Exploring multi-dimensional spaces

R code

Arnald Puy

Contents

1 Preliminary

```
# PRELIMINARY ##################################################################

# Function to read in all required packages in one go:
load_packages <- function(x) {
  for(i in x) {
    if(!require(i, character.only = TRUE)) {
      install.packages(i, dependencies = TRUE)
      library(i, character.only = TRUE)
    }
  }
}

# Define theme for plots
theme_AP <- function() {
  theme_bw() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.background = element_rect(fill = "transparent",
                                           color = NA),
          legend.margin=margin(0, 0, 0, 0),
          legend.box.margin=margin(-7,-7,-7,-7),
          legend.key = element_rect(fill = "transparent",
                                    color = NA),
          legend.title = element_text(size = 9),
          legend.text = element_text(size = 8),
          strip.background = element_rect(fill = "white"),
          axis.title.x = element_text(size = 9),
          axis.title.y = element_text(size = 9))
}

# Load the packages
load_packages(c("sensobol", "data.table", "tidyverse", "parallel",
                "scales", "doParallel", "benchmarkme",
                "cowplot", "wesanderson", "logitnorm"))


# Set checkpoint
dir.create(".checkpoint")

library("checkpoint")

checkpoint("2022-11-21",
           R.version ="4.2.1",
           checkpointLocation = getwd())
```

2 Define functions

2.1 Random distributions

```
# FUNCTION TO SAMPLE DISTRIBUTIONS ############################################

# Define function to random sample distributions
sample_distributions_PDF <- list(

  "uniform" = function(x) dunif(x, 0, 1),
  "normal" = function(x) dnorm(x, 0.5, 0.15),
  "beta" = function(x) dbeta(x, 8, 2),
  "beta2" = function(x) dbeta(x, 2, 8),
  "beta3" = function(x) dbeta(x, 2, 0.8),
  "beta4" = function(x) dbeta(x, 0.8, 2),
  "logitnormal" = function(x) dlogitnorm(x, 0, 3.16)
)

# Quantile function
sample_distributions <- list(

  "uniform" = function(x) qunif(x, -1, 1),
  "normal" = function(x) qnorm(x, 0, 0.3),
  "beta" = function(x) qbeta(x, 8, 2),
  "beta2" = function(x) qbeta(x, 2, 8),
  "beta3" = function(x) qbeta(x, 2, 0.8),
  "beta4" = function(x) qbeta(x, 0.8, 2),
  "logitnormal" = function(x) qlogitnorm(x, 0, 3.16)
)

random_distributions <- function(mat, phi, epsilon) {
  names_ff <- names(sample_distributions)
  if (!phi == length(names_ff) + 1) {
    out <- sample_distributions[[names_ff[phi]]](mat)
  } else {
    set.seed(epsilon)
    temp <- sample(names_ff, ncol(mat), replace = TRUE)
    out <- sapply(seq_along(temp), function(x) sample_distributions[[temp[x]]](mat[, x]))
  }
  return(out)
}
```

2.2 Metafunction

```
# METAFUNCTION ################################################################

meta_fun <- function(data, epsilon, n) {
```

```r
# Define list of functions included in metafunction
function_list <- list(
  Linear = function(x) x,
  Quadratic = function(x) x ^ 2,
  Cubic = function(x) x ^ 3,
  Exponential = function(x) exp(1) ^ x / (exp(1) - 1),
  Periodic = function(x) sin(2 * pi * x) / 2,
  Discontinuous = function(x) ifelse(x > 0.5, 1, 0),
  Non.monotonic = function(x) 4 * (x - 0.5) ^ 2,
  Inverse = function(x) (10 - 1 / 1.1) ^ -1 * (x + 0.1) ^ - 1,
  No.effect = function(x) x * 0,
  Trigonometric = function(x) cos(x),
  Piecewise.large = function(x) ((-1) ^ as.integer(4 * x) * (0.125 - (x %% 0.25)) + 0.125),
  Piecewise.small = function(x) ((-1) ^ as.integer(32 * x) * (0.03125 - 2 * (x %% 0.03125)) +
  Oscillation = function(x) x ^ 2 - 0.2 * cos(7 * pi * x)
)

# Sample list of functions
set.seed(epsilon)
all_functions <- sample(names(function_list), ncol(data), replace = TRUE)

# Compute model output first order effects
mat.y <- sapply(seq_along(all_functions), function(x)
  function_list[[all_functions[x]]](data[, x]))

# Compute first-order effects
y1 <- Rfast::rowsums(mat.y)

if (n >= 2) { # Activate interactions

  # Define matrix with all possible interactions up to the n-th order
  interactions <- lapply(2:n, function(x) RcppAlgos::comboGeneral(1:n, x, nThreads = 4))

  out <- lapply(1:length(interactions), function(x) {
    lapply(1:nrow(interactions[[x]]), function(y) {
      Rfast::rowprods(mat.y[, interactions[[x]][y, ]])
    })
  })

  y2 <- lapply(out, function(x) do.call(cbind, x)) %>%
    do.call(cbind, .) %>%
    Rfast::rowsums(.)

} else {

  y2 <- 0
}
```

```
  y <- y1 + y2

  return(y)
}

# Add stopping rule for precaution --------------
model <- function(data, epsilon, n) {

  k <- ncol(data)

  if (n > k) {

    stop("level_interactions should be smaller or equal than \n
         the number of parameters")
  }

  y <- meta_fun(data = data, epsilon = epsilon, n = n)

  return(y)
}

# Finalize metafunction --------------------------------
model_fun <- function(k, epsilon, phi, model.runs, n, type, matrices) {

  params <- paste("X", 1:k, sep = "")
  set.seed(epsilon)
  mat <- sobol_matrices(N = model.runs, params = params, type = type,
                        matrices = matrices)
  mat <- random_distributions(mat = mat, phi = phi, epsilon = epsilon)
  y <- model(data = mat, epsilon = epsilon, n = n)
  indices <- sobol_indices(Y = y, N = model.runs, params = params, matrices = matrices,
                           first = "azzini", total = "azzini")
  model.output <- mean(y[1:model.runs])

  output <- list(model.output, indices)
  names(output) <- c("output", "indices")

  return(output)
}
```

3 Settings

```
# DEFINE SETTINGS OF THE ANALYSIS ##################################################

params <- c("k", "epsilon", "n", "phi")
N <- 2^10
```

```r
matrices <- "A"
max.k <- 10 # maximum number of explored inputs
```