Value of information - Toy model

The DARTH workgroup

Developed by the Decision Analysis in R for Technologies in Health (DARTH) workgroup:

Fernando Alarid-Escudero, PhD (1)

Eva A. Enns, MS, PhD (2)

M.G. Myriam Hunink, MD, PhD (3,4)

Hawre J. Jalal, MD, PhD (5)

Eline M. Krijkamp, MSc (3)

Petros Pechlivanoglou, PhD (6,7)

Alan Yang, MSc (7)

In collaboration of:

1. Drug Policy Program, Center for Research and Teaching in Economics (CIDE) - CONACyT, Aguascalientes, Mexico
2. University of Minnesota School of Public Health, Minneapolis, MN, USA
3. Erasmus MC, Rotterdam, The Netherlands
4. Harvard T.H. Chan School of Public Health, Boston, USA
5. University of Pittsburgh Graduate School of Public Health, Pittsburgh, PA, USA
6. University of Toronto, Toronto ON, Canada
7. The Hospital for Sick Children, Toronto ON, Canada

Please cite our publications when using this code:

- Jalal H, Pechlivanoglou P, Krijkamp E, Alarid-Escudero F, Enns E, Hunink MG. An Overview of R in Health Decision Sciences. Med Decis Making. 2017; 37(3): 735-746. https://journals.sagepub.com/doi/abs/10.1177/0272989X16686559

- Krijkamp EM, Alarid-Escudero F, Enns EA, Jalal HJ, Hunink MGM, Pechlivanoglou P. Microsimulation modeling for health decision sciences using R: A tutorial. Med Decis Making. 2018;38(3):400–22. https://journals.sagepub.com/doi/abs/10.1177/0272989X18754513

- Krijkamp EM, Alarid-Escudero F, Enns E, Pechlivanoglou P, Hunink MM, Jalal H. A Multidimensional Array Representation of State-Transition Model Dynamics. Med Decis Making. Online First https://doi.org/10.1177/0272989X19893973

Change `eval` to `TRUE` if you want to knit this document.

```r
rm(list = ls())       # clear memory (removes all the variables from the workspace)
```

01 Load packages

```
if (!require('pacman')) install.packages('pacman'); library(pacman) # use this package to conveniently
# load (install if required) packages from CRAN
p_load("here", "dplyr", "devtools", "matrixStats", "scales", "ggplot2", "grid", "mgcv", "gridExtra", "g
# load (install if required) packages from GitHub
# install_github("DARTH-git/dampack", force = TRUE) Uncomment if there is a newer version
p_load_gh("DARTH-git/dampack")
```

02 Load functions

```
source("VOI_Functions.R") # VOI functions
source("GA_functions.R")  # Gaussian Approximation Approach functions
```

03 Input model parameters

```
# Load simulation file
# Read the `.csv` simulation file into `R`.
toy   <- read.csv("PSA.csv", header = TRUE)[, -1]
# Strategy A = includes parameter uncertainty
# Strategy B = includes paramter uncertainty
# Strategy C = no parameter uncertainty -> same NBM for all PSA runs
n_sim <- nrow(toy)

# Display first five observations of the data fram using the command `head`
head(toy)

# Net Monetary Benefit (NMB)
# Create NMB matrix
nmb <- toy[, 5:7]
head(nmb) # print the first six rows

# Number of Strategies
n_strategies <- ncol(nmb)
n_strategies

# Assign name of strategies
strategies    <- c("Strategy A", "Strategy B", "Strategy C")
colnames(nmb) <- strategies
head(nmb)  # print the first six rows

# Format data frame suitably for plotting
nmb_gg <- melt(nmb,
               variable.name = "Strategy",
               value.name = "NMB")

# Plot NMB for different strategies
```

```r
# Faceted plot by Strategy
ggplot(nmb_gg, aes(x = NMB/1000)) +
  geom_histogram(aes(y =..density..), col = "black", fill = "gray") +
  geom_density(color = "red") +
  facet_wrap(~ Strategy, scales = "free_y") +
  xlab("Net Monetary Benefit (NMB) x10^3") +
  scale_x_continuous(breaks = number_ticks(5), labels = dollar) +
  scale_y_continuous(breaks = number_ticks(5)) +
  theme_bw()
```

04 Incremental NMB (INMB)

```r
# Calculate INMB of B vs A
# Only B vs A but we could have plotted all combinations
inmb <- data.frame(Simulation = 1:n_sim,
                   `Strategy B vs Strategy A` = nmb$`Strategy B` - nmb$`Strategy A`)

## Format data frame suitably for plotting
inmb_gg <- melt(inmb, id.vars = "Simulation",
                variable.name = "Comparison",
                value.name = "INMB")
txtsize <- 16

## Plot INMB
ggplot(inmb_gg, aes(x = INMB/1000)) +
  geom_histogram(aes(y =..density..), col = "black", fill = "gray") +
  geom_density(color = "red") +
  geom_vline(xintercept = 0, col = 4, size = 1.5, linetype = "dashed") +
  facet_wrap(~ Comparison, scales = "free_y") +
  xlab("Incremental Net Monetary Benefit (INMB) in thousand $") +
  scale_x_continuous(breaks = number_ticks(5), limits = c(-100, 100)) +
  scale_y_continuous(breaks = number_ticks(5)) +
  theme_bw(base_size = 14)
```

05 Loss Matrix

```r
# Find optimal strategy (d*) based on the highest expected NMB
d_star <- which.max(colMeans(nmb))
d_star

# Compute Loss matrix iterating over all strategies
# Initialize loss matrix of dimension: number of simulation by number of strategies
loss <- matrix(0, n_sim, n_strategies)
for (d in 1:n_strategies){ # d <- 1
  loss[, d] <- nmb[, d] - nmb[, d_star]
}
head(loss)
```

```
# Or without iterating (much faster!)
loss <- as.matrix(nmb - nmb[, d_star])
head(loss)
```

06 EVPI

```
# Find maximum loss overall strategies at each state of the world
# (i.e., PSA sample)
max_loss_i <- rowMaxs(loss)  # Only the positive values are a loss. Negative values show we selected th
head(max_loss_i)
## Average expected loss across all states of the world
## Expected loss = expected value of perfect information
evpi <- mean(max_loss_i)
evpi
```

07 EVPPI

```
names_params <- c("Mean No. Visits (A)",
                   "Mean No. Visits (B)",
                   "Prob. Failing (A)",
                   "Prob. Failing (B)")
# Matrix with parameters
x <- toy[, 1:4]
colnames(x) <- names_params
head(x)

# Number and names of parameters
n_params <- ncol(x)
n_params

# Histogram of parameters
# Format data suitably for plotting
params <- melt(x, variable.name = "Parameter")
head(params)
# Make parameter names as factors (helps with plotting formatting)
params$Parameter <- factor(params$Parameter,
                           levels = names_params,
                           labels = names_params)
# Facet plot of parameter distributions
ggplot(params, aes(x = value)) +
  geom_histogram(aes(y =..density..), col="black", fill = "gray") +
  geom_density(color = "red") +
  facet_wrap(~ Parameter, scales = "free") +
  scale_x_continuous(breaks = number_ticks(5)) +
  scale_y_continuous(breaks = number_ticks(5)) +
  theme_bw(base_size = 14)
```

Construct Spline metamodel

```
# Splines
# Initialize EVPPI vector
evppi_splines <- matrix(0, n_params)
lmm1 <- vector("list", n_params)
lmm2 <- vector("list", n_params)
lmm3 <- vector("list", n_params)
for (p in 1:n_params){ # p <- 1
  print(paste("Computing EVPPI of parameter", names_params[p]))
  # Estimate Splines
  lmm1[[p]] <- gam(loss[, 1] ~ s(x[, p]))
  lmm2[[p]] <- gam(loss[, 2] ~ s(x[, p]))
  lmm3[[p]] <- gam(loss[, 2] ~ s(x[, p]))

  # Predict Loss using Splines
  Lhat_splines <- cbind(lmm1[[p]]$fitted, lmm2[[p]]$fitted, lmm3[[p]]$fitted)

  # Compute EVPPI
  evppi_splines[p] <- mean(rowMaxs(Lhat_splines))
}
# Ploting EVPPI using of order polynomial
evppi_splines_gg <- data.frame(Parameter = names_params, EVPPI = evppi_splines)
evppi_splines_gg$Parameter <- factor((evppi_splines_gg$Parameter),
                              levels = names_params[order(evppi_splines_gg$EVPPI, decreasing = TRUE)])

# Plot EVPPI using ggplot2 package
ggplot(data = evppi_splines_gg, aes(x = Parameter, y = EVPPI)) +
  geom_bar(stat = "identity") +
  ylab("EVPPI ($)") +
  scale_y_continuous(breaks = number_ticks(6), labels = comma) +
  theme_bw(base_size = 14)
```

08 Expected value of sample information (EVSI)

```
# Effective (prior) Sample size
n0 <- c(10, # MeanNumVisitsA
        10, # MeanNumVisitsB
        10, # ProbFailA
        10) # ProbFailB
n <- c(0, 1, 5, 10, seq(20, 200, by = 20))
n_samples <- length(n)

# Each parameter individually (only assuming linear relationship)
# Initialize EVSI matrix for each parameters
evsi <- data.frame(N = n, matrix(0, nrow = n_samples, ncol = n_params))

# Name columns of EVPSI matrix with parameter names
colnames(evsi)[-1] <- names_params
```

```r
# Compute EVSI for all parameters separately
for (p in 1:n_params){ # p <- 1
  print(paste("Computing EVSI of parameter", names_params[p]))
    # Update loss based on gaussian approximation for each sample of interest
    for (nSamp in 1:n_samples){ # nSamp <- 10
      Ltilde1 <- predict.ga(lmm1[[p]], n = n[nSamp], n0 = n0[p])
      Ltilde2 <- predict.ga(lmm2[[p]], n = n[nSamp], n0 = n0[p])
      Ltilde3 <- predict.ga(lmm3[[p]], n = n[nSamp], n0 = n0[p])
      ## Combine losses into one matrix
      Ltilde <- cbind(Ltilde1, Ltilde2, Ltilde3)
      ### Apply EVSI equation
      evsi[nSamp, p+1] <- mean(rowMaxs(Ltilde))
    }
}

# Plotting EVSI
# Create EVSI data frame for plotting in decreasing order of EVPPI
evsi_gg <- melt(evsi[1:21,], id.vars = "N",
                 variable.name = "Parameter",
                 value.name = "evsi")
evsi_gg$Parameter <- factor((evsi_gg$Parameter),
                              levels = names_params[order(evppi_splines_gg$EVPPI, decreasing = TRUE)])

# Plot evsi using ggplot2 package
ggplot(evsi_gg, aes(x = N, y = evsi)) +  # colour = Parameter
  geom_line() +
  geom_point() +
  facet_wrap(~ Parameter) +  # scales = "free_y"
  ggtitle("Expected Value of Sample Information (EVSI)") +
  xlab("Sample size (n)") +
  ylab("$") +
  scale_x_continuous(breaks = number_ticks(5)) +
  scale_y_continuous(breaks = number_ticks(6), labels = dollar) +
  theme_bw(base_size = 14)

# Adding EVPPI
ggplot(evsi_gg, aes(x = N, y = evsi)) +  # colour = Parameter
  geom_line(aes(linetype = "EVSI")) +
  geom_point() +
  facet_wrap(~ Parameter) +  # scales = "free_y"
  geom_hline(aes(yintercept = EVPPI, linetype = "EVPPI"), data = evppi_splines_gg) +
  scale_linetype_manual(name="",
                        values = c("EVSI" = "solid", "EVPPI" = "dashed")) +
  xlab("Sample size (n)") +
  ylab("$") +
  #ggtitle("Expected Value of Sample Information (EVSI)") +
  scale_x_continuous(breaks = number_ticks(5)) +
  scale_y_continuous(breaks = number_ticks(6), labels = dollar) +
  theme_bw(base_size = 14)
```

09 Combination of parameters

09.1 Assuming an observational study

```
sel_params_obs <- c(1, 2)
# Vector with samples to evaluate EVPSI for an Observational design
n_obs <- c(0, 1, 5, 10, seq(20, 200, by = 20), 300, 400, 500, 600, 700, 800) #seq(0, 1000, by = 20)
n_obs_samples <- length(n_obs)
# Initailize EVPSI matrix for a combination of parameters
evsi_obs <- data.frame(Study = "Observational",
                       N = n_obs,
                       EVSI = matrix(0, nrow = n_obs_samples, ncol = 1))

# Estimate linear metamodel of two parameters
lmm1_obs <- gam(loss[, 1] ~ s(x[, sel_params_obs[1]]) +
              s(x[, sel_params_obs[2]]) +
              ti(x[, sel_params_obs[1]], x[, sel_params_obs[2]]))
lmm2_obs <- gam(loss[, 2] ~ s(x[, sel_params_obs[1]]) +
                s(x[, sel_params_obs[2]]) +
                ti(x[, sel_params_obs[1]], x[, sel_params_obs[2]]))
lmm3_obs <- gam(loss[, 3] ~ s(x[, sel_params_obs[1]]) +
                s(x[, sel_params_obs[2]]) +
                ti(x[, sel_params_obs[1]], x[, sel_params_obs[2]]))
# Predict Loss using Splines
Lhat_obs_splines <- cbind(lmm1_obs$fitted, lmm2_obs$fitted, lmm3_obs$fitted)

# Compute EVPPI
evppi_obs <- mean(rowMaxs(Lhat_obs_splines))
evppi_obs

for (nSamp in 1:n_obs_samples){
  Ltilde1_obs <- predict.ga(lmm1_obs, n = n_obs[nSamp], n0 = n0[sel_params_obs])
  Ltilde2_obs <- predict.ga(lmm2_obs, n = n_obs[nSamp], n0 = n0[sel_params_obs])
  Ltilde3_obs <- predict.ga(lmm3_obs, n = n_obs[nSamp], n0 = n0[sel_params_obs])
  # Combine losses into one matrix
  Ltilde_obs <- cbind(Ltilde1_obs, Ltilde2_obs, Ltilde3_obs)
  # Apply EVSI equation
  evsi_obs$EVSI[nSamp] <- mean(rowMaxs(Ltilde_obs))
}
```

09.2 Assuming an RCT

```
sel_params_rct <- c(3, 4)
# Vector with samples to evaluate EVPSI for a RCT
n_rct <- c(0, 1, 5, 10, seq(20, 200, by = 20))
n_rct_samples <- length(n_rct)
# Initailize EVPSI matrix for a combination of parameters
evsi_rct <- data.frame(Study = "RCT",
                       N = n_rct,
                       EVSI = matrix(0, nrow = n_rct_samples, ncol = 1))
```

```r
# Estimate linear metamodel of two parameters
lmm1_rct <- gam(loss[, 1] ~ s(x[, sel_params_rct[1]]) +
                  s(x[, sel_params_rct[2]]) +
                  ti(x[, sel_params_rct[1]], x[, sel_params_rct[2]]))
lmm2_rct <- gam(loss[, 2] ~ s(x[, sel_params_rct[1]]) +
                  s(x[, sel_params_rct[2]]) +
                  ti(x[, sel_params_rct[1]], x[, sel_params_rct[2]]))
lmm3_rct <- gam(loss[, 3] ~ s(x[, sel_params_rct[1]]) +
                  s(x[, sel_params_rct[2]]) +
                  ti(x[, sel_params_rct[1]], x[, sel_params_rct[2]]))
# Predict Loss using Splines
Lhat_rct_splines <- cbind(lmm1_rct$fitted, lmm2_rct$fitted, lmm3_rct$fitted)

# Compute EVPPI
evppi_rct <- mean(rowMaxs(Lhat_rct_splines))
evppi_rct

# Compute EVSI over different sample sizes
for (nSamp in 1:n_rct_samples){
  Ltilde1_rct <- predict.ga(lmm1_rct, n = n_rct[nSamp], n0 = n0[sel_params_rct])
  Ltilde2_rct <- predict.ga(lmm2_rct, n = n_rct[nSamp], n0 = n0[sel_params_rct])
  Ltilde3_rct <- predict.ga(lmm3_rct, n = n_rct[nSamp], n0 = n0[sel_params_rct])
  # Combine losses into one matrix
  Ltilde_rct <- cbind(Ltilde1_rct, Ltilde2_rct, Ltilde3_rct)
  # Apply EVSI equation
  evsi_rct$EVSI[nSamp] <- mean(rowMaxs(Ltilde_rct))
}
```

Plot EVSI for both study designs.

```r
# Combine both study designs
evppi_combo <- data.frame(Study = c("Observational", "RCT"),
                          EVPPI = c(evppi_obs, evppi_rct))
evsi_combo  <- rbind(evsi_obs,
                     evsi_rct)

# Plot EVSI by study design
ggplot(evsi_combo, aes(x = N, y = EVSI)) +  # colour = Parameter
  geom_line() +
  geom_point() +
  facet_wrap(~ Study, scales = "free_x") +
  geom_hline(aes(yintercept = EVPPI, linetype = "EVPPI"), data = evppi_combo) +
  scale_linetype_manual(name="",
                        values = c("EVSI" = "solid", "EVPPI" = "dashed")) +
  ggtitle("EVPSI for different study designs") +
  xlab("Sample size (n)") +
  ylab("$") +
  scale_x_continuous(breaks = number_ticks(5)) +
  scale_y_continuous(breaks = number_ticks(6), labels = dollar) +
  theme_bw(base_size = 14) +
  theme(legend.position = "bottom")
```

10 ENBS

```
# Population Values
# Discount rate
disc <- c(0.03)
# Technology lifetime
LT   <- 10
time <- seq(0, LT)
# Per Annum Number of Individuals to Be Treated With Urate Lowering Therapy
# Present prevalence
prev <- 0.010 # In millions(1e6)
# Annual Incidence
incid <- 147*1e-6 # In millions: 0.005*29.376e-3
# Total population afectd by technology calculated with `TotPop` function in Millions
tot_pop <- TotPop(time,    # Function
                  prev,
                  incid,
                  disc)

# Population EVPSI
# Obervational study
pop_evsi_obs <- evsi_obs
pop_evsi_obs$popEVSI <- pop_evsi_obs$EVSI*tot_pop
# RCT
pop_evsi_rct <- evsi_rct
pop_evsi_rct$popEVSI <- pop_evsi_rct$EVSI*tot_pop

# Cost of research
# Obervational study
cost_res_obs <- CostRes(fixed.cost = 10000e-6,
                        samp.size = n_obs,  # vector
                        cost.per.patient = 500e-6, # In Million $
                        INMB = 0,
                        clin.trial = FALSE)
# Data frame with cost of trial in Millions
cost_obs <- data.frame(N = n_obs, CS = cost_res_obs)
# RCT
cost_res_rct <- CostRes(fixed.cost = 8000000e-6,
                        samp.size = n_rct,  # vector
                        cost.per.patient = 8500e-6, # In Million $
                        INMB = 0,
                        clin.trial = TRUE)
# Data frame with cost of trial in Millions
cost_rct <- data.frame(N = n_rct, CS = cost_res_rct)

# Create ENBS data frame
enbs_obs <- merge(pop_evsi_obs, cost_obs, by = "N")
enbs_rct <- merge(pop_evsi_rct, cost_rct, by = "N")
# Compute ENBS
enbs_obs$ENBS <- enbs_obs$popEVSI - enbs_obs$CS
enbs_rct$ENBS <- enbs_rct$popEVSI - enbs_rct$CS
# Compute OSS (n*)
enbs_obs$nstar <- enbs_obs$N[which.max(enbs_obs$ENBS)]
```

```r
enbs_rct$nstar <- enbs_rct$N[which.max(enbs_rct$ENBS)]
# Append data frames
enbs_all <- rbind(enbs_obs,
                  enbs_rct)

oss <- summarise(group_by(enbs_all, Study),
                 MaxENBS = max(ENBS),
                 Nstar   = N[which.max(ENBS)])

# Plot ENBS, EVPSI and n*
# Create suitable data frames for plotting
enbs_obs_gg <- melt(enbs_obs[, -3], id.vars = c("Study", "N", "nstar"), value.name = "Million")
enbs_rct_gg <- melt(enbs_rct[, -3], id.vars = c("Study", "N", "nstar"), value.name = "Million")
# Append data frames for plotting
enbs_all_gg <- rbind(enbs_obs_gg,
                     enbs_rct_gg)
levels(enbs_all_gg$Study) <- c(paste("Observational; n* = ", comma(oss$Nstar[1]), sep=""),
                               paste("RCT; n* = ", comma(oss$Nstar[2]), sep=""))

ggplot(enbs_all_gg, aes(x = N, y = Million, colour = variable, group = variable)) +
  facet_wrap(~ Study, scales = "free_x") +
  # geom_segment(data = oss, aes(x = Nstar, y = 0, xend = Nstar, yend = MaxENBS)) +
  geom_hline(aes(yintercept=0), size = 0.7, linetype = 2, colour = "gray") +
  geom_vline(aes(xintercept = nstar), size = 0.7, linetype = 2, colour = "gray") +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = number_ticks(6), labels = comma)+
  scale_y_continuous(breaks = number_ticks(6), labels = comma, limits = c(0, 40))+
  scale_colour_hue("Study design ", l=50,
                   labels=c("popEVPSI(n) ", "Cost of Research(n) ", "ENBS(n) ")) +
  xlab("Sample size (N)") +
  ylab("Value (Million $)") +
  theme_bw(base_size = 14) +
  theme(legend.position = "bottom",
        panel.spacing = unit(2, "lines"))
```