# Survival Analysis - Sick-Sicker 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)

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. The Hospital for Sick Children, Toronto and 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. BioRxiv 670612 2019.https://www.biorxiv.org/content/10.1101/670612v1

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

```
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/darthtools", force = TRUE) Uncomment if there is a newer version
p_load_gh("DARTH-git/darthtools")
```

## 02 Load functions

```
source(here("functions", "VOI_Functions.R"))
source(here("functions", "GA_functions.R"))
```

## 03 Input model parameters

```
# Load simulation file
# Read the `.csv` simulation file into `R`.
toy <- read.csv(here("data", "psa_sick_sicker.csv"), header = TRUE)[, -1]
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
wtp <- 120000
toy$NMB_NoTrt <- wtp * toy$QALY_NoTrt - toy$Cost_NoTrt
toy$NMB_Trt <- wtp * toy$QALY_Trt - toy$Cost_Trt

nmb <- toy[, c("NMB_NoTrt", "NMB_Trt")]
head(nmb)

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

# Assign name of strategies
strategies <- c("No Trt", "Trt")
colnames(nmb) <- strategies
head(nmb)
```

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

# Plot NMB for different strategies
# 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 NoTrt vs Trt
inmb <- data.frame(Simulation = 1:n_sim,
                   `Trt vs. No Trt` = nmb$Trt - nmb$`No Trt`)

## 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 = txtsize)
```

# 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
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)
head(max_loss_i)

# Average across all states of the world
evpi <- mean(max_loss_i)
evpi
```

## 07 EVPPI

```
# Matrix with parameters
x <- toy[, c(1:14)]
head(x)

# Number and names of parameters
n_params <- ncol(x)
n_params
names_params <- colnames(x)
names_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)
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]))

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

```
# Select parameters with positive EVPPI
sel_params <- c(3, 4, 10, 12, 14)
n_params <- length(sel_params)
# Effective (prior) Sample size
n0 <- numeric(length(sel_params))
n0[1] <- 84+800    # p.S1S2 ~ Beta(84, 800)
n0[2] <- 10+2000   # p.HD ~ Beta(10,2000)
n0[3] <- 73.5      # cTrt ~ Gamma(73.5, 163.3) -> likelihood ~ Exponential
n0[4] <- 50        # u.S1 ~ N(.75, .02 / sqrt(50) = )
n0[5] <- 20        # u.Trt ~ N(.95, 0.02)

n <- c(0, 100, seq(200, 2000, by = 200))
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[sel_params]

# 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[[sel_params[p]]], n = n[nSamp], n0 = n0[p])
    Ltilde2 <- predict.ga(lmm2[[sel_params[p]]], n = n[nSamp], n0 = n0[p])
    ## Combine losses into one matrix
    Ltilde <- cbind(Ltilde1, Ltilde2)
    ### 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, 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[sel_params, ]) +
  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)
```