

Discrepancy as a sensitivity measure in mathematical modeling

R code

Arnald Puy

Contents

1	Preliminary functions	2
2	Goal	3
3	Required functions	3
3.1	C++ functions	3
3.2	Savage ranks function	3
3.3	Discrepancy function	3
3.4	Jansen estimator	5
4	The metamodel	8
4.1	Define the model	11
4.2	Run the model	12
4.3	Arrange model output	13
5	Model for discretization	13
5.1	Define the model	13
5.2	Run the model	14
5.3	Bratley et al. function	16
5.4	Plot Bratley et al.	17
5.5	Plot uncertainty meta-model	18
6	Session information	24
	References	26

1 Preliminary functions

```
# PRELIMINARY -----

# 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)
    }
  }
}

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),
          strip.background = element_rect(fill = "white"))
}

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

# CPP CODE -----

# Source cpp code -----
cpp_functions <- c("cpp_functions.cpp", "L2star_functions.cpp",
                  "L2_functions.cpp", "L2centered_functions.cpp",
                  "L2wraparound_functions.cpp", "L2modified_functions.cpp")

for(i in 1:length(cpp_functions)) {
  Rcpp::sourceCpp(cpp_functions[i])
}
```

2 Goal

We aim at checking whether discrepancy measures can be used as a sensitivity measure to reliably rank parameters in terms of their influence in the model output. To that aim, we explore how well the symmetric L2-discrepancy measure ranks the most important parameters of a set of functions, including for the moment the Oakley and O’Hagan 2004 and the Bratley et al. 1992 functions, and compare its efficacy with that of the Jansen estimator. We do the comparison on Savage-score transformed ranks (Iman and Conover 1987).

The discrepancy is computed directly on the plane of the X_i, Y scatterplots where X_i is a coordinate $(0, 1)$ of the input hypercube and Y is the output rescaled in $0, 1$. For each of the k scatterplots, the points are sorted by one of the X_i . The idea is that a “pattern” (=effect) corresponds to a deviation from a uniform distribution of points (noise) for a non-influential variable: the larger the effect, the larger the discrepancy, e.g., there are areas in the plot where points are denser and areas where they are more rarefied.

One might wonder why use a discrepancy measure when there are the Sobol’ indices. The reply is that many do not understand the theory behind them (and the same applies to moment independent methods or VARS), while everyone understands scatterplots. ‘Explaining’ discrepancy as a measure of the non-uniformity of the scatterplots appears an avenue likely to appeal to some users.

3 Required functions

3.1 C++ functions

3.2 Savage ranks function

```
# SAVAGE SCORES FUNCTION -----

savage_scores <- function(x) {
  true.ranks <- rank(-x)
  p <- sort(1 / true.ranks)
  mat <- matrix(rep(p, length(p)), nrow = length(p), byrow = TRUE)
  mat[upper.tri(mat)] <- 0
  out <- sort(rowSums(mat), decreasing = TRUE)[true.ranks]
  return(out)
}
```

3.3 Discrepancy function

```
# DISCREPANCY FUNCTION -----

# This is based on the function discrepancyCriteria_cpp of
# the sensitivity package

# Function to rescale -----
rescale_fun <- function(x) (x - min(x)) / (max(x) - min(x))

# Function -----
```

```

discrepancy_fun <- function (design, type) {

  X <- as.matrix(design)
  dimension <- ncol(X)
  n <- nrow(X)

  # Rescale if needed-----

  if (min(X) < 0 || max(X) > 1) {

    X <- apply(X, 2, rescale_fun)
  }

  # Compute discrepancy

  if (type == "symmetric") {

    P <- 1 + 2 * X - 2 * X^2
    s1 <- DisS2_Rowprod(t(P), dimension)
    s2 <- DisS2_Crossprod(c(t(X)), dimension)
    R <- sqrt(((4/3)^dimension) - ((2/n) * s1) + ((2^dimension/n^2) * s2))

  } else if (type == "star") {

    dL2 <- DisL2star_Crossprod(t(X), dimension)
    R <- sqrt(3^(-dimension) + dL2)

  } else if (type == "L2") {

    P <- X * (1 - X)
    s1 <- DisL2_Rowprod(t(P), dimension)
    s2 <- DisL2_Crossprod(c(t(X)), dimension)
    R <- sqrt(12^(-dimension) - (((2^(1 - dimension))/n) * s1) + ((1/n^2) * s2))

  } else if (type == "centered") {

    P <- 1 + 0.5 * abs(X - 0.5) - 0.5 * (abs(X - 0.5)^2)
    s1 <- DisC2_Rowprod(t(P), dimension)
    s2 <- DisC2_Crossprod(c(t(X)), dimension)
    R <- sqrt(((13/12)^dimension) - ((2/n) * s1) + ((1/n^2) * s2))

  } else if (type == "wraparound") {

    s1 <- DisW2_Crossprod(t(X), dimension)
    R <- sqrt(-(4/3)^dimension + (1/n^2) * s1)

  } else if (type == "modified") {

```

```

P <- 3 - X^2
s1 <- DisM2_Rowprod(t(P), dimension)
s2 <- DisM2_Crossprod(c(t(X)), dimension)
R <- sqrt(((4/3)^dimension) - (((2^(1 - dimension))/n) * s1) + ((1/n^2) * s2))

}
return(R)
}

# Discrepancy function -----
discrepancy <- function(mat, y, params, type) {
  value <- sapply(1:ncol(mat), function(j) {
    design <- cbind(mat[, j], y)
    value <- discrepancy_fun(design = design, type = type)
  })
  return(value)
}

```

3.4 Jansen estimator

```

# FUNCTION TO COMPUTE JANSEN TI -----

jansen_ti <- function(d, N, params) {
  m <- matrix(d, nrow = N)
  k <- length(params)
  Y_A <- m[, 1]
  Y_AB <- m[, -1]
  f0 <- (1 / length(Y_A)) * sum(Y_A)
  VY <- 1 / length(Y_A) * sum((Y_A - f0) ^ 2)
  value <- (1 / (2 * N) * Rfast::colsums((Y_A - Y_AB) ^ 2)) / VY
  return(value)
}

# LIST OF FUNCTIONS -----

# NEW FUNCTIONS -----

f1_fun <- function(x) 10 * x[, 1] + 0.2 * x[, 2]^3

f2_fun <- function(x) 2 * x[, 1] - x[, 2]^2

f3_fun <- function(x) x[, 1]^2 + x[, 2]^4 + x[, 1] * x[, 2] + x[, 2] * x[, 3]^4

f4_fun <- function(x) 0.2 * exp(x[, 1] - 3) + 2.2 * abs(x[, 2]) + 1.3 * x[, 2]^6 -
  2 * x[, 2]^2 - 0.5 * x[, 2]^4 - 0.5 * x[, 1]^4 + 2.5 * x[, 1]^2 + 0.7 * x[, 1]^3 +
  3 / ((8 * x[, 1] - 2)^2 + (5 * x[, 2] - 3)^2 + 1) + sin(5 * x[, 1]) * cos(3 * x[, 1]^2)

```

```

f6_fun <- function(x) 0.2 * exp(x[, 1] + 2 * x[, 2])

fun_vec <- paste("f", 1:4, "_fun", sep = "")

f_list <- list(f1_fun, f2_fun, f3_fun, f4_fun)
names(f_list) <- fun_vec

# RUN FUNCTIONS -----

output <- ind <- list()

for (i in names(f_list)) {

  if(i == "f3_fun") {

    k <- 3

  } else {

    k <- 2
  }
  params <- paste("$x_", 1:k, "$", sep = "")
  N <- 100
  mat <- sobol_matrices(N = N, params = params, scrambling = 1)
  y <- f_list[[i]](mat)
  y <- rescale_fun(y)
  ind[[i]] <- sobol_indices(Y = y, N = N, params = params)
  output[[i]] <- plot_scatter(data = mat, N = N, Y = y, params = params) +
    labs(x = "$x$", y = "$y$") +
    scale_x_continuous(breaks = pretty_breaks(n = 3)) +
    scale_y_continuous(breaks = pretty_breaks(n = 3)) +
    theme(plot.margin = unit(c(0, 0.2, 0, 0), "cm"))
}

ind

```

```

## $f1_fun
##
## First-order estimator: saltelli | Total-order estimator: jansen
##
## Total number of model runs: 400
##
## Sum of first order indices: 1.002007
##      original sensitivity parameters
## 1: 1.0003862784      Si      $x_1$
## 2: 0.0016209664      Si      $x_2$
## 3: 1.0135578141      Ti      $x_1$

```

```

## 4: 0.0003635617      Ti      $x_2$
##
## $f2_fun
##
## First-order estimator: saltelli | Total-order estimator: jansen
##
## Total number of model runs: 400
##
## Sum of first order indices: 0.9938603
##      original sensitivity parameters
## 1: 0.8111121      Si      $x_1$
## 2: 0.1827481      Si      $x_2$
## 3: 0.8062738      Ti      $x_1$
## 4: 0.2011789      Ti      $x_2$
##
## $f3_fun
##
## First-order estimator: saltelli | Total-order estimator: jansen
##
## Total number of model runs: 500
##
## Sum of first order indices: 0.9551073
##      original sensitivity parameters
## 1: 0.39436570      Si      $x_1$
## 2: 0.51742917      Si      $x_2$
## 3: 0.04331248      Si      $x_3$
## 4: 0.46291049      Ti      $x_1$
## 5: 0.49048129      Ti      $x_2$
## 6: 0.04838575      Ti      $x_3$
##
## $f4_fun
##
## First-order estimator: saltelli | Total-order estimator: jansen
##
## Total number of model runs: 400
##
## Sum of first order indices: 0.7475612
##      original sensitivity parameters
## 1: 0.65955374      Si      $x_1$
## 2: 0.08800745      Si      $x_2$
## 3: 0.85677289      Ti      $x_1$
## 4: 0.32296577      Ti      $x_2$

```

```

# PLOT LIST OF FUNCTIONS -----

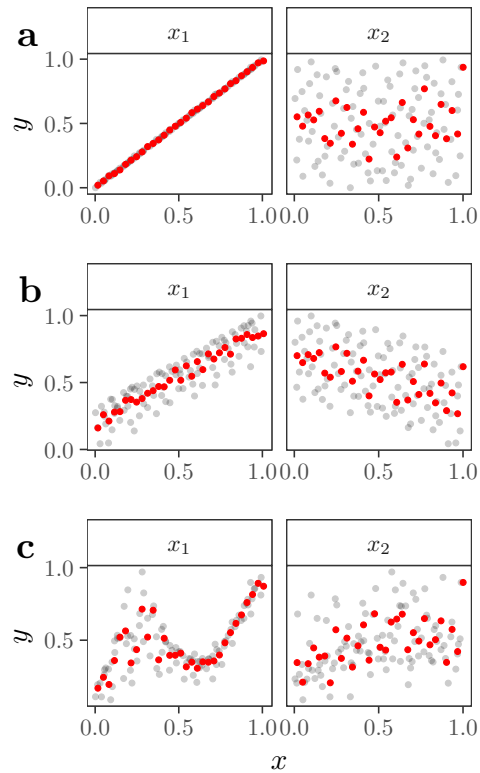
```

```

plot_list <- list(output[[1]] + labs(x = "", y = "$y$"),
                  output[[2]] + labs(x = "", y = "$y$"),
                  output[[4]])

```

```
scat_plot <- plot_grid(plotlist = plot_list, ncol = 1, labels = "auto")
scat_plot
```



4 The metamodel

```
# RANDOM FUNCTIONS AND DISTRIBUTIONS -----

# Functions -----
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)) + 0.03125) / 2,
  Oscillation = function(x) x ^ 2 - 0.2 * cos(7 * pi * x)
```



```

)

# Random distributions -----
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) logitnorm::qlogitnorm(x, 0, 3.16)
  # Logit-normal, Bates too?
)

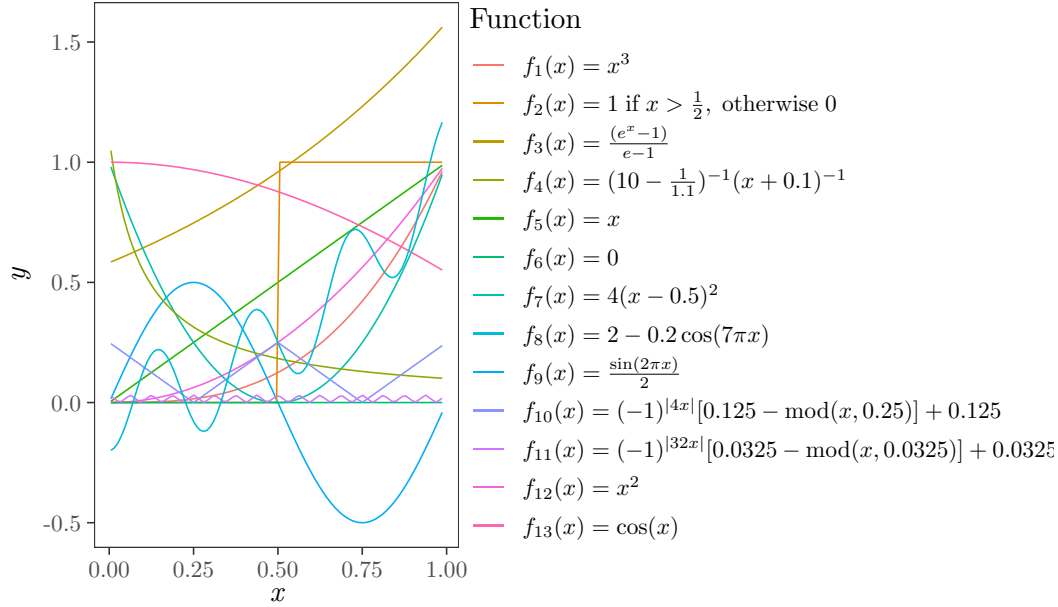
random_distributions <- function(X, phi) {
  names_ff <- names(sample_distributions)
  if(!phi == length(names_ff) + 1) {
    out <- sample_distributions[[names_ff[phi]]](X)
  } else {
    temp <- sample(names_ff, ncol(X), replace = TRUE)
    out <- sapply(seq_along(temp), function(x)
      sample_distributions[[temp[x]]](X[,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

# PLOT METAFUNCTION -----

ggplot(data.frame(x = runif(100)), aes(x)) +
  map(1:length(function_list), function(nn) {
    stat_function(fun = function_list[[nn]],
      geom = "line",
      aes_(color = factor(names(function_list[nn]))))
  }) +
  labs(color= "Function", linetype = "Function",
    x = expression(italic(x)),
    y = expression(italic(y))) +
  scale_color_discrete(labels = c("$f_1(x) = x^3$",
    "$f_2(x) = 1 \\hspace{1mm} \\mbox{if} \\hspace{1mm} x > \\fr",
    "$f_3(x) = \\frac{(e^x - 1)}{e-1}$",
    "$f_4(x) = (10-\\frac{1}{1.1})^{-1}(x + 0.1)^{-1}$",
    "$f_5(x) = x$",
    "$f_6(x) = 0$",
    "$f_7(x) = 4(x - 0.5)^2$",
    "$f_8(x) = 2 - 0.2 \\cos(7 \\pi x)$",
    "$f_9(x) = \\frac{\\sin(2 \\pi x)}{2}$",
    "$f_{10}(x) = (-1)^{|4x|} [0.125- \\mbox{mod}(x, 0.25)] + 0

```

```
theme_AP() +
theme(legend.text.align = 0)
```



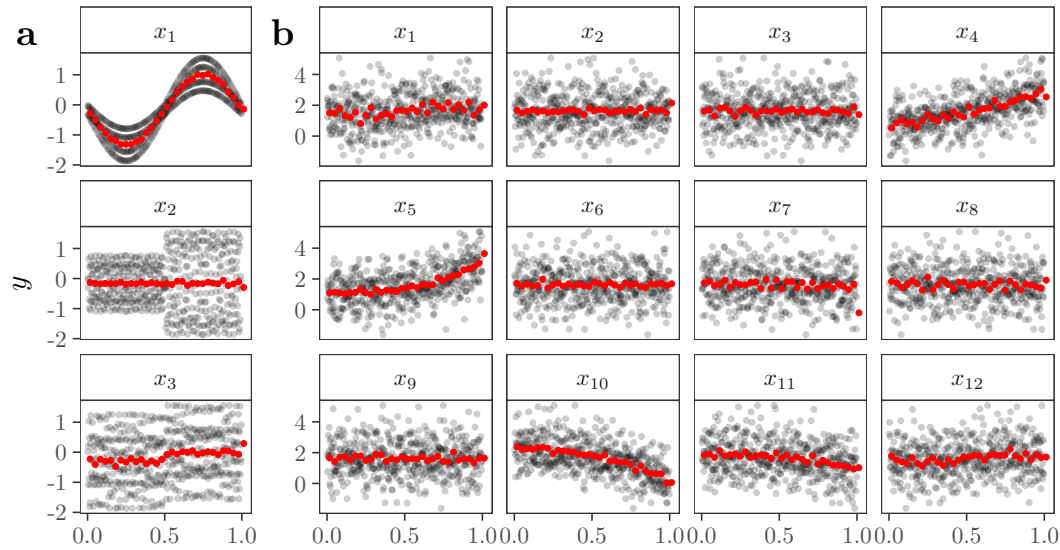
PLOT SCATTERS FROM METAFUNCTION -----

```
k_epsilon <- list(c(3, 2), c(12, 6))
out <- list()

for(i in 1:length(k_epsilon)) {
  N <- 2^9
  params <- paste("$x_{", 1:k_epsilon[[i]][[1]], "}$", sep = "")
  mat <- sobol_matrices(N = N, params = params)
  y <- metafunction(mat, epsilon = k_epsilon[[i]][[2]])
  out[[i]] <- plot_scatter(data = mat, N = N, Y = y, params = params) +
    scale_x_continuous(breaks = pretty_breaks(n = 3)) +
    theme(plot.margin = unit(c(0, 0.1, 0, 0), "cm")) +
    labs(x = "", y = "$y$")
}

scatters_plot <- plot_grid(out[[1]] + facet_wrap(~variable, ncol = 1),
  out[[2]] + facet_wrap(~variable, ncol = 4) + labs(y = ""),
  ncol = 2, labels = "auto", rel_widths = c(0.24, 0.76))

scatters_plot
```



4.1 Define the model

```
# DEFINE MODEL -----

model_fun <- function(tau, epsilon, base.sample.size, cost.discrepancy, phi, k) {

  params <- paste("X", 1:k, sep = "")

  if (tau == 1) {

    type <- "R"

  } else if (tau == 2) {

    type <- "QRN"
  }
  set.seed(epsilon)
  discrepancy.mat <- sobol_matrices(matrices = "A", N = cost.discrepancy,
                                   params = params,
                                   type = type)

  set.seed(epsilon)
  jansen.mat <- sobol_matrices(matrices = c("A", "AB"), N = base.sample.size,
                              params = params,
                              type = type)

  all.matrices <- rbind(discrepancy.mat, jansen.mat)
  set.seed(epsilon)
  transformed.all.matrices <- random_distributions(X = all.matrices, phi = phi)

  y <- sensobol::metafunction(data = transformed.all.matrices, epsilon = epsilon)
```

```

type <- c("symmetric", "star", "L2", "centered", "wraparound", "modified")

discrepancy.value <- lapply(type, function(x)
  discrepancy(mat = all.matrices[1:cost.discrepancy, ],
    y = y[1:cost.discrepancy], params = params, type = x))

jansen.value <- jansen_ti(d = y[(cost.discrepancy + 1):length(y)],
  N = base.sample.size, params = params)

savage.discrepancy <- lapply(discrepancy.value, savage_scores)
jansen.discrepancy <- savage_scores(jansen.value)

out <- lapply(savage.discrepancy, function(x)
  cor(x, jansen.discrepancy))

return(out)
}

```

4.2 Run the model

```

# CREATE SAMPLE MATRIX -----

N <- 2^10
params <- c("epsilon", "phi", "k", "tau", "base.sample.size")
mat <- sobol_matrices(matrices = "A", N = N, params = params)

# Define distributions -----

mat[, "epsilon"] <- floor(qunif(mat[, "epsilon"], 1, 200))
mat[, "phi"] <- floor(mat[, "phi"] * 8) + 1
mat[, "k"] <- floor(qunif(mat[, "k"], 3, 50))
mat[, "tau"] <- floor(mat[, "tau"] * 2) + 1
mat[, "base.sample.size"] <- floor(qunif(mat[, "base.sample.size"], 10, 100))

cost.jansen <- mat[, "base.sample.size"] * (mat[, "k"] + 1)
cost.discrepancy <- cost.jansen

final.mat <- cbind(mat, cost.jansen, cost.discrepancy)

# RUN MODEL -----

y <- mclapply(1:nrow(final.mat), function(i) {
  model_fun(tau = final.mat[i, "tau"],
    epsilon = final.mat[i, "epsilon"],
    base.sample.size = final.mat[i, "base.sample.size"],
    cost.discrepancy = final.mat[i, "cost.discrepancy"],

```

```

    phi = final.mat[i, "phi"],
    k = final.mat[i, "k"]}},
mc.cores = floor(detectCores() * 0.75))

```

4.3 Arrange model output

```

# ARRANGE DATA -----

y <- lapply(y, unlist)
output <- data.table(do.call(rbind, y))
discrepancy_methods <- c("symmetric", "star", "L2", "centered",
                        "wraparound", "modified")
colnames(output) <- discrepancy_methods

final.output <- data.table(cbind(final.mat, output))

# Write model output
fwrite(final.output, "final.output.csv")

```

5 Model for discretization

5.1 Define the model

```

# DISCRETIZATION FUNCTIONS -----

plot_ti <- function(ind, params) {

  out <- data.table(ind) %>%
    .[, parameters:= params] %>%
    ggplot(., aes(parameters, ind)) +
    geom_bar(stat = "identity", fill = "#00BFC4", color = "black") +
    labs(x = "", y = "$T_i$") +
    theme_AP()

  return(out)
}

# FUNCTION TO DISCRETIZE THE BRATLEY ET AL. FUNCTION -----

discretization_fun <- function(N, k, epsilon, plot.scatterplot = FALSE,
                              plot.jansen = FALSE, savage.scores = TRUE) {

  params <- paste("$x_", 1:k, "$", sep = "")
  mat <- sobol_matrices(N = N, params = params, matrices = c("A", "AB"))
  mat2 <- mat # mat2 = matrix for discretization

```

```

# Determine how many parameters to discretize (n.discrete),
# which parameters to discretize (n.parameters), and the
# number of levels (n.levels)
set.seed(epsilon)
n.discrete <- sample(1:ceiling(k / 2), size = 1)
set.seed(epsilon)
n.parameters <- sample(1:k, size = n.discrete)
set.seed(epsilon)
n.levels <- sample(2:5, size = 1)

# Discretization of the sampled parameters
mat2[, n.parameters] <- floor(mat2[, n.parameters] * n.levels) + 1

# Compute output and sobol indices (Jansen estimator)
y <- bratley1992_Fun(mat2)
ind2 <- jansen_ti(d = y, N = N, params = params)

# Conditional
if (plot.scatterplot == TRUE) {
  out <- plot_scatter(data = mat2, N = N, Y = y, params = params)
}
if (plot.jansen == TRUE) {
  out <- plot_ti(ind = ind2, params = params)
}
if (savage.scores == TRUE) {
  sobol_ranks <- savage_scores(ind2)

# Compute discrepancy
type <- c("symmetric", "star", "L2", "centered", "wraparound", "modified")
discrepancy.out <- lapply(type, function(x)
  discrepancy(mat = mat[1:N, ], y = y[1:N], params = params, type = x))

discrepancy.ranks <- lapply(discrepancy.out, savage_scores)
# Correlation on savage scores
out <- lapply(discrepancy.ranks, function(x) cor(sobol_ranks, x))
}
return(out)
}

```

5.2 Run the model

```

# CREATE SAMPLE MATRIX -----
N <- 2^9
params <- c("k", "epsilon")
mat <- sobol_matrices(matrices = "A", N = N, params = params)

```

```

# Define distributions -----

mat[, "epsilon"] <- floor(qunif(mat[, "epsilon"], 1, N))
mat[, "k"] <- floor(qunif(mat[, "k"], 6, 50))

# RUN MODEL -----

y.discrete <- mclapply(1:nrow(mat), function(i) {
  discretization_fun(N = 2^12,
    k = mat[i, "k"],
    epsilon = mat[i, "epsilon"],
    plot.scatterplot = FALSE,
    plot.jansen = FALSE,
    savage.scores = TRUE)},
  mc.cores = floor(detectedCores() * 0.75))

# ARRANGE OUTPUT -----

y.discrete <- lapply(y.discrete, unlist)
output <- data.table(do.call(rbind, y.discrete))
discrepancy_methods <- c("symmetric", "star", "L2", "centered",
  "wraparound", "modified")
colnames(output) <- discrepancy_methods

final.output.discrete <- data.table(cbind(mat, output))

melt(final.output.discrete, measure.vars = discrepancy_methods) %>%
  .[, .(mean = mean(value, na.rm = TRUE),
    median = median(value, na.rm = TRUE)), variable] %>%
  .[order(-median)]

##      variable      mean    median
## 1: wraparound 0.7359107 0.8223486
## 2: symmetric  0.6722620 0.7701982
## 3: centered  0.5956070 0.7384221
## 4:          L2 0.6365500 0.7201984
## 5:          star 0.5889423 0.6375529
## 6: modified  0.5889365 0.6375529

# EXPORT OUTPUT DISCRETE -----

fwrite(final.output.discrete, "final.output.discrete.csv")

# RUN MODEL TO GET SCATTERPLOTS -----

N.scatter <- 2^6
N.indices <- 2^10
row.selected <- 63

```

```

y.scatters <- discretization_fun(N = N.scatter,
                                k = mat[row.selected, "k"],
                                epsilon = mat[i, "epsilon"],
                                plot.scatterplot = TRUE,
                                plot.jansen = FALSE,
                                savage.scores = FALSE)

# RUN MODEL TO GET JANSEN TI -----

y.indices <- discretization_fun(N = N.indices,
                                k = mat[row.selected, "k"],
                                epsilon = mat[i, "epsilon"],
                                plot.scatterplot = FALSE,
                                plot.jansen = TRUE,
                                savage.scores = FALSE)

```

5.3 Bratley et al. function

```

# PLOT ORIGINAL BRATLEY ET AL. FUNCTION -----

# Settings -----
N <- 2^6
k <- 6
params <- paste("$x_", 1:k, "$", sep = "")
matrices <- c("A", "AB")

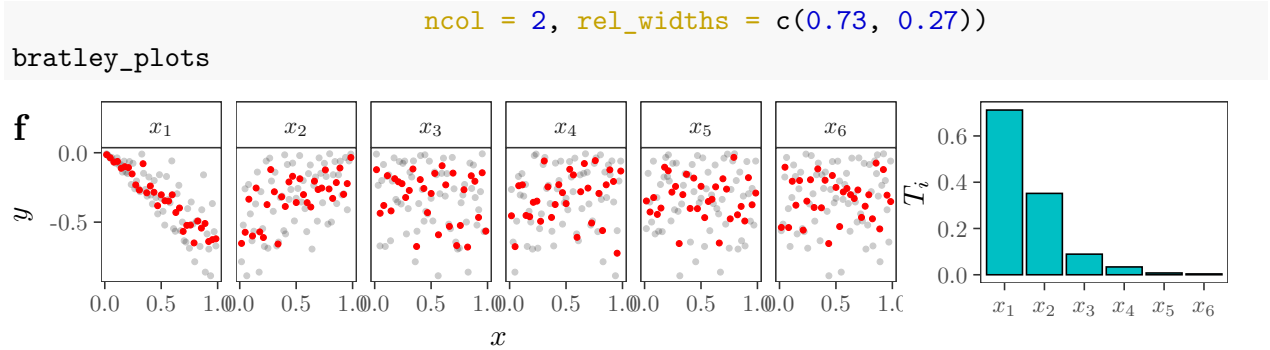
# Compute -----
mat <- sobol_matrices(N = N, params = params, matrices = matrices)
y <- bratley1992_Fun(mat)
ind <- jansen_ti(d = y, N = N, params = params)

# Plot scatter -----
bratley_scatter <- plot_scatter(data = mat, N = N, Y = y, params = params) +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  facet_wrap(~variable, ncol = 6) +
  labs(x = "$x$", y = "$y$") +
  theme(plot.margin = unit(c(0, 0.1, 0, 0), "cm"))

# Plot Ti -----
bratley_ti <- plot_ti(ind = ind, params = params) +
  theme(plot.margin = unit(c(0, 0.1, 0, 0), "cm")) +
  labs(x = "", y = "$T_i$") +
  scale_x_discrete(guide = guide_axis(n.dodge = 1))

# Merge -----
bratley_plots <- plot_grid(bratley_scatter, bratley_ti, labels = c("f", ""),

```

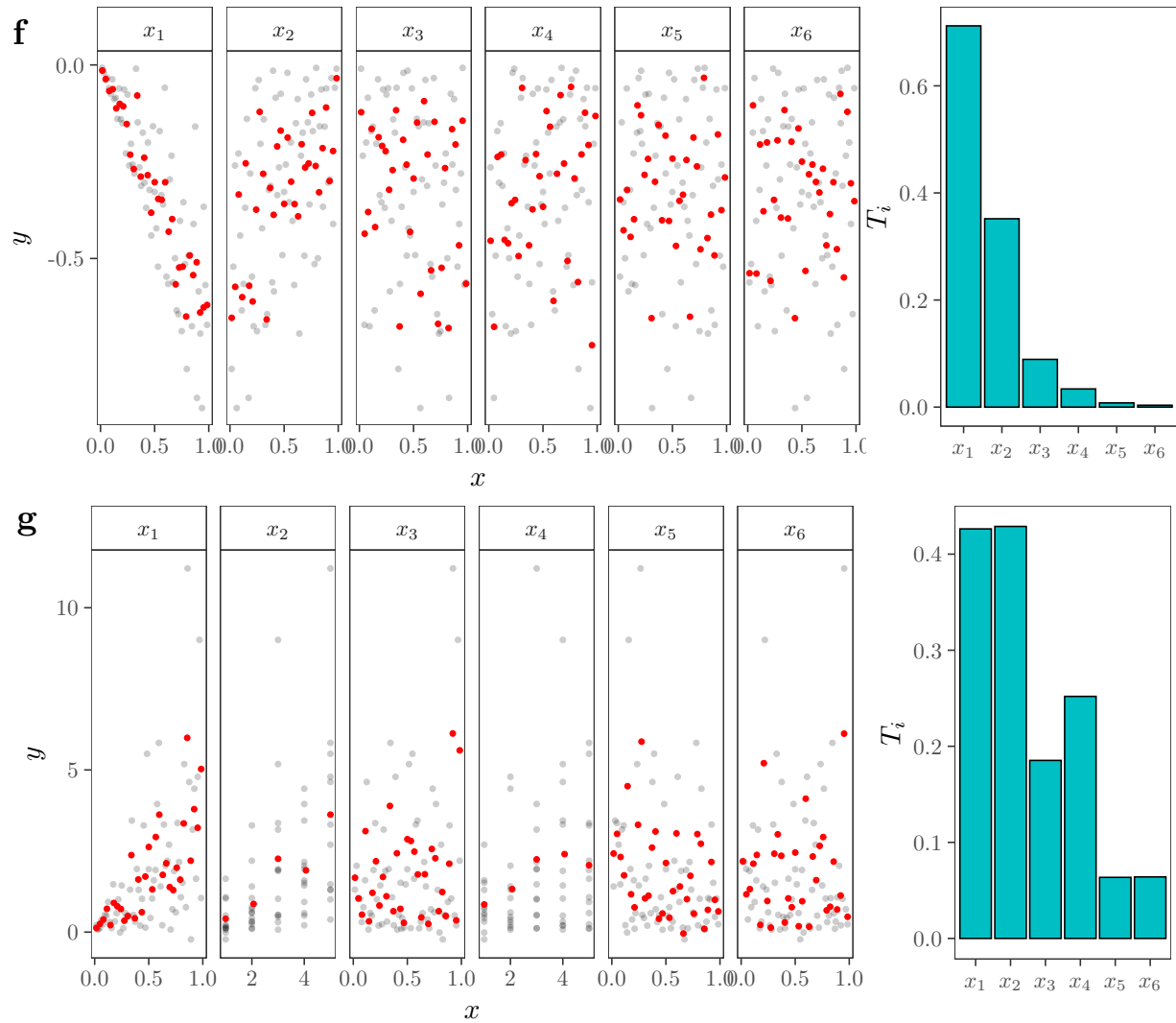
5.4 Plot Bratley et al.

```
# PLOT BRATLEY ET AL. DISCRETIZED -----

bottom.plots <- plot_grid(y.scatters +
  facet_wrap(~variable, ncol = 6, scales = "free_x") +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  labs(x = "$x$", y = "$y$"),
  y.indices,
  ncol = 2, rel_widths = c(0.73, 0.27))

all.bottom.plots <- plot_grid(bratley_plots, bottom.plots, ncol = 1, labels = c("", "g"),
  rel_heights = c(0.47, 0.53))

all.bottom.plots
```



5.5 Plot uncertainty meta-model

```
# PLOT UNCERTAINTY -----

# SCATTERPLOT -----

supp.labs <- c("Symmetric", "Star", "$L_2$", "Centered", "Wrap-around", "Modified")
names(supp.labs) <- discrepancy_methods

# SCATTERPLOT -----

a <- melt(final.output, measure.vars = discrepancy_methods) %>%
  .[, variable:= factor(variable, levels = c("symmetric", "wraparound",
                                             "centered", "L2",
                                             "star", "modified"))] %>%
  ggplot(., aes(cost.discrepancy, k, color = value)) +
```

```

geom_point(size = 0.4) +
scale_colour_gradientn(colours = c("black", "purple", "red", "orange",
                                   "yellow", "lightgreen"),
                      name = expression(italic(r)),
                      breaks = pretty_breaks(n = 3)) +
scale_x_continuous(breaks = pretty_breaks(n = 3)) +
scale_y_continuous(breaks = pretty_breaks(n = 3)) +
labs(x = "N° of model runs", y = "$d$") +
facet_wrap(~variable, ncol = 6, labeller = labeller(variable = supp.labs)) +
theme_AP() +
theme(legend.position = "none",
      strip.background = element_rect(fill = "white"),
      strip.text.x = element_text(size = 8))

# BOXPLOTS -----

discrete.dt <- melt(final.output.discrete, measure.vars = discrepancy_methods) %>%
  .[, Function:= "Bratley et al. 1992"] %>%
  .[, .(variable, value, Function)]

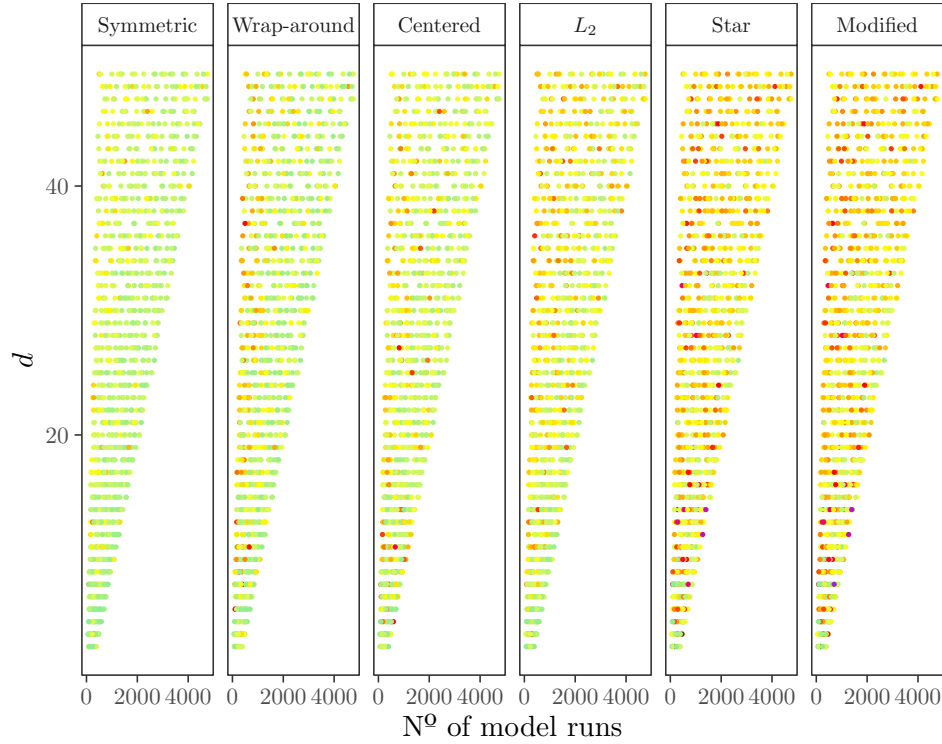
final.dt <- melt(final.output, measure.vars = discrepancy_methods) %>%
  .[, Function:= "Meta-model"] %>%
  .[, .(variable, value, Function)]

b <- rbind(discrete.dt, final.dt) %>%
  ggplot(., aes(reorder(variable, -value), value, fill = Function)) +
  geom_boxplot(outlier.size = 0.5) +
  scale_x_discrete(labels = supp.labs) +
  scale_fill_manual(values = wes_palette(2, name = "Chevalier1")) +
  labs(x = "", y = "$r$") +
  theme_AP() +
  theme(legend.position = "none",
        plot.margin = unit(c(0, 0.1, 0, 0), "cm"))

# PLOT SCATTERPLOT -----

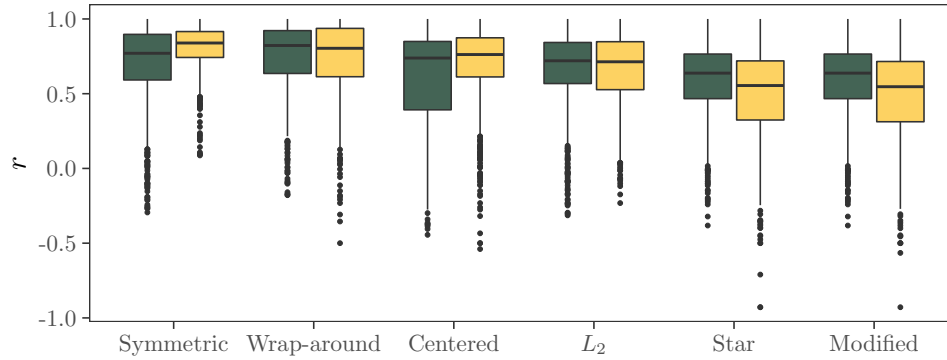
a

```



PLOT BOXPLOTS

b



MERGE ALL PLOTS

```

legend <- get_legend(a + theme(legend.position = "top",
                                legend.margin=margin(0, 0, 0, 0),
                                legend.box.margin=margin(-7,-7,-7,-7)))

legend.boxplot <- get_legend(b + theme(legend.position = "top",
                                         legend.margin=margin(0, 0, 0, 0),
                                         legend.box.margin=margin(-7,-7,-7,-7)))

boxplots.plot <- plot_grid(legend.boxplot, b, ncol = 1, rel_heights = c(0.15, 0.85),
                           labels = c("e", ""))

scatters.plot <- plot_grid(legend, a, ncol = 1, rel_heights = c(0.18, 0.82),

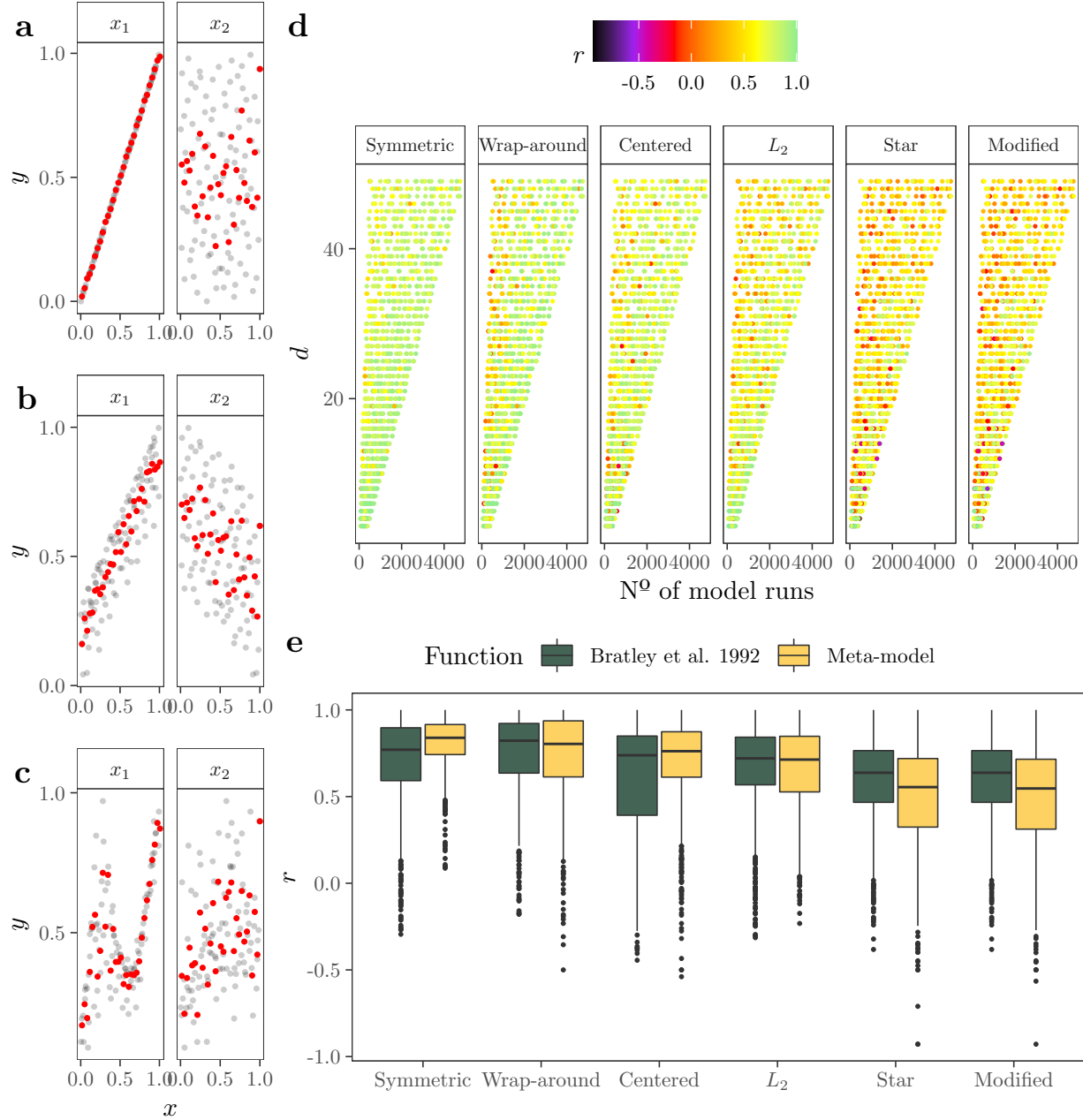
```

```

labels = c("d", "")
da <- plot_grid(scatters.plot, boxplots.plot, ncol = 1, rel_heights = c(0.55, 0.45))

full.plot <- plot_grid(scat_plot, da, ncol = 2, rel_widths = c(0.25, 0.75))
full.plot

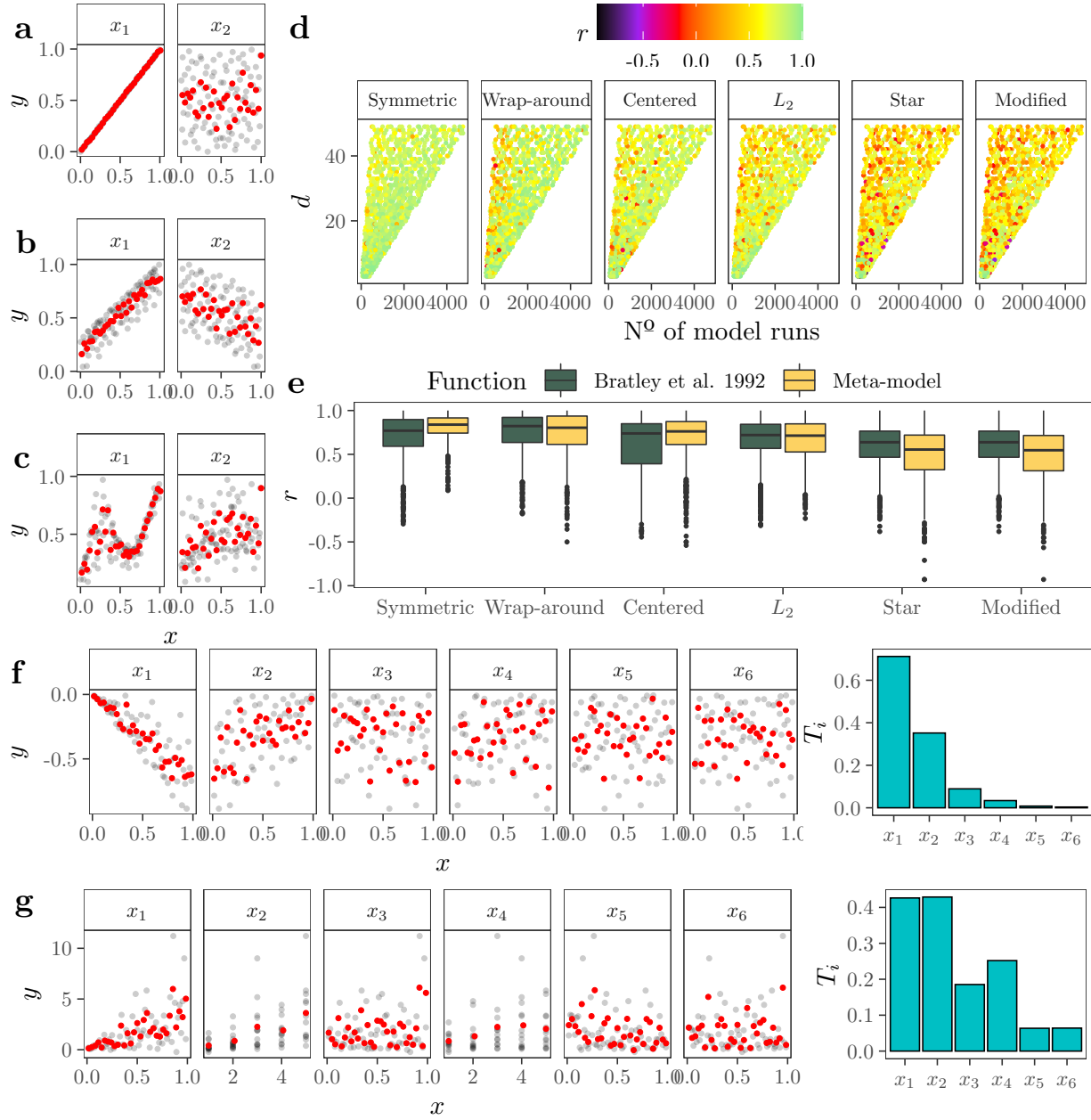
```



```

plot_grid(full.plot, all.bottom.plots, ncol = 1, rel_heights = c(0.6, 0.45),
labels = c("", ""))

```



STATISTICS

```

rbind(discrete.dt, final.dt) %>%
  .[, .(mean = mean(value), median = median(value)), .(Function, variable)] %>%
  .[order(-mean, variable)]

```

```

##           Function  variable    mean    median
## 1:      Meta-model symmetric 0.8115546 0.8391197
## 2:      Meta-model wraparound 0.7400277 0.8036229
## 3: Bratley et al. 1992 wraparound 0.7359107 0.8223486
## 4:      Meta-model  centered 0.7028626 0.7618533
## 5: Bratley et al. 1992 symmetric 0.6722620 0.7701982
## 6:      Meta-model      L2 0.6652102 0.7133971

```

## 7: Bratley et al. 1992	L2	0.6365500	0.7201984
## 8: Bratley et al. 1992	centered	0.5956070	0.7384221
## 9: Bratley et al. 1992	star	0.5889423	0.6375529
## 10: Bratley et al. 1992	modified	0.5889365	0.6375529
## 11: Meta-model	star	0.5000740	0.5545182
## 12: Meta-model	modified	0.4916879	0.5466692

6 Session information

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

```
sessionInfo()
```

```
## R version 4.2.0 (2022-04-22)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Monterey 12.4
##
## 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] wesanderson_0.3.6 cowplot_1.1.1 benchmarkme_1.0.7 doParallel_1.0.17
## [5] iterators_1.0.14 foreach_1.5.2 scales_1.2.0 RcppAlgos_2.5.3
## [9] forcats_0.5.1 stringr_1.4.0 dplyr_1.0.9 purrr_0.3.4
## [13] readr_2.1.2 tidyr_1.2.0 tibble_3.1.7 ggplot2_3.3.6
## [17] tidyverse_1.3.1 data.table_1.14.2 sensobol_1.1.1
##
## loaded via a namespace (and not attached):
## [1] Rcpp_1.0.8.3 lattice_0.20-45 lubridate_1.8.0
## [4] gmp_0.6-5 assertthat_0.2.1 digest_0.6.29
## [7] utf8_1.2.2 R6_2.5.1 cellranger_1.1.0
## [10] backports_1.4.1 reprex_2.0.1 evaluate_0.15
## [13] httr_1.4.3 pillar_1.7.0 Rdpack_2.3
## [16] rlang_1.0.2 readxl_1.4.0 rstudioapi_0.13
## [19] Matrix_1.4-1 tikzDevice_0.12.3.1 rmarkdown_2.14
## [22] munsell_0.5.0 broom_0.8.0 compiler_4.2.0
## [25] modelr_0.1.8 xfun_0.31 pkgconfig_2.0.3
## [28] htmltools_0.5.2 tidyselect_1.1.2 codetools_0.2-18
## [31] fansi_1.0.3 crayon_1.5.1 tzdb_0.3.0
## [34] dbplyr_2.1.1 withr_2.5.0 rbibutils_2.2.8
## [37] grid_4.2.0 jsonlite_1.8.0 gtable_0.3.0
## [40] lifecycle_1.0.1 DBI_1.1.2 magrittr_2.0.3
## [43] cli_3.3.0 stringi_1.7.6 fs_1.5.2
## [46] benchmarkmeData_1.0.4 xml2_1.3.3 ellipsis_0.3.2
## [49] generics_0.1.2 vctrs_0.4.1 tools_4.2.0
## [52] glue_1.6.2 hms_1.1.1 fastmap_1.1.0
## [55] yaml_2.3.5 colorspace_2.0-3 filehash_2.4-3
```



```
## [58] rvest_1.0.2          knitr_1.39             haven_2.5.0
```

```
## 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
```

References

Iman, Ronald L., and W. J. Conover. 1987. "A measure of top-down correlation." *Technometrics* 29 (3): 351–57.