the multivariate normal distribution, the MVT can capture the feature of the thick tail and may give better approximation of the covariance structure of the MCMC samples. Here is the comparison results between the two distributions of the EVPPI for NOAC simple trial:

	DIFF	DIFF sd	EVPPI	EVPPI sd	Bias
Loghr 8:21 MCMC	**215.01**	8.01	196.93	15.76	11.36
Loghr 8:21 MCMC t	**227.34**	8.73	203.48	17.66	9.72

Table 8: EVPPI value 1024*32

	DIFF	DIFF sd	EVPPI	EVPPI sd	Bias
Loghr 8:21 MCMC	**218.01**	7.25	184.51	16.01	3.85
Loghr 8:21 MCMC t	**232.33**	7.75	189.13	16.89	3.24

Table 9: EVPPI value 1024*128

The MLMC convergence test shows

```
**********************************************************
*** Convergence tests, kurtosis, telescoping sum check ***
**********************************************************

 l   ave(Pf-Pc)    ave(Pf)   var(Pf-Pc)    var(Pf)    kurtosis     check
```

```
-----------------------------------------------------------------------

 0   8.0919e+01  8.0919e+01  5.4203e+04  5.4203e+04  0.0000e+00  0.00e+00
 1   5.4755e+01  1.5288e+02  2.1013e+04  9.2621e+04  1.8635e+01  9.51e-02
 2   2.9143e+01  2.3733e+02  8.2890e+03  1.2676e+05  1.3665e+01  2.77e-01
 3   1.6829e+01  2.2709e+02  5.0631e+03  1.1435e+05  4.7186e+01  1.33e-01

******************************************************
*** Linear regression estimates of MLMC parameters ***
******************************************************

 alpha in 0.792219  (exponent for (MLMC weak convergence)
 beta  in 0.711176  (exponent for (MLMC variance)
 gamma in 1.000000  (exponent for (MLMC cost)

*****************************
*** MLMC complexity tests ***
*****************************

  eps        value   mlmc_cost   std_cost  savings     N_l
---------------------------------------------------------
20.0000  2.6284e+02  2.190e+04  7.806e+04     3.57      1958      1025
         411       160        67        32        16         7
```

which we can see that the confidence interval is consistent with the results when using multivariate normal distribution.

9 Summary

In this testcase, we calculate 1 EVPI value and 16 EVPPI values. The typical features of this case are high-dimensional inputs random variables (99 in total: 34 normal, 8 uniform, 15 beta, MCMC 7 (baseline), MCMC 28 (loghr), MCMC 7 (no treatment)) and a really time-consuming function evaluation (approximately 0.72s for one calculation on one core).

For **EVPI** calculation, parallel computing is necessary and desirable. Overall, using a 32 core computer can be at least 20 times faster.

For **EVPPI** calculation, parallel computing is still important. **MLMC** can save a lot computational cost in 4 out of 16 cases. Fortunately, 2 of them are the cases we are really interested in. Actually this is not an accident. MLMC can help the parameters which enjoy a smaller difference of EVPI and EVPPI and at the same time, a larger EVPPI value. In medical funding research, the parameter with a large EVPPI value is usually we are concern about and deserve more calculations. For other parameters with low EVPPI value, we only adopt standard Monte Carlo method to calculate it. In 11 out of 12 cases, all the EVPPI samples are 0 and practically, with the information of this parameter, we still choose the treatment which is same as the one without this information. These cases then will be much less of interest for the funding decision. For more accurate calculations, we may need to employ the importance sampling technique to try to capture the non-zero samples.

Therefore, for **EVPPI**, parallel computing and MLMC together will be at least 100-600 times faster. This savings will increase when the required accuracy become smaller.

The last issue we investigate is the covariance structure between the MCMC samples. Initially we use multivariate normal distribution (MVN) to approximate it and get good convergence results. However, when we use the statistical test to see whether MVN is a good approximation of the MCMC samples, all the tests reject it. Then, we use multivariate t distribution (MVT) to approximate it and get the results in section 8, which is consistent with the results we get from the MVN approximation. Therefore, although MVN does not give a good approximation to the distribution of the MCMC samples, but it gives a good covariance structure and we should use it due to its simplicity and tractability.

As for the use of the mixture model, it often requires machine learning technique to estimate the parameters and it is also difficult to generate the conditional samples which exactly are what we want. In addition, from the form of the posterior distribution of the MCMC samples, it should be unimodal and have the tail which is the mixture of MVT and MVN. Plus, in our case, the conditional sampling is important not the good approximation of the distribution itself. Therefore, we do not consider the mixture model.

As for the conditional Monte Carlo idea, it is still quite difficult to construct the joint distribution (except for MVN) to reduce the variance. Plus, the random selection of the MCMC samples has given a good result compared with the direct MCMC generators. For the conditional distribution, it can correct the bias but we still need the MVN approximation and the weights for the correction are quite extremely distributed. Therefore, if we want to use MVN, then this idea will fail due to the extreme weights, otherwise, it will cause difficulty in generating the conditional samples.