The Revenge Paper

Arnald Puy, Samuele Lo Piano, Andrea Saltelli

Contents

1 Preliminary functions

```
# PRELIMINARY FUNCTIONS ----------------------------------------------------

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

# Load the packages
loadPackages(c("tidyverse"))

# Create custom theme
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.key = element_rect(fill = "transparent",
                                    color = NA))
}

# Set checkpoint

dir.create(".checkpoint")
library("checkpoint")

checkpoint("2020-03-09",
           R.version ="3.6.1",
           checkpointLocation = getwd())
```

Here we define the code to produce STAR-VARS matrices using Sobol' quasi-random number sequences:

```
# FUNCTION TO CREATE STAR-VARS ------------------------------------------------

star_vars <- function(N, params, h) {
  out <- center <- sections <- A <- B <- AB <- X <- out <- list()
  mat <- randtoolbox::sobol(n = N, dim = length(params))
  for(i in 1:nrow(mat)) {
    center[[i]] <- mat[i, ]
    sections[[i]] <- sapply(center[[i]], function(x) {
      all <- seq(x %% h, 1, h)
      non.zeros <- all[all!= 0] # Remove zeroes
      })
    B[[i]] <- sapply(1:ncol(mat), function(x)
      sections[[i]][, x][!sections[[i]][, x] %in% center[[i]][x]])
    A[[i]] <- matrix(center[[i]], nrow = nrow(B[[i]]),
                     ncol = length(center[[i]]), byrow = TRUE)
    X[[i]] <- rbind(A[[i]], B[[i]])
    for(j in 1:ncol(A[[i]])) {
      AB[[i]] <- A[[i]]
      AB[[i]][, j] <- B[[i]][, j]
      X[[i]] <- rbind(X[[i]], AB[[i]])
    }
    AB[[i]] <- X[[i]][(2 * nrow(B[[i]]) + 1):nrow(X[[i]]), ]
    out[[i]] <- rbind(unname(center[[i]]), AB[[i]])
  }
  return(do.call(rbind, out))
}

# Function to split matrices in n chunks
CutBySize <- function(m, block.size, nb = ceiling(m / block.size)) {
  int <- m / nb
  upper <- round(1:nb * int)
  lower <- c(1, upper[-nb] + 1)
  size <- c(upper[1], diff(upper))
  cbind(lower, upper, size)
}
```

Now we create a STAR-VARS sample matrix with two star centers, $h = 0.1$ and three model inputs.

```
# DEFINE THE SETTINGS FOR A STAR-VARS SAMPLE MATRIX ---------------------------

N <- 2
params <- paste("X", 1:3, sep = "")
h <- 0.1
```

```
# CREATE STAR-VARS ------------------------------------------------------------

mat <- star_vars(N = N, params = params, h = h)
```

```r
print(mat)
```

```
##       [,1] [,2] [,3]
##  [1,] 0.50 0.50 0.50
##  [2,] 0.10 0.50 0.50
##  [3,] 0.20 0.50 0.50
##  [4,] 0.30 0.50 0.50
##  [5,] 0.40 0.50 0.50
##  [6,] 0.60 0.50 0.50
##  [7,] 0.70 0.50 0.50
##  [8,] 0.80 0.50 0.50
##  [9,] 0.90 0.50 0.50
## [10,] 1.00 0.50 0.50
## [11,] 0.50 0.10 0.50
## [12,] 0.50 0.20 0.50
## [13,] 0.50 0.30 0.50
## [14,] 0.50 0.40 0.50
## [15,] 0.50 0.60 0.50
## [16,] 0.50 0.70 0.50
## [17,] 0.50 0.80 0.50
## [18,] 0.50 0.90 0.50
## [19,] 0.50 1.00 0.50
## [20,] 0.50 0.50 0.10
## [21,] 0.50 0.50 0.20
## [22,] 0.50 0.50 0.30
## [23,] 0.50 0.50 0.40
## [24,] 0.50 0.50 0.60
## [25,] 0.50 0.50 0.70
## [26,] 0.50 0.50 0.80
## [27,] 0.50 0.50 0.90
## [28,] 0.50 0.50 1.00
## [29,] 0.75 0.25 0.75
## [30,] 0.05 0.25 0.75
## [31,] 0.15 0.25 0.75
## [32,] 0.25 0.25 0.75
## [33,] 0.35 0.25 0.75
## [34,] 0.45 0.25 0.75
## [35,] 0.55 0.25 0.75
## [36,] 0.65 0.25 0.75
## [37,] 0.85 0.25 0.75
## [38,] 0.95 0.25 0.75
## [39,] 0.75 0.05 0.75
## [40,] 0.75 0.15 0.75
## [41,] 0.75 0.35 0.75
## [42,] 0.75 0.45 0.75
## [43,] 0.75 0.55 0.75
## [44,] 0.75 0.65 0.75
```

```
## [45,] 0.75 0.75 0.75
## [46,] 0.75 0.85 0.75
## [47,] 0.75 0.95 0.75
## [48,] 0.75 0.25 0.05
## [49,] 0.75 0.25 0.15
## [50,] 0.75 0.25 0.25
## [51,] 0.75 0.25 0.35
## [52,] 0.75 0.25 0.45
## [53,] 0.75 0.25 0.55
## [54,] 0.75 0.25 0.65
## [55,] 0.75 0.25 0.85
## [56,] 0.75 0.25 0.95
```

Note that the first star center is in row 1; the second star center is in line 29. All stars thus have $k((1/h) - 1) + 1$ points (including the center), 28 rows per star in this specific case.

Now we compute the model output with the Ishigami function:

```
# MODEL OUTPUT ------------------------------------------------------------

Y <- sensobol::ishigami_Fun(mat)
```

Now we reorganize the data:

```
# REARRANGE DATA ----------------------------------------------------------

n.cross.points <- length(params) * ((1 / h) - 1) + 1
index.centers <- seq(1, nrow(mat), n.cross.points)
mat.nocenters <- matrix(Y[-index.centers], ncol = N)
mat.centers <- matrix(Y[index.centers],
                      nrow = nrow(mat.nocenters) + length(params),
                      ncol = N,
                      byrow = TRUE)
location.centers <- seq(1, nrow(mat.nocenters), 1 / h)
mat.centers[-location.centers, ] <- mat.nocenters

indices <- CutBySize(nrow(mat.centers), nb = length(params))
out <- da <- list()
for(i in 1:nrow(indices)) {
  out[[i]] <- mat.centers[indices[i, "lower"]:indices[i, "upper"], ]
}

print(out)
```

```
## [[1]]
##                 [,1]       [,2]
##  [1,]  0.000000e+00  9.0880682
##  [2,] -5.877853e-01 -0.1903335
##  [3,] -9.510565e-01 -3.7343676
##  [4,] -9.510565e-01 -5.0880682
```

```
##  [5,] -5.877853e-01 -3.7343676
##  [6,]  5.877853e-01 -0.1903335
##  [7,]  9.510565e-01  4.1903335
##  [8,]  9.510565e-01  7.7343676
##  [9,]  5.877853e-01  7.7343676
## [10,]  1.224647e-16  4.1903335
##
## [[2]]
##               [,1]     [,2]
##  [1,] 0.000000e+00 9.088068
##  [2,] 6.909830e-01 7.279051
##  [3,] 1.809017e+00 8.397085
##  [4,] 1.809017e+00 8.397085
##  [5,] 6.909830e-01 7.279051
##  [6,] 6.909830e-01 7.279051
##  [7,] 1.809017e+00 8.397085
##  [8,] 1.809017e+00 9.088068
##  [9,] 6.909830e-01 8.397085
## [10,] 2.999520e-32 7.279051
##
## [[3]]
##       [,1]      [,2]
##  [1,]    0  9.088068
##  [2,]    0 66.910105
##  [3,]    0 26.387923
##  [4,]    0  9.088068
##  [5,]    0  3.789014
##  [6,]    0  3.009741
##  [7,]    0  3.009741
##  [8,]    0  3.789014
##  [9,]    0 26.387923
## [10,]    0 66.910105
```

The list of matrices above includes three slots for three model inputs. Each slot has two columns, each column being a star. The first row of each slot includes the star center.

According to Razavi and Gupta, now we have to take pairs of points for each dimension, using all of the stars. Below I only print the output for the first parameter (first slot), where the two columns reflect all possible combinations between pairs of points, for one star and the other.

```r
# EXTRACT PAIRS OF POINTS ----------------------------------------------------

pairs.points <- lapply(1:length(params), function(x)
  lapply(1:ncol(out[[x]]), function(y)
    t(combn(out[[x]][, y], 2))))

pairs.points <- lapply(pairs.points, function(x) do.call(rbind, x))

print(pairs.points[[1]])
```

```
##             [,1]          [,2]
##  [1,]  0.0000000 -5.877853e-01
##  [2,]  0.0000000 -9.510565e-01
##  [3,]  0.0000000 -9.510565e-01
##  [4,]  0.0000000 -5.877853e-01
##  [5,]  0.0000000  5.877853e-01
##  [6,]  0.0000000  9.510565e-01
##  [7,]  0.0000000  9.510565e-01
##  [8,]  0.0000000  5.877853e-01
##  [9,]  0.0000000  1.224647e-16
## [10,] -0.5877853 -9.510565e-01
## [11,] -0.5877853 -9.510565e-01
## [12,] -0.5877853 -5.877853e-01
## [13,] -0.5877853  5.877853e-01
## [14,] -0.5877853  9.510565e-01
## [15,] -0.5877853  9.510565e-01
## [16,] -0.5877853  5.877853e-01
## [17,] -0.5877853  1.224647e-16
## [18,] -0.9510565 -9.510565e-01
## [19,] -0.9510565 -5.877853e-01
## [20,] -0.9510565  5.877853e-01
## [21,] -0.9510565  9.510565e-01
## [22,] -0.9510565  9.510565e-01
## [23,] -0.9510565  5.877853e-01
## [24,] -0.9510565  1.224647e-16
## [25,] -0.9510565 -5.877853e-01
## [26,] -0.9510565  5.877853e-01
## [27,] -0.9510565  9.510565e-01
## [28,] -0.9510565  9.510565e-01
## [29,] -0.9510565  5.877853e-01
## [30,] -0.9510565  1.224647e-16
## [31,] -0.5877853  5.877853e-01
## [32,] -0.5877853  9.510565e-01
## [33,] -0.5877853  9.510565e-01
## [34,] -0.5877853  5.877853e-01
## [35,] -0.5877853  1.224647e-16
## [36,]  0.5877853  9.510565e-01
## [37,]  0.5877853  9.510565e-01
## [38,]  0.5877853  5.877853e-01
## [39,]  0.5877853  1.224647e-16
## [40,]  0.9510565  9.510565e-01
## [41,]  0.9510565  5.877853e-01
## [42,]  0.9510565  1.224647e-16
## [43,]  0.9510565  5.877853e-01
## [44,]  0.9510565  1.224647e-16
## [45,]  0.5877853  1.224647e-16
## [46,]  9.0880682 -1.903335e-01
## [47,]  9.0880682 -3.734368e+00
```

```
## [48,]  9.0880682 -5.088068e+00
## [49,]  9.0880682 -3.734368e+00
## [50,]  9.0880682 -1.903335e-01
## [51,]  9.0880682  4.190334e+00
## [52,]  9.0880682  7.734368e+00
## [53,]  9.0880682  7.734368e+00
## [54,]  9.0880682  4.190334e+00
## [55,] -0.1903335 -3.734368e+00
## [56,] -0.1903335 -5.088068e+00
## [57,] -0.1903335 -3.734368e+00
## [58,] -0.1903335 -1.903335e-01
## [59,] -0.1903335  4.190334e+00
## [60,] -0.1903335  7.734368e+00
## [61,] -0.1903335  7.734368e+00
## [62,] -0.1903335  4.190334e+00
## [63,] -3.7343676 -5.088068e+00
## [64,] -3.7343676 -3.734368e+00
## [65,] -3.7343676 -1.903335e-01
## [66,] -3.7343676  4.190334e+00
## [67,] -3.7343676  7.734368e+00
## [68,] -3.7343676  7.734368e+00
## [69,] -3.7343676  4.190334e+00
## [70,] -5.0880682 -3.734368e+00
## [71,] -5.0880682 -1.903335e-01
## [72,] -5.0880682  4.190334e+00
## [73,] -5.0880682  7.734368e+00
## [74,] -5.0880682  7.734368e+00
## [75,] -5.0880682  4.190334e+00
## [76,] -3.7343676 -1.903335e-01
## [77,] -3.7343676  4.190334e+00
## [78,] -3.7343676  7.734368e+00
## [79,] -3.7343676  7.734368e+00
## [80,] -3.7343676  4.190334e+00
## [81,] -0.1903335  4.190334e+00
## [82,] -0.1903335  7.734368e+00
## [83,] -0.1903335  7.734368e+00
## [84,] -0.1903335  4.190334e+00
## [85,]  4.1903335  7.734368e+00
## [86,]  4.1903335  7.734368e+00
## [87,]  4.1903335  4.190334e+00
## [88,]  7.7343676  7.734368e+00
## [89,]  7.7343676  4.190334e+00
## [90,]  7.7343676  4.190334e+00
```

Now I assume that I can compute the variogram, using the following formula:

$$\gamma_i(h) = \frac{1}{2N_h} \sum_{j=1}^{N_h} (y_j^{i(h)} - y_j)^2 \tag{1}$$

where N_h is the number of pairs differing by a distance h.

```
# COMPUTE VARIOGRAM ------------------------------------------------------

# Define formula
variogram <- function(x) 1/ (2 * nrow(x)) * sum(x[, 1] - x[, 2]) ^ 2

# Compute
variogr <- unlist(lapply(pairs.points, variogram))

variogr
```

```
## [1] 86.4416043  0.1388889 60.9403561
```

These three numbers reflect the variogram for the three parameters.

Now, the covariogram. According to Razavi and Gupta, it should be computed as follows:

$$C(h) = \frac{1}{2} COV(y_j^{i(h)}, y_j) \tag{2}$$

```
# COMPUTE COVARIOGRAM ----------------------------------------------------

covariog <- unlist(lapply(pairs.points, function(x) 1 / (2 * nrow(x)) * cov(x[, 1], x[, 2])))

covariog
```

```
## [1] 0.0003195397 0.0702550209 0.4909264736
```

I am not sure if the covariogram results make much sense...

How do I compute the total variance?