EVSI calculation for model development

Abdollah Safari

2025-04-18

Introduction

This is a step-by-step vignette for EVSI calculation for model development. The methodology used in this vignette can be found from our manscript titled “Expected value of sample information calculations for risk prediction model development” available at https://arxiv.org/abs/2410.03096 (https://arxiv.org/abs/2410.03096).

Dataset

As a case study, we apply our findings to GUSTO-I, a clinical trial of multiple thrombolytic strategies for acute myocardial infarction (AMI, or ‘heart attack’) (GUSTO Investigators, 1993). The GUSTO trial was a four-arm randomized clinical trial that compared the efficacy of different intravenous thrombolytic regimens for AMI. This dataset has been widely used for methodological research in predictive analytics.

We use these data to build a prediction model for predicting 30-day mortality for AMI. We used sex, age, diabetes (yes or no), milocc (which part of the heart muscle is affected by the blocked artery and coded as 3=anterior, 4=inferior, 5=other, and 6=no MI), previous MI, hypertension, smoking, killip class, and treatment as covariates in the model. Here is the top a few rows of this dataset:

```
data("gusto")

gusto %>%
  select(day30, sex, age, dia, miloc, pmi, hyp, smk, Killip, tx) %>%
  head
```

```
##   day30  sex    age dia    miloc pmi hyp     smk Killip     tx
## 1     0 male 19.027   0 Anterior  no   0   never      I     SK
## 2     0 male 19.031   0 Anterior yes   0   never     II SK+tPA
## 3     0 male 20.289   0 Inferior  no   0 current      I SK+tPA
## 4     0 male 20.781   0 Inferior  no   0   never      I    tPA
## 5     0 male 20.969   0 Anterior  no   0    quit      I     SK
## 6     0 male 20.984   0 Inferior  no   0   never      I    tPA
```

We set a risk threshold of 1% as the default threshold, but will evaluate the model at the entire possible range of thresholds. Given the relatively large sample size (n=40830), the uncertainty in predictions when the model is fitted using the full sample is negligible. Indeed, a previous study reported an EVPI of 0 with the full dataset

(thus, EVSI will be zero for all sample sizes). Therefore, we took, without replacement, a random sub-sample of 1000 patients from GUSTO data and treat that as our development sample (source of current information).

Initial setup

```
bootstrap <- function (n, Bayesian = T)
{
  if(Bayesian)
  {
    u <- c(0, sort(runif(n - 1)), 1)
    return((u[-1] - u[-length(u)]) * n)
  }
  else
  {
    u <- rmultinom(1, n, rep(1 / n, n))
    return(u[ , 1])
  }
}

winner_finder <- function(x) {
  max(x)
}


settings <- list()
settings$master_formula <-
  "y ~ sex + age + dia + miloc + pmi + htn + smk + kill + tx - 1"
settings$n_rep <- 10000 # of times repeating whole process
settings$initial_n <- 1000 # The development initial sample size
settings$future_n <- c(100, 500, 1000, 2000, 4000, 8000, 16000, 32000)

## development sample (a subsample of gusto)
set.seed(1234)
dev_rct <-
  gusto %>%
  filter(complete.cases(.),
         regl %in% c(1, 7, 9, 10, 11, 12, 14, 15)) %>% ## US region
  mutate(kill = (as.numeric(Killip) > 1) * 1) %>%
  sample_n(size = settings$initial_n, replace = F) %>%
  rename(y = day30)# %>%

dev_glm <- glm(as.formula(settings$master_formula),
               family = binomial(link = "logit"),
               data = dev_rct)
dev_rct$pi <- predict(dev_glm, type = "response")
```

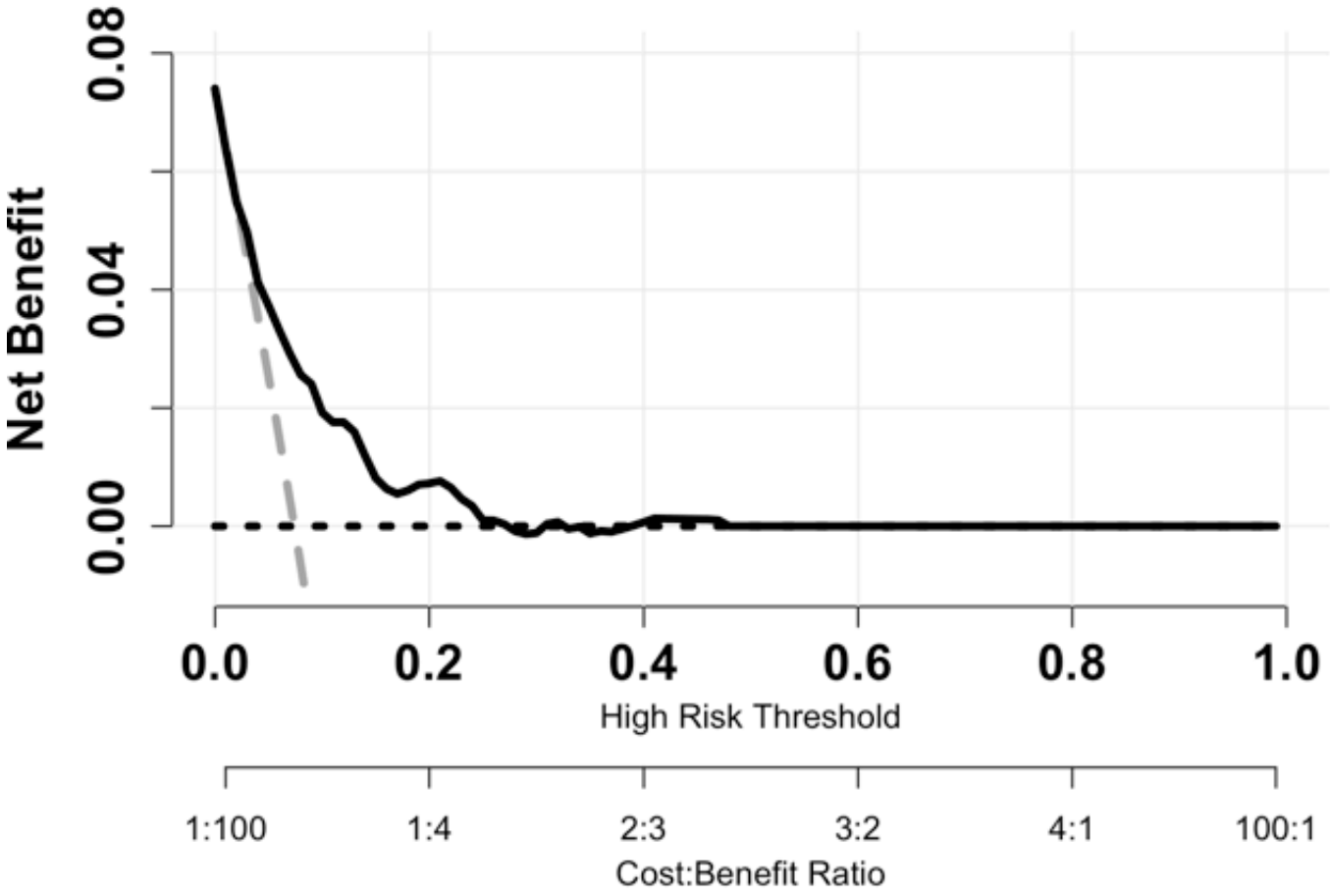
DCA

For DCA, we estimated NB of the model and the alternative default strategies (treat all and treat none) at different threshold values as well as estimated NB increments of model versus the best alternative.

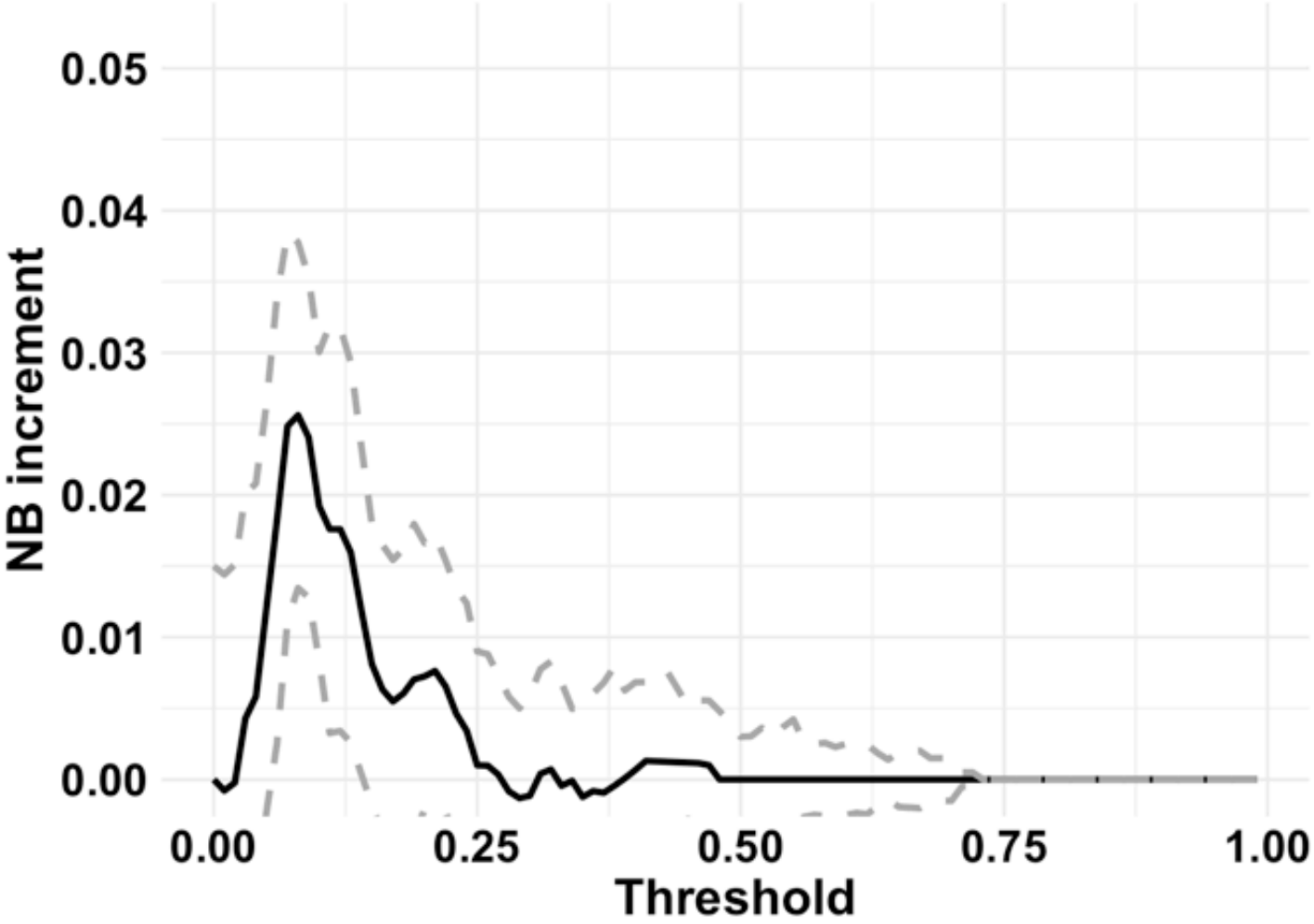
```
library(rmda)

set.seed(123)
# first use rmda with the default settings (set bootstraps = 50 here to reduce comput
ation time).
baseline.model <- decision_curve(as.formula(settings$master_formula),
                                 data = dev_rct,
                                 study.design = "cohort",
                                 policy = "opt-in",  #default
                                 bootstraps = 50)
```

```
## Note:  The data provided is used to both fit a prediction model and to estimate th
e respective decision curve. This may cause bias in decision curve estimates leading
to over-confidence in model performance.
```

```
#plot the curve
plot_decision_curve(baseline.model,
                    lty = c(1 : 3),
                    standardize = FALSE,
                    col = "black",
                    curve.names = "Using model",
                    legend.position = "none",
                    confidence.intervals = FALSE,
                    ylim = c(-0.01, 0.08),
                    lwd = rep(4, 3),
                    cex.lab = 1.5, cex.axis = 1.5, cex.sub = 1.5,
                    font = 2, font.lab = 2, font.sub = 2,
                    fg = "black")
```


Net Benefit
0.08
0.04
0.00
0.0
0.2
0.4
0.6
0.8
1.0
High Risk Threshold
1:100
1:4
2:3
3:2
4:1
100:1
Cost:Benefit Ratio

```
baseline.model$derived.data %>%
  select(model, thresholds,
         # sNB, sNB_lower, sNB_upper,
         NB, NB_lower, NB_upper) %>%
  mutate(NB_sd = (NB_upper - NB_lower) / (2 * 1.96),
         model = ifelse(model %in% c("All", "None"),
                        model, "Model")) %>%
  arrange(thresholds) %>%
  filter(thresholds < 1) %>%
  group_by(thresholds) %>%
  mutate(#NB_max = max(NB),
         NB_alt_max = max(NB[model != "Model"]),
         NB_alt_argmax = which.max(NB[model != "Model"]),
         NB_alt_max_sd = (NB_sd[model != "Model"])[NB_alt_argmax]) %>%
  ungroup() %>%
  select(! c(NB_upper, NB_lower, NB_alt_argmax)) %>%
  filter(model == "Model") %>%
  mutate(NB_inc = NB - NB_alt_max,
         NB_inc_sd = ifelse(NB_inc == 0,
                            0,
                            # sqrt(NB_sd ^ 2 + NB_alt_max_sd ^ 2)
                            NB_sd)) %>%
  ggplot(aes(x = thresholds, y = NB_inc)) +
  geom_line(aes(color = "Model NB increment"), linewidth = 1.2,
            show.legend = TRUE, linetype = 1) +
  geom_line(aes(y = NB_inc + 1.96 * NB_sd, color = "95% CI"),
            show.legend = TRUE, linetype = 2, linewidth = 1.2) +
  geom_line(aes(y = NB_inc - 1.96 * NB_sd), linetype = 2,
            color = "darkgray", linewidth = 1.2) +
  labs(y = "NB increment", x = "Threshold", colour = "") +
  theme_minimal() +
  coord_cartesian(ylim = c(0, 0.052)) +
  scale_color_manual(breaks = c("Model NB increment", "95% CI"),
                     values = c("black", "darkgrey")) +
  theme(legend.position = "none",
        axis.text = element_text(size = 17,
                                 color = "black",
                                 face = "bold"),
        axis.title = element_text(size = 19,
                                  color = "black",
                                  face = "bold"),
        legend.text = element_text(size = 17,
                                   color = "black",
                                   face = "bold"),
        legend.title = element_text(size = 17,
                                    color = "black",
                                    face = "bold"))
```



The above figure shows the NB of the model as well as the alternative default strategies (treat none and treat all). The model had higher expected NB than the alternative strategies at the all threshold values of interest. Additionally, the second figure shows the difference between the estimated NB of the model and that of the best alternative default strategy, i.e., ENB(1,d)−max(0,ENB(2,d)), along with its bootstrap 95% CI. Again, the model had a higher estimated NB than the default alternatives at the entire range of threshold values of interest.

EVPI vs EVSI

Next, we will calculate EVPI and EVSI for model developemnt.

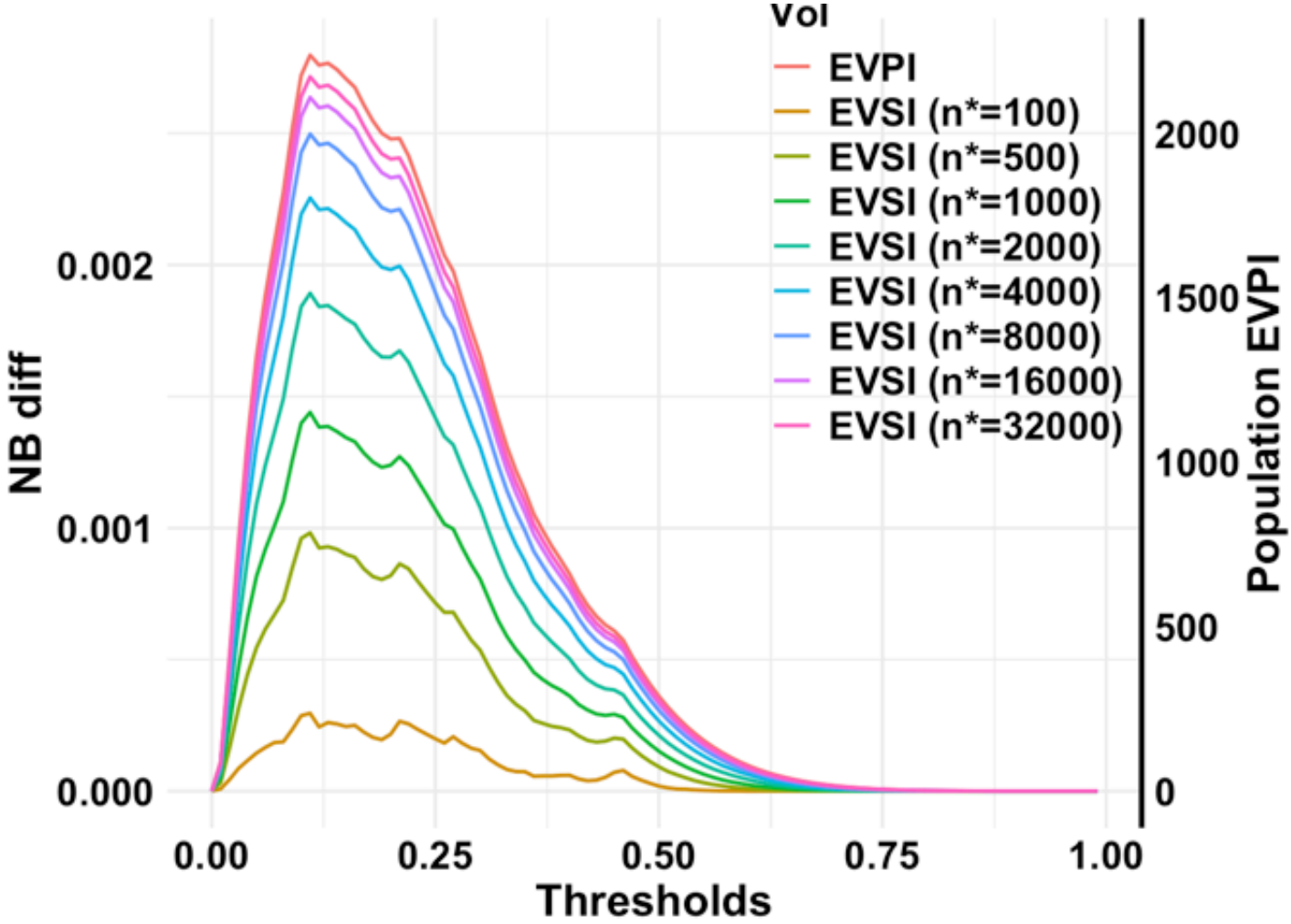
```
lambdas <- (0 : 99) / 100
strategy_names <- c("none", "model", "all", "max")
NB_array <-
  array(0, dim = c(length(strategy_names),
                   length(lambdas)),
        dimnames = list(strategy_names,
                        lambdas))
NB_array["none", ] <- 0
strategy_names_future <- c("none", "model", "all", "max")
NB_future <- array(0,
                   dim = c(length(strategy_names_future),
```

```
                          length(lambdas),
                          length(settings$future_n)),
                  dimnames = list(strategy_names_future,
                                  lambdas,
                                  settings$future_n))
NB_future["none", , ] <- 0

for(t in 1 : settings$n_rep) {
  dev_rct_t <- dev_rct

  dev_rct_t$w <- bootstrap(settings$initial_n, Bayesian = TRUE)
  true_model_t <- glm(settings$master_formula,
                      family = binomial(link = "logit"),
                      data = dev_rct_t, weights = dev_rct_t$w)

  dev_rct_t$p <- predict(true_model_t, type = "response") # true p's

  nb_model_tmp <- sapply(lambdas, function(x) {
    mean((dev_rct_t$p - (1 - dev_rct_t$p) * x /
            (1 - x)) * (dev_rct_t$pi > x))})
  NB_array["model", ] <- NB_array["model", ] + nb_model_tmp
  nb_all_tmp <- sapply(lambdas, function(x) {
    mean((dev_rct_t$p - (1 - dev_rct_t$p) * x / (1 - x)) * (1))})
  NB_array["all", ] <- NB_array["all", ] + nb_all_tmp

  NB_array["max", ] <- NB_array["max", ] +
    sapply(lambdas, function(x) {
      mean((dev_rct_t$p - (1 - dev_rct_t$p) * x /
              (1 - x)) * (dev_rct_t$p > x))})

  future_rct_n_max <- max(settings$future_n)
  future_rct_indices <- sample(c(1 : settings$initial_n),
                               size = future_rct_n_max,
                               prob = dev_rct_t$w,
                               replace = TRUE)
  future_rct_all <- dev_rct_t[future_rct_indices , ]
  # new Y's only for future rct
  future_rct_all$y <- rbinom(n = future_rct_n_max, size = 1,
                             future_rct_all$p)

  # s = 1
  for (s in 1 : length(settings$future_n))
  {
    dev_future_rct <-
      rbind(dev_rct_t, future_rct_all[c(1 : settings$future_n[s]) , ])

    dev_future_model <- glm(settings$master_formula,
                            family = binomial(link = "logit"),
                            data = dev_future_rct)
    dev_future_rct$pi_future <- predict(dev_future_model, type = "response")
```

```
    dev_rct_t$pi_from_future <- predict(dev_future_model,
                                        newdata = dev_rct_t,
                                        type = "response")

    nb_future_model_tmp <- sapply(lambdas, function(x) {
      mean((dev_rct_t$p - (1 - dev_rct_t$p) *
              x / (1 - x)) * (dev_rct_t$pi_from_future > x))})
    NB_future["model", , as.character(settings$future_n[s])] <-
      NB_future["model", , as.character(settings$future_n[s])] +
      nb_future_model_tmp
    nb_future_all_tmp <- sapply(lambdas, function(x) {
      mean((dev_rct_t$p - (1 - dev_rct_t$p) *
              x / (1 - x)) * (1))})
    NB_future["all", , as.character(settings$future_n[s])] <-
      NB_future["all", , as.character(settings$future_n[s])] +
      nb_future_all_tmp

    NB_future["max", , as.character(settings$future_n[s])] <-
      NB_future["max", , as.character(settings$future_n[s])] +
      apply(cbind(nb_future_model_tmp, nb_future_all_tmp, 0), 1, max)

    dev_rct_t %<>%
      select(! c(pi_from_future))
  }
}

NB_array_mean <- NB_array / settings$n_rep
evpi_max <-
  apply(NB_array_mean, 2, function(x) x[4] - max(x[-c(4, 5)]))


NB_future_mean <- NB_future / settings$n_rep
evsi <- matrix(NA, nrow = length(lambdas), ncol = length(settings$future_n))
evsi_orgSample <-
  apply(NB_array_mean, 2, function(x) winner_finder(x[-c(4 : 5)]))
for (i in c(1 : length(settings$future_n))) {
  NB_future_mean_tmp <- NB_future_mean[ , , as.character(settings$future_n[i])]
  evsi_futSample_temp <- apply(NB_future_mean_tmp, 2, function(x) winner_finder(x[-
4]))
  evsi[ , i] <- evsi_futSample_temp - evsi_orgSample
}

tibble(Thresholds = lambdas,
       evpi = evpi_max,
       evsi_1 = evsi[ , 1],
       evsi_2 = evsi[ , 2],
       evsi_3 = evsi[ , 3],
       evsi_4 = evsi[ , 4],
       evsi_5 = evsi[ , 5],
       evsi_6 = evsi[ , 6],
```

```
      evsi_7 = evsi[ , 7],
      evsi_8 = evsi[ , 8]) %>%
  pivot_longer(cols = c(evpi, evsi_1 : evsi_8),
               names_to = "VoI",
               values_to = "NB diff") %>%
  mutate(VoI = factor(VoI,
                      levels = c("evpi",
                                 paste0("evsi_", c(1 : 8))),
                      labels = c("EVPI",
                                 paste0("EVSI (n*=", settings$future_n, ")")))) %>%
  ggplot(aes(x = Thresholds, y = `NB diff`,
             group = VoI, color = VoI)) +
  geom_line(size = 0.7) +
  lims(y = c(0, 0.003)) +
  scale_y_continuous(sec.axis = sec_axis(~ 800000 * .,
                                         name = "Population EVPI")) +
  theme_minimal() +
  theme(legend.position = c(0.8, 0.75),
        axis.text = element_text(size = 15,
                                 color = "black",
                                 face = "bold"),
        axis.title = element_text(size = 17,
                                  color = "black",
                                  face = "bold"),
        axis.line.y.right = element_line(color = "black",
                                         linetype = 1,
                                         linewidth = 1),
        axis.title.y.right = element_text(angle = 90),
        legend.text = element_text(size = 15,
                                   color = "black",
                                   face = "bold"),
        legend.title = element_text(size = 15,
                                    color = "black",
                                    face = "bold"))
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```



The above EVPI figure illustrates computed EVSI (right Y-axis) for various sample sizes (dashed and dotted lines) at different threshold values, z, alongside the computed EVPI (black curve). Both EVPI and EVSI quantify the expected NB loss per decision each time the model is applied. The overall NB loss attributable to uncertainty is influenced by the expected frequency of the medical decision being made (Willan and others, 2012). To scale the VoI metrics (EVPI and EVSI) to the population level, given that approximately 800,000 acute myocardial infarctions (AMIs) occur annually in the United States (Virani and others, 2021), we provide scaled VoI values in terms of true positive units on the right Y-axis of the figure. At a threshold of 0.01, where the EVPI is 0.0001, a future development study could yield a maximum benefit equivalent to identifying 89 additional true positive cases (individuals who will die within 30 days and are correctly classified as high risk) per year, or avoiding 8,761 false positive cases (individuals who will not die within 30 days but are incorrectly classified as high risk) annually.

EVSI curve

Next, we Compute EVSI (left Y-axis) and its scaling at population (right Y-axis) with different future sample sizes (red curve) at threshold value of z= 0.01 compared to its corresponding computed EVPI (black curve). Dotted/dashed vertical curves are the estimated sample sizes based on the three criteria of Riley and others (2024).

```
dev_rct_lp <- predict(dev_glm, type = "link")
library(pROC)
dev_roc <- roc(dev_rct$y ~ dev_rct$pi, print.auc = TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
library(pmsampsize)
Riley_def_ss <-
  pmsampsize(type = "b",
             cstatistic = dev_roc$auc,
             parameters = 13,
             prevalence = mean(dev_rct$y),
             shrinkage = 0.9)

(thresh_z <- which(names(evpi_max) == "0.01"))
```

```
## [1] 2
```

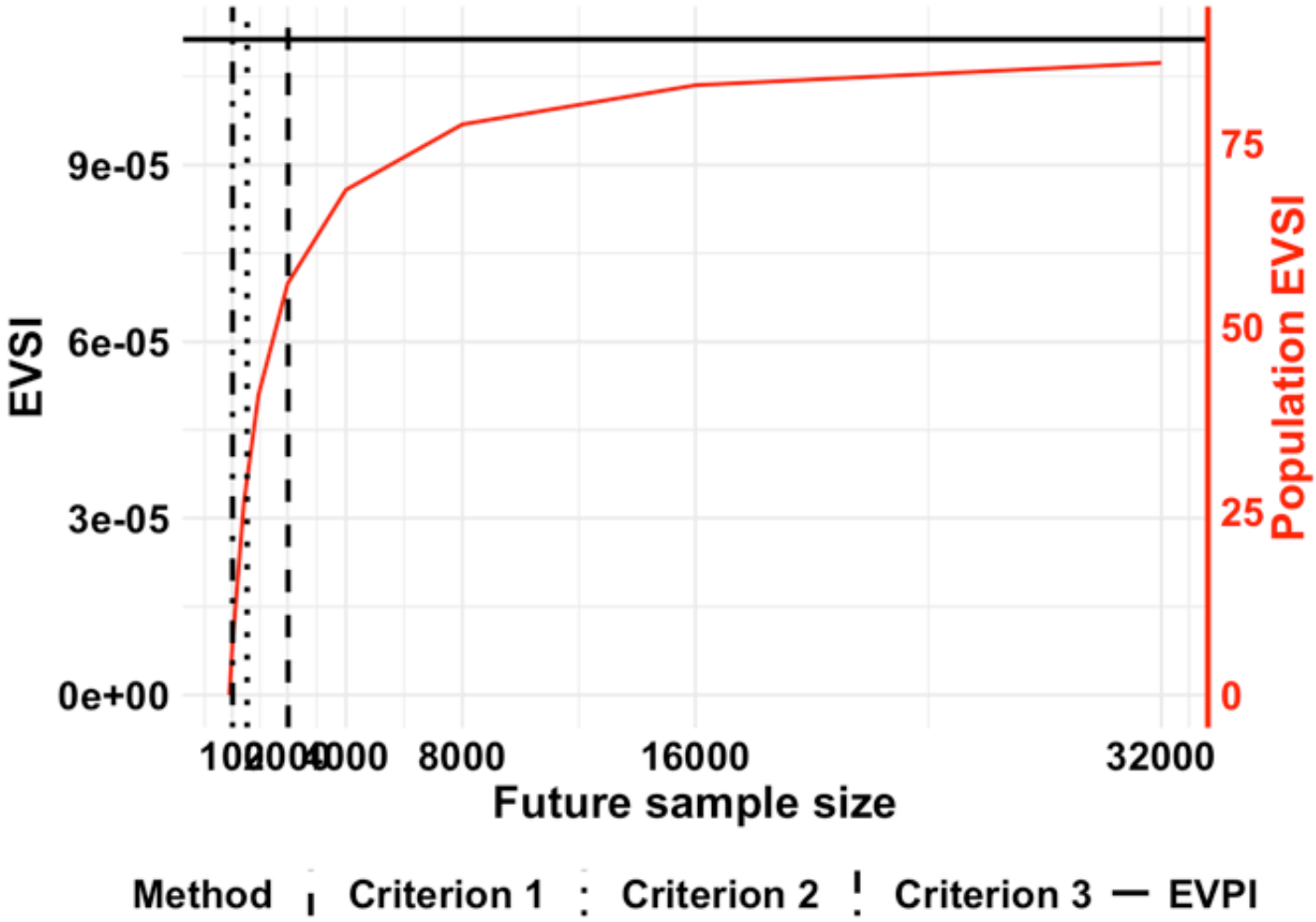
```
lambdas[thresh_z]
```

```
## [1] 0.01
```

```
evpi_max[thresh_z]
```

```
##         0.01
## 0.0001112755
```

```
tibble(Sample_size = c(0, settings$future_n),
       evsi = c(0, evsi[thresh_z , ])) %>%
  ggplot(aes(x = Sample_size, y = evsi)) +
  geom_line(size = 0.7, color = "red") +
  geom_hline(aes(yintercept = evpi_max[thresh_z],
                 linetype = "EVPI"),
             linewidth = 1) +
  geom_vline(aes(xintercept = Riley_def_ss$results_table[1, 1],
                 linetype = "Criterion 1"),
             linewidth = 1) +
  geom_vline(aes(xintercept = Riley_def_ss$results_table[2, 1],
                 linetype = "Criterion 2"),
             linewidth = 1) +
  geom_vline(aes(xintercept = Riley_def_ss$results_table[3, 1],
                 linetype = "Criterion 3"),
             linewidth = 1) +
  scale_linetype_manual(name = "Method",
                        values = c(EVPI = 1,
                                   # `Riley et al. (2018) - Criterion 1` = 2,
                                   `Criterion 1` = 2,
                                   `Criterion 2` = 3,
                                   `Criterion 3` = 4)) +
  scale_x_continuous(breaks = settings$future_n[-c(2, 3)]) +
  scale_y_continuous(sec.axis = sec_axis(~800000 * .,
                                         name = "Population EVSI")) +
  theme_minimal() +
  labs(x = "Future sample size", y = "EVSI") +
  theme(legend.position = "bottom",
        axis.text = element_text(size = 15,
                                 color = "black",
                                 face = "bold"),
        axis.title = element_text(size = 17,
                                  color = "black",
                                  face = "bold"),
        axis.line.y.right = element_line(color = "red",
                                         linetype = 1,
                                         linewidth = 1),
        axis.title.y.right = element_text(angle = 90,
                                          color = "red"),
        axis.text.y.right = element_text(color = "red"),
        legend.text = element_text(size = 15,
                                   color = "black",
                                   face = "bold"),
        legend.title = element_text(size = 15,
                                    color = "black",
                                    face = "bold"),
        axis.minor.ticks.x.bottom = element_blank())
```



As anticipated, EVSI increases with larger future sample sizes, eventually asymptoting to the EVPI. The expected NB gain (left Y-axis) exhibits a steep increase at smaller sample sizes, followed by a plateau as n_* grows, reflecting a “diminishing return” pattern. Similar to the EVPI plot in Panel A, we scale the EVSI to the population level, with the corresponding values displayed on the right Y-axis of the figure. At 0.01 threshold, a future study of size n_*= 1,000 has a per-decision EVSI value of 0.000049, corresponding to population value of 39 in true positive cases gained (or 3,881 in false positive cases averted), while a future study of size n= 16,000 has a per-decision EVSI value of 0.00010, corresponding to population value of 82 in true positive cases gained (or 8,141 in false positive cases averted).