Simulated example

Sami Haidar-Wehbe, Sam Emerson, Louis Aslett, James Liley

2025-04-08

Introduction

Risk scores to predict future events are widely used in many fields. Deployment of a predictive risk score is usually with the intent of avoiding such future events (or sometimes bringing them about), in which case the risk score is often intended to be used to plan anticipatory interventions which avoid the event in an efficient way (e.g., Hippisley-Cox, Coupland, and Brindle (2017), Nashef et al. (2012), Koopman and Mainous (2008)).

This risk-score-directed anticipatory action can mean that, when analysed retrospectively, it appears the risk score is performing poorly, since high-risk samples have undergone risk-score directed intervention (Lenert, Matheny, and Walsh (2019),Sperrin et al. (2019)). This leads to difficulties updating risk scores (Perdomo et al. 2020). A simple 'fix' allowing risk scores to be updated safely is to with-hold risk score generation and distribution for a random subset of the population, on whom an updated risk score can then be trained (Liley et al. 2021).

Such a 'hold-out' set of individuals should not be too small, since this will lead to excessive inaccuracy in the risk score trained on it, nor too large, since those individuals in the hold-out set miss out on any benefit. Our manuscript (partly) and this vignette (predominantly) serve to demonstrate the emergence of an optimal holdout set size under 'natural' conditions. We will also demonstrate the dependence of this optimal holdout size on system parameters, particularly the 'learning curve' of the risk score.

Setup

Assumptions

Suppose we begin with a population of N samples at which we observe p covariates, represented by an $N \times p$ matrix X. Each sample has an outcome of either 1 or 0. We denote by Y the set of N outcomes which would occur should no risk score guided intervention take place. We presume that our N samples represents our entire population, so we have the option of fitting a risk score to some 'hold-out' subset of the N samples, and using this risk score it to pre-empt the outcome on the remaining 'intervention' subset.

Although this obviously does not make sense temporally, we can think of the 'intervention' set as coming from a subsequent 'epoch' representing a period in which a particular risk score is used. At each epoch, we hold out a fixed proportion of the N samples as a hold-out set.

We presume that we may take a binary action to attempt to avoid the event $Y = 1$, and that we take this action if assessed risk is > 0.5. Thus the eventual cost for any sample can be summarised with a contingency table (true positive, false negative etc). We assume that in the intervention set, some better-than-random 'baseline' assessment of risk is made, corresponding to whatever the protocol was before a risk score was introduced. For this better-than random assessment, we use the Bayes-optimal (oracle) predictor on a single covariate.

We consider an underlying generative model for $f(x) = \text{logit}\{P(Y = 1|X = x)\}$ as either

- **(a)** linear or
- **(b)** non-linear (including interaction terms)

in x. We consider risk scores fitted using either

- **(i)** logistic models with no interaction terms or
- **(ii)** random forests

Model type **(i)** can approximate the oracle (Bayes-optimal) predictor for $Y|X$ arbitrarily well in setting **(a)** but not **(b)**. Model type **(ii)** can approximate the Bayes-optimal arbitrarily well in either setting, but will be less accurate than model type **(i)** in setting **(a)**.

General setup

We briefly set up general parameters for this simulation

```
# Set random seed
seed=1234
set.seed(seed)

# Packages
library("ranger")            # Random forests
library("OptHoldoutSize")    # Functions for OHS estimation

# Initialisation of patient data
n_iter <- 500           # Number of point estimates to be calculated
nobs <- 5000            # Number of observations, i.e patients (N)
npreds <- 7             # Number of predictors (p)

# Parameters to loop over
families <- c("log_reg","rand_forest")    # Model family: (i,ii) respectively
interactions <- c(0,10)                   # Number of interaction terms: (a,b) respectively

# The baseline assessment uses an oracle (Bayes-optimal) predictor on max_base_powers covariates.
max_base_powers <- 1

# Check holdout sizes within these two proportions of total
min_frac <- 0.02
max_frac <- 0.15
num_sizes <- 50

# Fraction of patients assigned to the holdout set
frac_ho <- seq(min_frac, max_frac, length.out = num_sizes)


# Cost function: total cost is defined from the continengency table, as follows:
c_tn <- 0 # Cost of true neg
c_tp <- 0.5 # Cost of true pos
c_fp <- 0.5 # Cost of false pos
c_fn <- 1 # Cost of false neg
```

Results

Emergence of optimal holdout size

For each holdout size, we simulate 500 datasets X, Y, using the same underlying function for $P(Y = 1|X = x)$ (that is, the same coefficients) and evaluate the costs in the holdout and intervention sets and in total.

Detailed code to run simulation is shown below (not run)

Show detailed code

```r
# Cost of true negative, false positive etc for computing total cost
cost_mat <- rbind(c(c_tn, c_fp), c(c_fn, c_tp))

# Generate coefficients for underlying model and for predictions in h.o. set
set.seed(seed + 1)

# Set ground truth coefficients, and the accuracy at baseline
coefs_general <- rnorm(npreds,sd=1/sqrt(npreds))
coefs_base <- gen_base_coefs(coefs_general, max_base_powers = max_base_powers)

# Run simulation
old_progress=0
for (ninters in interactions) {

  # If ninters>0, append coefficients for interaction terms to coefficients
  if (ninters>0)
    coefs_generate=c(coefs_general,rnorm(ninters,sd=2/sqrt(npreds)))
  else
    coefs_generate=coefs_general

  for (family in families) {

    ## Initialise arrays to populate

    # Record total cost in intervention and holdout sets of various sizes over n_iter resamplings
    costs_inter_resample <- costs_ho_resample <- matrix(nrow = n_iter, ncol = num_sizes)

    # Record total cost (cost in intervention set + cost in holdout set)
    costs_tot_resample <- array(0, dim = c(max_base_powers, n_iter, num_sizes))

    # Standard deviations of cost estimates at each holdout size
    costs_sd <- array(0, dim = c(max_base_powers, num_sizes))


    # Sweep through h.o. set sizes of interest
    for (i in 1:num_sizes) {

      # Sweep through resamplings
      for (b in 0:n_iter) {

        progress <- 100 * ((((i - 1) * n_iter) + b) / (num_sizes * (n_iter + 1))
        if (abs(floor(old_progress) - floor(progress)) > 0) {
          cat(floor(progress), "%\n")
        }

        # Set random seed
        set.seed(seed + b + i*n_iter)

        # Generate dataset
        X <- gen_preds(nobs, npreds, ninters)

        # Generate labels
        newdata <- gen_resp(X, coefs = coefs_generate)
```

```r
      Y <- newdata$classes

      # This contains only non-interaction columns and is used to train models.
      Xtrain <- X[,1:npreds]

      # Combined dataset
      pat_data <- cbind(Xtrain, Y)
      pat_data$Y = factor(pat_data$Y)

      # For each holdout size, split data into intervention and holdout set
      mask <- split_data(pat_data, frac_ho[i])
      data_interv <- pat_data[!mask,]
      data_hold <- pat_data[mask,]

      # Train model
      trained_model <- model_train(data_hold, model_family = family)

      # Predict
      class_pred <- model_predict(data_interv, trained_model,
                                  return_type = "class",
                                  threshold = 0.5, model_family = family)

      # Predictions made at baseline (oracle/Bayes optimal predictor on subset of covariates)
      base_pred <- oracle_pred(data_hold, coefs_base[base_vars, ])


      ## Generalised cost function with a specific cost for fn, fp, tp and tn

      # Generate confusion matrices
      confus_inter <- table(factor(data_interv$Y, levels=0:1),
                            factor(class_pred, levels=0:1))

      confus_hold <- table(factor(data_hold$Y, levels=0:1),
                           factor(base_pred, levels=0:1))

      costs_inter_resample[b, i] <- sum(confus_inter * cost_mat)
      costs_ho_resample[b, i] <- sum(confus_hold * cost_mat)
      costs_tot_resample[base_vars, b, i] <- costs_ho_resample[b, i] + costs_inter_resample[b, i]

      # Progress
      old_progress <- progress
    }

}

# Calculate Standard Deviations and Errors
for (base_vars in 1:max_base_powers) {
  costs_sd[base_vars, ] <-
    apply(costs_tot_resample[base_vars, , ], 2, sd)
}
costs_se <- costs_sd / sqrt(n_iter)

# Store data for this model family and number of interactions
```

```r
    data_list=list(max_base_powers=max_base_powers,
                   frac_ho=frac_ho,
                   costs_tot_resample=costs_tot_resample,
                   costs_ho_resample=costs_ho_resample,
                   costs_inter_resample=costs_inter_resample,
                   base_vars=base_vars,
                   costs_sd=costs_sd,
                   costs_se=costs_se)
    assign(paste0("data_",family,"_inter",ninters),data_list)
  }
}

# Collate and save all data
data_example_simulation=list()
ind=1
for (family in families) {
  for (ninters in interactions) {
    xname=paste0("data_",family,"_inter",ninters)
    data_example_simulation[[ind]]=get(xname)
    names(data_example_simulation)[ind]=xname
    ind=ind+1
  }
}
save(data_example_simulation,file="data/data_example_simulation.RData")
```

The following figure shows the emergence of an optimal holdout size in setting (a), (i). The leftmost panel (panel 1) shows the empirical cost function $L(n)$ and standard deviation. The two rightmost panels (panels 2 and 3) show the constituents of $L(n)$: namely, the contribution from the intervention set $k_2(n)(N-n)$ and the contribution from the holdout set $k_1 n$. The middle panel additionally shows the empirical function $k_2(n)$, effectively the 'learning curve' of the predictive score.

It is evident that the location of the optimal holdout size depends on a balance of the slope of the blue-coloured curves in panels 2 and 3.
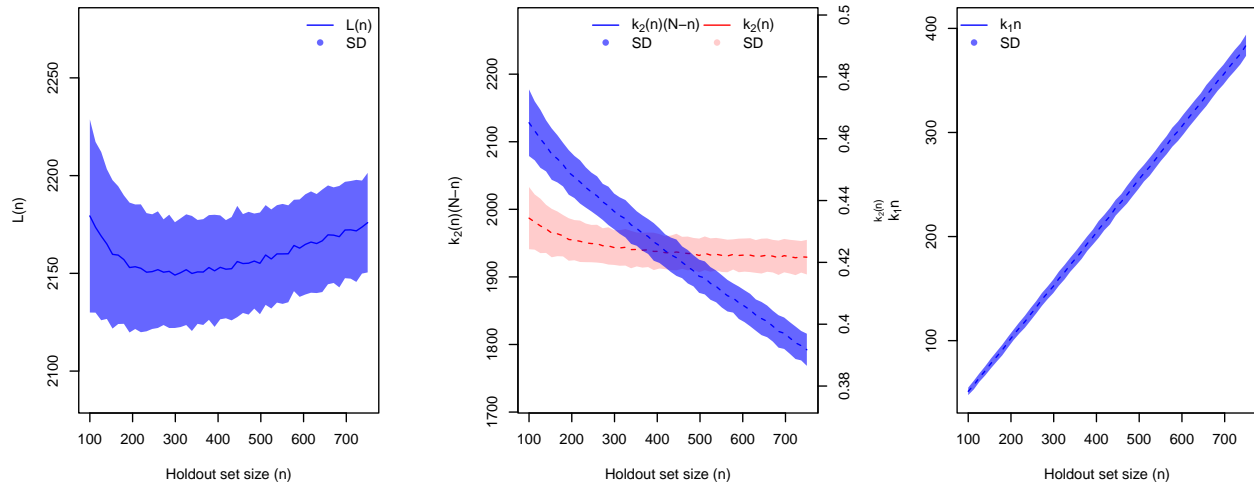
Show code

```r
# Load data
data(data_example_simulation)
X=data_example_simulation$data_log_reg_inter0
```

```r
for (i in 1:length(X)) assign(names(X)[i],X[[i]])


# Holdout set size in absolute terms
n_ho=frac_ho*nobs

# Colour intensities
r_alpha=0.2
b_alpha=0.6


oldpar=par(mfrow=c(1,3))



## Draw cost function

# Cost values
l_n=colMeans(costs_tot_resample[1,,])
l_sd=costs_sd[1, ]

# Initialise plot for cost function
oldpar1=par(mar=c(4,4,3,4))
plot(0, 0, type = "n",
     ylab = "L(n)",
     xlab = "Holdout set size (n)",
     xlim=range(n_ho),
     ylim = range(l_n + 2*l_sd*c(1,-1),na.rm=T)
)

# Plot Cost line
lines(n_ho,l_n,pch = 16,lwd = 1,col = "blue")

# Standard Deviation Bands
polygon(c(n_ho, rev(n_ho)),
        c(l_n - l_sd,
          rev(l_n + l_sd)),
        col = rgb(0,0,1,alpha=b_alpha),
        border = NA)

# Add legend
legend("topright", legend = c("L(n)", "SD"),
       col = c("blue",rgb(0,0,1, alpha = b_alpha)),
       lty=c(1,NA),
       pch=c(NA,16),
       bty="n",bg="white",
       ncol=1
)

par(oldpar1)
```

```
## Draw k2 function

# Evaluate k2(n)*(N-n)
k2n=colMeans(costs_inter_resample)
sd_k2n=colSds(costs_inter_resample)

# Evaluate k2
k2=k2n/(nobs-n_ho)
sd_k2=sd_k2n/(nobs-n_ho)

# Scalinge factor (for plotting)
sc=mean(nobs-n_ho)


# Initialise plot for k2 function
oldpar2=par(mar=c(4,4,3,4))
yr_k2=range(k2n+3*sd_k2n*c(1,-1),na.rm=F)
plot(0, 0, type = "n",
     ylab = expression(paste("k"[2],"(n)(N-n)")),
     xlab = "Holdout set size (n)",
     xlim=range(n_ho),
     ylim = yr_k2
)

# Line for estimated k2(n)
lines(n_ho,k2*sc,col="red",lty=2)

# Add standard deviation of k2(n).
polygon(c(n_ho, rev(n_ho)),
        c(k2 + sd_k2,rev(k2-sd_k2))*sc,
        col = rgb(1,0,0,alpha=r_alpha),
        border = NA)

axis(4,at=sc*pretty(yr_k2/sc),labels=pretty(yr_k2/sc))
mtext(expression(paste("k"[2],"(n)")), side=4, line=3,cex=0.5)

# Line for estimated k2(n)*(N-n)
lines(n_ho,k2n,col="blue",lty=2)

# Add standard deviation for k2(n)*(N-n)
polygon(c(n_ho, rev(n_ho)),
        c(k2n + sd_k2n,rev(k2n-sd_k2n)),
        col = rgb(0,0,1,alpha=b_alpha),
        border = NA)

# Add legend
legend("topright", legend = c(expression(paste("k"[2],"(n)(N-n)")), "SD",
                              expression(paste("k"[2],"(n)")), "SD"),
       col = c("blue",rgb(0,0,1, alpha = b_alpha),
               "red",rgb(1,0,0, alpha = r_alpha)),
       lty=c(1,NA,1,NA),
       pch=c(NA,16,NA,16),
```

```
        bty="n",bg="white",
        ncol=2
)

par(oldpar2)




## Draw k1*n function

# Evaluate k1
k1n=colMeans(costs_ho_resample)
sd_k1n=colSds(costs_ho_resample)

# Initialise plot for k2 function
oldpar3=par(mar=c(4,4,3,4))
plot(0, 0, type = "n",
     ylab = expression(paste("k"[1],"n")),
     xlab = "Holdout set size (n)",
     xlim=range(n_ho),
     ylim = range(k1n+3*sd_k1n*c(1,-1),na.rm=F)
)

# Line for estimated k1*n
lines(n_ho,k1n,col="blue",lty=2)

# Add standard deviation. Note this is scaled by (nobs-n_ho)
polygon(c(n_ho, rev(n_ho)),
        c(k1n + sd_k1n,rev(k1n-sd_k1n)),
        col = rgb(0,0,1,alpha=b_alpha),
        border = NA)

# Add legend
legend("topleft", legend = c(expression(paste("k"[1],"n")), "SD"),
       col = c("blue",rgb(0,0,1, alpha = b_alpha)),
       lty=c(1,NA),
       pch=c(NA,16),
       bty="n",bg="white",
       ncol=1
)
par(oldpar3)

par(oldpar)
```

Behaviour of empirical cost function and optimal holdout size

The following figure shows empirical cost functions $L(n)$ and empirical curves $k_2(n)$ using both underlying models (**(a)** and **(b)**) and risk score types (**(i)** and **(ii)**). Parametric approximations to empirical cost functions and empirical $k_2(n)$ are additionally shown, as well as optimal holdout size and minimum cost estimates derived from these approximations.

It is evident that the random forest (risk score **(ii)**) leads to better performance in setting **(b)** but worse performance in setting **(a)**. A more accurate risk score does not necessarily lead to a lower optimal holdout size.

It should be noted that the optimal holdout size often occurs at a value of n at which the performance of the risk score is substantially worse than its optimal performance.
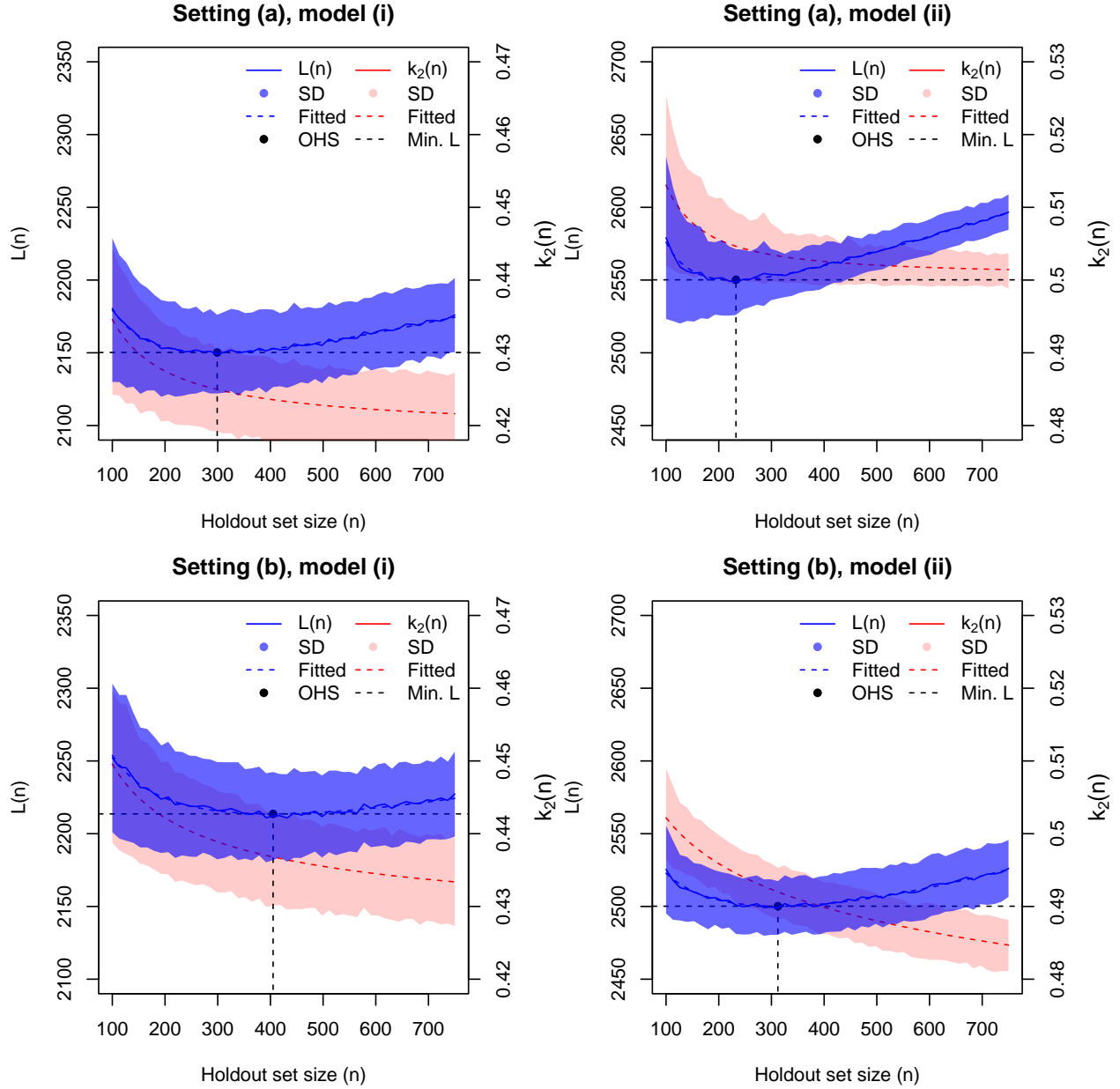
Show code

```
data(data_example_simulation)

oldpar=par(mfrow=c(2,2))


# Loop through values of 'families' and 'interactions'
for (xf in 1:length(families)) {
```

```r
for (xi in 1:length(interactions)) {
  family=families[xf]
  ninters=interactions[xi]

  X=data_example_simulation[[paste0("data_",family,"_inter",ninters)]]
  for (i in 1:length(X)) assign(names(X)[i],X[[i]])

  # Plot title
  if (xi==1 & xf==1) ptitle="Setting (a), model (i)"
  if (xi==2 & xf==1) ptitle="Setting (a), model (ii)"
  if (xi==1 & xf==2) ptitle="Setting (b), model (i)"
  if (xi==2 & xf==2) ptitle="Setting (b), model (ii)"

  # Y axis range
  if (xi==1) yl=c(2100,2350) else yl=c(2450,2700)

  # Holdout set size in absolute terms
  n_ho=frac_ho*nobs

  # Colour intensities
  r_alpha=0.2
  b_alpha=0.6

  # Create figure
  oldpar1=par(mar=c(4,4,3,4))
  plot(0, 0, type = "n",
       ylab = "L(n)",
       xlab = "Holdout set size (n)",
       main=ptitle,
       xlim=range(n_ho),
       ylim = yl
  )


  ### Draw learning curve with axis on right

  # Compute
  k2=colMeans(costs_inter_resample)/(nobs-n_ho)

  # Scaling
  if (xi==2) scvec=c(0.48,20) else scvec=c(0.42,20)
  k2_sc=function(x) yl[1] + (yl[2]-yl[1])*(x-scvec[1])*scvec[2] # Scaling transform
  i_k2_sc=function(x) (x-yl[1])/(scvec[2]*(yl[2]-yl[1])) + scvec[1] # Inverse

  # Add standard deviation. Note this is scaled by (nobs-n_ho)
  sd_k2=colSds(costs_inter_resample)/(nobs-n_ho)
  polygon(c(n_ho, rev(n_ho)),
          k2_sc(c(k2 + sd_k2,rev(k2-sd_k2))),
          col = rgb(1,0,0,alpha=r_alpha),
          border = NA)

  # Add axis and label
  axis(4,at=k2_sc(pretty(i_k2_sc(yl))),labels=pretty(i_k2_sc(yl)))
  mtext(expression(paste("k"[2],"(n)")), side=4, line=3)
```

```r
# Estimate parameters.
theta=c(1,1,1);
for (i in 1:5) theta=powersolve(n_ho,k2,init=theta)$par
k1=median(colMeans(costs_ho_resample)/(nobs*frac_ho))
k2=theta[1]*(n_ho)^(-theta[2]) + theta[3]

# Line for estimated k2 curve
lines(n_ho,
      k2_sc(k2),
      col="red",lty=2)

# Line for estimated cost function
xcost=n_ho*k1 + k2*(nobs-n_ho)
lines(n_ho,
      xcost,
      col="blue",lty=2)

# Add minimum L and OHS
ohs=n_ho[which.min(xcost)]
minL=min(xcost)
lines(c(ohs,ohs),c(0,minL),lty=2)
abline(h=minL,lty=2)
points(ohs,minL,pch=16)

# Finally, add cost function (on top)
# Plot Cost line
lines(n_ho,
      colMeans(costs_tot_resample[base_vars, , ]),
      pch = 16,
      lwd = 1,
      col = "blue")

# Standard Deviation Bands
polygon(c(n_ho, rev(n_ho)),
        c(colMeans(costs_tot_resample[1,,]) - costs_sd[1, ],
          rev(colMeans(costs_tot_resample[1,,]) + costs_sd[1, ])),
        col = rgb(0,0,1,alpha=b_alpha),
        border = NA)

# Add legend
legend("topright", legend = c("L(n)", "SD","Fitted",
                              "OHS",
                              expression(paste("k"[2],"(n)")), "SD","Fitted",
                              "Min. L"),
       col = c("blue",rgb(0,0,1, alpha = b_alpha),"blue","black",
               "red",rgb(1,0,0, alpha = r_alpha),"red","black"),
       lty=c(1,NA,2,NA,
             1,NA,2,2),
       pch=c(NA,16,NA,16,
             NA,16,NA,NA),
       bty="n",bg="white",
       ncol=2
)
```

```
    par(oldpar1)
  }
}

par(oldpar)
```

References

Hippisley-Cox, Julia, Carol Coupland, and Peter Brindle. 2017. "Development and Validation of Qrisk3 Risk Prediction Algorithms to Estimate Future Risk of Cardiovascular Disease: Prospective Cohort Study." *Bmj* 357.

Koopman, Richelle J, and AG Mainous. 2008. "Evaluating Multivariate Risk Scores for Clinical Decision Making." *Family Medicine* 40 (6): 412.

Lenert, Matthew C, Michael E Matheny, and Colin G Walsh. 2019. "Prognostic Models Will Be Victims of Their Own Success, Unless. . . ." *Journal of the American Medical Informatics Association* 26 (12): 1645–50.

Liley, James, Samuel R Emerson, Bilal A Mateen, Catalina A Vallejos, Louis JM Aslett, and Sebastian J Vollmer. 2021. "Model Updating After Interventions Paradoxically Introduces Bias." *AISTATS Proceedings*.

Nashef, Samer AM, Francois Roques, Linda D Sharples, Johan Nilsson, Christopher Smith, Antony R Goldstone, and Ulf Lockowandt. 2012. "Euroscore II." *European Journal of Cardio-Thoracic Surgery* 41 (4): 734–45.

Perdomo, Juan, Tijana Zrnic, Celestine Mendler-Dunner, and Moritz Hardt. 2020. "Performative Prediction." In *International Conference on Machine Learning*, 7599–609. PMLR.

Sperrin, Matthew, David Jenkins, Glen P Martin, and Niels Peek. 2019. "Explicit Causal Reasoning Is Needed to Prevent Prognostic Models Being Victims of Their Own Success." *Journal of the American Medical Informatics Association* 26 (12): 1675–76.