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) x,
  "normal" = function(x) qnorm(x, 0.5, 0.15),
  "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,
                      first, total) {

  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 = first, total = total)
  model.output <- mean(y[1:model.runs])

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

  return(output)
}
```

3 Analysis

3.1 Settings

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

params <- c("k", "epsilon", "n", "phi")
N <- 2^10
matrices <- "A"
max.k <- 10 # maximum number of explored inputs
```

3.2 Sample matrix

```r
# DEFINE SAMPLE MATRIX #########################################################

# Creation of sample matrix --------------------
mat <- sobol_matrices(N = N, params = params, matrices = matrices)

# Transformation to appropriate distributions ---
mat[, "k"] <- floor(mat[, "k"] * (max.k - 2 + 1) + 2)
mat[, "epsilon"] <- floor(mat[, "epsilon"] * (N - 1 + 1) + 1)
mat[, "phi"] <- floor(mat[, "phi"] * ((length(sample_distributions) + 1) - 1 + 1) + 1)
mat[, "n"] <- floor(mat[, "n"] * (max.k - 1 + 1) + 1)

# Constrain n as a function of k
mat[, "n"] <- ifelse(mat[, "n"] > mat[, "k"], mat[, "k"], mat[, "n"])

# Define an increasing sample size --------------
exponents <- 7:15
sample.sizes <- 2^exponents

# Replicate the sample matrix for each sample size
n.times <- length(sample.sizes)
mat <- matrix(rep(t(mat), n.times), ncol = ncol(mat), byrow = TRUE)
model.runs <- rep(sample.sizes, each = N)

mat <- cbind(mat, model.runs)
colnames(mat) <- c(params, "model.runs")
mat <- data.table(mat)
sampling.methods <- c("R", "QRN", "LHS")

# Replicate matrix for each sampling method -----
final.mat <- mat[rep(mat[, .I], length(sampling.methods))]
final.mat <- final.mat[, sample.method:= rep(sampling.methods, each = N * n.times)]
```

3.3 Simulations

```r
# RUN SIMULATIONS ##############################################################

# Define parallel computing --------------------
```

```
matrices <- c("A", "B", "AB", "BA")
n_cores <- detectCores() * 0.75
cl <- makeCluster(n_cores)
registerDoParallel(cl)

# Compute ---------------------------------------
y <- foreach(i=1:nrow(final.mat),
             .packages = c("Rfast", "sensobol", "dplyr", "RcppAlgos",
                           "logitnorm")) %dopar%
  {
    model_fun(k = final.mat[[i, "k"]],
              epsilon = final.mat[[i, "epsilon"]],
              n = final.mat[[i, "n"]],
              phi = final.mat[[i, "phi"]],
              model.runs = final.mat[[i, "model.runs"]],
              type = final.mat[[i, "sample.method"]],
              matrices = matrices,
              first = "azzini",
              total = "azzini")
  }

# Stop parallel cluster -------------------------
stopCluster(cl)
```

3.4 Arrange results

```
# ARRANGE RESULTS ##############################################################

dt.output <- do.call("c", lapply(y, function(x) x$output))
dt.indices <- lapply(y, function(x) x$indices)
sum.si <- do.call("c", lapply(dt.indices, function(x) x$si.sum))
mean.dimension <- lapply(dt.indices, function(x) x$results %>%
                           .[sensitivity == "Ti"] %>%
                           .[, sum(original)]) %>%
  do.call("c", .)
kt <- lapply(dt.indices, function(x) x$results %>%
               .[sensitivity == "Ti"] %>%
               .[original > 0.05] %>%
               nrow(.)) %>%
  do.call("c", .)

dt.dimensions <- cbind(final.mat, sum.si, kt, mean.dimension)
dt.largest.sample <- dt.dimensions[model.runs == sample.sizes[length(sample.sizes)]]
fwrite(dt.largest.sample, "dt.largest.sample.csv")

# ---------------------
```

```
# Merge output with sample matrix ---------------

dt <- cbind(final.mat, dt.output)
dt.benchmark <- dt[model.runs == sample.sizes[length(sample.sizes)]]

dt.tmp <- split(dt, dt$model.runs) %>%
  lapply(., function(x) merge(x, dt.benchmark, by = c("k", "epsilon",
                                                      "sample.method", "n"))) %>%
  rbindlist(., idcol = "sample.size") %>%
  setnames(., c("dt.output.y", "dt.output.x"), c("y", "y.highest")) %>%
  .[, `:=` (model.runs.x = NULL, model.runs.y = NULL)] %>%
  .[, .(RMSE = sqrt(mean((y.highest - y)^2))), .(sample.size, sample.method)]

fwrite(dt.tmp, "dt.tmp.csv")
```

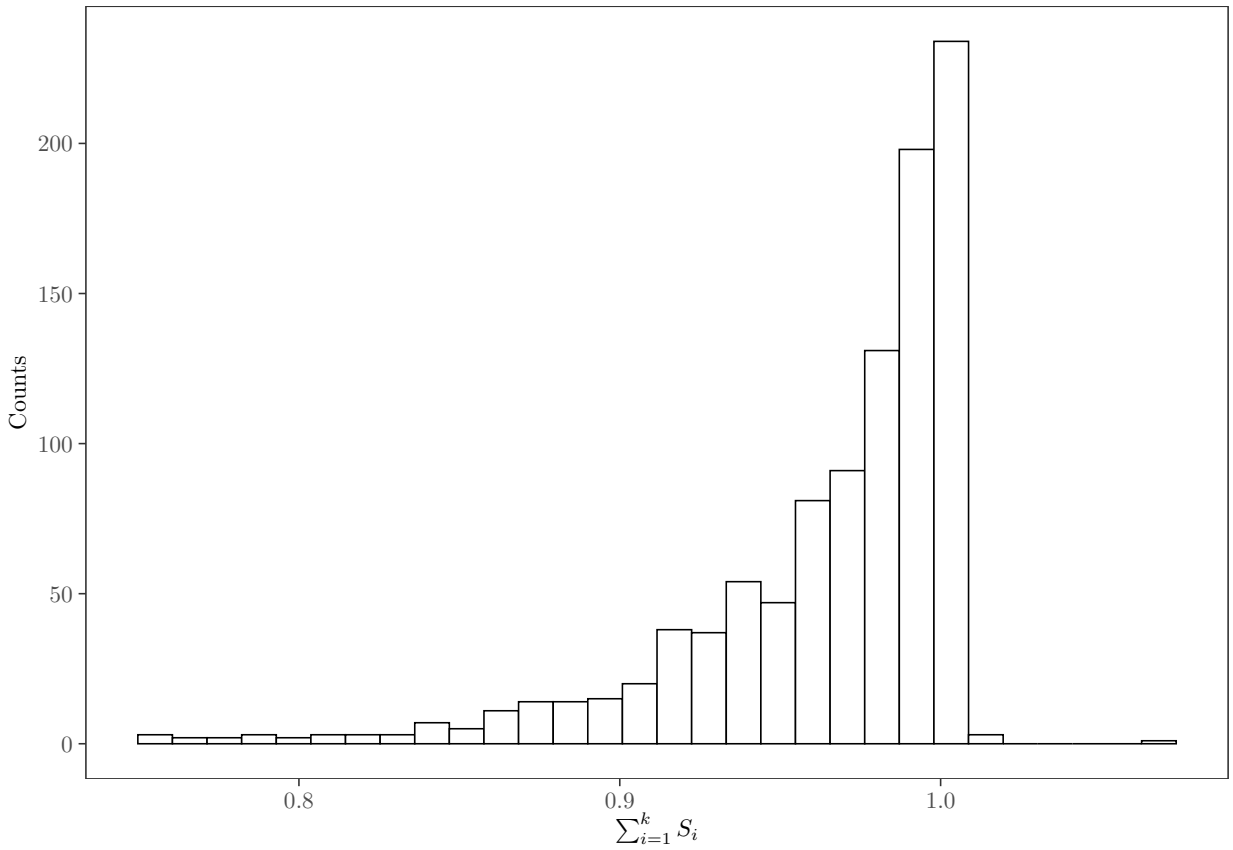
```
dt.largest.sample[, .(summary = sum.si < 0)] %>%
  .[, .N, summary]
```

```
##    summary    N
## 1:   FALSE 3066
## 2:      NA    6
```

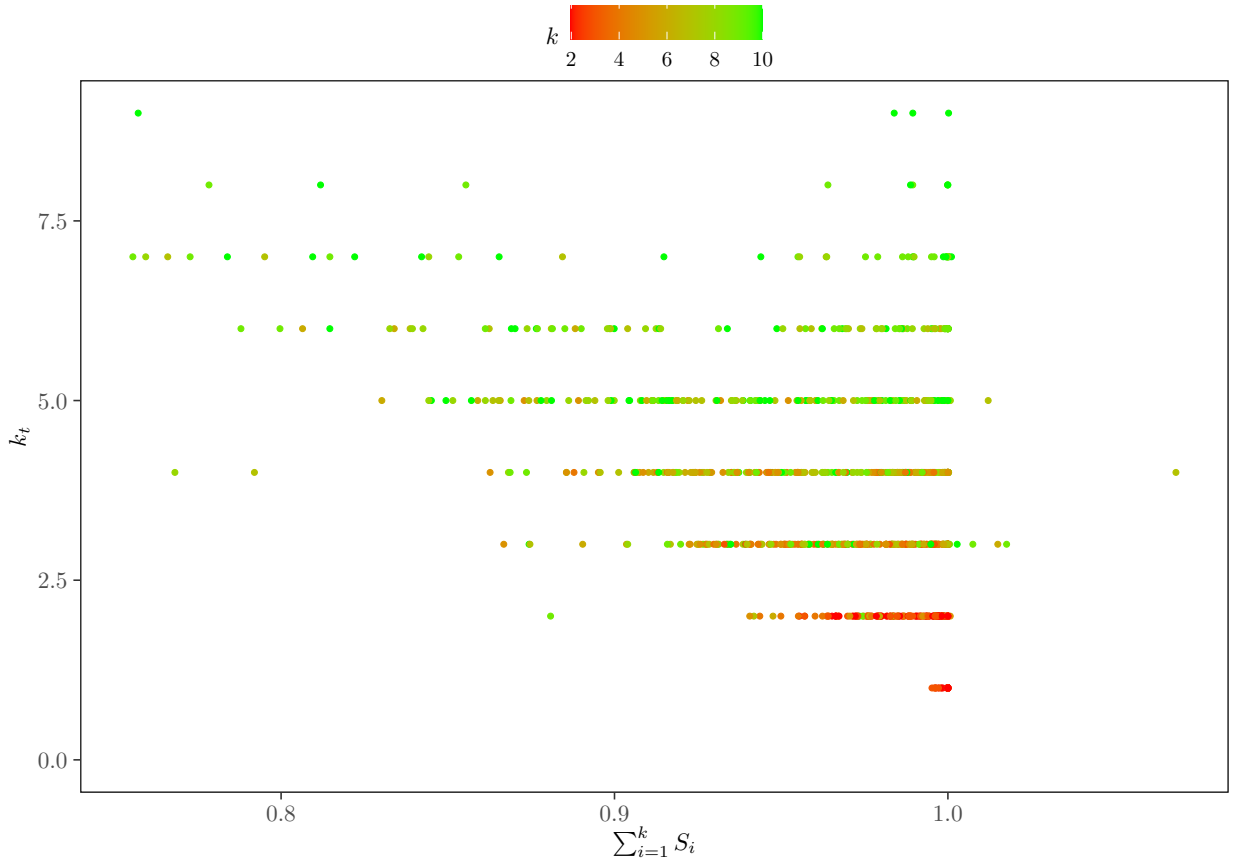
3.5 Figures

```
# PLOT ###############################################################################

# Plot distribution of functions based on sum S_i
plot.histogram <- dt.largest.sample[sample.method == "QRN"] %>%
  ggplot(., aes(sum.si)) +
  geom_histogram(fill = "white", color = "black") +
  theme_AP() +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  labs(x = "$\\sum_{i=1}^{k} S_i$",  y = "Counts") +
  theme(legend.position = c(0.35, 0.7))

plot.histogram
```

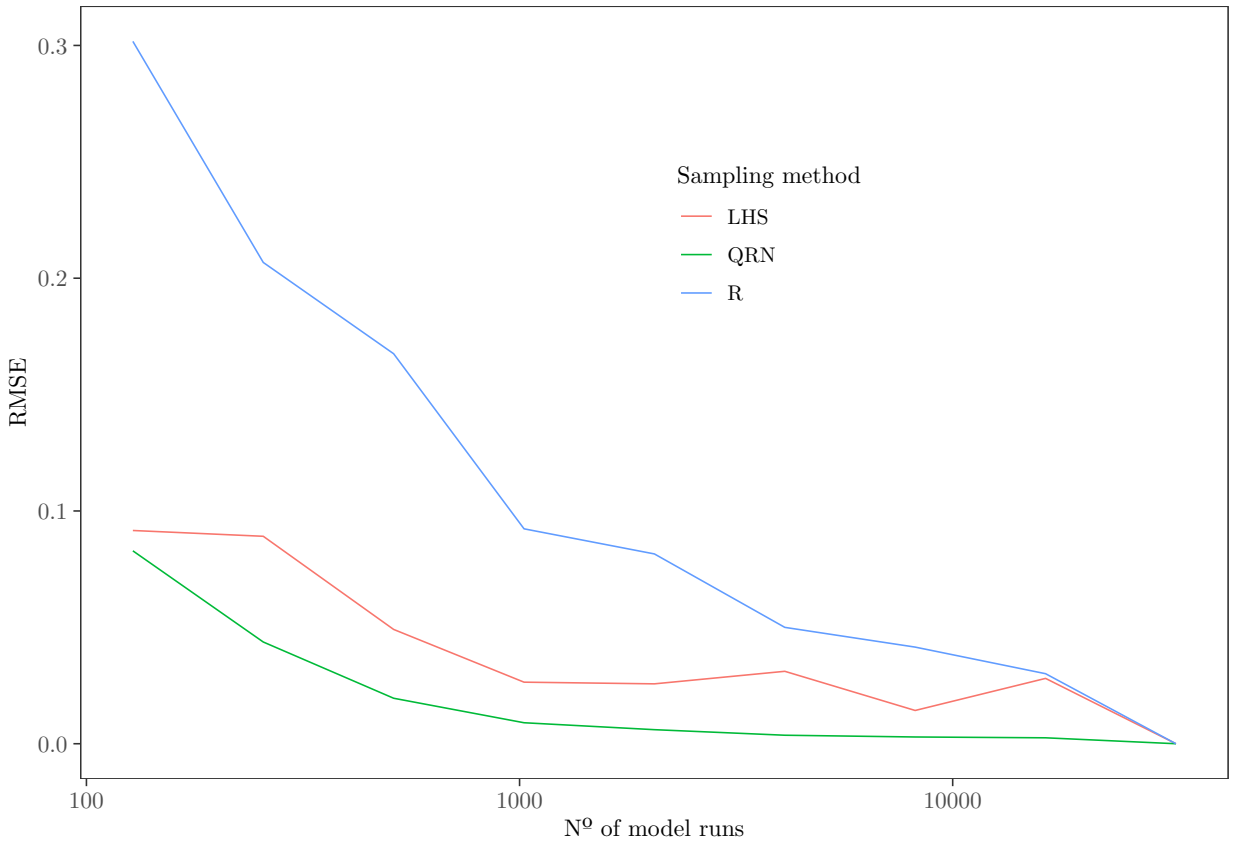
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Plot sum of S_i against k and colored by k_t
plot.scatter <- dt.largest.sample[sample.method == "QRN"] %>%
  ggplot(., aes(sum.si, kt, color = k)) +
  geom_point(size = 0.7) +
  scale_color_gradient(low = "red", high = "green",
                       name = "$k$") +
  theme_AP() +
  theme(legend.position = "top") +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  labs(x = "$\\sum_{i=1}^{k} S_i$", y = "$k_t$")

plot.scatter
```



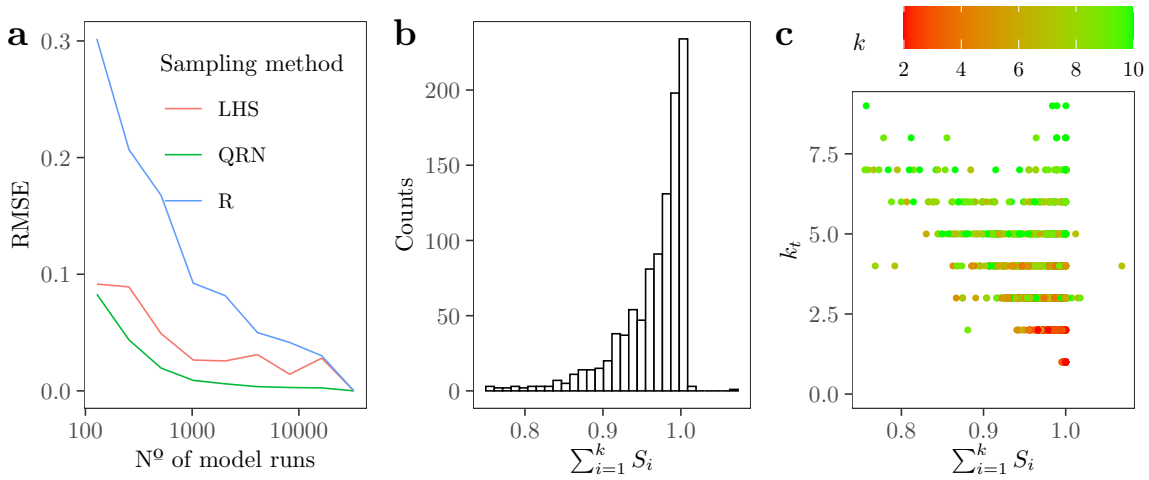

```
plot.convergence <- dt.tmp %>%
  .[, sample.size:= as.numeric(sample.size)] %>%
  ggplot(., aes(sample.size, RMSE, color = sample.method, group = sample.method)) +
  geom_line() +
  scale_x_log10() +
  theme_AP() +
  labs(x = "Nº of model runs", y = "RMSE") +
  scale_color_discrete(name = "Sampling method") +
  theme(legend.position = c(0.6, 0.7))

plot.convergence
```




```
# MERGE FIGURES ######################################################################
```

```
plot_grid(plot.convergence, plot.histogram, plot.scatter, ncol = 3, labels = "auto")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
# SESSION INFORMATION ############################################################

sessionInfo()
```

```
## R version 4.2.1 (2022-06-23)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Monterey 12.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] checkpoint_1.0.2  logitnorm_0.8.38  wesanderson_0.3.6 cowplot_1.1.1
##  [5] benchmarkme_1.0.8 doParallel_1.0.17 iterators_1.0.14  foreach_1.5.2
##  [9] scales_1.2.1      forcats_0.5.2     stringr_1.4.1     dplyr_1.0.10
## [13] purrr_0.3.5       readr_2.1.3       tidyr_1.2.1       tibble_3.1.8
## [17] ggplot2_3.4.0     tidyverse_1.3.2   data.table_1.14.6 sensobol_1.1.2
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.4            jsonlite_1.8.3        modelr_0.1.10
##  [4] Rdpack_2.4            assertthat_0.2.1      tikzDevice_0.12.3.1
##  [7] googlesheets4_1.0.1   cellranger_1.1.0      yaml_2.3.6
## [10] pillar_1.8.1          backports_1.4.1       lattice_0.20-45
## [13] glue_1.6.2            digest_0.6.30         rbibutils_2.2.10
## [16] rvest_1.0.3           colorspace_2.0-3      htmltools_0.5.3
## [19] Matrix_1.5-3          pkgconfig_2.0.3       broom_1.0.1
## [22] haven_2.5.1           tzdb_0.3.0            timechange_0.1.1
## [25] googledrive_2.0.0     generics_0.1.3        ellipsis_0.3.2
## [28] withr_2.5.0           cli_3.4.1             magrittr_2.0.3
## [31] crayon_1.5.2          readxl_1.4.1          evaluate_0.18
## [34] fs_1.5.2              fansi_1.0.3           xml2_1.3.3
## [37] tools_4.2.1           hms_1.1.2             gargle_1.2.1
## [40] lifecycle_1.0.3       munsell_0.5.0         reprex_2.0.2
## [43] compiler_4.2.1        rlang_1.0.6           grid_4.2.1
## [46] rstudioapi_0.14       filehash_2.4-3        rmarkdown_2.18
## [49] gtable_0.3.1          codetools_0.2-18      DBI_1.1.3
## [52] benchmarkmeData_1.0.4 R6_2.5.1              lubridate_1.9.0
## [55] knitr_1.41            fastmap_1.1.0         utf8_1.2.2
## [58] stringi_1.7.8         Rcpp_1.0.9.5          vctrs_0.5.1
```

```
## [61] dbplyr_2.2.1          tidyselect_1.2.0      xfun_0.35
```

```
## Return the machine CPU
cat("Machine:     "); print(get_cpu()$model_name)
```

```
## Machine:
```

```
## [1] "Apple M1 Max"
```

```
## Return number of true cores
cat("Num cores:   "); print(detectCores(logical = FALSE))
```

```
## Num cores:
```

```
## [1] 10
```

```
## Return number of threads
cat("Num threads: "); print(detectCores(logical = FALSE))
```

```
## Num threads:
```

```
## [1] 10
```

```
############################END SIMULATIONS#######################################
##############################################################################
```