The topology of software risk in scientific models

2. The uncertainty and sensitivity analysis

Arnald Puy

Contents

1 Preliminary

```r
# PRELIMINARY FUNCTIONS #######################################################
################################################################################

sensobol::load_packages(c("data.table", "tidyverse", "openxlsx", "scales",
                          "cowplot", "readxl", "ggrepel", "tidytext", "here",
                          "tidygraph", "igraph", "foreach", "parallel", "ggraph",
                          "tools", "purrr", "sensobol", "benchmarkme"))

# 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),
          strip.background = element_rect(fill = "white"),
          legend.text = element_text(size = 7.3),
          axis.title = element_text(size = 10),
          legend.key.width = unit(0.4, "cm"),
          legend.key.height = unit(0.4, "cm"),
          legend.key.spacing.y = unit(0, "lines"),
          legend.box.spacing = unit(0, "pt"),
          legend.title = element_text(size = 7.3),
          axis.text.x = element_text(size = 7),
          axis.text.y = element_text(size = 7),
          axis.title.x = element_text(size = 7.3),
          axis.title.y = element_text(size = 7.3),
          plot.title = element_text(size = 8),
          strip.text.x = element_text(size = 7.4),
          strip.text.y = element_text(size = 7.4))
}

# Source all .R files in the "functions" folder --------------------------------

r_functions <- list.files(path = here("functions"),
                          pattern = "\\.R$", full.names = TRUE)

lapply(r_functions, source)

# Set seed ---------------------------------------------------------------------

seed <- 123

# Define labels for better plotting --------------------------------------------
```

```r
lab_expr <- c(b1 = expression(C %in% "(" * 0 * ", 10" * "]"),
              b2 = expression(C %in% "(" * 10 * ", 20" * "]"),
              b3 = expression(C %in% "(" * 20 * ", 50" * "]"),
              b4 = expression(C %in% "(" * 50 * ", " * infinity * ")"))
```

2 The uncertainty and sensitivity analysis

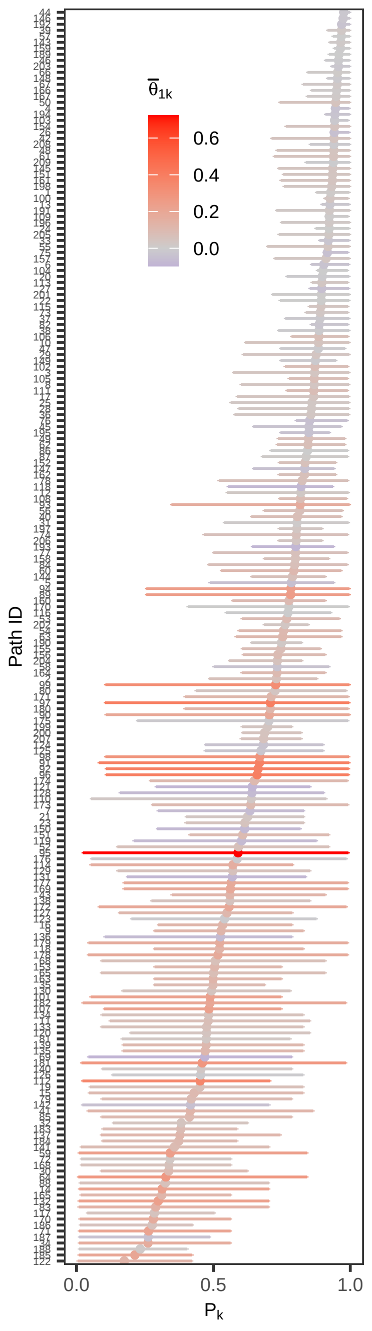
```r
# UNCERTAINTY AND SENSITIVITY ANALYSIS ##########################################
#################################################################################

# Load the nodes_df and the paths_tbl from the synthetic example ---------------

objs <- readRDS("graph_objects.rds")
nodes_df  <- objs$nodes_df
paths_tbl <- objs$paths_tbl

# Define settings ---------------------------------------------------------------

N <- 10^4
order <- "second"

# Run an UA/SA ------------------------------------------------------------------

output_ua.sa <- full_ua_sa_risk_fun(node_df = nodes_df,
                                    paths_tbl = paths_tbl,
                                    N = N, order = order)
```

3 Figures

```r
# PLOT ERRORBARS ################################################################

# Plot risk slope ---------------------------------------------------------------

a <- output_ua.sa$paths %>%
  ggplot(., aes(P_k_mean, reorder(path_id, P_k_mean), color = risk_slope))  +
  geom_point(size = 1) +
  geom_errorbar(aes(xmin = P_k_min, xmax = P_k_max), height = 0.2) +
  scale_color_gradient2(low = "blue", mid = "grey80", high = "red", midpoint = 0,
                        name = expression(bar(theta)[1*k])) +
  labs(y = "Path ID", x = expression(P[k])) +
  theme_AP() +
  scale_x_continuous(breaks = breaks_pretty(n = 3)) +
  theme(axis.text.y = element_text(size = 4),
        legend.position = c(0.4, 0.87))
```

```
a
```

```
## `height` was translated to `width`.
```

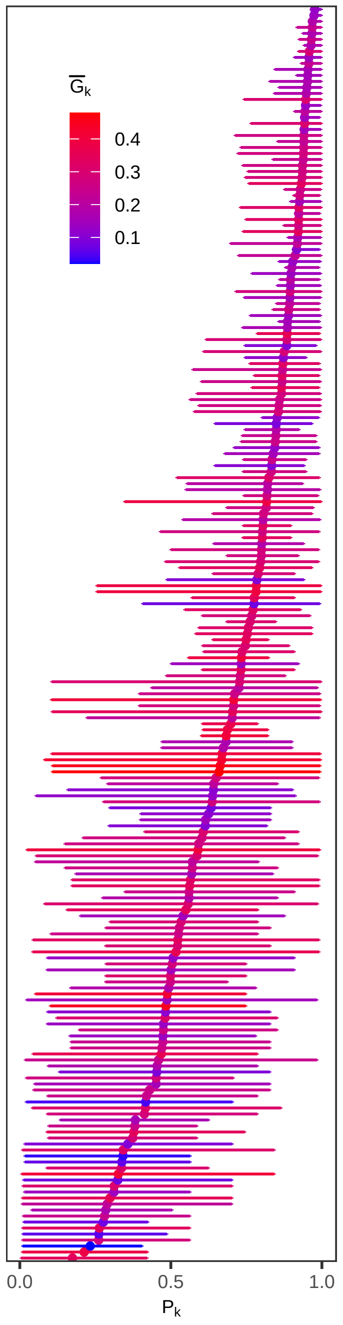


```
# Plot Gini index ---------------------------------------------------------------

b <- output_ua.sa$paths %>%
  ggplot(., aes(P_k_mean, reorder(path_id, P_k_mean), color = gini_node_risk))  +
  geom_point(size = 1) +
  geom_errorbarh(aes(xmin = P_k_min, xmax = P_k_max), height = 0.2) +
  scale_color_gradient(low = "blue",high = "red", name = expression(bar(G)[k])) +
  labs(y = "", x = expression(P[k])) +
  theme_AP() +
  scale_x_continuous(breaks = breaks_pretty(n = 3)) +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = c(0.3, 0.87))
```

```
## Warning: `geom_errorbarh()` was deprecated in ggplot2 4.0.0.
## i Please use the `orientation` argument of `geom_errorbar()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

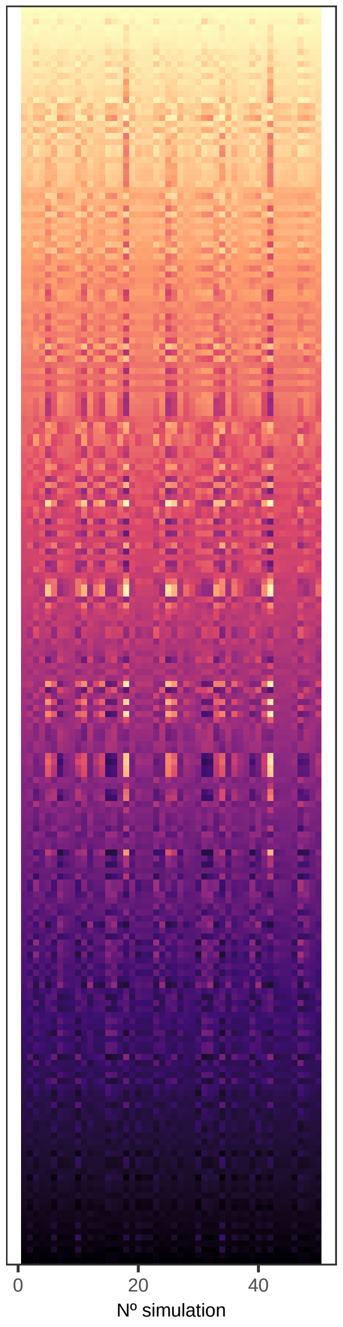
```
b
```

```
## `height` was translated to `width`.
```



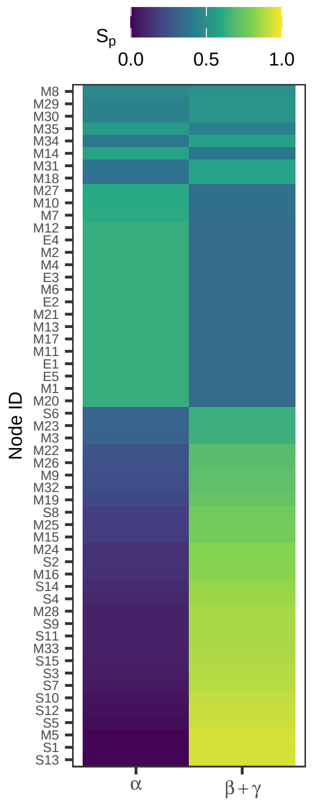
```
# SUPER TILE PLOT ###################################################################

paths_long <- output_ua.sa$paths[, .(sample_id = seq_along(P_k_vec[[1]]),
                                      P_k = unlist(P_k_vec)), path_id]

# Rank paths within each monte carlo run (1 = highest risk) -----------------

paths_long[, rank:= frank(-P_k, ties.method = "average"), sample_id]

# Sort based on previous plots ---------------------------------------------------
```

```r
paths_long <- paths_long[, path_id:= factor(path_id, levels = output_ua.sa$paths$path_id)] %>%
  .[sample_id %in% 1:50] # Use only the first 50 samples rather than the 2N

# Plot: ranking instability of paths under weight uncertainty ------------------

plot_supertile <- ggplot(paths_long, aes(sample_id, factor(path_id), fill = rank)) +
  geom_tile() +
  scale_fill_viridis_c(option = "magma", direction = -1, name = "Rank") +
  labs(x = "Nº simulation", y = "") +
  theme_AP() +
  scale_x_continuous(breaks = breaks_pretty(n = 3)) +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = "none")

plot_supertile
```

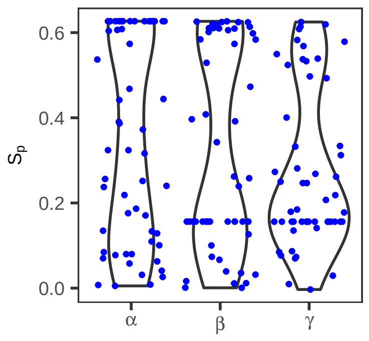


```
# PLOT FIRST ORDER OF ALPHA AND BETA + GAMMA ###################################

# Unnest results -----------------------------------------------------------------

sa_dt <- output_ua.sa$nodes %>%
  unnest(sensitivity_indices) %>%
  data.table() %>%
  .[, .(name, original, parameters, sensitivity)]

# First-order Si -----------------------------------------------------------------
```

```r
si_dt <- sa_dt[sensitivity == "Si"]
si_wide <- dcast(si_dt,name ~ parameters, value.var = "original")
setnames(si_wide, c("a_raw", "b_raw", "c_raw"), c("S_a",  "S_b",  "S_c"))

# Get S_bc (interaction between b_raw and c_raw) ------------------------------

sbc_dt <- sa_dt[sensitivity == "Sij" & parameters == "b_raw.c_raw",
                .(S_bc = original), name]

group_dt <- merge(si_wide, sbc_dt, by = "name", all.x = TRUE)
group_dt[is.na(S_bc), S_bc:= 0]

# Group effect of (b,c) --------------------------------------------------------

group_dt[, S_group_bc := S_b + S_c + S_bc]

# Long format for tile plot ----------------------------------------------------

tile_dt <- melt(group_dt, id.vars = "name", measure.vars = c("S_a", "S_group_bc"),
                variable.name = "factor", value.name   = "Si")

# Change the labels ------------------------------------------------------------

tile_dt[factor == "S_a", factor:= "alpha"]
tile_dt[factor == "S_group_bc", factor:= "beta_gamma"]

# Structure the heatmap --------------------------------------------------------

tile_dt[, name:= factor(name, levels = unique(name[order(-Si)]))]

# Tile plot --------------------------------------------------------------------

plot.sa <- ggplot(tile_dt, aes(x = factor, y = name, fill = Si)) +
  geom_tile() +
  scale_fill_viridis_c(name = expression(S[p]), limits = c(0, 1),
                       breaks = c(0, 0.5, 1)) +
  scale_x_discrete(labels = c("alpha" = expression(alpha),
                              "beta_gamma" = expression(beta + gamma))) +
  labs(x = "", y = "Node ID") +
  theme_AP() +
  theme(axis.text.y = element_text(size = 5),
        legend.position = "top")

plot.sa
```



```
# Violin plot -------------------------------------------------------------------

plot_violin <- si_dt %>%
  mutate(param = fct_recode(parameters,
                            alpha = "a_raw",
                            beta  = "b_raw",
                            gamma = "c_raw")) %>%
  ggplot(., aes(x = param, y = original)) +
  geom_violin() +
  geom_jitter(size = 0.5, color = "blue") +
  scale_x_discrete(labels = c("alpha" = expression(alpha),
                              "beta" = expression(beta),
                              "gamma" = expression(gamma))) +
  labs(x = "", y = expression(S[p])) +
  theme_AP()

plot_violin
```



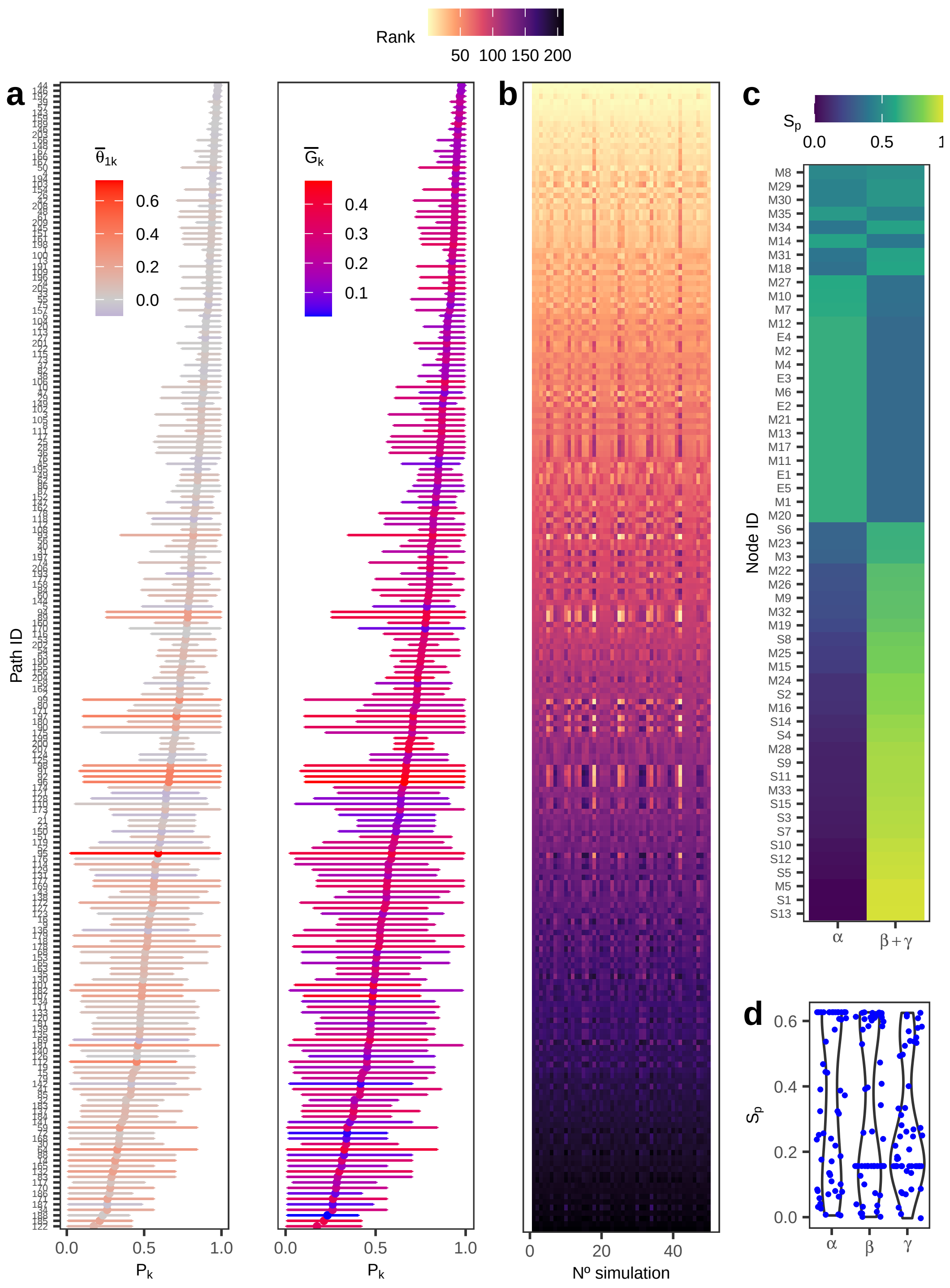
```r
# MERGE AND PLOT ###############################################################

left.plot <- plot_grid(a, b, plot_supertile, ncol = 3, labels = c("a", "", "b"))
```

```
## `height` was translated to `width`.
## `height` was translated to `width`.
```

```r
right.plot <- plot_grid(plot.sa, plot_violin, ncol = 1, rel_heights = c(0.75, 0.25),
                        labels = c("c", "d"))
all_plots <- plot_grid(left.plot, right.plot, rel_widths = c(0.7, 0.2))

legend <- get_legend_fun(plot_supertile + theme(legend.position = "top"))
```

```
## Warning: `is.ggplot()` was deprecated in ggplot2 3.5.2.
## i Please use `is_ggplot()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
plot_grid(legend, all_plots, ncol = 1, rel_heights = c(0.05, 0.95))
```


a
b
c
d
Rank
50 100 150 200
$\overline{\theta}_{1k}$
0.6
0.4
0.2
0.0
$\overline{G}_k$
0.4
0.3
0.2
0.1
Path ID
P_k
0.0
0.5
1.0
Nº simulation
0
20
40
S_p
0.0
0.5
1
Node ID
M8
M29
M30
M35
M34
M14
M31
M18
M27
M10
M7
M12
E4
M2
M4
E3
M6
E2
M21
M13
M17
M11
E1
E5
M1
M20
S6
M23
M3
M22
M26
M9
M32
M19
S8
M25
M15
M24
S2
M16
S14
S4
M28
S9
S11
M33
S15
S3
S7
S10
S12
S5
M5
S1
S13
α
β+γ
0.6
0.4
0.2
0.0
α
β
γ