|  |  |
| --- | --- |
|  | #' @param titles A list of two strings: a title and a subtitle for produced graph. |
|  | #' |
|  | #' @return A list of two \code{data.frame}s, one for removed space and one for misclassifications. |
|  | #' @export |
|  | #' |
|  | #' @examples |
|  | #' ranges <- list(aSI = c(0.1, 0.8), aIR = c(0, 0.5), aSR = c(0, 0.05)) |
|  | #' targets <- list( |
|  | #' list(val = 281, sigma = 10.43), |
|  | #' list(val = 30, sigma = 11.16), |
|  | #' list(val = 689, sigma = 14.32) |
|  | #' ) |
|  | #' outputs <- c('nS','nI','nR') |
|  | #' ems <- emulator\_from\_data(GillespieSIR, outputs, ranges, deltas = rep(0.1, 3), quadratic = TRUE) |
|  | #' t\_ems <- purrr::map(seq\_along(ems), ~ems[[.]]$adjust(GillespieSIR, outputs[[.]])) |
|  | #' names(t\_ems) <- outputs |
|  | #' removal <- space\_removed(ems, GillespieValidation, targets, |
|  | #' n\_points = 5, u\_mod = seq(0.75, 1.25, by = 0.25), intervals = seq(2, 6, by = 0.1)) |
|  | #' |

|  |
| --- |
| space\_removed <- function(emulators, validation\_points, z, n\_points = 10, u\_mod = seq(0.8, 1.2, by = 0.1), intervals = seq(0, 10, length.out = 200), modified = 'disc',  titles = NULL) { |
|  | value <- variable <- NULL |
|  | in\_names <- names(emulators[[1]]$ranges) |
|  | z\_vals <- purrr::map\_dbl(z, ~.$val) |
|  | z\_sigs <- purrr::map\_dbl(z, ~.$sigma) |
|  | on\_grid <- setNames(expand.grid(purrr::map(emulators[[1]]$ranges, ~seq(.[[1]], .[[2]], length.out = n\_points))), in\_names) |
|  | imp\_array <- array(0, dim = c(length(intervals), length(u\_mod))) |
|  | if (!(modified %in% c('disc', 'var', 'corr'))) { |
|  | warning("Unrecognised varying parameter. Setting to structural discrepancy (disc).") |
|  | modified = 'disc' |
|  | } |
|  | if (modified == 'disc') { |
|  | em\_exps <- data.frame(purrr::map(emulators, ~.$get\_exp(on\_grid))) |
|  | em\_vars <- data.frame(purrr::map(emulators, ~.$get\_cov(on\_grid))) |
|  | #valid\_exps <- data.frame(purrr::map(emulators, ~.$get\_exp(validation\_points[,in\_names]))) |
|  | #valid\_vars <- data.frame(purrr::map(emulators, ~.$get\_cov(validation\_points[,in\_names]))) |
|  | #misclass\_arr <- array(0, dim = c(length(intervals), length(u\_mod))) |
|  | for (i in u\_mod) { |
|  | num <- data.frame(t(apply(em\_exps, 1, function(x) abs(x - z\_vals)))) |
|  | denom <- data.frame(t(apply(em\_vars, 1, function(x) sqrt(x + (i \* z\_sigs)^2)))) |
|  | imps <- num/denom |
|  | m\_imps <- apply(imps, 1, max) |
|  | cutoff <- purrr::map\_dbl(intervals, ~1-length(m\_imps[m\_imps <= .])/length(m\_imps)) |
|  | #em\_imps <- abs(valid\_exps - z\_vals)/sqrt(valid\_vars + (z\_sigs \* i)^2) |
|  | #sim\_imps <- abs(validation\_points[,!names(validation\_points) %in% in\_names] - z\_vals)/(z\_sigs \* i) |
|  | #misc <- purrr::map\_dbl(intervals, ~sum(apply(em\_imps > . & sim\_imps <= ., 1, purrr::some, isTRUE), na.rm = TRUE))/length(validation\_points[,1]) |
|  | imp\_array[, match(i, u\_mod)] <- cutoff |
|  | #misclass\_arr[, match(i, u\_mod)] <- misc |
|  | } |
|  | } |
|  | else { |
|  | for (i in u\_mod) { |
|  | if (modified == 'var') |
|  | ems <- purrr::map(emulators, ~.$set\_sigma(i\*.$u\_sigma)) |
|  | else { |
|  | ems <- purrr::map(emulators, ~.$o\_em$clone()) |
|  | for (j in 1:length(ems)) ems[[j]]$corr <- function(x, xp) exp\_sq(x, xp, i\*ems[[j]]$theta) ## This will only work if the correlation function WAS exp\_sq!! |
|  | ems <- purrr::map(seq\_along(ems), ~ems[[.]]$adjust(setNames(cbind(eval\_funcs(scale\_input, emulators[[.]]$in\_data, emulators[[.]]$ranges, FALSE), emulators[[.]]$out\_data), c(names(emulators[[.]]$in\_data), "out")), 'out')) |
|  | } |
|  | imps <- nth\_implausible(ems, on\_grid, z) |
|  | cutoff <- purrr::map\_dbl(intervals, ~1-length(imps[imps <= .])/length(imps)) |
|  | imp\_array[, match(i, u\_mod)] <- cutoff |
|  | } |
|  | } |
|  | df1 <- setNames(data.frame(imp\_array), u\_mod) |
|  | df1$cutoff <- intervals |
|  | #df2 <- setNames(data.frame(misclass\_arr), u\_mod) |
|  | #df2$cutoff <- intervals |
|  | melted\_df1 <- reshape2::melt(df1, id.vars = 'cutoff') |
|  | #melted\_df2 <- reshape2::melt(df2, id.vars = 'cutoff') |
|  | if (titles == NULL) {  tit <- switch(modified, 'disc' = 'structural discrepancy', 'var' = 'variance inflation', 'corr' = 'correlation length inflation') |
|  | subtit <- switch(modified, 'disc' = '% Structural\nDiscrepancy', 'var' = '% Variance\nInflation', 'corr' = '% Theta\nInflation')  }  else {  tit <- titles[1]  subtit <- titles[2]  } |
|  | g <- ggplot(data = melted\_df1, aes(x = cutoff, y = value, group = variable, colour = variable)) + |
|  | geom\_line(lwd = 1.5) + |
|  | #geom\_line(data = plyr::mutate(melted\_df2, value = value/max(value)), lwd = 0.5) + |
|  | scale\_colour\_viridis(discrete = TRUE, option = 'cividis', labels = function(b) {paste0(round(as.numeric(b)\*100, 0), "%")}) + |
|  | scale\_x\_continuous("Implausibility cut-off", labels = function(b) {round(b, 1)}) + |
|  | scale\_y\_continuous("Removed", labels = function(b) { |
|  | paste0(round(b\*100,0),"%") |
|  | } #, |
|  | #sec.axis = sec\_axis(~.\*max(melted\_df2$value), name = "Misclassified", labels = function(b) { |
|  | # paste0(round(b\*100,0), "%") |
|  | #}) |
|  | ) + |
|  | labs(title = paste("Space removed as a function of implausibility cut-off and", tit), colour = subtit, x = "Cut-off", y = "% removed") + |
|  | theme\_minimal() |
|  | print(g) |
|  | return(df1) |
|  | #return(list(reduced = df1, misclassed = df2)) |
|  | } |

|  |
| --- |
|  |
| #' Space Removal |
|  | #' |
|  | #' Finds the proportion of space removed as a function of implausibility cut-off, and of structural |
|  | #' discrepancy, or changed variance. |
|  | #' |
|  | #' The reduction in space is found by evaluating over a p^d regular grid, where p is chosen by |
|  | #' \code{n\_points} and d is the dimension of the input space. Larger values of \code{n\_points} |
|  | #' will give a more accurate reflection of removed space, at high computational cost. For the |
|  | #' purpose of quick diagnostics, \code{n\_points = 5} is acceptable. |
|  | #' |
|  | #' @import ggplot2 |
|  | #' @importFrom stats setNames |
|  | #' @importFrom viridis scale\_colour\_viridis |
|  | #' @importFrom plyr mutate |
|  | #' |
|  | #' @param emulators A set of \code{\link{Emulator}} objects. |
|  | #' @param validation\_points The validation set used in this wave. |
|  | #' @param z The observations with which to match, as \code{list(val, sigma)} pairs. |
|  | #' @param n\_points The number of points in each dimension of the grid. |
|  | #' @param u\_mod The percentage differences in structural discrepancy to examine. |
|  | #' @param intervals The set of implausibility cut-offs to consider. |
|  | #' @param modified What should be changed: model discrepancy (disc), emulator variance (var), correlation length (corr) or ensemble variability (ev)? |
|  | #' |
|  | #' @return A list of two \code{data.frame}s, one for removed space and one for misclassifications. |
|  | #' @export |
|  | #' |
|  | #' @examples |
|  | #' ranges <- list(aSI = c(0.1, 0.8), aIR = c(0, 0.5), aSR = c(0, 0.05)) |
|  | #' targets <- list( |
|  | #' list(val = 281, sigma = 10.43), |
|  | #' list(val = 30, sigma = 11.16), |
|  | #' list(val = 689, sigma = 14.32) |
|  | #' ) |
|  | #' outputs <- c('nS','nI','nR') |
|  | #' ems <- emulator\_from\_data(GillespieSIR, outputs, ranges, deltas = rep(0.1, 3), quadratic = TRUE) |
|  | #' t\_ems <- purrr::map(seq\_along(ems), ~ems[[.]]$adjust(GillespieSIR, outputs[[.]])) |
|  | #' names(t\_ems) <- outputs |
|  | #' removal <- space\_removed(ems, GillespieValidation, targets, |
|  | #' n\_points = 5, u\_mod = seq(0.75, 1.25, by = 0.25), intervals = seq(2, 6, by = 0.1)) |
|  | #' |
|  | space\_removed <- function(emulators, validation\_points, z, n\_points = 10, u\_mod = seq(0.8, 1.2, by = 0.1), intervals = seq(0, 10, length.out = 200), modified = 'disc') { |
|  | value <- variable <- NULL |
|  | in\_names <- names(emulators[[1]]$ranges) |
|  | z\_vals <- purrr::map\_dbl(z, ~.$val) |
|  | z\_sigs <- purrr::map\_dbl(z, ~.$sigma) |
|  | on\_grid <- setNames(expand.grid(purrr::map(emulators[[1]]$ranges, ~seq(.[[1]], .[[2]], length.out = n\_points))), in\_names) |
|  | imp\_array <- array(0, dim = c(length(intervals), length(u\_mod))) |
|  | if (!(modified %in% c('disc', 'var', 'corr'))) { |
|  | warning("Unrecognised varying parameter. Setting to structural discrepancy (disc).") |
|  | modified = 'disc' |
|  | } |
|  | if (modified == 'disc') { |
|  | em\_exps <- data.frame(purrr::map(emulators, ~.$get\_exp(on\_grid))) |
|  | em\_vars <- data.frame(purrr::map(emulators, ~.$get\_cov(on\_grid))) |
|  | #valid\_exps <- data.frame(purrr::map(emulators, ~.$get\_exp(validation\_points[,in\_names]))) |
|  | #valid\_vars <- data.frame(purrr::map(emulators, ~.$get\_cov(validation\_points[,in\_names]))) |
|  | #misclass\_arr <- array(0, dim = c(length(intervals), length(u\_mod))) |
|  | for (i in u\_mod) { |
|  | num <- data.frame(t(apply(em\_exps, 1, function(x) abs(x - z\_vals)))) |
|  | denom <- data.frame(t(apply(em\_vars, 1, function(x) sqrt(x + (i \* z\_sigs)^2)))) |
|  | imps <- num/denom |
|  | m\_imps <- apply(imps, 1, max) |
|  | cutoff <- purrr::map\_dbl(intervals, ~1-length(m\_imps[m\_imps <= .])/length(m\_imps)) |
|  | #em\_imps <- abs(valid\_exps - z\_vals)/sqrt(valid\_vars + (z\_sigs \* i)^2) |
|  | #sim\_imps <- abs(validation\_points[,!names(validation\_points) %in% in\_names] - z\_vals)/(z\_sigs \* i) |
|  | #misc <- purrr::map\_dbl(intervals, ~sum(apply(em\_imps > . & sim\_imps <= ., 1, purrr::some, isTRUE), na.rm = TRUE))/length(validation\_points[,1]) |
|  | imp\_array[, match(i, u\_mod)] <- cutoff |
|  | #misclass\_arr[, match(i, u\_mod)] <- misc |
|  | } |
|  | } |
|  | else { |
|  | for (i in u\_mod) { |
|  | if (modified == 'var') |
|  | ems <- purrr::map(emulators, ~.$set\_sigma(i\*.$u\_sigma)) |
|  | else { |
|  | ems <- purrr::map(emulators, ~.$o\_em$clone()) |
|  | for (j in 1:length(ems)) ems[[j]]$corr <- function(x, xp) exp\_sq(x, xp, i\*ems[[j]]$theta) ## This will only work if the correlation function WAS exp\_sq!! |
|  | ems <- purrr::map(seq\_along(ems), ~ems[[.]]$adjust(setNames(cbind(eval\_funcs(scale\_input, emulators[[.]]$in\_data, emulators[[.]]$ranges, FALSE), emulators[[.]]$out\_data), c(names(emulators[[.]]$in\_data), "out")), 'out')) |
|  | } |
|  | imps <- nth\_implausible(ems, on\_grid, z) |
|  | cutoff <- purrr::map\_dbl(intervals, ~1-length(imps[imps <= .])/length(imps)) |
|  | imp\_array[, match(i, u\_mod)] <- cutoff |
|  | } |
|  | } |
|  | df1 <- setNames(data.frame(imp\_array), u\_mod) |
|  | df1$cutoff <- intervals |
|  | #df2 <- setNames(data.frame(misclass\_arr), u\_mod) |
|  | #df2$cutoff <- intervals |
|  | melted\_df1 <- reshape2::melt(df1, id.vars = 'cutoff') |
|  | #melted\_df2 <- reshape2::melt(df2, id.vars = 'cutoff') |
|  | tit <- switch(modified, 'disc' = 'structural discrepancy', 'var' = 'variance inflation', 'corr' = 'correlation length inflation', 'ev' = 'ensemble variability') |
|  | subtit <- switch(modified, 'disc' = '% Structural\nDiscrepancy', 'var' = '% Variance\nInflation', 'corr' = '% Theta\nInflation', 'ev' = '% Ensemble\nVariability' ) |
|  | g <- ggplot(data = melted\_df1, aes(x = cutoff, y = value, group = variable, colour = variable)) + |
|  | geom\_line(lwd = 1.5) + |
|  | #geom\_line(data = plyr::mutate(melted\_df2, value = value/max(value)), lwd = 0.5) + |
|  | scale\_colour\_viridis(discrete = TRUE, option = 'cividis', labels = function(b) {paste0(round(as.numeric(b)\*100, 0), "%")}) + |
|  | scale\_x\_continuous("Implausibility cut-off", labels = function(b) {round(b, 1)}) + |
|  | scale\_y\_continuous("Removed", labels = function(b) { |
|  | paste0(round(b\*100,0),"%") |
|  | } #, |
|  | #sec.axis = sec\_axis(~.\*max(melted\_df2$value), name = "Misclassified", labels = function(b) { |
|  | # paste0(round(b\*100,0), "%") |
|  | #}) |
|  | ) + |
|  | labs(title = paste("Space removed as a function of implausibility cut-off and", tit), colour = subtit, x = "Cut-off", y = "% removed") + |
|  | theme\_minimal() |
|  | print(g) |
|  | return(df1) |
|  | #return(list(reduced = df1, misclassed = df2)) |
|  | } |
|  |  |