Sensitivity auditing of global irrigation water withdrawal estimates

R code

Arnald Puy

Contents

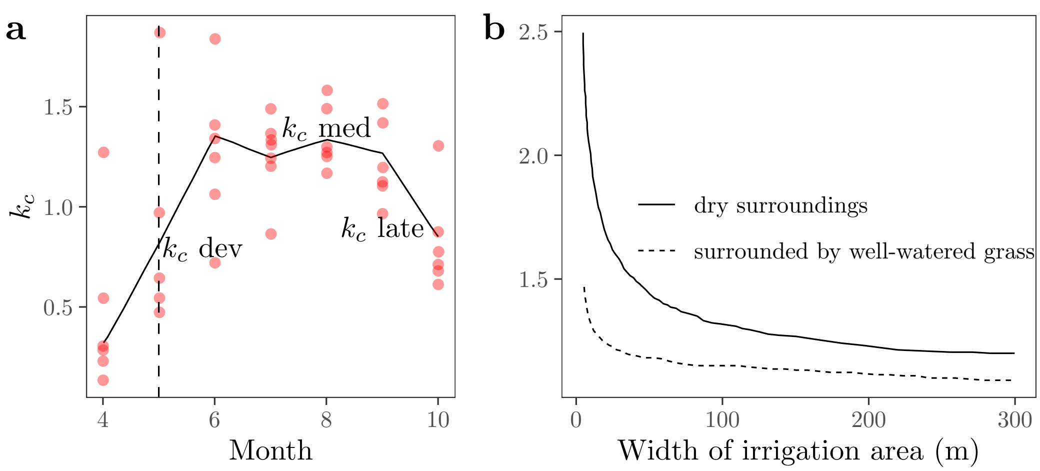
```r
# PRELIMINARY STEPS --------------------------------------------------------------

# Install and load 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(
  "data.table", "tidyverse", "sensobol", "wesanderson",
  "triangle", "scales", "cowplot", "fitdistrplus",
  "parallel", "doParallel", "foreach", "sp", "sf", "Rfast",
  "raster", "rworldmap", "countrycode"))

# 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
      ),
      legend.position = "top",
      strip.background = element_rect(fill = "white")
    )
}
```

1 ET__0 and ET__c

2 k__c values for salt cedar

```r
# K_C VALUES FOR SALT CEDAR --------------------------------------------------------

# Read in dataset
kc_evolution_cedar <- fread("kc_evolution_cedar.csv")
kc_evolution_point <- fread("kc_evolution_point.csv")
```

```r
# Retrieve minimum and maximmum values for month 5
cedar.min.max <- kc_evolution_point[x > 5 & x < 6]

# Plot
a <- ggplot(kc_evolution_cedar, aes(x = x, y = y)) +
  geom_line() +
  geom_point(
    data = kc_evolution_point, aes(x = x, y = y),
    color = "red", alpha = 0.4
  ) +
  scale_color_discrete(name = "") +
  annotate("text", x = 5.78, y = 0.8, label = "$k_c$ dev") +
  annotate("text", x = 8, y = 1.4, label = "$k_c$ med") +
  annotate("text", x = 9, y = 0.9, label = "$k_c$ late") +
  geom_vline(xintercept = 5, lty = 2) +
  theme_AP() +
  labs(x = "Month", y = "$k_c$")

# Read in data to show oasis effect
oasis <- fread("oasis_effect.csv")

# Plot and merge
b <- ggplot(oasis, aes(x = x, y = y, group = name, linetype = name)) +
  geom_line() +
  scale_linetype_discrete(name = "") +
  theme_AP() +
  labs(x = "Width of irrigation area (m)", y = "") +
  theme(legend.position = c(0.6, 0.5))

cowplot::plot_grid(a, b, ncol = 2, labels = "auto", rel_widths = c(0.45, 0.55))
```




```r
# Define settings
# ------------------------------------------------
```

```r
N <- 2^12
R <- 10^3

# Define ET_0 Equations
# ------------------------------------------------

# Priestley and Taylor
pt_fun <- function(alpha, delta, gamma, A, k_c = 1) {
  out <- k_c * (alpha * ((delta * A) / (gamma + delta)))
  return(out)
}

# Penman Monteith
pm_fun <- function(delta, A, gamma, t, w, v, k_c = 1) {
  out <- k_c * ((0.408 * delta * A + gamma * (900 / (t + 273)) * w * v) /
    delta + gamma * (1 + 0.34 * w))
  return(out)
}

# Define sampling matrices
# ------------------------------------------------

# Priestley and Taylor

params <- c("alpha", "delta", "gamma", "A")
mat <- sobol_matrices(N = N, params = params)
mat[, "alpha"] <- qunif(mat[, "alpha"], 1.26 + 1.26 * -0.1, 1.26 + 1.26 * 0.1)
mat[, "delta"] <- qunif(mat[, "delta"], 0.21 + 0.21 * -0.005, 0.21 + 0.21 * 0.005)
mat[, "gamma"] <- qunif(mat[, "gamma"], 0.059 + 0.059 * -0.001, 0.059 + 0.059 * 0.001)
mat[, "A"] <- qunif(mat[, "A"], 350 + 350 * -0.15, 350 + 350 * 0.15)

# Run model
y.pt <- pt_fun(
  alpha = mat[, "alpha"],
  delta = mat[, "delta"],
  gamma = mat[, "gamma"],
  A = mat[, "A"]
)

# Compute Sobol' indices
params.pt <- c("$\\alpha$", "$\\Delta$", "$\\gamma$", "$A$")
ind.pt <- sobol_indices(
  Y = y.pt, N = N, params = params.pt,
  first = "jansen", R = R, boot = TRUE
)[
  , method := "PT"
]
```

```r
# Penman Monteith
params <- c("delta", "gamma", "A", "t", "w", "v")
# Define matrix
mat <- sobol_matrices(N = N, params = params)
mat[, "delta"] <- qunif(mat[, "delta"], 0.21 + 0.21 * -0.005, 0.21 + 0.21 * 0.005)
mat[, "gamma"] <- qunif(mat[, "gamma"], 0.059 + 0.059 * -0.001, 0.059 + 0.059 * 0.001)
mat[, "A"] <- qunif(mat[, "A"], 350 + 350 * -0.15, 350 + 350 * 0.15)
mat[, "t"] <- qunif(mat[, "t"], 27 + 27 * -0.01, 27 + 27 * 0.01)
mat[, "w"] <- qunif(mat[, "w"], 2.5 + 2.5 * -0.05, 2.5 + 2.5 * 0.05)
mat[, "v"] <- qunif(mat[, "v"], 2.49 + 2.49 * -0.04, 2.49 + 2.49 * 0.04)

# Run model
y.pm <- pm_fun(
  delta = mat[, "delta"],
  A = mat[, "A"],
  gamma = mat[, "gamma"],
  t = mat[, "t"],
  w = mat[, "w"],
  v = mat[, "v"]
)

# Compute Sobol' indices
params.pm <- c("$\\Delta$", "$\\gamma$", "$A$", "$t$", "$w$", "$v$")
ind.pm <- sobol_indices(
  Y = y.pm, N = N, params = params.pm,
  first = "jansen", R = R, boot = TRUE
)[
  , method := "PM"
]

# Plot results
# ----------------------------------------------------

dt.pt <- data.table(y.pt)[, method := "PT"] %>%
  setnames(., "y.pt", "value")
dt.pm <- data.table(y.pm)[, method := "PM"] %>%
  setnames(., "y.pm", "value")

# Uncertainty
a1 <- rbind(dt.pt, dt.pm) %>%
  ggplot(., aes(value, fill = method)) +
  geom_histogram(alpha = 0.3, color = "black") +
  scale_fill_manual(
    name = "Method",
    values = wes_palette(n = 2, name = "Chevalier1")
  ) +
```

```
  labs(x = "$ET_0$ (mm d$^{-1}$)", y = "Counts") +
  theme_AP() +
  theme(legend.position = "none")

# Sobol' indices
b1 <- rbind(ind.pt, ind.pm) %>%
  .[, parameters := factor(parameters, levels = c(
    "$\\Delta$", "$\\gamma$",
    "$A$", "$t$", "$w$", "$v$", "$\\alpha$"
  ))] %>%
  plot_sobol(.) +
  theme(legend.position = "none") +
  facet_grid(~method,
    scales = "free_x",
    space = "free_x"
  )

top <- plot_grid(a1, b1, ncol = 2, labels = "auto")
```

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

```
# ASSESS ET_C
# ------------------------------------------------------------

# Priestley Taylor
params <- c("alpha", "delta", "gamma", "A", "k_c")

# Define matrix
mat <- sobol_matrices(N = N, params = params)
mat[, "alpha"] <- qunif(mat[, "alpha"], 1.26 + 1.26 * -0.1, 1.26 + 1.26 * 0.1)
mat[, "delta"] <- qunif(mat[, "delta"], 0.21 + 0.21 * -0.005, 0.21 + 0.21 * 0.005)
mat[, "gamma"] <- qunif(mat[, "gamma"], 0.059 + 0.059 * -0.001, 0.059 + 0.059 * 0.001)
mat[, "A"] <- qunif(mat[, "A"], 350 + 350 * -0.15, 350 + 350 * 0.15)
mat[, "k_c"] <- qunif(mat[, "k_c"], min(cedar.min.max$y), max(cedar.min.max$y))


# Run model
y.pt <- pt_fun(
  alpha = mat[, "alpha"],
  delta = mat[, "delta"],
  gamma = mat[, "gamma"],
  A = mat[, "A"],
  k_c = mat[, "k_c"]
)

# Compute Sobol' indices
params.pt <- c("$\\alpha$", "$\\Delta$", "$\\gamma$", "$A$", "$k_c$")
ind.pt <- sobol_indices(
```

```r
  Y = y.pt, N = N, params = params.pt,
  first = "jansen", R = R, boot = TRUE
)[
  , method := "PT"
]

# Penman Monteith
params <- c("delta", "gamma", "A", "t", "w", "v", "k_c")

# Define matrix
mat <- sobol_matrices(N = N, params = params)
mat[, "delta"] <- qunif(mat[, "delta"], 0.21 + 0.21 * -0.005, 0.21 + 0.21 * 0.005)
mat[, "gamma"] <- qunif(mat[, "gamma"], 0.059 + 0.059 * -0.001, 0.059 + 0.059 * 0.001)
mat[, "A"] <- qunif(mat[, "A"], 350 + 350 * -0.15, 350 + 350 * 0.15)
mat[, "t"] <- qunif(mat[, "t"], 27 + 27 * -0.01, 27 + 27 * 0.01)
mat[, "w"] <- qunif(mat[, "w"], 2.5 + 2.5 * -0.05, 2.5 + 2.5 * 0.05)
mat[, "v"] <- qunif(mat[, "v"], 2.49 + 2.49 * -0.04, 2.49 + 2.49 * 0.04)
mat[, "k_c"] <- qunif(mat[, "k_c"], min(cedar.min.max$y), max(cedar.min.max$y))

# Run model
y.pm <- pm_fun(
  delta = mat[, "delta"],
  A = mat[, "A"],
  gamma = mat[, "gamma"],
  t = mat[, "t"],
  w = mat[, "w"],
  v = mat[, "v"],
  k_c = mat[, "k_c"]
)

# Compute Sobol' indices
params.pm <- c("$\\Delta$", "$\\gamma$", "$A$", "$t$", "$w$", "$v$", "$k_c$")
ind.pm <- sobol_indices(
  Y = y.pm, N = N, params = params.pm,
  first = "jansen", R = R, boot = TRUE
)[
  , method := "PM"
]

# Plot and merge
# ------------------------------------------------------

dt.pt <- data.table(y.pt)[, method := "PT"] %>%
  setnames(., "y.pt", "value")
dt.pm <- data.table(y.pm)[, method := "PM"] %>%
  setnames(., "y.pm", "value")
```

```r
# Uncertainty
c <- rbind(dt.pt, dt.pm) %>%
  ggplot(., aes(value, fill = method)) +
  geom_histogram(alpha = 0.3, color = "black", position = "identity") +
  scale_fill_manual(
    name = "Method",
    values = wes_palette(n = 2, name = "Chevalier1")
  ) +
  labs(x = "$ET_c$ (mm d$^{-1}$)", y = "Counts") +
  theme_AP() +
  theme(legend.position = "none")

# Sobol' indices
d <- rbind(ind.pt, ind.pm) %>%
  .[, parameters := factor(parameters, levels = c(
    "$\\Delta$", "$\\gamma$",
    "$A$", "$t$", "$w$", "$v$", "$\\alpha$", "$k_c$"
  ))] %>%
  plot_sobol(.) +
  theme(legend.position = "none") +
  facet_grid(~method,
    scales = "free_x",
    space = "free_x"
  )
```

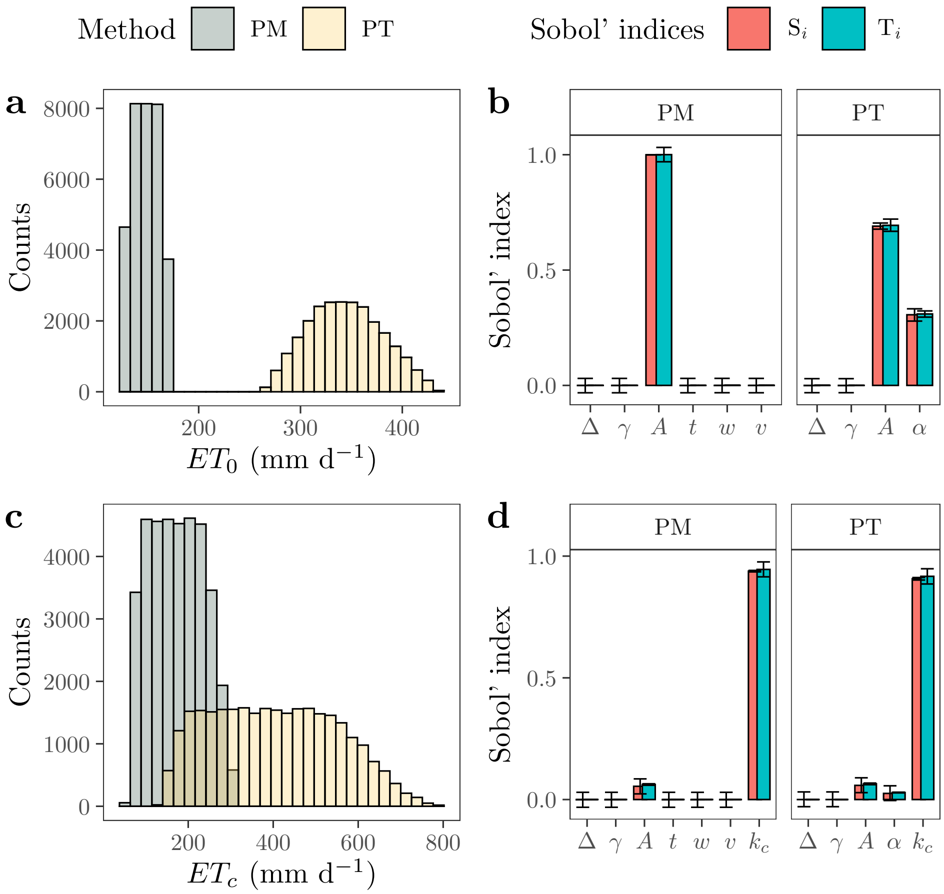
```r
bottom <- plot_grid(c, d, ncol = 2, labels = c("c", "d"))
```

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

```r
all_plot <- plot_grid(top, bottom, ncol = 1)
legend_ua <- cowplot::get_legend(a1 + theme(legend.position = "top"))
```

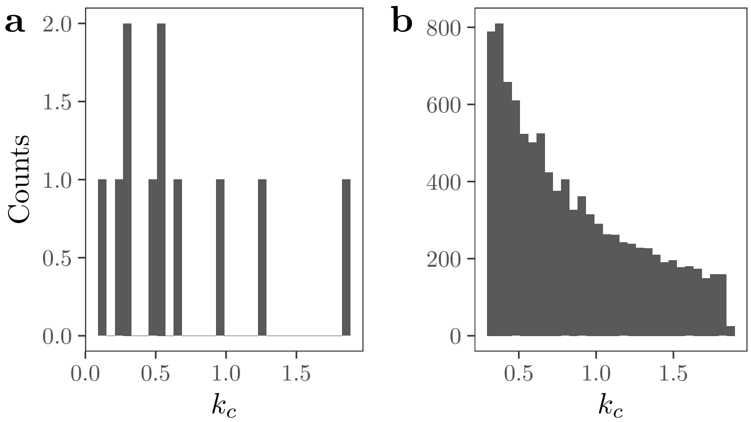
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```r
legend_sa <- cowplot::get_legend(b1 + theme(legend.position = "top"))
all_legend <- plot_grid(legend_ua, legend_sa, ncol = 2, rel_heights = c(0.01, 0.01))
plot_grid(all_legend, all_plot, ncol = 1, rel_heights = c(0.1, 0.9))
```




```r
#
a <- kc_evolution_point[x < 6] %>%
  ggplot(., aes(y)) +
  geom_histogram() +
  labs(x = "$k_c$", y = "Counts") +
  theme_AP()

v <- EnvStats::rlnormTrunc(10^4, meanlog = 0.1, sdlog = 15, min = 0.3, max = 1.85)
b <- ggplot(data.table(v), aes(v)) +
  geom_histogram() +
  labs(x = "$k_c$", y = "") +
  theme_AP()

cowplot::plot_grid(a, b, ncol = 2, labels = "auto")
```

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




```
# UNCERTAINTY IN E_C (K_C * ET_0)

pt_fun <- function(alpha, delta, gamma, A, k_c) {
  out <- k_c * (alpha * ((delta * A) / (gamma + delta)))
  return(out)
}

# Define settings
N <- 2^12
params <- c("alpha", "delta", "gamma", "A", "k_c")
R <- 10^3

# Define matrix
mat <- sobol_matrices(N = N, params = params)
mat[, "alpha"] <- qunif(mat[, "alpha"], 1.26 + 1.26 * -0.1, 1.26 + 1.26 * 0.1)
mat[, "delta"] <- qunif(mat[, "delta"], 0.21 + 0.21 * -0.005, 0.21 + 0.21 * 0.005)
mat[, "gamma"] <- qunif(mat[, "gamma"], 0.059 + 0.059 * -0.001, 0.059 + 0.059 * 0.001)
mat[, "A"] <- qunif(mat[, "A"], 350 + 350 * -0.15, 350 + 350 * 0.15)
mat[, "k_c"] <- EnvStats::qlnormTrunc(mat[, "k_c"],
  meanlog = 0.1, sdlog = 15, min = 0.3, max = 1.85
)


# Run model
y.pt <- pt_fun(
  alpha = mat[, "alpha"],
  delta = mat[, "delta"],
  gamma = mat[, "gamma"],
  A = mat[, "A"],
  k_c = mat[, "k_c"]
)

# Compute Sobol' indices
params.pt <- c("$\\alpha$", "$\\Delta$", "$\\gamma$", "$A$", "$k_c$")
ind.pt <- sobol_indices(
```

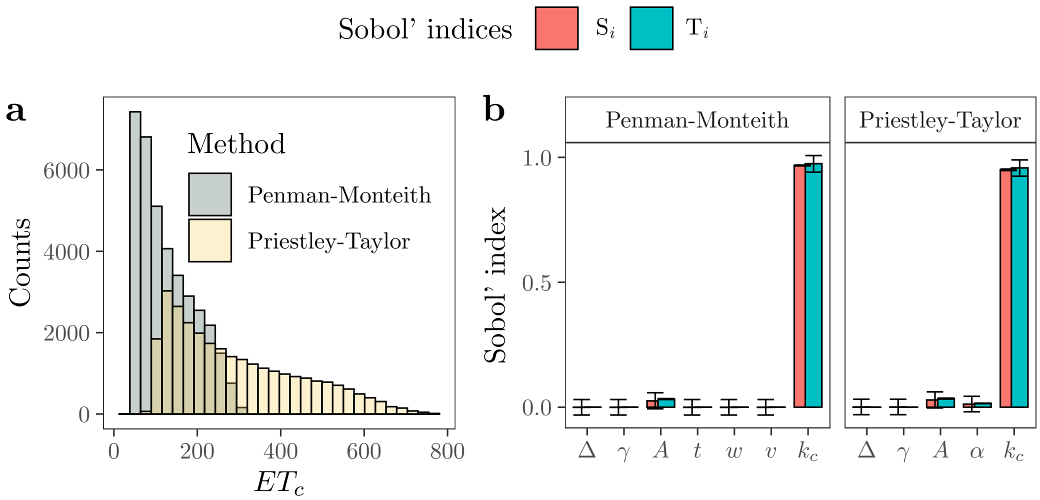
```
  Y = y.pt, N = N, params = params.pt,
  first = "jansen", R = R, boot = TRUE
)[
  , method := "Priestley-Taylor"
]


# UA / SA OF THE FAO 56 PENMAN-MONTEITH
################################################

pm_fun <- function(delta, A, gamma, t, w, v, k_c) {
  out <- k_c * ((0.408 * delta * A + gamma * (900 / (t + 273)) * w * v) /
    delta + gamma * (1 + 0.34 * w))
  return(out)
}

params <- c("delta", "gamma", "A", "t", "w", "v", "k_c")
R <- 10^3

# Define matrix
mat <- sobol_matrices(N = N, params = params)
mat[, "delta"] <- qunif(mat[, "delta"], 0.21 + 0.21 * -0.005, 0.21 + 0.21 * 0.005)
mat[, "gamma"] <- qunif(mat[, "gamma"], 0.059 + 0.059 * -0.001, 0.059 + 0.059 * 0.001)
mat[, "A"] <- qunif(mat[, "A"], 350 + 350 * -0.15, 350 + 350 * 0.15)
mat[, "t"] <- qunif(mat[, "t"], 27 + 27 * -0.01, 27 + 27 * 0.01)
mat[, "w"] <- qunif(mat[, "w"], 2.5 + 2.5 * -0.05, 2.5 + 2.5 * 0.05)
mat[, "v"] <- qunif(mat[, "v"], 2.49 + 2.49 * -0.04, 2.49 + 2.49 * 0.04)
mat[, "k_c"] <- EnvStats::qlnormTrunc(mat[, "k_c"],
  meanlog = 0.1, sdlog = 15, min = 0.3, max = 1.85
)

# Run model
y.pm <- pm_fun(
  delta = mat[, "delta"],
  A = mat[, "A"],
  gamma = mat[, "gamma"],
  t = mat[, "t"],
  w = mat[, "w"],
  v = mat[, "v"],
  k_c = mat[, "k_c"]
)

# Compute Sobol' indices
params.pm <- c("$\\Delta$", "$\\gamma$", "$A$", "$t$", "$w$", "$v$", "$k_c$")
ind.pm <- sobol_indices(
  Y = y.pm, N = N, params = params.pm,
  first = "jansen", R = R, boot = TRUE
```

```
)[
  , method := "Penman-Monteith"
]
```

```
dt.pt <- data.table(y.pt)[, method := "Priestley-Taylor"] %>%
  setnames(., "y.pt", "value")
dt.pm <- data.table(y.pm)[, method := "Penman-Monteith"] %>%
  setnames(., "y.pm", "value")

# Uncertainty
a <- rbind(dt.pt, dt.pm) %>%
  ggplot(., aes(value, fill = method)) +
  geom_histogram(alpha = 0.3, color = "black", position = "identity") +
  scale_fill_manual(
    name = "Method",
    values = wes_palette(n = 2, name = "Chevalier1")
  ) +
  labs(x = "$ET_c$", y = "Counts") +
  theme_AP() +
  theme(legend.position = c(0.6, 0.7))

# Sobol' indices
b <- rbind(ind.pt, ind.pm) %>%
  .[, parameters := factor(parameters, levels = c(
    "$\\Delta$", "$\\gamma$",
    "$A$", "$t$", "$w$", "$v$", "$\\alpha$", "$k_c$"
  ))] %>%
  plot_sobol(.) +
  theme(legend.position = "none") +
  facet_grid(~method,
    scales = "free_x",
    space = "free_x"
  )

legend <- cowplot::get_legend(b + theme(legend.position = "top"))
bottom <- cowplot::plot_grid(a, b, ncol = 2, labels = "auto", rel_widths = c(0.45, 0.55))
```

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

```
cowplot::plot_grid(legend, bottom, ncol = 1, rel_heights = c(0.2, 0.8))
```



3 Efficiencies

```
# PLOT EFFICIENCIES ----------------------------------------------------------------

# USA data
usa.dt <- fread("usa_efficiency.csv")
usa.dt <- usa.dt[, Efficiency:= consumptive.use / total.withdrawal]

a <- ggplot(usa.dt, aes(Efficiency)) +
  geom_histogram() +
  geom_vline(xintercept = 0.6, lty = 2) +
  labs(x = "$E_p$", y = "Counts") +
  theme_AP()

# FAO 1997 (Irrigation potential in Africa)
fao_dt <- fread("fao_1997.csv")
fao_dt <- fao_dt[, Efficiency:= Efficiency / 100]

b <- ggplot(fao_dt, aes(Efficiency)) +
  geom_histogram() +
  labs(x = "$E_p$", y = "") +
  theme_AP()

# Bos and Nugteren data
bos.dt <- fread("bos.dt.csv")
col_names <- colnames(bos.dt)[2:7]
setnames(bos.dt, col_names, paste("$", col_names, "$", sep = ""))

# Create data set with E_a values as defined by Rohwer
dt_e_a <- data.table("Type" = c("Sprinkler", "Surface"),
                     "Value" = c(0.75, 0.6))
```

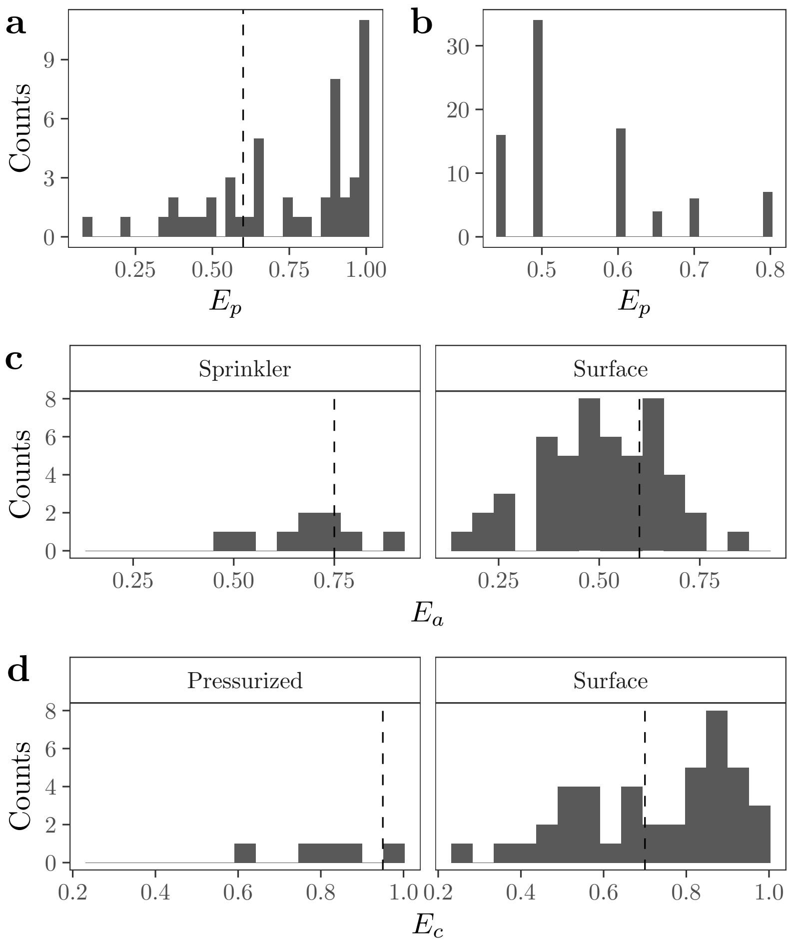
```r
c <- ggplot(bos.dt, aes(`$E_a$`)) +
  geom_histogram(bins = 15) +
  geom_vline(data = dt_e_a, aes(xintercept = Value), lty = 2) +
  facet_grid(~ Type) +
  labs(x = "$E_a$", y = "Counts") +
  theme_AP()

# Create data set with E_c values as defined by Rohwer
dt_e_c <- data.table("Type" = c("Surface", "Pressurized"),
                     "Value" = c(0.7, 0.95))

bos.dt.copy <- copy(bos.dt)

d <- bos.dt.copy[, Type:= ifelse(Type == "Surface", "Surface", "Pressurized")] %>%
  ggplot(., aes(`$E_c$`)) +
  geom_histogram(bins = 15) +
  geom_vline(data = dt_e_c, aes(xintercept = Value), lty = 2) +
  facet_grid(~ Type) +
  labs(x = "$E_c$", y = "Counts") +
  theme_AP()

# Merge all plots
top <- cowplot::plot_grid(a, b, ncol = 2, labels = "auto")
```

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

## Warning: Removed 3 rows containing non-finite values (stat_bin).

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

```r
bottom <- cowplot::plot_grid(c, d, ncol = 1, labels = c("c", "d"))
```

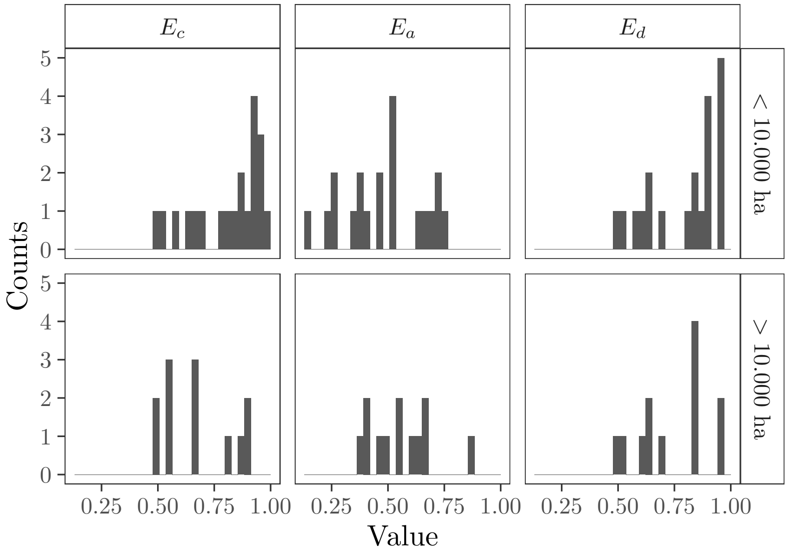
```
## Warning: Removed 31 rows containing non-finite values (stat_bin).

## Warning: Removed 43 rows containing non-finite values (stat_bin).
```

```r
cowplot::plot_grid(top, bottom, ncol = 1, rel_heights = c(0.35, 0.65))
```




```
# EFFICIENCIES AS A FUNCTION OF SCALE ---------------------------------------------------

dt.tmp <- bos.dt[, Scale := ifelse(Irrigated_area < 10000,
  "$<10.000$ ha", "$>10.000$ ha"
)] %>%
  na.omit()

melt(dt.tmp, measure.vars = c("$E_c$", "$E_a$", "$E_d$")) %>%
  ggplot(., aes(value)) +
  geom_histogram() +
  labs(x = "Value", y = "Counts") +
  facet_grid(Scale ~ variable) +
  theme_AP()
```

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



4 Uncertainty of Rohwer factorial design

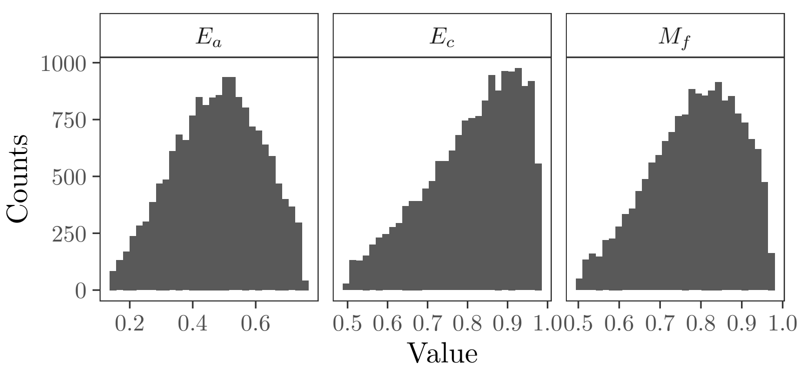
```r
# DEFINE SETTINGS --------------------------------------------------------------------

N <- 2^14
params <- c("E_a", "E_c", "M_f")
R <- 10^3
```

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

mat <- sobol_matrices(N = N, params = params, type = "R")

# Truncated Weibull for E_a
shape <- 3.502469
scale <- 0.5444373
minimum <- 0.14
maximum <- 0.75
Fa.weibull <- pweibull(minimum, shape = shape, scale = scale)
Fb.weibull <- pweibull(maximum, shape = shape, scale = scale)
mat[, "E_a"] <- qunif(mat[, "E_a"], Fa.weibull, Fb.weibull)
mat[, "E_a"] <- qweibull(mat[, "E_a"], shape, scale)

# Truncated Beta for E_c
shape1 <- 5.759496
shape2 <- 1.403552
minimum <- 0.5
maximum <- 0.98
Fa.beta <- pbeta(minimum, shape1 = shape1, shape2 = shape2)
Fb.beta <- pbeta(maximum, shape1 = shape1, shape2 = shape2)
mat[, "E_c"] <- qunif(mat[, "E_c"], Fa.beta, Fb.beta)
mat[, "E_c"] <- qbeta(mat[, "E_c"], shape1, shape2)
```

```r
# Truncated Weibull for M_f
shape <- 6.844793
scale <- 0.8481904
minimum <- 0.5
maximum <- 0.97
Fa.weibull <- pweibull(minimum, shape = shape, scale = scale)
Fb.weibull <- pweibull(maximum, shape = shape, scale = scale)
mat[, "M_f"] <- qunif(mat[, "M_f"], Fa.weibull, Fb.weibull)
mat[, "M_f"] <- qweibull(mat[, "M_f"], shape, scale)
```

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

dt <- data.table(mat)
params_plot <- paste("$", params, "$", sep = "")
dt <- setnames(dt, params, params_plot)
dt[1:N] %>%
  melt(., measure.vars = params_plot) %>%
  ggplot(., aes(value)) +
  geom_histogram() +
  labs(x = "Value", y = "Counts") +
  scale_x_continuous(breaks = pretty_breaks(n = 4)) +
  facet_wrap(~variable, scales = "free_x") +
  theme_AP()
```

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




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

y <- mat[, "E_a"] * mat[, "M_f"] * mat[, "E_c"]

# Some statistics
data.table(y)[, .(min = min(y), max = max(y))]
```

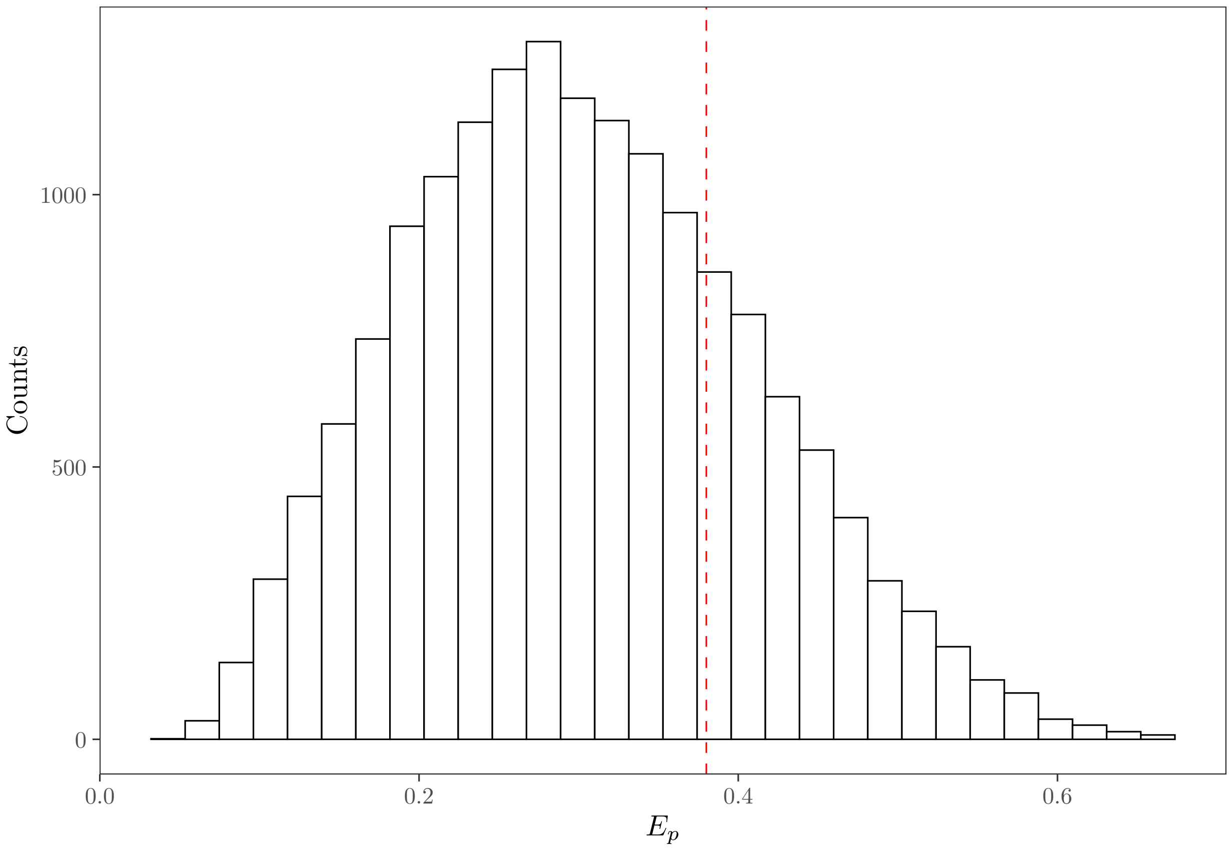
```
##           min       max
## 1: 0.04575103 0.6885467
```

```r
quantile(y, probs = c(0.025, 0.975))
```

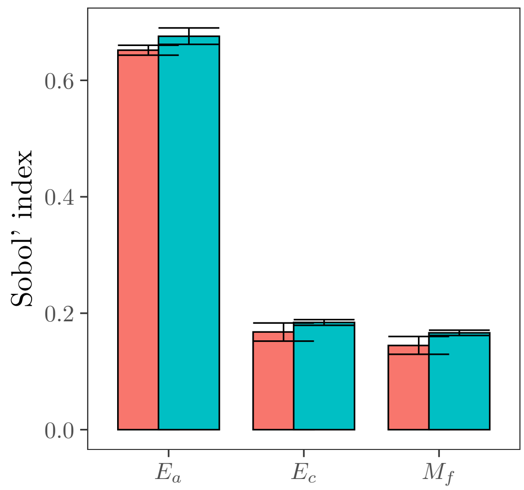
```
##      2.5%     97.5%
## 0.1154830 0.5293579
```

```r
# PLOT UNCERTAINTY -----------------------------------------------------------

a <- plot_uncertainty(Y = y, N = N) +
  labs(x = "$E_p$", y = "Counts") +
  geom_vline(xintercept = 0.38, lty = 2, color = "red")
a
```

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



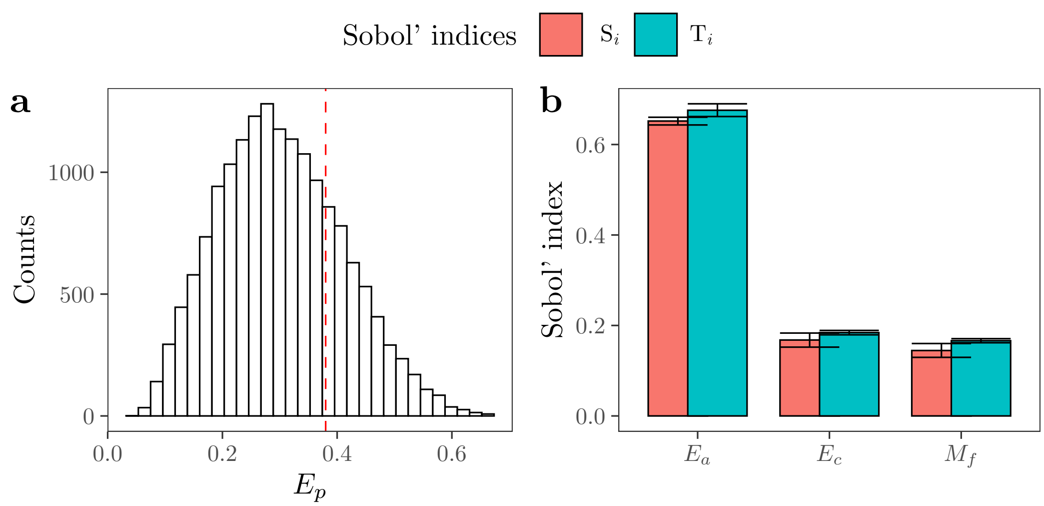

```r
# SENSITIVITY ANALYSIS -------------------------------------------------------

ind <- sobol_indices(
  Y = y, N = N, params = params_plot, R = R, boot = TRUE,
  first = "jansen"
)
b <- plot_sobol(ind) +
  theme(legend.position = "none")
b
```




```r
# MERGE PLOTS -----------------------------------------------------------------------

legend <- get_legend(b + theme(legend.position = "top"))
bottom <- cowplot::plot_grid(a, b, ncol = 2, labels = "auto")
```

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

```r
cowplot::plot_grid(legend, bottom, ncol = 1, rel_heights = c(0.15, 0.85))
```



5 Irrigated areas

```r
# PLOT UNCERTAINTY IN IRRIGATED AREAS ----------------------------------------------

# Read data compiled by Meier
irrigated.areas <- fread("meier.csv")

# Arrange and drop Oceania and 0's
dt <- melt(irrigated.areas, measure.vars = 4:9) %>%
  .[!Continent == "Oceania"] %>%
```

```
  .[!value == 0]

# List to plot
continent_list <- list(c("Africa", "Americas"), c("Asia", "Europe"))

# Plot
gg <- list()
for (i in 1:length(continent_list)) {
  gg[[i]] <- ggplot(dt[Continent %in% continent_list[[i]]], aes(reorder(Country, value), value
    geom_point(stat = "identity", aes(color = variable)) +
    scale_y_log10(
      breaks = trans_breaks("log10", function(x) 10^x),
      labels = trans_format("log10", math_format(10^.x))
    ) +
    coord_flip() +
    scale_color_manual(
      name = "Dataset",
      values = c(
        "yellowgreen", "seagreen4", "magenta3",
        "sienna3", "turquoise2", "khaki3"
      )
    ) +
    labs(
      x = "",
      y = "Irrigated area (ha)"
    ) +
    facet_wrap(~Continent, scales = "free_y") +
    theme_AP()
}
```

```
gg[[1]]
```


```r
gg[[2]]
```

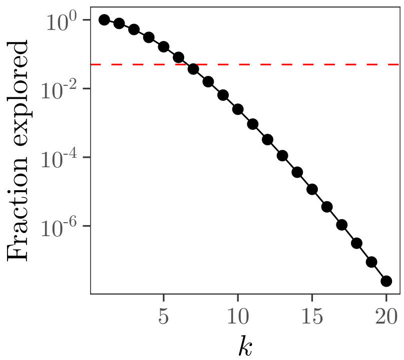
```
## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
```

```
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.
```

```
## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.

## Warning in (function (texString, cex = 1, face = 1, engine =
## getOption("tikzDefaultEngine"), : Attempting to calculate the width of a Unicode
## stringusing the pdftex engine. This may fail! See the Unicodesection of ?
## tikzDevice for more information.
```

6 Ratio of sphere to hypercube

```r
# ASSESS THE FRACTION OF THE UNCERTAIN SPACE EXAMINED BY OAT -----------------------

# Formula: Ratio of the hypercube to the hypersphere
oat_exploration <- function(x) pi^((x) / 2) * (0.5)^(x) / gamma(1 + x / 2)

# Check from 1 to 20 dimensions
out <- sapply(1:20, function(x) oat_exploration(x))

# Plot
data.table(k = 1:20, x = out) %>%
  ggplot(., aes(k, x)) +
  geom_point() +
  scale_y_log10(
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))
  ) +
  geom_hline(yintercept = 0.05, lty = 2, color = "red") +
  geom_line() +
  labs(
    y = "Fraction explored",
    x = "$k$"
  ) +
  theme_bw() +
  theme(
    legend.position = "top",
    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
    )
  )
```



7 Assesement of the entire uncertain space

7.1 Uncertainty in the extension of irrigated areas in Texas

```r
# FUNCTIONS TO CONVERT LON LAT TO USA STATES AND COUNTRIES --------------------------

# Lon Lat to USA states
# (extracted from https://stackoverflow.com/questions/
# 8751497/latitude-longitude-coordinates-to-state-code-in-r)
lonlat_to_state <- function(pointsDF,
                            states = spData::us_states,
                            name_col = "NAME") {
  ## Convert points data.frame to an sf POINTS object
  pts <- st_as_sf(pointsDF, coords = 1:2, crs = 4326)

  ## Transform spatial data to some planar coordinate system
  ## (e.g. Web Mercator) as required for geometric operations
  states <- st_transform(states, crs = 3857)
  pts <- st_transform(pts, crs = 3857)

  ## Find names of state (if any) intersected by each point
  state_names <- states[[name_col]]
  ii <- as.integer(st_intersects(pts, states))
  state_names[ii]
}

# Lon Lat to countries
coords2country <- function(points) {
  countriesSP <- rworldmap::getMap(resolution = "low")
  pointsSP <- sp::SpatialPoints(points, proj4string = CRS(proj4string(countriesSP)))
  indices <- sp::over(pointsSP, countriesSP)
  indices$ADMIN
}
```

```r
# READ IN RASTERS -------------------------------------------------------------------

# Define parallel computing
```

```r
n_cores <- floor(detectCores() * 0.75)
cl <- makeCluster(n_cores)
registerDoParallel(cl)

# Vector with the name of the files
c("fao_gmia.asc", "meier_map.tif", "iwmi_giam.tif")
```

```
## [1] "fao_gmia.asc"  "meier_map.tif" "iwmi_giam.tif"
```

```r
vec_rasters <- c("fao_gmia.asc", "GRIPC_irrigated_area.asc")

# Load rasters and transform to csv in parallel
out <- foreach(
  i = 1:length(vec_rasters),
  .packages = c(
    "raster", "data.table", "sf",
    "rworldmap", "sp"
  )
) %dopar% {
  rs <- raster(vec_rasters[i])
  dt <- data.table(rasterToPoints(rs,
    fun = function(r) {
      r > 0
    }
  ))
  states_vector <- lonlat_to_state(dt[, 1:2])
  dt[, states := cbind(states_vector)]
  dt[, country := coords2country(dt[, c("x", "y")])]
  setnames(dt, 3, "area")
}

# Stop parallel cluster
stopCluster(cl)
```

```r
# ARRANGE DATASET -----------------------------------------------------------------

# Read the meier map
meier.map <- fread("meier.map.csv")

# Arrange dataset
names(out) <- c("FAO-GMIA", "GRIPC")
rasters.dt <- rbindlist(out, idcol = "Map")

# Rbind GRIPC, FAO-GMIA and Meier map
all.rasters <- rbind(rasters.dt, meier.map)

# Check differences in global irrigated areas
all.rasters[, sum(area) / 10^6, Map] # Million ha
```

```
##          Map       V1
## 1: FAO-GMIA 307.6357
## 2:    GRIPC 248.5050
## 3:    Meier 368.0746
```

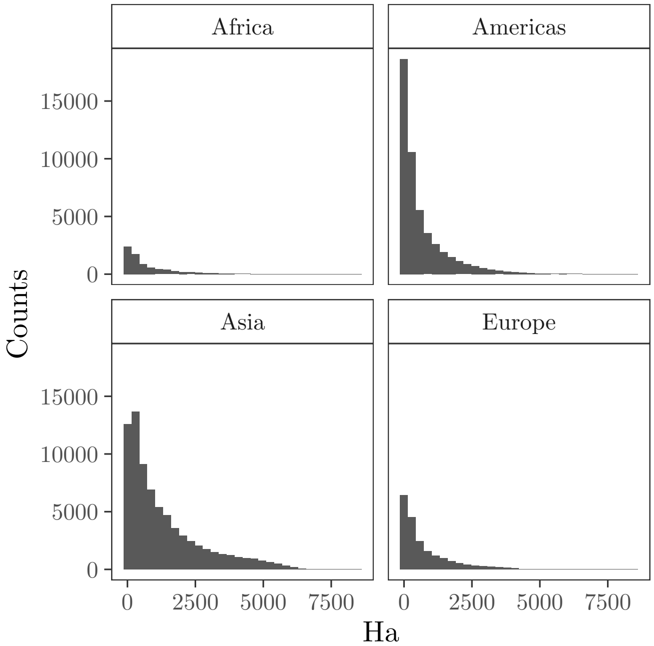
```r
# Export
fwrite(all.rasters, "all.rasters.csv")

# Coordinates of Uvalde, Texas
# 29.209684, -99.786171.

# keep for later: [x< -99.6 & x > -99.8 & y > 29 & y < 29.5]

# meier map: x = -99.70416, y = 29.34583
rasters.merge <- copy(rasters.dt)
rasters.merge <- rasters.merge[, c("x", "y") := round(.SD, 4), .SDcols = c("x", "y")]
rasters.uvalde <- rbind(
  rasters.merge[x == -99.7083 & y == 29.4583],
  meier.map[x >= -99.71 & x <= -99.70 &
    y >= 29.2 & y <= 29.46]
)
```

```r
# CHECK DIFFERENCES AT THE GRID CELL LEVEL PER CONTINENT ---------------------------

rasters.dt <- rasters.dt[, c("x", "y"):= round(.SD, 4), .SDcols = c("x", "y")]
tmp <- merge(rasters.dt[Map == "FAO-GMIA"], rasters.dt[Map == "GRIPC"], by = c("x", "y"))

# Compute absolute difference at the grid level
tmp <- tmp[, abs:= abs(area.x - area.y)]

# Get continent
tmp <- tmp[, continent:= countrycode(tmp[, country.x],
                                     origin = "country.name",
                                     destination = "continent")]
```

```
## Warning in countrycode(tmp[, country.x], origin = "country.name", destination = "continent")
```

```r
# Plot
tmp[!continent == "Oceania"] %>%
ggplot(., aes(abs)) +
  geom_histogram() +
  labs(x = "Ha", y = "Counts") +
  facet_wrap(~continent) +
  theme_AP()
```

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



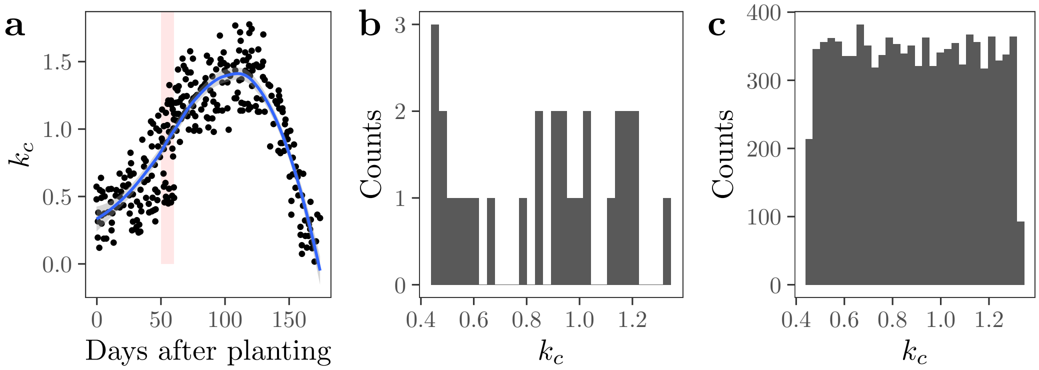
7.2 Uncertainty in k_c coefficients for wheat in Texas

```
# CHECK THE UNCERTAINTY IN KC COEFFICIENTS FOR WHEAT IN TEXAS -----------------

kc_wheat <- fread("kc_wheat_new.csv")

# Define the time frame of the data
min.day <- 50
max.day <- 60

# Retrieve the filtered data
kc_wheat.dt <- kc_wheat[x > min.day & x < max.day][y > 0]

# Uniform distribution
v.final <- runif(10^4, min = min(kc_wheat.dt$y), max = max(kc_wheat.dt$y))
```

```
# PLOT DATA, EMPIRICAL DISTRIBUTION AND MODELED DISTRIBUTION -----------------------

a <- ggplot(kc_wheat, aes(x = x, y = y)) +
  annotate("rect",
    xmin = min.day, xmax = max.day,
    ymin = 0, ymax = Inf, alpha = 0.1, fill = "red"
  ) +
  geom_point(size = 0.6) +
  geom_smooth() +
  labs(x = "Days after planting", y = "$k_c$") +
  theme_AP()

b <- ggplot(kc_wheat.dt, aes(y)) +
```

```r
  geom_histogram() +
  labs(x = "$k_c$", y = "Counts") +
  theme_AP()

c <- ggplot(data.table(v.final), aes(v.final)) +
  geom_histogram() +
  labs(x = "$k_c$", y = "Counts") +
  theme_AP()

plot_grid(a, b, c, ncol = 3, labels = "auto")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

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



7.3 Uncertainty and sensitivity analysis

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

N <- 2^15
R <- 10^3
type <- "R"
order <- "second"
params <- c(
  "I_a", "delta", "gamma", "A", "t", "w", "v",
  "k_c", "P", "E_a", "E_c", "M_f"
)

# Vector with the name of the parameters modified for better plotting
params.plot <- c(
  "$I_a$", "$\\delta$", "$\\gamma$", "$A$", "$t$", "$w$",
  "$v$", "$k_c$", "$P$", "$E_a$", "$E_c$", "$M_f$"
)

# I_a (ha)
# P (mm)
```

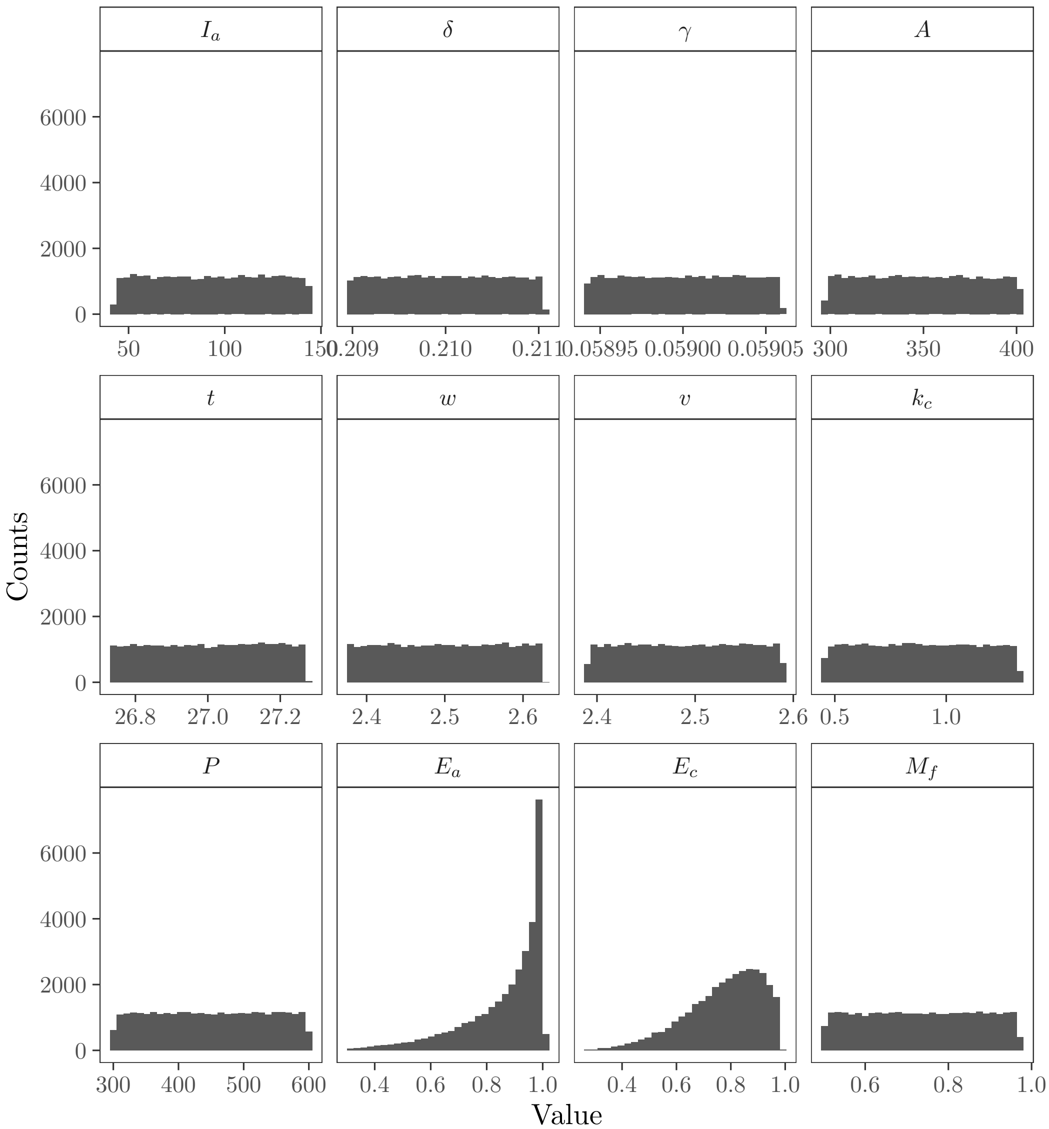
```r
# ET_0 (mm)
```

```r
# DEFINE SAMPLE MATRIX AND TRANSFORM TO APPROPRIATE DISTRIBUTIONS -----------------

# Define sampling matrix
mat <- sobol_matrices(N = N, params = params, order = order, type = type)

# Transform to appropriate probability distributions
mat[, "I_a"] <- qunif(mat[, "I_a"], rasters.uvalde[, min(area)], rasters.uvalde[, max(area)])
mat[, "delta"] <- qunif(mat[, "delta"], 0.21 + 0.21 * -0.005, 0.21 + 0.21 * 0.005)
mat[, "gamma"] <- qunif(mat[, "gamma"], 0.059 + 0.059 * -0.001, 0.059 + 0.059 * 0.001)
mat[, "A"] <- qunif(mat[, "A"], 350 + 350 * -0.15, 350 + 350 * 0.15)
mat[, "t"] <- qunif(mat[, "t"], 27 + 27 * -0.01, 27 + 27 * 0.01)
mat[, "w"] <- qunif(mat[, "w"], 2.5 + 2.5 * -0.05, 2.5 + 2.5 * 0.05)
mat[, "v"] <- qunif(mat[, "v"], 2.49 + 2.49 * -0.04, 2.49 + 2.49 * 0.04)
mat[, "P"] <- qunif(mat[, "P"], 300, 600)
mat[, "M_f"] <- qunif(mat[, "M_f"], 0.5, 0.97)
mat[, "k_c"] <- qunif(mat[, "k_c"], min(kc_wheat.dt$y), max(kc_wheat.dt$y))

# Truncated weibull for E_a
shape.weibull <- 3.659042
scale.weibull <- 0.590995
minimum <- 0.14
maximum <- 0.88
Fa.weibull <- pweibull(minimum, shape = shape.weibull, scale = scale.weibull)
Fb.weibull <- pweibull(maximum, shape = shape.weibull, scale = scale.weibull)
mat[, "E_a"] <- qunif(mat[, "E_a"], Fa.weibull, Fb.weibull)
mat[, "E_a"] <- qbeta(mat[, "E_a"], shape.weibull, scale.weibull)

# Truncated beta for E_c
shape1 <- 5.974774
shape2 <- 1.712115
minimum <- 0.26
maximum <- 0.98
Fa.beta <- pbeta(minimum, shape1 = shape1, shape2 = shape2)
Fb.beta <- pbeta(maximum, shape1 = shape1, shape2 = shape2)
mat[, "E_c"] <- qunif(mat[, "E_c"], Fa.beta, Fb.beta)
mat[, "E_c"] <- qbeta(mat[, "E_c"], shape1, shape2)
```

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

data.table(mat[1:N, ]) %>%
  setnames(., params, params.plot) %>%
  melt(., measure.vars = params.plot) %>%
  ggplot(., aes(value)) +
  geom_histogram() +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  labs(x = "Value", y = "Counts") +
```

```r
  theme_AP() +
  facet_wrap(~variable, scales = "free_x")
```

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




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

full_model <- function(I_a, delta, A, gamma, t, w, v, k_c, P, E_a, E_c, M_f) {
  out <- (I_a * (k_c * ((0.408 * delta * A + gamma * (900 / (t + 273)) * w * v) /
    delta + gamma * (1 + 0.34 * w))) - P) / (E_a * E_c * M_f)
  return(out)
}
```

```r
# RUN THE MODEL -----------------------------------------------------------

y <- full_model(
  I_a = mat[, "I_a"],
```

```r
  delta = mat[, "delta"],
  A = mat[, "A"],
  gamma = mat[, "gamma"],
  t = mat[, "t"],
  w = mat[, "w"],
  v = mat[, "v"],
  k_c = mat[, "k_c"],
  P = mat[, "P"],
  E_a = mat[, "E_a"],
  E_c = mat[, "E_c"],
  M_f = mat[, "M_f"]
)
```

```r
# ASSESS UNCERTAINTIES -------------------------------------------------------

unc <- plot_uncertainty(Y = y, N = N)
```

```r
# ASSESS SENSITIVITIES ------------------------------------------------------------

# Compute sobol' indices
ind <- sobol_indices(
  Y = y, N = N, params = params.plot,
  order = order, boot = TRUE, R = R,
  parallel = "multicore"
)

# Everything is explained by first and second-order effects
ind[sensitivity %in% c("Si", "Sij"), sum(original)]
```

```
## [1] 0.9878623
```

```r
# Plot sobol' indices
sobol.plot <- plot_sobol(ind) +
  theme(legend.position = c(0.83, 0.5))
```

```r
# PLOT SECOND-ORDER INDICES -------------------------------------------------------

second.order <- plot_sobol(ind, "second")
```

```r
# PLOT SCATTERPLOTS ---------------------------------------------------------------
```

7.4 One-at-a-time (OAT)

```r
# CONSTRUCT SAMPLE MATRIX ----------------------------------------------------------

A <- mat[1:N, ]
B <- matrix(rep(Rfast::colmeans(A), each = N), nrow = N)

X <- B
```

```r
for (j in 1:ncol(A)) {
  AB <- B
  AB[, j] <- A[, j]
  X <- rbind(X, AB)
}

mat.oat <- X[(N + 1):nrow(X), ]
colnames(mat.oat) <- params
```

```r
# RUN THE MODEL ------------------------------------------------------------------

y.oat <- full_model(
  I_a = mat.oat[, "I_a"],
  delta = mat.oat[, "delta"],
  A = mat.oat[, "A"],
  gamma = mat.oat[, "gamma"],
  t = mat.oat[, "t"],
  w = mat.oat[, "w"],
  v = mat.oat[, "v"],
  k_c = mat.oat[, "k_c"],
  P = mat.oat[, "P"],
  E_a = mat.oat[, "E_a"],
  E_c = mat.oat[, "E_c"],
  M_f = mat.oat[, "M_f"]
)
```

```r
# COMPUTE A SINGLE-POINT ESTIMATE USING MEAN VALUES--------------------------------

vec_means <- colMeans(A)

y.point <- full_model(
  I_a = vec_means[["I_a"]],
  delta = vec_means[["delta"]],
  A = vec_means[["A"]],
  gamma = vec_means[["gamma"]],
  t = vec_means[["t"]],
  w = vec_means[["w"]],
  v = vec_means[["v"]],
  k_c = vec_means[["k_c"]],
  P = vec_means[["P"]],
  E_a = vec_means[["E_a"]],
  E_c = vec_means[["E_c"]],
  M_f = vec_means[["M_f"]]
)

y.point
```

```
## [1] 24130.3
```

7.5 Compare OAT and global sensitivity analysis

```
# ASSESS UNCERTAINTIES -------------------------------------------------------

unc.oat <- plot_uncertainty(Y = y.oat, N = N)

full.unc <- data.table(cbind(y[1:N], y.oat))
colnames(full.unc) <- c("Global", "OAT")

a <- melt(full.unc, measure.vars = colnames(full.unc)) %>%
  ggplot(., aes(value, fill = variable)) +
  geom_histogram(position = "identity", alpha = 0.3, color = "black") +
  labs(x = "$y$", y = "Counts") +
  scale_x_log10(
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", scales::math_format(10^.x))
  ) +
  geom_vline(xintercept = y.point, lty = 2, color = "red", size = 2) +
  scale_fill_manual(values = wes_palette(2, name = "Chevalier1"), name = "Uncertainty analysis"
  theme_AP() +
  theme(legend.position = c(0.2, 0.5))
```

```
# SOME STATISTICS ------------------------------------------------------------

stat.full.unc <- melt(full.unc, measure.vars = c("Global", "OAT"))
stat.full.unc[, .(min = min(value), max = max(value)), variable]
```

```
##    variable       min       max
## 1:   Global  2868.174 282749.56
## 2:      OAT 10557.436  71577.32
```

```
# Quantiles
probs.quantile <- c(0, 0.025, 0.1, 0.5, 0.9, 0.975, 1)
stat.full.unc[, .(value = quantile(value, probs = probs.quantile)), variable] %>%
  .[, quantile := rep(probs.quantile, 2)] %>%
  dcast(., variable ~ quantile, value.var = "value") %>%
  print()
```

```
##    variable         0    0.025      0.1      0.5     0.9    0.975         1
## 1:   Global  2868.174  7016.82 10539.97 22949.88 48470.0 71615.64 282749.56
## 2:      OAT 10557.436 15048.53 20710.22 24130.14 28983.3 35074.37  71577.32
```

```
# Sum of first and second-order indices
ind[sensitivity == "Si", sum(original)]
```

```
## [1] 0.8714554
```

```
ind[sensitivity %in% c("Si", "Sij"), sum(original)]
```

```
## [1] 0.9878623
```

```r
# MERGE UNCERTAINTY AND SOBOL' INDICES ----------------------------------------

plot_grid(a, sobol.plot, second.order, ncol = 1, labels = "auto")
```

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