The battle of total-order sensitivity estimators

Arnald Puy, Samuele Lo Piano, William Becker and Andrea Saltelli

Contents

```r
# 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)
    }
  }
}

# Install development version of sensobol
remotes::install_github("arnaldpuy/sensobol")

# Load the packages
loadPackages(c("Rcpp", "RcppArmadillo", "tidyverse", "parallel", "foreach",
               "doParallel", "Rfast", "data.table", "scales", "cowplot",
               "benchmarkme", "logitnorm", "sensobol", "ggrepel"))

# 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("2020-01-23",
           R.version ="3.6.1",
           checkpointLocation = getwd())
```

1 Functions

1.1 Savage scores

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

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)
}
```

1.2 Sensitivity indices

```
# VARS FUNCTIONS -----------------------------------------------------------------

vars_matrices <- function(star.centers, params, h) {
  out <- center <- sections <- A <- B <- AB <- X <- out <- list()
  mat <- randtoolbox::sobol(n = star.centers, 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 cut by size
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)
}

# Function to compute VARS-TI
vars_ti <- function(Y, star.centers, params, h) {
  n.cross.points <- length(params) * ((1 / h) - 1) + 1
  index.centers <- seq(1, length(Y), n.cross.points)
  mat <- matrix(Y[-index.centers], ncol = star.centers)
  indices <- CutBySize(nrow(mat), nb = length(params))
  out <- list()
  for(i in 1:nrow(indices)) {
    out[[i]] <- mat[indices[i, "lower"]:indices[i, "upper"], ]
  }
  d <- lapply(1:length(params), function(x)
    lapply(1:ncol(out[[x]]), function(j) {
      da <- c(out[[x]][, j][1],
              rep(out[[x]][, j][-c(1, length(out[[x]][, j]))], each = 2),
              out[[x]][, j][length(out[[x]][, j])])
    }))
  out <- lapply(d, function(x) lapply(x, function(y) matrix(y, nrow = length(y) / 2, byrow = TI
  variogr <- unlist(lapply(out, function(x) lapply(x, function(y)
    mean(0.5 * (y[, 1] - y[, 2]) ^ 2))) %>%
      lapply(., function(x) do.call(rbind, x)) %>%
      lapply(., mean))
  covariogr <- unlist(lapply(out, function(x)
    lapply(x, function(y) cov(y[, 1], y[, 2]))) %>%
      lapply(., function(x) Rfast::colmeans(do.call(rbind, x))))
  VY <- var(Y[index.centers])
  Ti <- (variogr + covariogr) / VY
  output <- data.table::data.table(Ti)
  output[, `:=`(parameters = params)]
  return(output)
}
```

```
# COMPUTATION OF SOBOL' Ti INDICES -----------------------------------------------

sobol_Ti <- function(d, N, params, total) {
  m <- matrix(d, nrow = N)
  k <- length(params)
  if(total == "jansen" | total == "homma" | total == "sobol" | total == "monod" |
     total == "glen") {
    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)
```

```
    # VY <- 1 / length(Y_A) * (sum(Y_A ^ 2) -
    # (1 / N * sum(Y_A ^ 2))) ((Variance used by Becker))
  }
  if(total == "jansen") {
    Ti <- (1 / (2 * N) * Rfast::colsums((Y_A - Y_AB) ^ 2)) / VY
  } else if(total == "homma") {
    Ti <- (VY - (1 / N) * Rfast::colsums(Y_A * Y_AB) + f0 ^ 2) / VY
  } else if(total == "sobol") {
    Ti <- ((1 / N) * Rfast::colsums(Y_A * (Y_A - Y_AB))) / VY
  } else if(total == "monod") {
    Ti <- 1 - (1 / N * Rfast::colsums(Y_A * Y_AB) -
                 (1/ N * Rfast::colsums((Y_A + Y_AB) / 2)) ^ 2) /
      (1 / N * Rfast::colsums((Y_A ^ 2 + Y_AB ^ 2) / 2) -
         (1/ N * Rfast::colsums((Y_A + Y_AB) / 2)) ^ 2)
  } else if(total == "glen") {
    Ti <- 1 - (1 / (N - 1) *
                Rfast::colsums(((Y_A - mean(Y_A)) * (Y_AB - Rfast::colmeans(Y_AB))) /
                                 sqrt(var(Y_A) * Rfast::colVars(Y_AB))))
  }
  if(total == "azzini" | total == "lamboni") {
    Y_A <- m[, 1]
    Y_B <- m[, 2]
    Y_AB <- m[, 3:(3 + k - 1)]
    Y_BA <- m[, (ncol(m) - k + 1):ncol(m)]
    f0 <- 1 / (2 * N) * sum(Y_A + Y_B)
    VY <- 1 / (2 * N - 1) * sum((Y_A - f0) ^ 2 + (Y_B - f0) ^ 2)
  }
  if(total == "azzini") {
    Ti <- 1 - abs(Rfast::colsums((Y_A - Y_BA) * (Y_B - Y_AB)) /
                    (1 / 2 * Rfast::colsums((Y_A - Y_B) ^ 2 + (Y_AB - Y_BA) ^ 2)))
  } else if(total == "lamboni") {
    Ti <- (1 / (4 * N) * colSums((Y_A - Y_AB) ^ 2 + (Y_B - Y_BA) ^ 2, na.rm = TRUE)) / VY
  }
  if(total == "owen") {
    Y_A <- m[, 1]
    Y_B <- m[, 2]
    Y_BA <- m[, 3:(3 + k - 1)]
    Y_CB <- m[, (ncol(m) - k + 1):ncol(m)]
    VY <- sapply(1:k, function(j)
      mean(Rfast::rowmeans(m[,c(1, 2, 2 + j, 2 + j + k)]^ 2)) -
        mean(Rfast::rowmeans(m[,c(1, 2, 2 + j, 2 + j + k)])) ^2)
    Ti <- (VY - (1 / N * Rfast::colsums((Y_B - Y_CB) * (Y_BA - Y_A)))) / VY
  }
  output <- data.table(Ti)
  output[, `:=`(parameters = paste("X", 1:k, sep = ""))]
  return(output)
}
```

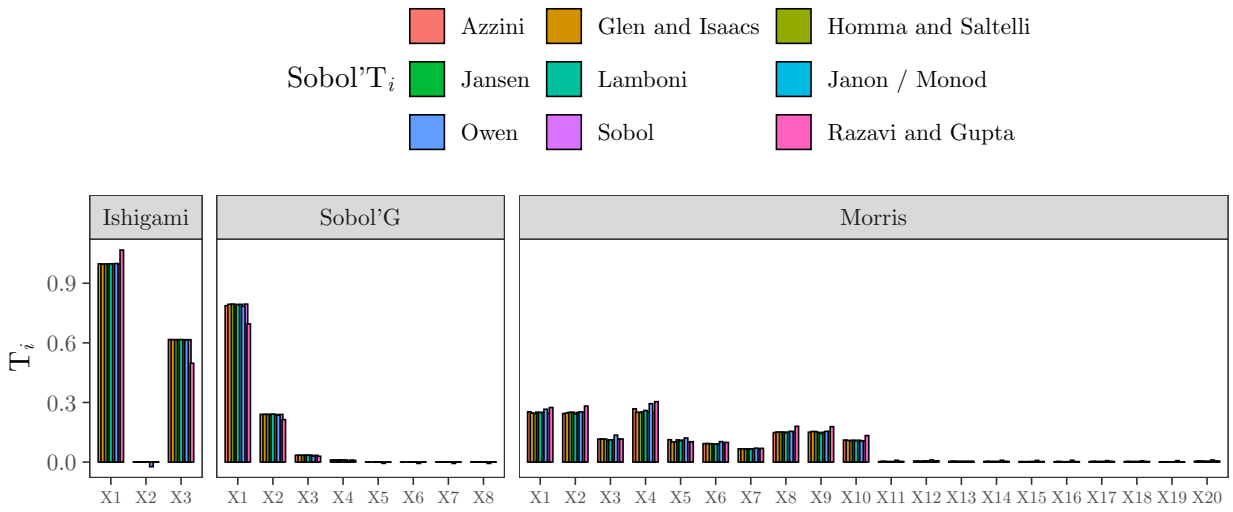
```r
# CHECK THAT ALL TI ESTIMATORS WORK ----------------------------------------------

# Settings
estimators <- c("jansen", "sobol", "homma", "azzini", "monod", "lamboni", "glen", "owen")
test_functions <- c("Ishigami", "Sobol'G", "Morris")
N <- 2 ^ 11

# Run model
ind <- Y <- mt <- list()
for(i in estimators) {
  for(j in test_functions) {
    if(i == "jansen" | i == "sobol" | i == "homma" | i == "monod" | i == "glen") {
      matrices <- c("A", "AB")
    } else if(i == "azzini" | i == "lamboni"){
      matrices <- c("A", "B", "AB", "BA")
    } else if(i == "owen") {
      matrices <- c("A", "B", "BA", "CB")
    }
    if(j == "Ishigami") {
      k <- 3
      modelRun <- sensobol::ishigami_Fun
    } else if(j == "Sobol'G") {
      k <- 8
      modelRun <- sensobol::sobol_Fun
    } else if(j == "Morris") {
      k <- 20
      modelRun <- sensitivity::morris.fun
    }
    mt[[i]][[j]] <- sobol_matrices(N = N, params = paste("X", 1:k, sep = ""), matrices = matri
    Y[[i]][[j]] <- modelRun(mt[[i]][[j]])
    ind[[i]][[j]] <- sobol_Ti(d = Y[[i]][[j]], params = paste("X", 1:k, sep = ""),
                              N = N, total = i)
  }
}
```

```
## Registered S3 method overwritten by 'sensitivity':
##   method    from
##   print.src dplyr
```

```r
# Run model for VARS
star.centers <- 200
h <- 0.2
vars.ind <- Y <- list()
for(j in test_functions) {
  if(j == "Ishigami") {
    k <- 3
    modelRun <- sensobol::ishigami_Fun
  } else if(j == "Sobol'G") {
```

```
    k <- 8
    modelRun <- sensobol::sobol_Fun
  } else if(j == "Morris") {
    k <- 20
    modelRun <- sensitivity::morris.fun
  }
  mt[[j]] <- vars_matrices(star.centers = star.centers,
                           params = paste("X", 1:k, sep = ""),
                           h = h)
  Y[[j]] <- modelRun(mt[[j]])
  vars.ind[[j]] <- vars_ti(Y = Y[[j]], star.centers = star.centers,
                           params = paste("X", 1:k, sep = ""), h = h)
}

vars.ind <- rbindlist(vars.ind, idcol = "Function")[
  , estimator:= "vars"] %>%
  setcolorder(., c("estimator", "Function", "Ti", "parameters"))
```

```
# PLOT SENSITIVITY INDICES ------------------------------------------------------

lapply(ind, function(x) rbindlist(x, idcol = "Function")) %>%
  rbindlist(., idcol = "estimator") %>%
  rbind(vars.ind) %>%
  .[, parameters:= factor(parameters, levels = paste("X", 1:20, sep = ""))] %>%
  .[, Function:= factor(Function, levels = test_functions)] %>%
  ggplot(., aes(parameters, Ti, fill = estimator)) +
  geom_bar(stat = "identity",
           position = position_dodge(0.7),
           color = "black") +
  scale_fill_discrete(name = expression(paste("Sobol' ", T[italic(i)])),
                      labels = c("Azzini", "Glen and Isaacs", "Homma and Saltelli",
                                 "Jansen", "Lamboni", "Janon / Monod", "Owen",
                                 "Sobol", "Razavi and Gupta")) +
  facet_grid(~Function,
             scales = "free_x",
             space = "free_x") +
  labs(x = "",
       y = expression(T[italic(i)])) +
  theme_AP() +
  theme(axis.text.x = element_text(size = 6.5),
        legend.position = "top") +
  guides(fill = guide_legend(nrow = 3,
                             byrow = TRUE))
```



2 The metafunction

```
# CREATE 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)
)
```

```
# PLOT METAFUNCTION --------------------------------------------------------------

a <- 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])),
                       linetype = factor(names(function_list[nn]))))
  }) +
  labs(color= "Function", linetype = "Function",
       x = expression(italic(x)),
       y = expression(italic(y))) +
  theme_AP() +
  theme(legend.position = "right")
```

```
## Warning: `mapping` is not used by stat_function()
```

```
## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()
```

```
a
```

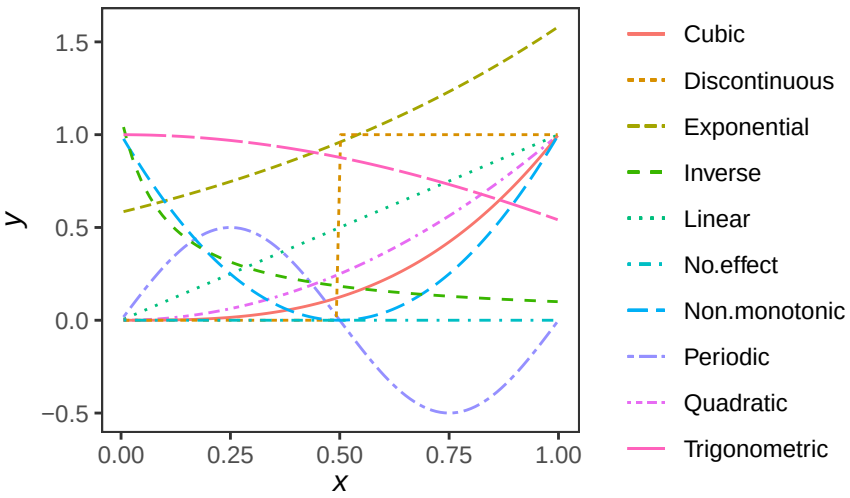


Figure 1: Functions used in the metafunction of Becker (2019).

```
# CREATE FUNCTION FOR RANDOM DISTRIBUTIONS ------------------------------------

sample_distributions <- list(
  "uniform" = function(x) x,
  "normal" = function(x) qnorm(x, 0.5, 0.2),
  "beta" = function(x) qbeta(x, 8, 2),
  "beta2" = function(x) qbeta(x, 2, 8),
  "beta3" = function(x) qbeta(x, 2, 0.5),
  "beta4" = function(x) qbeta(x, 0.5, 2),
  "logitnormal" = function(x) qlogitnorm(x, 0, 3.16)
  # Logit-normal, Bates too?
)
```

```r
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)
}
```

```r
# PLOT DISTRIBUTIONS ------------------------------------------------------------

names_ff <- names(sample_distributions)
prove <- randtoolbox::sobol(n = 1000, dim = length(names_ff))

out <- data.table(sapply(seq_along(names_ff), function(x)
  sample_distributions[[names_ff[x]]](prove[, x])))

b <- data.table::melt(out) %>%
  ggplot(., aes(value, group = variable, colour = variable)) +
  geom_density() +
  scale_color_discrete(labels = c("U(0, 1)",
                                  "N(0.5, 0.2)",
                                  "Beta(8, 2)",
                                  "Beta(2, 8)",
                                  "Beta(2, 0.5)",
                                  "Beta(0.5, 2)",
                                  "Logitnormal(0, 3.16)"),
                       name = "") +
  labs(x = expression(italic(x)),
       y = "Density") +
  theme_AP()
```

```
## Warning in melt.data.table(out): id.vars and measure.vars are internally
## guessed when both are 'NULL'. All non-numeric/integer/logical type columns are
## considered id.vars, which in this case are columns []. Consider providing at
## least one of 'id' or 'measure' vars in future.
```

```r
b
```

```r
# MERGE METAFUNCTION PLOT AND DISTRIBUTIONS PLOT --------------------------------

plot_grid(a, b, ncol = 1, labels = "auto", align = "hv")
```

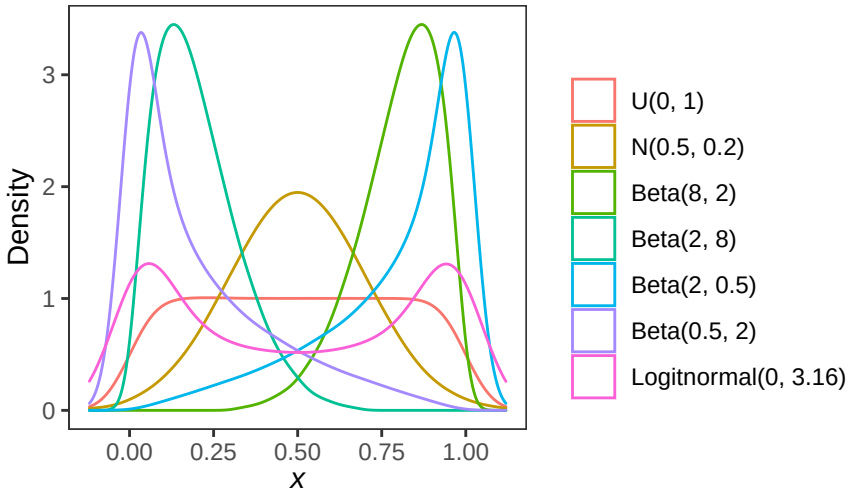
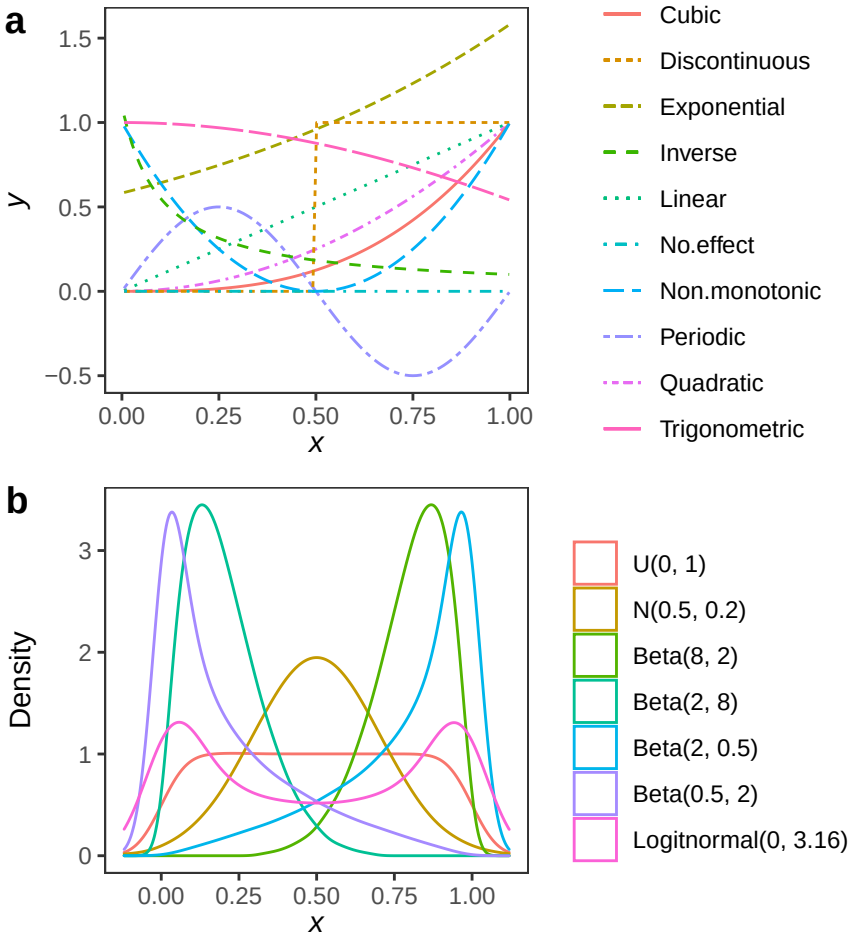


Figure 2: Distributions used in the metafunction of Becker (2019).




```
# EXEMPLIFY THE METAFUNCTION WITH AN EXAMPLE ------------------------------------

N <- 5000 # Sample size
R <- 100 # Number of bootstrap replications
k <- 55 # Number of model inputs
```

```r
k_2 <- 0.5 # Fraction of active pairwise interactions
k_3 <- 0.3 # Fraction of active three-wise interactions
epsilon <- 5 # to reproduce the results
params <- paste("X", 1:k, sep = "")
A <- sobol_matrices(N = N, params = params)

# Compute
Y <- metafunction(data = A, k_2 = k_2, k_3 = k_3, epsilon = epsilon)
ind <- sobol_indices(Y = Y, N = N, params = params, R = R, boot = TRUE)
```

3 The model

3.1 Settings

```r
# DEFINE SETTINGS ----------------------------------------------------------------

N <- 2 ^ 11 # Sample size of sample matrix
h <- 0.2 # step for VARS
R <- 500 # Number of bootstrap replicas
n_cores <- ceiling(detectCores() * 0.5)
order <- "first"
params <- c("k", "N_t", "k_2", "k_3", "epsilon", "phi", "delta")
N.high <- 2 ^ 11 # Maximum sample size of the large sample matrix
```

3.2 Sample matrix

```r
# CREATE SAMPLE MATRIX -----------------------------------------------------------

mat <- sobol_matrices(N = N, params = params, order = order)
mat[, 1] <- floor(qunif(mat[, 1], 3, 100)) # k
mat[, 2] <- floor(qunif(mat[, 2], 10, 1000)) # N_t
mat[, 3] <- round(qunif(mat[, 3], 0.3, 0.5), 2) # k_2
mat[, 4] <- round(qunif(mat[, 4], 0.1, 0.3), 2) # k_3
mat[, 5] <- floor(qunif(mat[, 5], 1, 200)) # Epsilon
mat[, 6] <- floor(mat[, 6] * (8 - 1 + 1)) + 1 # Phi
mat[, 7] <- floor(mat[, 7] * (3 - 1 + 1)) + 1 # Phi

colnames(mat) <- params

N.all <- apply(mat, 1, function(x) ceiling(x["N_t"] / (x["k"] + 1)))
N.azzini <- apply(mat, 1, function(x) ceiling(x["N_t"] / (2 * x["k"] + 2)))
N.vars <- apply(mat, 1, function(x) floor(x["N_t"] / ((x[["k"]] * ((1 / h) -1) + 1))))
# N.vars <- apply(mat, 1, function(x) floor((x["N_t"] * h) / x["k"]))

tmp <- cbind(mat, N.all, N.azzini, N.vars)
sel <- c("N.all", "N.azzini", "N.vars")
```

```r
mat <- as.matrix(data.table(tmp)[, (sel):= lapply(.SD, function(x)
  ifelse(x == 1 | x == 0, 2, x)), .SDcols = (sel)])

C.all <- apply(mat, 1, function(x) x["N.all"] * (x["k"] + 1))
C.azzini <- apply(mat, 1, function(x) x["N.azzini"] * (2 * x["k"] + 2))
C.vars <- apply(mat, 1, function(x) x["N.vars"] * (x["k"] * ((1 / h) - 1) + 1))

mat <- cbind(mat, C.all, C.azzini, C.vars)
```

3.3 Define the model

```r
# DEFINE MODEL ------------------------------------------------------------------

model_Ti <- function(k, N.all, N.azzini, N.vars, h,
                     N.high, k_2, k_3, epsilon, phi, delta) {
  ind <- list()
  estimators <- c("jansen", "sobol", "homma", "monod", "azzini",
                  "lamboni", "glen", "owen", "vars")
  all.but.azzini <- sobol_matrices(N = N.all, params = paste("X", 1:k, sep = ""),
                                   matrices = c("A", "AB"))
  azzini <- sobol_matrices(N = N.azzini, params = paste("X", 1:k, sep = ""),
                           matrices = c("A", "B", "AB", "BA"))
  owen.matrix <- sobol_matrices(N = N.azzini, params = paste("X", 1:k, sep = ""),
                           matrices = c("A", "B", "BA", "CB"))
  vars.matrix <- vars_matrices(star.centers = N.vars, params = paste("X", 1:k, sep = ""),
                               h = h)
  large.matrix <- sobol_matrices(N = N.high, params = paste("X", 1:k, sep = ""),
                                 matrices = c("A", "AB"))
  set.seed(epsilon)
  all.matrices <- random_distributions(X = rbind(all.but.azzini, azzini,
                                                 owen.matrix, vars.matrix,
                                                 large.matrix),
                                       phi = phi)
  output <- sensobol::metafunction(data = all.matrices,
                                   k_2 = k_2,
                                   k_3 = k_3,
                                   epsilon = epsilon)
  full.ind <- sobol_Ti(d = tail(output, nrow(large.matrix)),
                       N = N.high,
                       params = paste("X", 1:k, sep = ""),
                       total = "jansen")
  full.ind[, sample.size:= "N"]

  # Define indices of Y for estimators
  Nt.all.but.azzini <- N.all * (k + 1)
  Nt.azzini.owen <- N.azzini * ((2 * k) + 2)
```

```
  Nt.vars <- N.vars * (k * ((1 / h) - 1) + 1)
  lg.all.but.azzini <- 1:Nt.all.but.azzini
  lg.azzini <- (length(lg.all.but.azzini) + 1):(length(lg.all.but.azzini) + Nt.azzini.owen)
  lg.owen <- (max(lg.azzini) + 1):(max(lg.azzini) + Nt.azzini.owen)
  lg.vars <- (max(lg.owen) + 1): (max(lg.owen) + Nt.vars)

  for(i in estimators) {
    if(i == "jansen" | i == "sobol" | i == "homma" | i == "monod" | i == "glen") {
      y <- output[lg.all.but.azzini]
      n <- N.all
    } else if(i == "azzini" | i == "lamboni") {
      y <- output[lg.azzini]
      n <- N.azzini
    } else if(i == "owen") {
      y <- output[lg.owen]
      n <- N.azzini
    } else if(i == "vars") {
      y <- output[lg.vars]
      star.centers <- N.vars
    }
    if(!i == "vars") {
      ind[[i]] <- sobol_Ti(d = y, N = n, params = paste("X", 1:k, sep = ""), total = i)
    }
    if(i == "vars") {
      ind[[i]] <- vars_ti(Y = y, star.centers = star.centers,
                          params = paste("X", 1:k, sep = ""), h = h)
    }
    ind[[i]][, sample.size:= "n"]
    ind[[i]] <- rbind(ind[[i]], full.ind)
  }
  # Arrange data
  out <- rbindlist(ind, idcol = "estimator")
  out.wide <- dcast(out, estimator + parameters ~ sample.size, value.var = "Ti")
  if(delta == 1) { # Regular Pear
    final <- out.wide[, .(correlation = cor(N, n)), estimator]
  } else if(delta == 2) { # kendall tau
    final <- out.wide[, .(correlation = pcaPP::cor.fk(N, n)), estimator]
  } else { # Savage ranks
    final <- out.wide[, lapply(.SD, savage_scores), .SDcols = c("N", "n"), estimator][
      , .(correlation = cor(N, n)), estimator]
  }
  return(final)
}
```

3.4 Run the model

```
# RUN MODEL ------------------------------------------------------------------

# Define parallel computing
cl <- makeCluster(n_cores)
registerDoParallel(cl)

# Compute
Y.ti <- foreach(i=1:nrow(mat),
                .packages = c("sensobol", "data.table", "pcaPP",
                              "logitnorm", "dplyr")) %dopar%
  {
    model_Ti(k = mat[[i, "k"]],
             k_2 = mat[[i, "k_2"]],
             k_3 = mat[[i, "k_3"]],
             epsilon = mat[[i, "epsilon"]],
             phi = mat[[i, "phi"]],
             delta = mat[[i, "delta"]],
             N.all = mat[[i, "N.all"]],
             N.azzini = mat[[i, "N.azzini"]],
             N.vars = mat[[i, "N.vars"]],
             N.high = N.high,
             h = h)
  }

# Stop parallel cluster
stopCluster(cl)
```

3.5 Arrange output

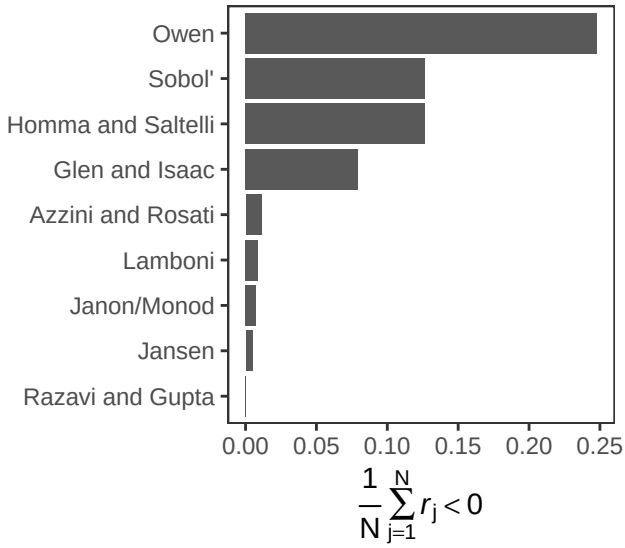
```
# ARRANGE OUTPUT -------------------------------------------------------------

out_cor <- rbindlist(Y.ti, idcol = "row")

mt.dt <- data.table(mat) %>%
  .[, row:= 1:.N]

full_output <- merge(mt.dt, out_cor) %>%
  .[, Nt:= ifelse(estimator == "azzini" | estimator == "lamboni"
                  | estimator == "owen", N.azzini * (2 * k + 2),
                  ifelse(estimator == "vars", N.vars * (k * ((1 / h) -1) + 1), N.all * (k + 1)))
  .[, estimator:= ifelse(estimator %in% "azzini", "Azzini and Rosati",
                         ifelse(estimator %in% "homma", "Homma and Saltelli",
                                ifelse(estimator %in% "monod", "Janon/Monod",
                                       ifelse(estimator %in% "jansen", "Jansen",
                                              ifelse(estimator %in% "glen", "Glen and Isaac",
```

```
                                                    ifelse(estimator %in% "lamboni", "Lamboni"
                                                           ifelse(estimator %in% "owen", "Owen
                                                                  ifelse(estimator %in% "vars"
                                                                         "Sobol'"))))))))] %>%
  .[, ratio:= Nt / k]

# Define A matrix
A <- full_output[,.SD[1:N], estimator]
```

```
# EXPORT OUTPUT ------------------------------------------------------------------

fwrite(A, "A.csv")
fwrite(full_output, "full_output.csv")
```

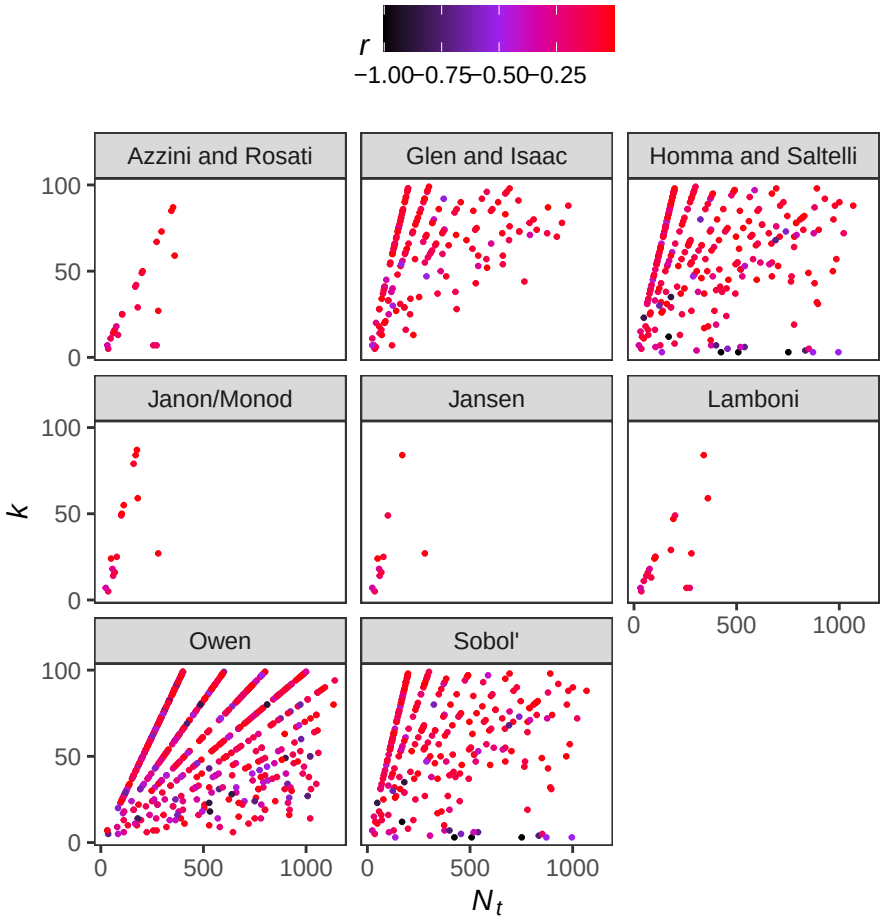
4 Uncertainty analysis

```
# PLOT PROPORTION OF NEGATIVE VALUES ---------------------------------------------

A[, sum(correlation < 0)/ .N, estimator] %>%
  ggplot(., aes(reorder(estimator, V1), V1)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(y = expression(frac(1, N)~sum(italic(r)[j] < 0, j==1, N)),
       x = "") +
  theme_AP()
```




```
# MAP VALUES WITH NEGATIVE R -----------------------------------------------------

index.neg <- A[, .I[correlation < 0]]

A[index.neg] %>%
```

```
  ggplot(., aes(Nt, k, color = correlation)) +
  geom_point(size = 0.5) +
  scale_colour_gradientn(colours = c("black", "purple", "red"),
                         name = expression(italic(r))) +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  labs(x = expression(italic(N[t])),
       y = expression(italic(k))) +
  facet_wrap(~estimator) +
  theme_AP() +
  theme(legend.position = "top")
```




```
# PLOT OUTPUT ------------------------------------------------------------------

# Compute median and quantiles
dt_median <- A[, .(median = median(correlation),
                                  low.ci = quantile(correlation, 0.25),
                                  high.ci = quantile(correlation, 0.75)), estimator]

A[, .(median = median(correlation),
      low.ci = quantile(correlation, 0.25),
```

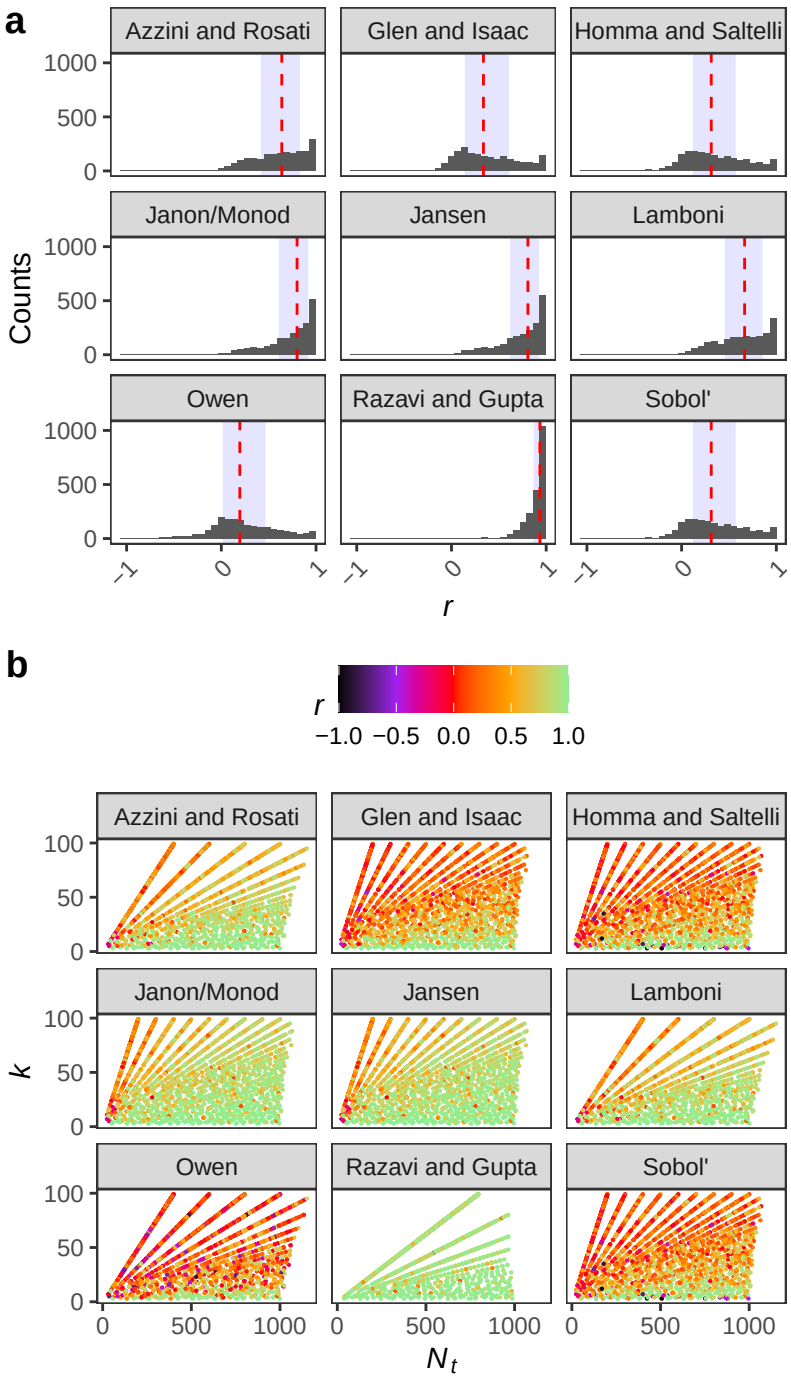
```
      high.ci = quantile(correlation, 0.75)), estimator][order(median)]
```

```
##             estimator    median    low.ci   high.ci
## 1:               Owen 0.1952707 0.0000000 0.4828150
## 2:             Sobol' 0.3125050 0.1039631 0.5867431
## 3: Homma and Saltelli 0.3128686 0.1042273 0.5867431
## 4:     Glen and Isaac 0.3375124 0.1261882 0.6229799
## 5:  Azzini and Rosati 0.6385908 0.4015177 0.8462274
## 6:            Lamboni 0.6652399 0.4407569 0.8695652
## 7:        Janon/Monod 0.8000000 0.5916509 0.9318265
## 8:             Jansen 0.8071220 0.6040774 0.9386104
## 9:   Razavi and Gupta 0.9336505 0.8512948 0.9798027
```

```
a <- ggplot(A, aes(correlation)) +
  geom_rect(data = dt_median,
            aes(xmin = low.ci,
                xmax = high.ci,
                ymin = -Inf,
                ymax = Inf),
            fill = "blue",
            color = "white",
            alpha = 0.1,
            inherit.aes = FALSE) +
  geom_histogram() +
  geom_vline(data = dt_median, aes(xintercept = median),
             lty = 2,
             color = "red") +
  facet_wrap(~estimator,
             ncol = 3) +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  labs(x = expression(italic(r)),
       y = "Counts") +
  theme_AP() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Scatterplot
b <- ggplot(A, aes(Nt, k, color = correlation)) +
  geom_point(size = 0.1) +
  scale_colour_gradientn(colours = c("black", "purple", "red", "orange", "lightgreen"),
                         name = expression(italic(r))) +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  labs(x = expression(italic(N[t])),
       y = expression(italic(k))) +
  facet_wrap(~estimator,
             ncol = 3) +
  theme_AP() +
```

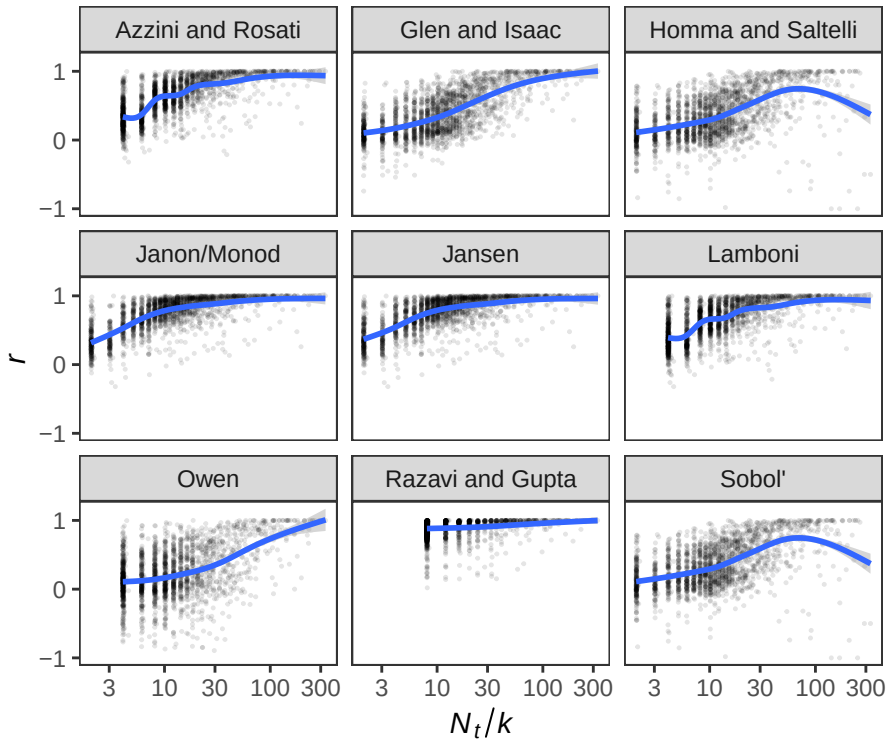
```
  theme(legend.position = "top")

# Ratio
c <- ggplot(A, aes(ratio, correlation)) +
  geom_point(alpha = 0.1, size = 0.2) +
  facet_wrap(~estimator,
             ncol = 3) +
  geom_smooth() +
  labs(x = expression(italic(N[t]/k)),
       y = expression(italic(r))) +
  scale_x_log10() +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  theme_AP()

# Merge plot
plot_grid(a, b, ncol = 1, labels = "auto", rel_heights = c(0.85, 1))
```

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

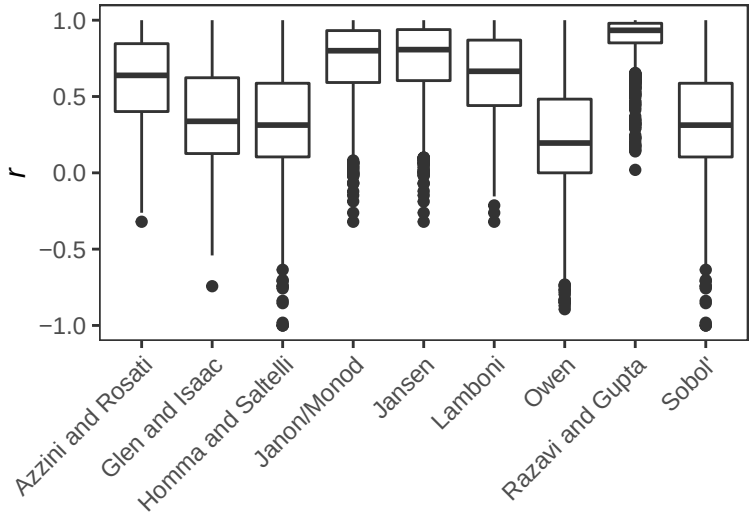



```
# DISPLAY THE RATIO NT/K FOR EACH SIMULATION AND ESTIMATOR -----------------------

c
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

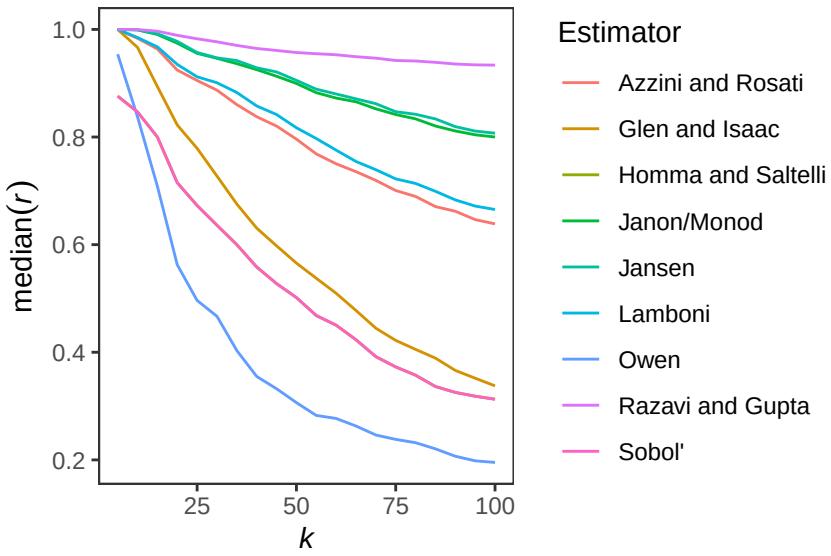



```
# PLOT BOXPLOT ------------------------------------------------------------------

ggplot(A, aes(estimator, correlation)) +
  geom_boxplot() +
  labs(x = "",
       y = expression(italic(r))) +
  theme_AP() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



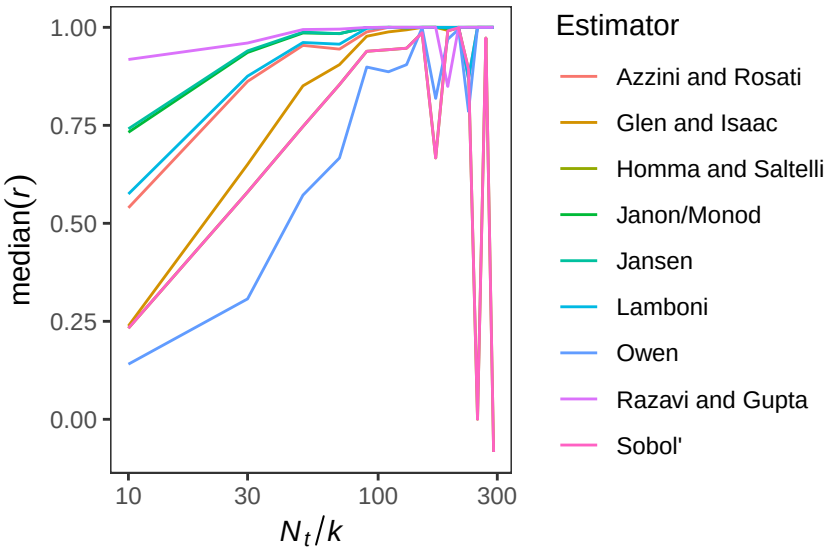

```
# PLOT MEDIANS ------------------------------------------------------------------

# Median k
```

```r
vv <- seq(5, 100, 5)

dt <- lapply(vv, function(x) A[k <= x, median(correlation), estimator])

names(dt) <- vv

rbindlist(dt, idcol = "k") %>%
  .[, k:= as.numeric(k)] %>%
  ggplot(., aes(k, V1, color = estimator)) +
  scale_color_discrete(name = "Estimator") +
  labs(x = expression(italic(k)),
       y = expression(median(italic(r)))) +
  geom_line() +
  theme_AP()
```




```r
# Median Nt/k
dt.tmp <- A[, .(min = min(ratio), max = max(ratio))]

v <-  seq(0, ceiling(dt.tmp$max), 20)
a <- c(v[1], rep(v[-c(1, length(v))], each = 2), v[length(v)])
indices <- matrix(a, ncol = 2 ,byrow = TRUE)


out <- list()
for(i in 1:nrow(indices)) {
  out[[i]] <- A[ratio > indices[i, 1] & ratio < indices[i, 2]]
}

names(out) <- rowmeans(indices)

# Plot
lapply(out, function(x) x[, median(correlation, na.rm = TRUE), estimator]) %>%
```

```
  rbindlist(., idcol = "N") %>%
  .[, N:= as.numeric(N)] %>%
  ggplot(., aes(N, V1, group = estimator, color = estimator)) +
  geom_line() +
  labs(x = expression(italic(N[t]/k)),
       y = expression(median(italic(r)))) +
  scale_color_discrete(name = "Estimator") +
  scale_x_log10() +
  theme_AP()
```



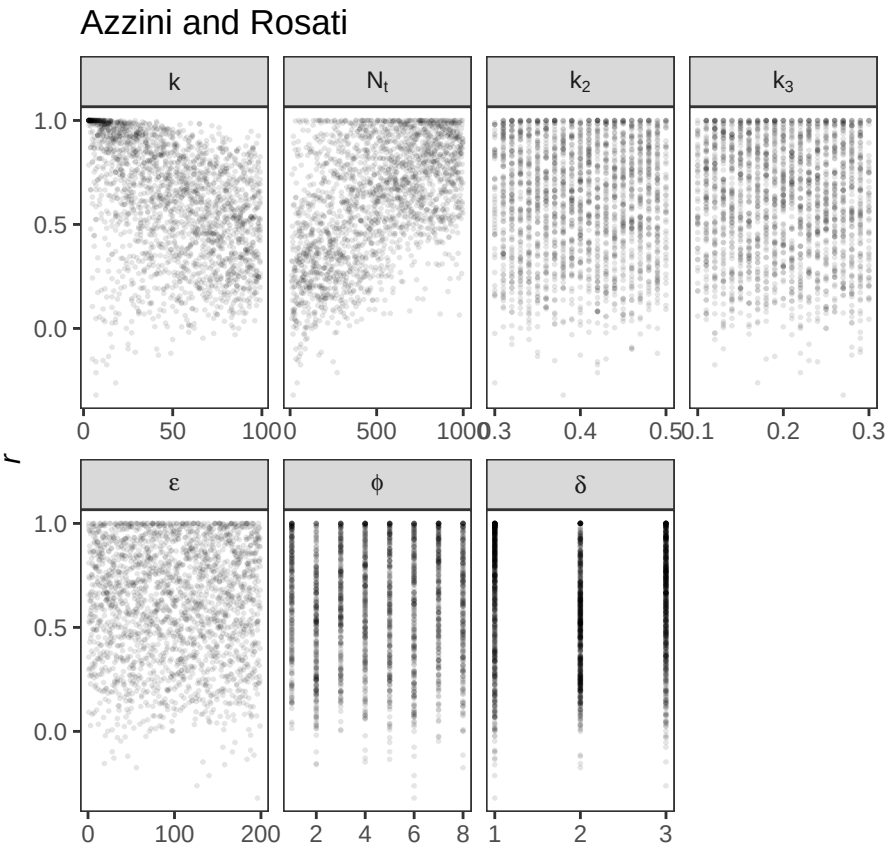
5 Sensitivity analysis

5.1 Scatterplots

```
# SCATTERPLOTS OF MODEL OUTPUT AGAINST PARAMETERS ---------------------------------

A <- setnames(A, c("N_t", "k_2", "k_3"), c("N[t]", "k[2]", "k[3]"))

scatter.dt <- melt(A, measure.vars = c("k", "N[t]", "k[2]", "k[3]",
                                       "epsilon", "phi", "delta")) %>%
  split(., .$estimator)

gg <- list()
for(i in names(scatter.dt)) {
  gg[[i]] <- ggplot(scatter.dt[[i]], aes(value, correlation)) +
    geom_point(alpha = 0.1, size = 0.3) +
    facet_wrap(~ variable,
               scales = "free_x",
               labeller = label_parsed,
               ncol = 4) +
    scale_color_manual(values = c("#00BFC4", "#F8766D")) +
```

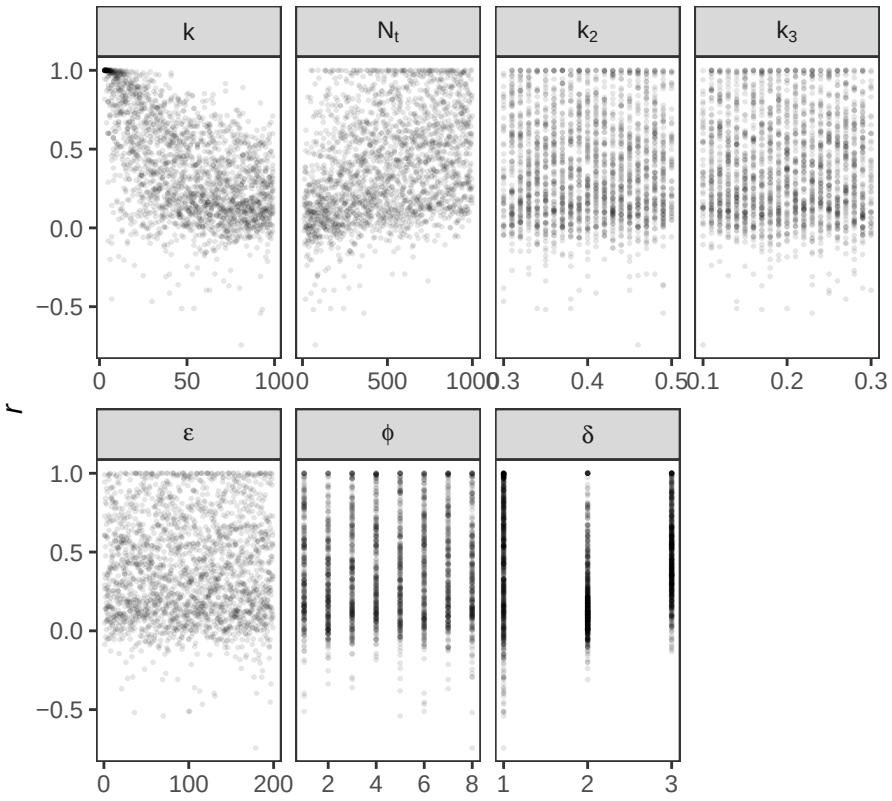
```
    scale_x_continuous(breaks = pretty_breaks(n = 3)) +
    theme_AP() +
    labs(x = "", y = expression(italic(r))) +
    theme() +
    ggtitle(names(scatter.dt[i]))
}

gg
```

```
## $`Azzini and Rosati`
```

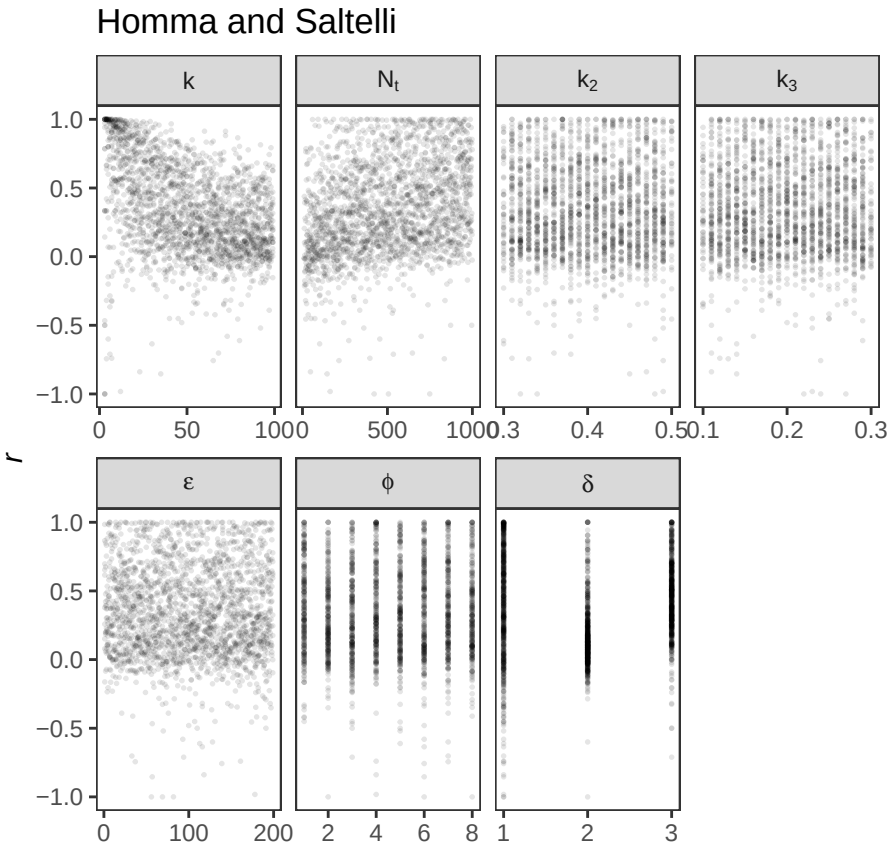



```
##
## $`Glen and Isaac`
```

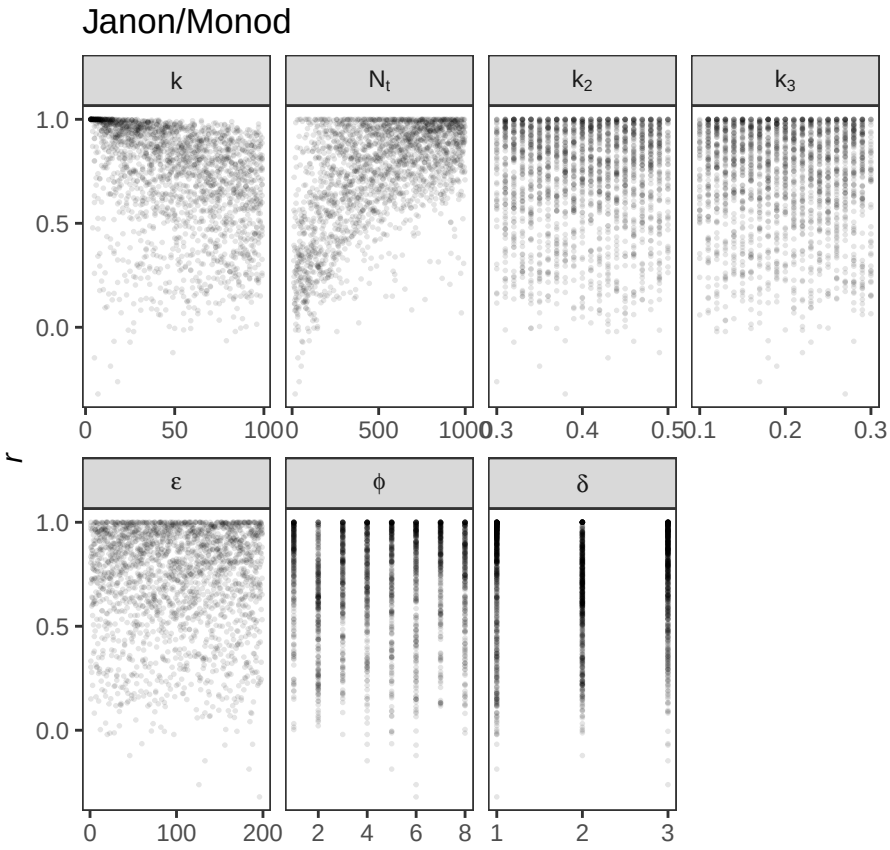
Glen and Isaac



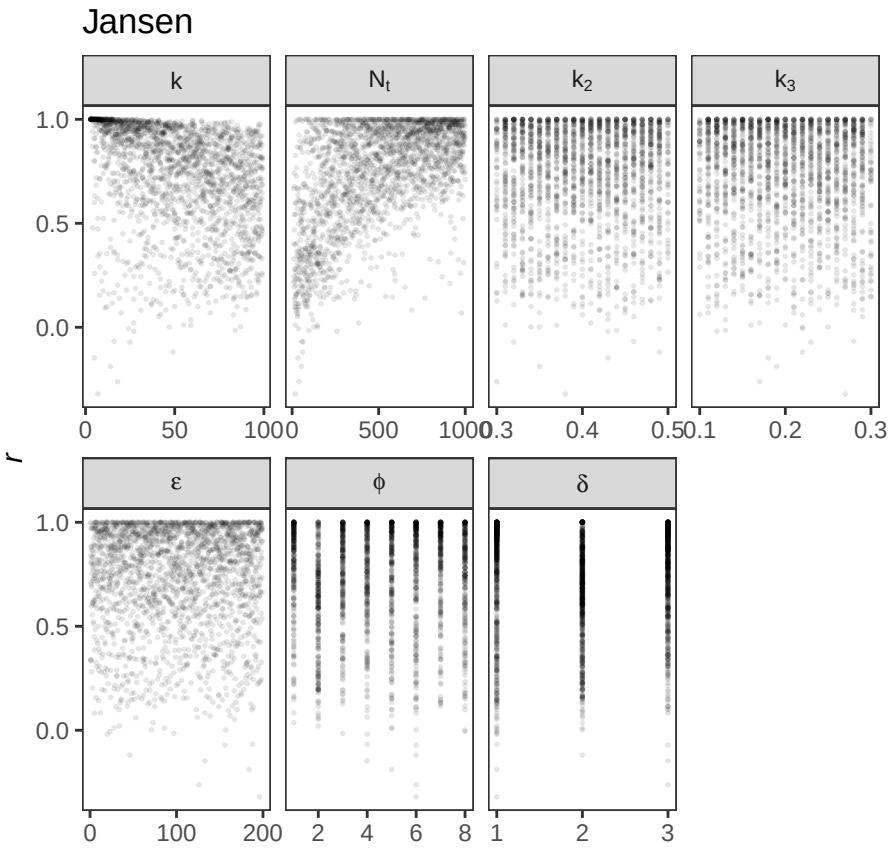

```
##
## $`Homma and Saltelli`
```


Homma and Saltelli
k
N_t
k_2
k_3
ε
ϕ
δ
r
1.0
0.5
0.0
−0.5
−1.0
0
50
100
500
1000
0.3
0.4
0.5
0.1
0.2
0.3
100
200
2
4
6
8
1
2
3

```
##
## $`Janon/Monod`
```

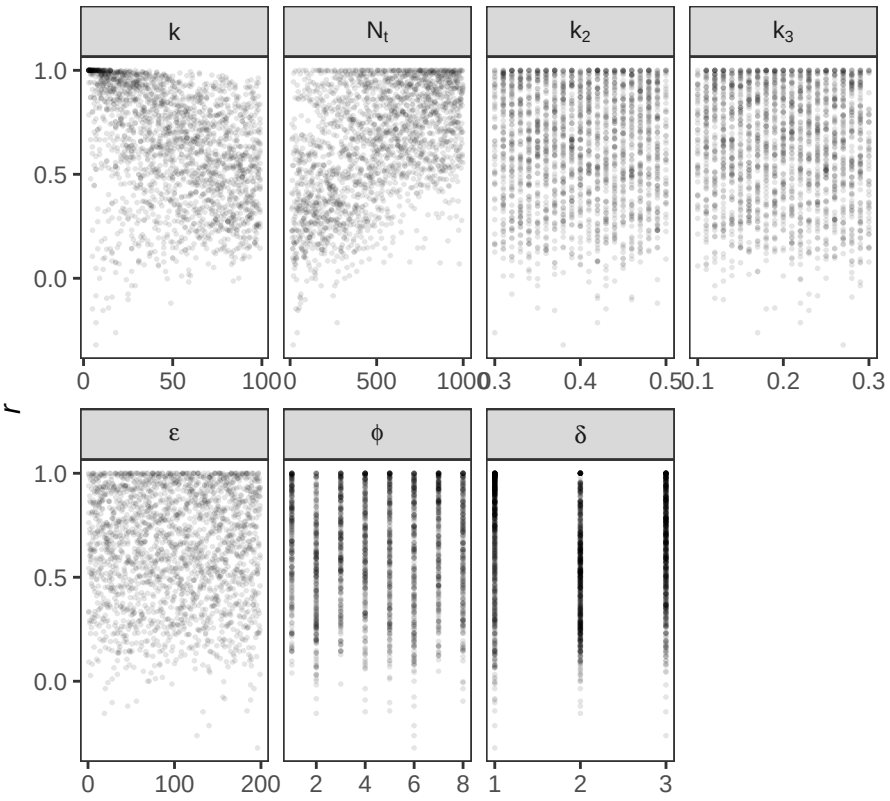

Janon/Monod
k
N_t
k_2
k_3
ε
ϕ
δ
r
1.0
0.5
0.0
0
50
100
500
1000
0.3
0.4
0.5
0.1
0.2
0.3
100
200
2
4
6
8
1
2
3

```
##
## $Jansen
```

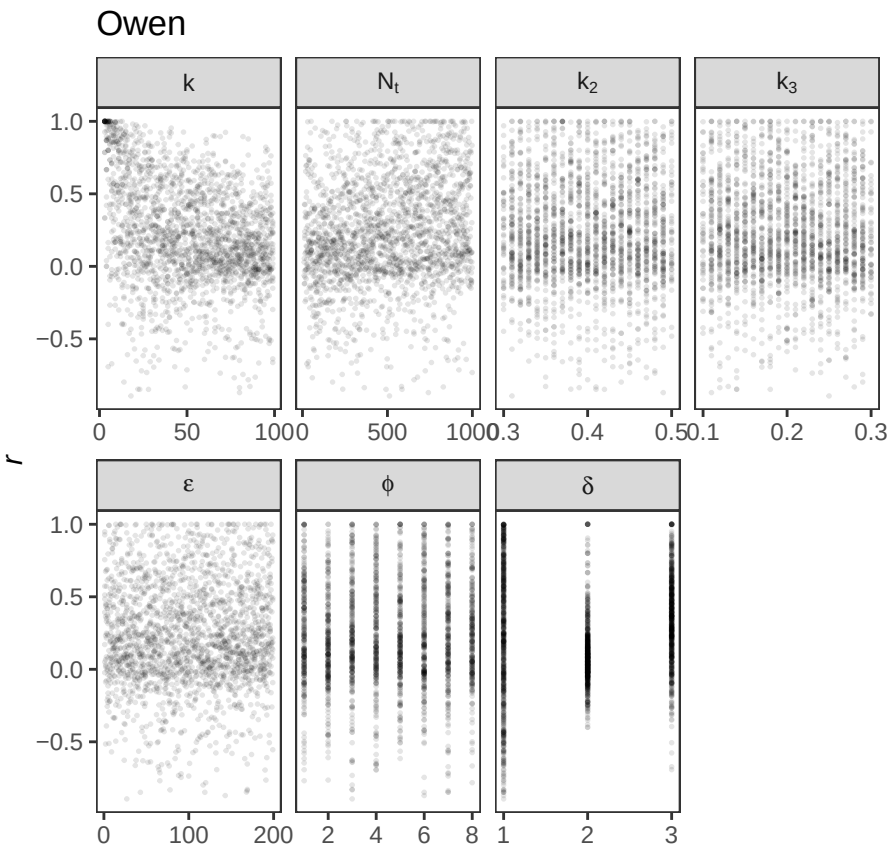

Jansen
k
N_t
k_2
k_3
ε
ϕ
δ
r
1.0
0.5
0.0
0
50
100
500
1000
0.3
0.4
0.5
0.1
0.2
0.3
100
200
2
4
6
8
1
2
3

```
##
## $Lamboni
```

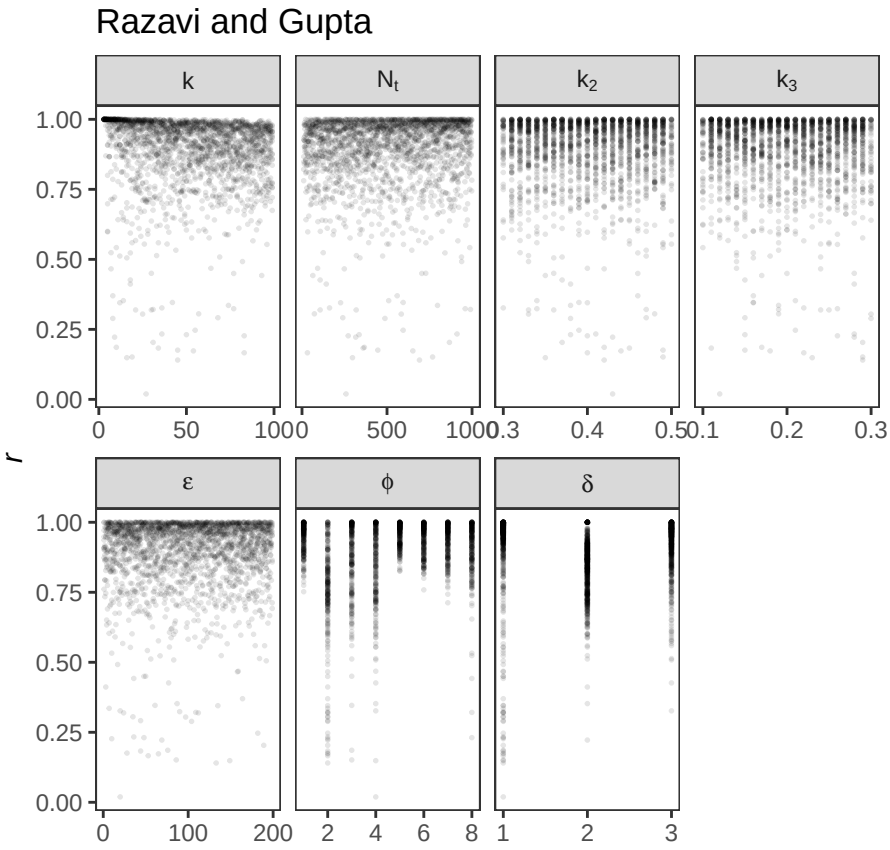
Lamboni



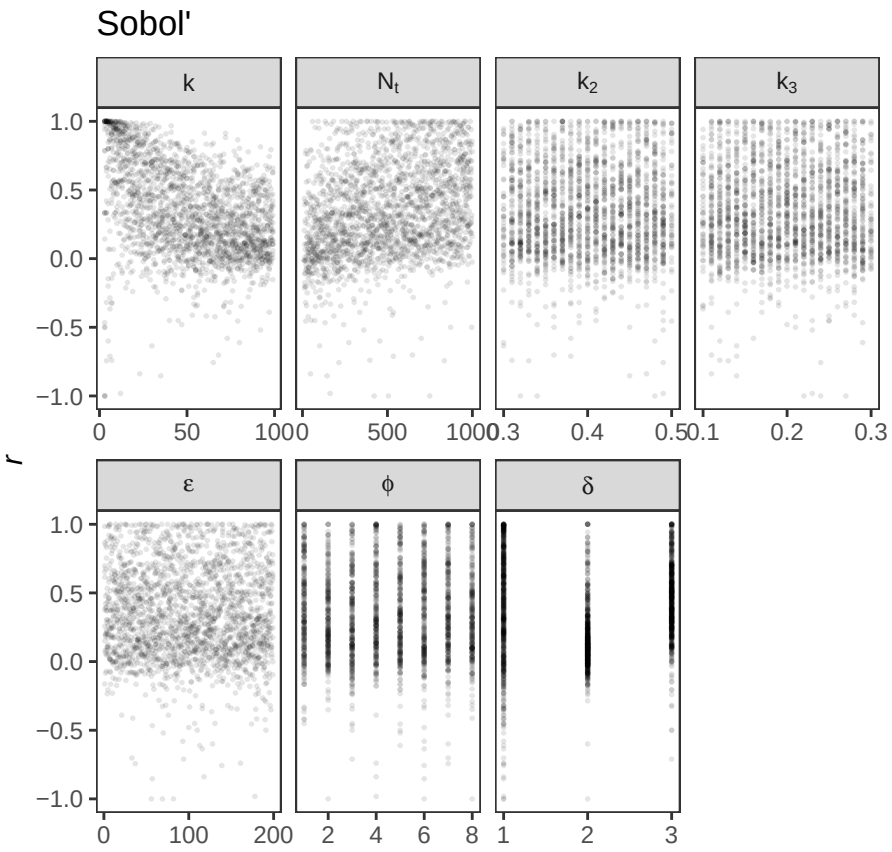

```
##
## $Owen
```




```
##
## $`Razavi and Gupta`
```


Razavi and Gupta
k
N_t
k_2
k_3
ε
ϕ
δ
r
1.00
0.75
0.50
0.25
0.00
0
50
100
500
0.3
0.4
0.5
0.1
0.2
0.3
100
200
2
4
6
8
1
2
3

```
##
## $`Sobol'`
```



5.2 Sobol' indices

```
# SENSITIVITY ANALYSIS ----------------------------------------------------------

params.plot <- c("$k$", "$N_t$", "$k_2$", "$k_3$", "$\\varepsilon$", "$\\phi$",
                 "$\\delta$")

# Show rows with NA
full_output[is.na(correlation), ]
```

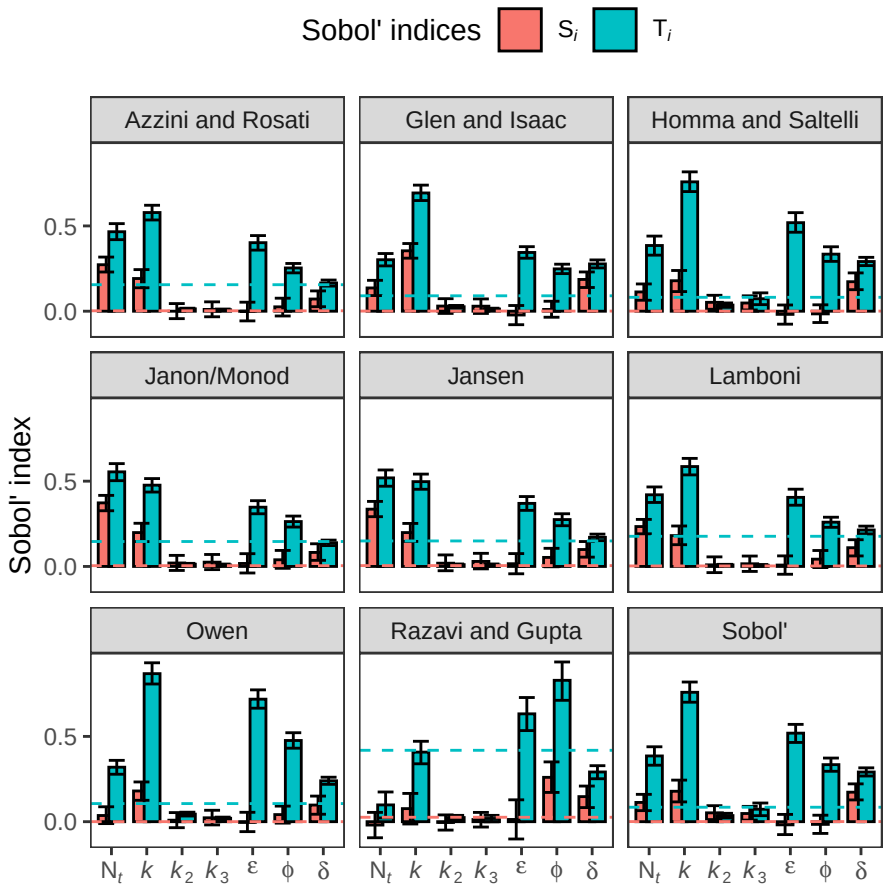
```
##     row k N_t  k_2  k_3 epsilon phi delta N.all N.azzini N.vars C.all C.azzini
## 1: 4164 9  33 0.46 0.29     167   2     1     4        2      2    40       40
## 2: 8032 7  27 0.46 0.22      37   3     3     4        2      2    32       32
## 3: 8032 7  27 0.46 0.22      37   3     3     4        2      2    32       32
##    C.vars estimator correlation Nt    ratio
## 1:     74      Owen          NA 40 4.444444
## 2:     58   Lamboni          NA 32 4.571429
## 3:     58      Owen          NA 32 4.571429
```

```
# Substitute NA by 0
full_output <- full_output[, correlation:= ifelse(is.na(correlation) == TRUE, 0, correlation)]

# Compute Sobol' indices
indices <- full_output[, sobol_indices(Y = correlation,
```

```
                                        N = N,
                                        params = params.plot,
                                        first = "jansen",
                                        R = R,
                                        boot = TRUE,
                                        order = order),
                        estimator]

# Compute Sobol' indices for dummy parameter
indicesD <- full_output[, sobol_dummy(Y = correlation,
                                      N = N,
                                      params = params.plot,
                                      boot = TRUE,
                                      R = R),
                         estimator]
```

```
# PLOT SOBOL' INDICES ----------------------------------------------------------

# Reorder the levels of the parameters
indices <- indices[, parameters:= factor(parameters,
                                         levels = c("$N_t$", "$k$", "$k_2$",
                                                    "$k_3$", "$\\varepsilon$",
                                                    "$\\phi$", "$\\delta$"))]

ggplot(indices, aes(parameters, original, fill = sensitivity)) +
  geom_bar(stat = "identity",
           position = position_dodge(0.6),
           color = "black") +
  geom_errorbar(aes(ymin = low.ci,
                    ymax = high.ci),
                position = position_dodge(0.6)) +
  geom_hline(data = indicesD,
             aes(yintercept = high.ci,
                 color = sensitivity),
             lty = 2,
             show.legend = FALSE) +
  scale_x_discrete(labels = c(expression(N[italic(t)]),
                              expression(italic(k)),
                              expression(italic(k[2])),
                              expression(italic(k[3])),
                              expression(epsilon),
                              expression(phi),
                              expression(delta))) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  facet_wrap(~estimator,
             ncol = 3) +
  labs(x = "",
```

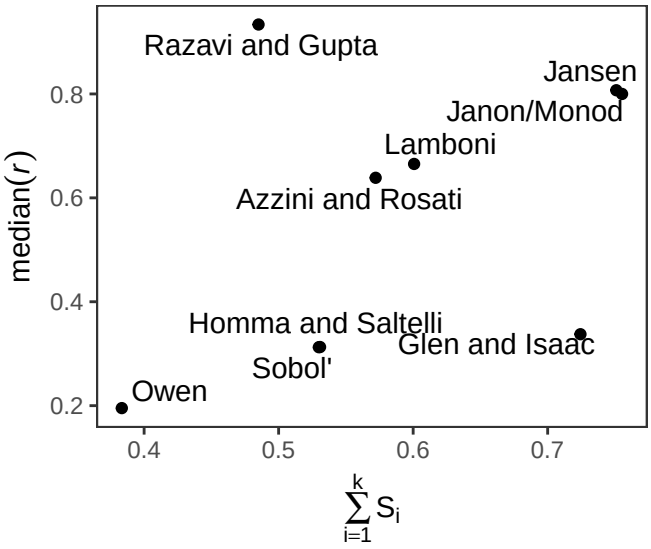
```
      y = "Sobol' index") +
  scale_fill_discrete(name = "Sobol' indices",
                      labels = c(expression(S[italic(i)]),
                                 expression(T[italic(i)]))) +
  theme_AP() +
  theme(legend.position = "top")
```




```
# SUM OF FIRST-ORDER INDICES ----------------------------------------------------

indices[sensitivity == "Si", sum(original), estimator]
```

```
##             estimator        V1
## 1:  Azzini and Rosati 0.5721296
## 2:     Glen and Isaac 0.7243739
## 3: Homma and Saltelli 0.5307930
## 4:             Jansen 0.7510287
## 5:            Lamboni 0.6007199
## 6:        Janon/Monod 0.7552958
## 7:               Owen 0.3834172
## 8:             Sobol' 0.5298899
## 9:   Razavi and Gupta 0.4850579
```

```r
# Plot
merge(indices[sensitivity == "Si", sum(original), estimator],
      dt_median, by = "estimator") %>%
  ggplot(., aes(V1, median)) +
  geom_point() +
  labs(x = expression(sum(S[i], i==1, k)),
       y = expression(median(italic(r)))) +
  geom_text_repel(aes(label = estimator)) +
  theme_AP()
```




```r
# PLOT SECOND-ORDER EFFECTS ------------------------------------------------------

indices[sensitivity == "Sij"] %>%
  .[low.ci > 0] %>%
  .[, parameters:= gsub(parameters,
                        pattern = ".",
                        replacement = "~",
                        fixed = TRUE)] %>%
  ggplot(., aes(parameters, original)) +
  geom_point() +
  geom_errorbar(aes(ymax = high.ci, ymin = low.ci)) +
  geom_hline(yintercept = 0,
             lty = 2,
             color = "red") +
  facet_grid(~estimator,
             scales = "free_x",
             space = "free_x") +
  scale_x_discrete(labels = ggplot2:::parse_safe) +
  labs(x = "",
       y = "Sobol' index") +
  theme_AP()
```

```r
# EXPORT SOBOL' INDICES ------------------------------------------------------

fwrite(indices, "indices.csv")
fwrite(dt_median, "dt_median.csv")
```

6 Session information

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

sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.4
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/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_0.4.8          ggrepel_0.8.2
##  [3] sensobol_0.3              logitnorm_0.8.37
##  [5] benchmarkme_1.0.3         cowplot_1.0.0
##  [7] scales_1.1.0              data.table_1.12.8
##  [9] Rfast_1.9.9               RcppZiggurat_0.1.5
## [11] doParallel_1.0.15         iterators_1.0.12
## [13] foreach_1.4.8             forcats_0.4.0
## [15] stringr_1.4.0             dplyr_0.8.3
## [17] purrr_0.3.3               readr_1.3.1
## [19] tidyr_1.0.0               tibble_2.1.3
## [21] ggplot2_3.3.0             tidyverse_1.3.0
## [23] RcppArmadillo_0.9.850.1.0 Rcpp_1.0.4
##
## loaded via a namespace (and not attached):
##  [1] lubridate_1.7.4       lattice_0.20-38       assertthat_0.2.1
##  [4] digest_0.6.25         R6_2.4.1              cellranger_1.1.0
##  [7] backports_1.1.5       reprex_0.3.0          evaluate_0.14
## [10] httr_1.4.1            pillar_1.4.3          Rdpack_0.11-1
## [13] rlang_0.4.5           curl_4.3              readxl_1.3.1
## [16] rstudioapi_0.11       Matrix_1.2-18         rmarkdown_2.1
## [19] munsell_0.5.0         broom_0.5.3           compiler_3.6.1
## [22] modelr_0.1.5          xfun_0.12             pkgconfig_2.0.3
## [25] htmltools_0.4.0       tidyselect_0.2.5      codetools_0.2-16
## [28] fansi_0.4.1           crayon_1.3.4          dbplyr_1.4.2
## [31] withr_2.1.2           grid_3.6.1            nlme_3.1-143
## [34] jsonlite_1.6          gtable_0.3.0          lifecycle_0.2.0
```

```
## [37] DBI_1.1.0             magrittr_1.5          bibtex_0.4.2.2
## [40] cli_2.0.2             stringi_1.4.6         fs_1.3.1
## [43] remotes_2.1.1         benchmarkmeData_1.0.3 xml2_1.2.2
## [46] generics_0.0.2        vctrs_0.2.4           tools_3.6.1
## [49] glue_1.3.2            hms_0.5.3             yaml_2.2.0
## [52] colorspace_1.4-1      gbRd_0.4-11           rvest_0.3.5
## [55] knitr_1.27            haven_2.2.0
```

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

```
## Machine:
```

```
## [1] "Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz"
```

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

```
## Num cores:
```

```
## [1] 8
```

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

```
## Num threads:
```

```
## [1] 16
```

```
## Return the machine RAM
cat("RAM:         "); print (get_ram()); cat("\n")
```

```
## RAM:
```

```
## 34.4 GB
```

References

Becker, William. 2019. “Sensitivity analysis on a shoestring : screening model inputs at low sample size.”