Expected value of more precise/context-relevant calibration target data

## Introduction

Calibration target data is one of the key elements of any calibration process. However, like any data, calibration targets are prone to precision and/or validity issues.

The Standard error (SE) is a measure of the precision of a statistical estimate. Specifically, it quantifies the variability or imprecision of a sample statistic, such as the mean or proportion, compared to the true population parameter. We quantify the Expected Value of Perfect Information (EVPI) as the value of reducing uncertainty in decision-making. The EVPI represents the maximum amount a decision-maker would be willing to pay for perfect information that eliminates all uncertainty associated with a decision, reducing the SE to zero.

Bias, on the other hand, refers to the systematic or consistent error in measurements or estimates. It can lead to inaccuracies in the results, and it can result from flaws in study design, data collection methods, or other systematic errors.

## Methods

### Model

We use a simple decision model with two inputs, x1 and x2. The model, shown in the code below, estimates mortality (used in the calibration process) and net benefit (to inform our decision).

# Define the model:  
model.function <- function(.x1, .x2) {  
 .model.mortality <- .x1 + .x2  
 .inb <- .x1 \* 100  
 return(list(model.mortality = .model.mortality, inb = .inb))  
}

### Calibration parameters

The prior of parameters x1 and x2 is a bivariate normal distribution.

# Define model parameters:  
## Priors:  
m1 <- 0  
m2 <- 190  
mu <- c(m1, m2)  
  
v1 <- 1000  
v2 <- 100  
rho <- 0  
Sigma <- matrix(c(v1, rho \* v1 \* v2, rho \* v1 \* v2, v2), 2, 2)  
  
set.seed(1)  
n <- 1e6  
x <- MASS::mvrnorm(n, mu, Sigma)  
x1 <- x[, 1]  
x2 <- x[, 2]

### Calibration target data

We simulate mortality data to be used as the calibration target. We start with d.n = 100 simulated population. The sample size of the simulated population affects the precision of the simulated calibration target.

# Simulate mortality data  
simulate\_mortatlity\_data <- function(d.mean = 200, d.var = 1000, d.n) {  
 d.sd <- sqrt(d.var)  
 d <- rnorm(d.n, d.mean, d.sd)  
   
 # compute calibration target from dummy data  
 c("mean" = mean(d), "sd" = sd(d))  
}  
set.seed(10)  
  
d.n = 100  
simulated\_mortatlity\_estimates <- simulate\_mortatlity\_data(d.n = d.n)  
  
target.mortality.mean <- simulated\_mortatlity\_estimates["mean"]  
target.mortality.mean

mean   
195.6819

target.mortality.sd <- simulated\_mortatlity\_estimates["sd"]  
target.mortality.sd

sd   
29.76449

We assume that the calibration target follows a normal distribution.

# Define a likelihood for calibration  
likelihood <- function(.parameter, .target.mean, .target.se) {  
 dnorm(.target.mean, .parameter, .target.se)  
}

## Calibration process

We employ the sampling importance re-sampling (SIR) method to sample from the joint posterior distribution of parameters x1 and x2.

# Evaluate the model using prior samples  
prior.output <- model.function(x1, x2)  
prior.inb <- prior.output$inb  
prior.evpi <- mean(pmax(prior.inb, 0)) - max(mean(prior.inb), 0)  
prior.evpi

[1] 1261.864

# Estimate the likelihood of prior samples  
likelihood.values <- likelihood(prior.output$model.mortality,  
 .target.mean = target.mortality.mean,  
 .target.se = target.mortality.sd / sqrt(d.n))  
  
# Re-sampling x1 and x2 with weights equal to the likelihood values  
  
posterior.index <- sample(1:n, n, replace = TRUE, prob = likelihood.values)  
posterior.x <- x[posterior.index, ]  
  
posterior.output <- model.function(posterior.x[, 1], posterior.x[, 2])  
  
posterior.inb <- posterior.output$inb  
  
mean(posterior.inb)

[1] 508.7615

sd(posterior.inb)

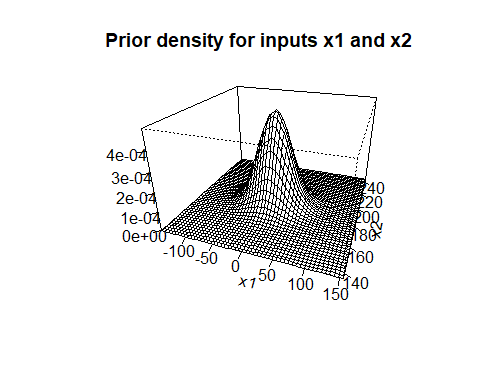
[1] 990.9914

posterior.evpi <- mean(pmax(posterior.inb, 0)) - max(mean(posterior.inb), 0)  
posterior.evpi

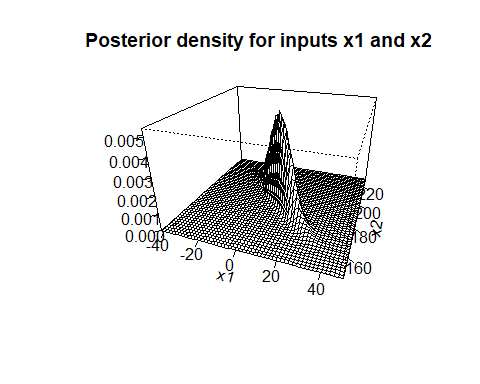
[1] 191.9165

### Graphical representations

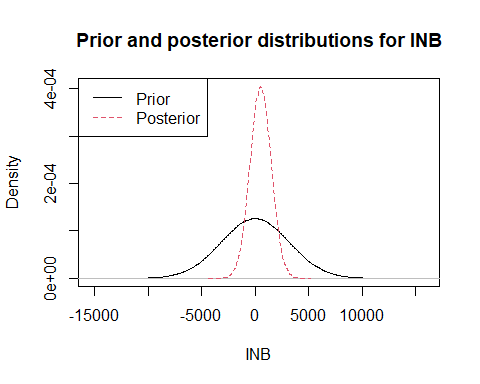
# Prior for inputs  
  
bivariate.density.prior <- MASS::kde2d(x1, x2, n = 50)  
  
#dev.new()  
persp(bivariate.density.prior,  
 theta = 20, phi = 25, expand = 0.6, ticktype = 'detailed',  
 xlab = "x1", ylab = "x2", zlab = "",  
 main = "Prior density for inputs x1 and x2")



# Save the parameters' posterior distribution:  
#savePlot(filename = paste0(report\_directoy, "1\_x1\_x2\_prior"), type = "tiff")  
  
# Posterior for inputs  
# note the induced correlation  
  
bivariate.density.posterior <- MASS::kde2d(posterior.x[, 1], posterior.x[, 2], n = 50)  
  
#dev.new()  
persp(bivariate.density.posterior,  
 theta = 20, phi = 25, expand = 0.6, ticktype = 'detailed',  
 xlab = "x1", ylab = "x2", zlab = "",  
 main = "Posterior density for inputs x1 and x2")



# Save the parameters' posterior distribution:  
#savePlot(filename = paste0(report\_directoy, "1\_x1\_x2\_post"), type = "tiff")  
  
# Prior and posterior for INB output  
  
#dev.new()  
den.prior <- density(prior.inb)  
den.post <- density(posterior.inb)  
plot(den.prior,  
 xlim = c(min(den.prior$x, den.post$x), max(den.prior$x, den.post$x)),  
 ylim = c(min(den.prior$y, den.post$y), max(den.prior$y, den.post$y)),  
 main = "Prior and posterior distributions for INB",  
 xlab = "INB"  
)  
  
lines(den.post, col = 2, lty = 2)  
legend(x = "topleft", legend = c("Prior", "Posterior"), col = c(1, 2), lty = c(1, 2))



# Save the INB:  
#savePlot(filename = paste0(report\_directoy, "1\_pre\_post\_inmb"), type = "tiff")

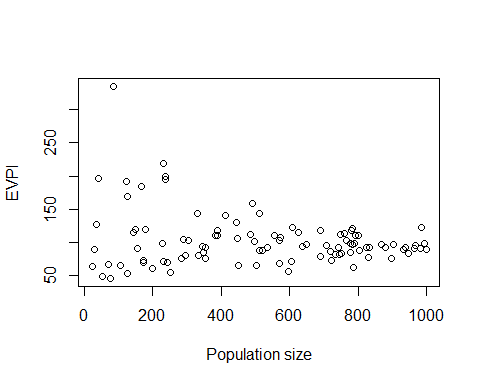
## Precision and EVPI

Below, we demonstrate the association between the sample size of the dummy calibration target data (which affects the SE) and the EVPI.

# Generate a vector of different population sizes:  
pop\_size <- runif(100, 20, 1000) |> round()  
# Empty results object:  
SE\_EVPI <- matrix(ncol = 2, nrow = length(pop\_size))  
# Loop over different precision values:  
set.seed(10)  
for (i in 1:length(pop\_size)) {  
 SE\_EVPI [i, 1] <- pop\_size[i]  
 d.n <- pop\_size[i]  
 # Estimate the target data moments:  
 simulated\_mortatlity\_estimates <- simulate\_mortatlity\_data(d.n = d.n)  
   
 target.mortality.mean <- simulated\_mortatlity\_estimates["mean"]  
   
 target.mortality.sd <- simulated\_mortatlity\_estimates["sd"]  
   
 # Estimate the likelihood of prior samples  
 likelihood.values <- likelihood(  
 prior.output$model.mortality,  
 .target.mean = target.mortality.mean,  
 .target.se = target.mortality.sd / sqrt(d.n)  
 )  
   
 # Re-sampling x1 and x2 with weights equal to the likelihood values  
   
 posterior.index <- sample(1:n, n, replace = TRUE, prob = likelihood.values)  
 posterior.x <- x[posterior.index, ]  
   
 posterior.output <- model.function(posterior.x[, 1], posterior.x[, 2])  
  
 # Compute posterior NMB and EVPI   
 posterior.inb <- posterior.output$inb  
  
 SE\_EVPI [i, 2] <- mean(pmax(posterior.inb, 0)) - max(mean(posterior.inb), 0)  
}  
  
head(SE\_EVPI)

[,1] [,2]  
[1,] 790 98.40224  
[2,] 510 143.10178  
[3,] 624 114.77717  
[4,] 830 78.06968  
[5,] 751 84.07079  
[6,] 382 110.70007

plot(SE\_EVPI,  
 xlab = "Population size",  
 ylab = "EVPI"  
)



## Expected value of perfect calibration target data

Below, we estimate the INB and EVPI from a nested loop Monte Carlo, where the outer loop allows the propagation of the uncertainty associated with the calibration target.

# Empty results object:  
target\_nmb <- matrix(ncol = 2, nrow = 100)  
# Loop over different target data mean values:  
set.seed(10)  
d.n = 100  
simulated\_mortatlity\_estimates <- simulate\_mortatlity\_data(d.n = d.n)  
  
target.mortality.mean <- simulated\_mortatlity\_estimates["mean"]  
  
target.mortality.sd <- simulated\_mortatlity\_estimates["sd"]  
target.mortality.se <- target.mortality.sd / sqrt(d.n)  
  
set.seed(10)  
for (i in 1:nrow(target\_nmb)) {  
 # Sample a target data mean.  
 target\_nmb[i, 1] <- rnorm(1, target.mortality.mean, target.mortality.se)  
   
 # Estimate the likelihood of prior samples  
 likelihood.values <- likelihood(  
 prior.output$model.mortality,  
 .target.mean = target\_nmb[i, 1],  
 .target.se = target.mortality.sd / sqrt(d.n)  
 )  
   
 # Re-sampling x1 and x2 with weights equal to the likelihood values  
   
 posterior.index <- sample(1:n, n, replace = TRUE, prob = likelihood.values)  
 posterior.x <- x[posterior.index, ]  
   
 posterior.output <- model.function(posterior.x[, 1], posterior.x[, 2])  
  
 # Compute posterior NMB  
 target\_nmb[i, 2] <- mean(posterior.output$inb)  
}  
  
head(target\_nmb)

[,1] [,2]  
[1,] 195.7377 510.8885  
[2,] 194.7867 427.3745  
[3,] 197.6558 685.1252  
[4,] 197.5476 676.9959  
[5,] 195.3825 479.8399  
[6,] 187.8057 -196.0719

# Compute posterior EVPI  
target\_data\_evpi <- mean(pmax(target\_nmb[i, 2], 0)) - max(mean(target\_nmb[i, 2]), 0)  
target\_data\_evpi

[1] 0

## Expected value of sample information

# Empty results object:  
target\_nmb <- matrix(ncol = 2, nrow = 100)  
# Loop over different target data mean values:  
set.seed(10)  
d.n = 100  
simulated\_mortatlity\_estimates <- simulate\_mortatlity\_data(d.n = d.n)  
  
target.mortality.mean <- simulated\_mortatlity\_estimates["mean"]  
  
target.mortality.sd <- simulated\_mortatlity\_estimates["sd"]  
  
set.seed(10)  
for (i in 1:nrow(target\_nmb)) {  
 # Sample a target data mean.  
 target\_nmb[i, 1] <- rnorm(1, target.mortality.mean, target.mortality.sd)  
   
 # Estimate the likelihood of prior samples  
 likelihood.values <- likelihood(  
 prior.output$model.mortality,  
 .target.mean = target\_nmb[i, 1],  
 .target.se = target.mortality.sd / sqrt(d.n)  
 )  
   
 # Re-sampling x1 and x2 with weights equal to the likelihood values  
   
 posterior.index <- sample(1:n, n, replace = TRUE, prob = likelihood.values)  
 posterior.x <- x[posterior.index, ]  
   
 posterior.output <- model.function(posterior.x[, 1], posterior.x[, 2])  
  
 # Compute posterior NMB  
 target\_nmb[i, 2] <- mean(posterior.output$inb)  
}  
  
head(target\_nmb)

[,1] [,2]  
[1,] 196.2399 557.3648  
[2,] 186.7296 -289.6901  
[3,] 215.4210 2294.9244  
[4,] 214.3381 2194.8717  
[5,] 192.6877 236.7353  
[6,] 116.9196 -6605.0033

# Compute posterior EVSI  
evsi <- mean(pmax(target\_nmb[i, 2], 0)) - max(mean(posterior.inb, 0))  
evsi

[1] -509.2903