The delusive accuracy of global irrigation water withdrawal estimates

R code

Arnald Puy

Contents

1 Preliminary functions **2**

2 Irrigated area (I_a) **2**

3 The crop evapotranspiration (ET_c) **3**

4 Irrigation efficiency (E_p) **13**
4.1 Rohwer et al. 's factorial design 15

5 Assessment of uncertainties **20**
5.1 OAT's perfunctory exploration of the uncertainty space 20
5.2 Assesment of uncertainties at the grid cell level 22
5.3 Uncertainty in the extension of irrigated areas in Uvalde, Texas 22
5.4 Uncertainty in k_c coefficients for wheat in Texas 25
5.5 Uncertainty and sensitivity analysis 27
5.6 One-at-a-time (OAT) 35
5.7 Compare OAT and global sensitivity analysis 36

References **41**

1 Preliminary functions

```
# 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"))
}

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

checkpoint("2021-02-22",
           R.version ="4.0.3",
           checkpointLocation = getwd())
```

2 Irrigated area (I_a)

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

```r
# 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()
}
```

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

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

3 The crop evapotranspiration (ET_c)

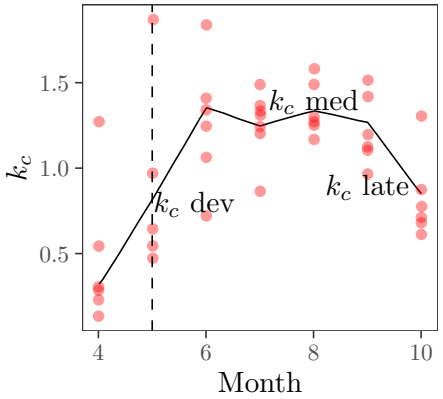
```r
# K_C VALUES FOR SALT CEDAR ----------------------------------------------------

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

Figure 1: Irrigated area estimates produced for Africa and the Americas by the FAO-GMIA (Siebert et al. 2013), the IWMI–GIAM (Thenkabail et al. 2009), the GRIPC (Salmon et al. 2015), Meier's map (Meier, Zabel, and Mauser 2018), Aquastat (FAO 2016) and FAOSTAT (FAO 2017). The data has been retrieved from Meier, Zabel, and Mauser (2018)

Figure 2: Irrigated area estimates produced for Asia and Europe by the FAO-GMIA (Siebert et al. 2013), the IWMI–GIAM (Thenkabail et al. 2009), the GRIPC (Salmon et al. 2015), Meier's map (Meier, Zabel, and Mauser 2018), Aquastat (FAO 2016) and FAOSTAT (FAO 2017). The data has been retrieved from Meier, Zabel, and Mauser (2018)

```
kc_evolution_point <- fread("kc_evolution_point.csv")

# 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))
```

```
# PLOT K_C VALUES FOR SALT CEDAR ONLY ----------------------------------------

a
```



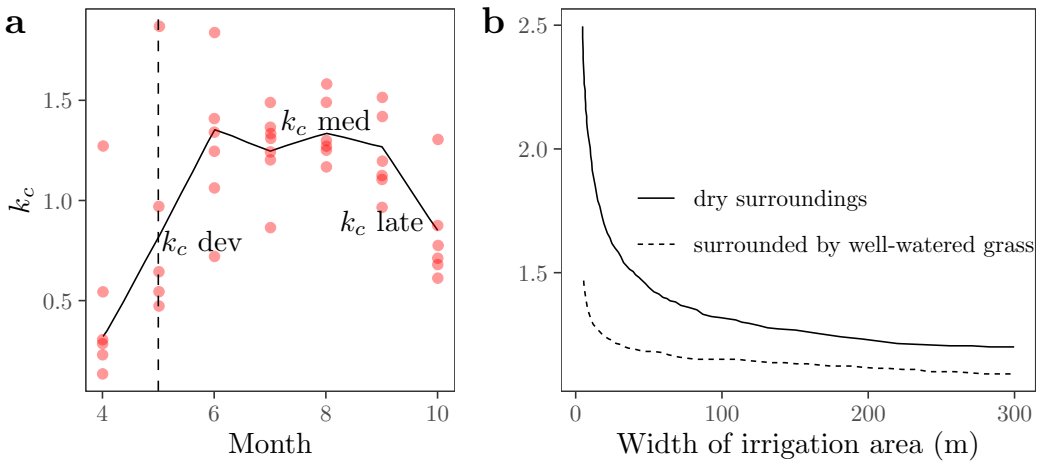


Figure 3: The crop coefficient. a) Evolution of k_c values for salt cedar. The solid black line is the mean k_c, the red dots show the individual measured values (adapted from Figure 10 in Nichols et al. (2004)), and the vertical, dashed black line marks the values selected for the global sensitivity analysis presented in Figure~1c and d. b) Oasis effect on k_c values (adapted from Figure 46 in Allen et al. (1998)).

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

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

# LISTED FUNCTIONS -----------------------------

et0_fun <- list(

  "pt" = function(alpha, delta, gamma, A, k_c = 1)
    k_c * (alpha * ((delta * A) / (gamma + delta))),

  "pm" = function(delta, A, gamma, T_a, w, v, k_c = 1)
    k_c * ((0.408 * delta * A + gamma * (900 / (T_a + 273)) * w * v) /
                    delta + gamma * (1 + 0.34 * w))
)

mat_transform <- function(mat, method, et_c = FALSE) {

  if (method == "pt") {
    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)

  } else if (method == "pm") {
    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_a"] <- qunif(mat[, "T_a"], 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)
  }

  if (et_c == TRUE) {
    mat[, "k_c"] <- qunif(mat[, "k_c"], min(cedar.min.max$y), max(cedar.min.max$y))
  }
  return(mat)
}

params.shared <- c("delta", "gamma", "A")
params.pt <- c(params.shared, "alpha")
params.pm <- c(params.shared, "T_a", "v", "w")

# COMPUTATION OF ET_0 ------------------------------------------------------------

y <- list()

for (i in list("pt", "pm")) {

  if(i == "pt") {

    mat <- sobol_matrices(N = N, params = params.pt)
    mat <- mat_transform(mat = mat, method = i, et_c = FALSE)
    y[[i]] <- et0_fun[[i]](alpha = mat[, "alpha"],
                           delta = mat[, "delta"],
                           gamma = mat[, "gamma"],
                           A = mat[, "A"])

  } else if (i == "pm") {

    mat <- sobol_matrices(N = N, params = params.pm)
    mat <- mat_transform(mat = mat, method = i, et_c = FALSE)
    y[[i]] <- et0_fun[[i]](delta = mat[, "delta"],
                           A = mat[, "A"],
                           gamma = mat[, "gamma"],
                           T_a = mat[, "T_a"],
                           w = mat[, "w"],
                           v = mat[, "v"])
  }
}

# COMPUTATION OF ET_C ------------------------------------------------------------

y.kc <- list()
```

```r
for(i in list("pt", "pm")) {

  if(i == "pt") {

    mat <- sobol_matrices(N = N, params = c(params.pt, "k_c"))
    mat <- mat_transform(mat = mat, method = i, et_c = TRUE)
    y.kc[[i]] <- et0_fun[[i]](alpha = mat[, "alpha"],
                              delta = mat[, "delta"],
                              gamma = mat[, "gamma"],
                              A = mat[, "A"],
                              k_c = mat[, "k_c"])

  } else if (i == "pm") {

    mat <- sobol_matrices(N = N, params = c(params.pm, "k_c"))
    mat <- mat_transform(mat = mat, method = i, et_c = TRUE)
    y.kc[[i]] <- et0_fun[[i]](delta = mat[, "delta"],
                              A = mat[, "A"],
                              gamma = mat[, "gamma"],
                              T_a = mat[, "T_a"],
                              w = mat[, "w"],
                              v = mat[, "v"],
                              k_c = mat[, "k_c"])
  }
}

# FUNCTION TO EXTRACT UNCERTAINTY ------------------------------------------

extract_unc <- function(out, N = N) {
  value_output <- unlist(lapply(out, function(x) x[1:N]), use.names = FALSE)
  da <- data.table(value_output)[, method:= rep(names(et0_fun), each = N)] %>%
    .[, method:= toupper(method)]
  gg <- ggplot(da, aes(value_output, fill = method)) +
    geom_histogram(alpha = 0.3, color = "black", position = "identity") +
    scale_fill_manual(name = "$ET_0$ formula",
                      values = wes_palette(n = 2, name = "Chevalier1")) +
    theme_AP() +
    theme(legend.position = "none")
  return(gg)
}

# PLOT UNCERTAINTY ---------------------------------------------------------

plots.unc <- lapply(list(y, y.kc), function(x) extract_unc(out = x, N = N))

plots.unc[[1]] <- plots.unc[[1]] + labs(x = "$ET_0$ (mm)", y = "Counts") +
  scale_x_continuous(limits = c(0, 800))
```

```r
plots.unc[[2]] <- plots.unc[[2]] + labs(x = "$ET_c$ (mm)", y = "Counts") +
  scale_x_continuous(limits = c(0, 800))

etc <- plots.unc[[2]] + theme(legend.position = "top") +
  scale_fill_manual(labels = c("FAO-56 Penman Monteith", "Priestley-Taylor"),
                    values = wes_palette(n = 2, name = "Chevalier1"))
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```r
  # For later

# SOBOL' INDICES ------------------------------------------------------------

params.pt.plot <- c("$\\Delta$", "$\\gamma$", "$A$", "$\\alpha$")
params.pm.plot <- c("$\\Delta$", "$\\gamma$", "$A$", "$T_a$", "$v$", "$w$")
first <- "jansen"

# ET0 ---------------------------

ind <- list()

for (i in list("pt", "pm")) {

  if (i == "pt") {
    params <- params.pt.plot

  } else if (i == "pm") {
    params <- params.pm.plot
  }

  ind[[i]] <- sobol_indices(Y = y[[i]], N = N, params = params,
                            first = first, boot = TRUE, R = R)
  ind[[i]]$results[, method:= i]
}

# ET0 ---------------------------

ind.kc <- list()

for( i in list("pt", "pm")) {

  if (i == "pt") {
    params <- c(params.pt.plot, "$k_c$")

  } else if (i == "pm") {
    params <- c(params.pm.plot, "$k_c$")
  }
```

```
  ind.kc[[i]] <- sobol_indices(Y = y.kc[[i]], N = N, params = params,
                               first = first, R = R, boot = TRUE)
  ind.kc[[i]]$results[, method:= i]
}

# FUNCTION TO EXTRACT SOBOL' INDICES ----------------------------------------

extract_sobol <- function(data) {
  out <- lapply(data, function(x) x$results) %>%
    rbindlist(.) %>%
    .[, method:= toupper(method)] %>%
    .[, parameters:= factor(parameters, levels = c("$\\Delta$", "$\\gamma$",
                                                   "$A$", "$T_a$", "$w$", "$v$",
                                                   "$\\alpha$", "$k_c$"))] %>%
    ggplot(., aes(parameters, original, fill = sensitivity)) +
    geom_bar(stat = "identity",
             position = position_dodge(0.6), color = "black") +
    scale_y_continuous(breaks = scales::pretty_breaks(n = 3)) +
    labs(x = "", y = "Sobol' index") +
    ggplot2::scale_fill_discrete(name = "Sobol' indices",
                                 labels = c(expression(S[italic(i)]), expression(T[italic(i)]))
    theme_AP() +
    theme(legend.position = "none") +
    facet_grid(~ method,
               scales = "free_x",
               space = "free_x")
  return(out)
}

# PLOT SOBOL  ---------------------------------------------------------------

plots.ind <- lapply(list(ind, ind.kc), function(x) extract_sobol(data = x))
```

```
# MERGE ALL PLOTS -----------------------------------------------------------

# Arrange legend
legend_ua <- cowplot::get_legend(plots.unc[[1]] + theme(legend.position = "top"))
legend_sa <- cowplot::get_legend(plots.ind[[1]] + theme(legend.position = "top"))
all_legend <- plot_grid(legend_ua, legend_sa, ncol = 2, rel_heights = c(0.01, 0.01))

# Arrange plots
top <- plot_grid(plots.unc[[1]], plots.ind[[1]], ncol = 2, labels = c("a", "b"))
bottom <- plot_grid(plots.unc[[2]], plots.ind[[2]], ncol = 2, labels = c("c", "d"))
all_plot <- plot_grid(top, bottom, ncol = 1)

# Merge
plot_grid(all_legend, all_plot, ncol = 1, rel_heights = c(0.1, 0.9))
```

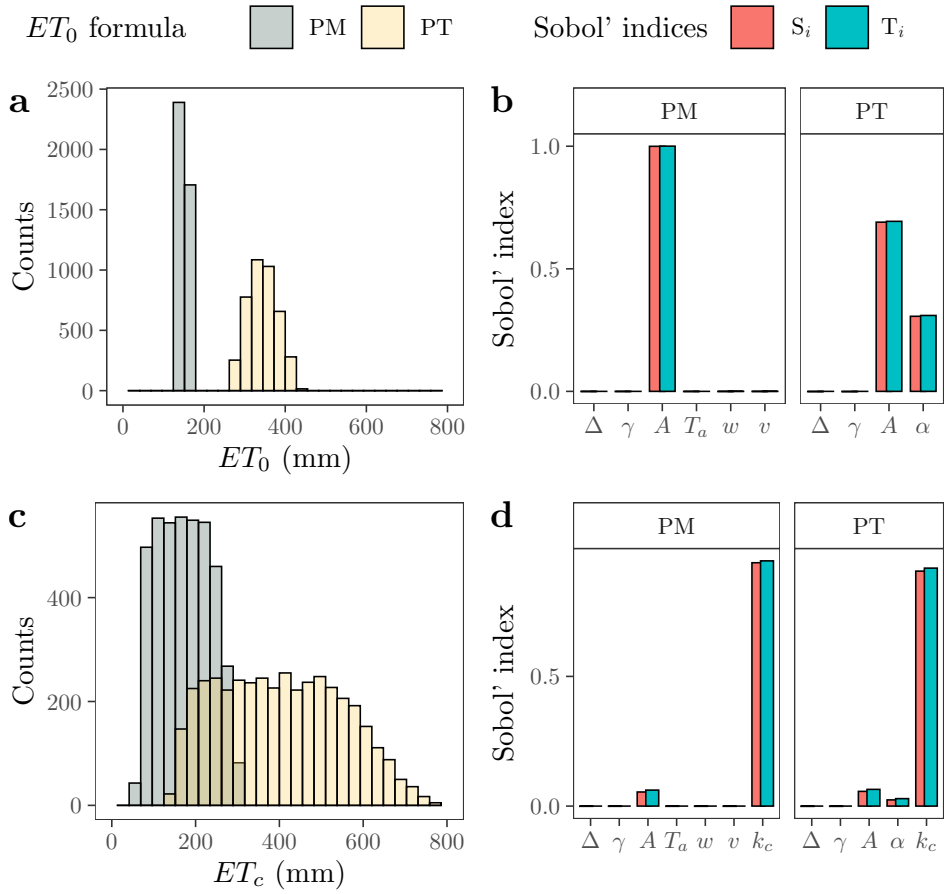


Figure 4: Monte-Carlo based, global uncertainty and sensitivity analysis of the Penman-Monteith (PM) and the Priestley-Taylor (PT) methods. The red bar indicates the first-order effect S_i, i.e. the proportion of variance conveyed by each model input. The blue bar reflects the total-order effect T_i, which includes the first-order effect of the parameter plus the effect derived of its interaction with the rest. The parameters were described with the probability distributions shown in Table~1. See the Supplementary Materials for a technical explanation of global uncertainty and sensitivity analysis. a) Empirical distribution of ET_0. b) Sobol' indices. c) Empirical distribution of ET_c. We described k_c as $U(0.47, 1.86)$ to reproduce the range reported at month 5 by Nichols et al. (2004) for developing salt cedars in the Bosque del Apache, New Mexico (see Figure~3a). d) Sobol' indices.

4 Irrigation efficiency (E_p)

```
# 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))

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")
bottom <- cowplot::plot_grid(c, d, ncol = 1, labels = c("c", "d"))
cowplot::plot_grid(top, bottom, ncol = 1, rel_heights = c(0.35, 0.65))
```

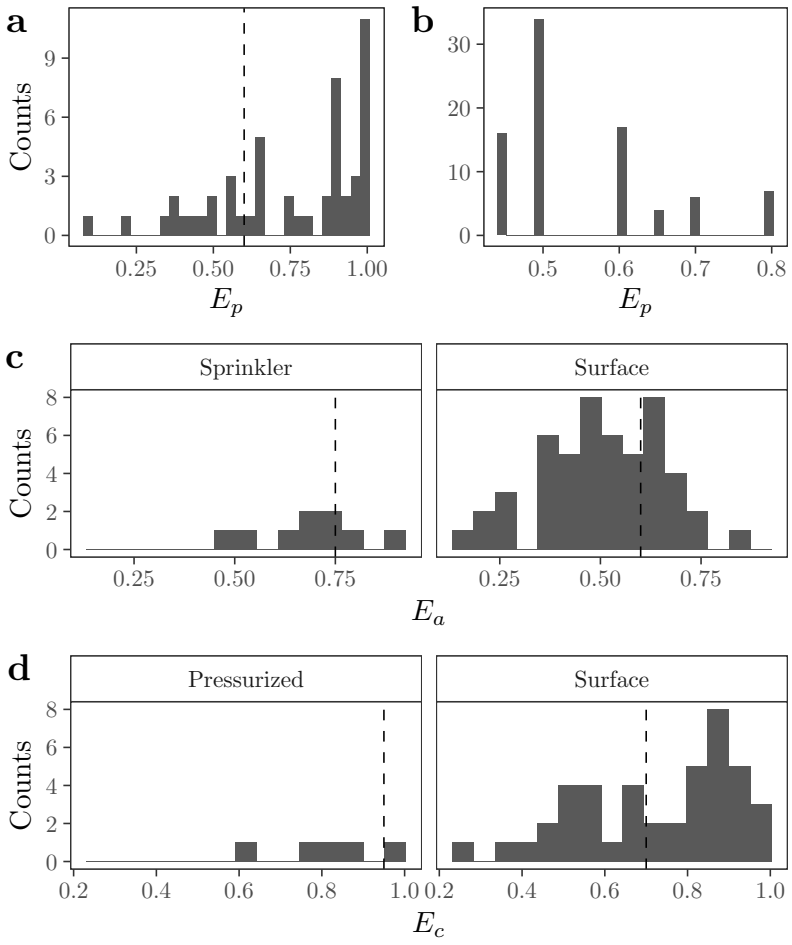


Figure 5: Data on irrigation efficiency. a) Distribution of project efficiencies (E_p) in USA according to Solley, Pierce, and Perlman (1998), calculated as Total water withdrawal / consumptive water use. The vertical dashed line is the E_p value used by Döll and Siebert (2002) to characterize the irrigation efficiency of USA. b) Distribution of project efficiencies (E_p) for Africa reported by FAO (1997). c) Distribution of field application efficiencies (E_a) in surface and sprinkler irrigation systems according to Bos and Nugteren (1990). `Surface` includes furrow, basin and border irrigation systems. The dashed vertical lines mark the E_a point estimates selected by Rohwer, Gerten, and Lucht (2007). d) Distribution of conveyance efficiencies (E_c) in surface and pressurized (sprinkler) irrigation systems according to Bos and Nugteren (1990). The vertical, dashed black lines show the E_c point estimates used by Rohwer, Gerten, and Lucht (2007).

```
# 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`.
```

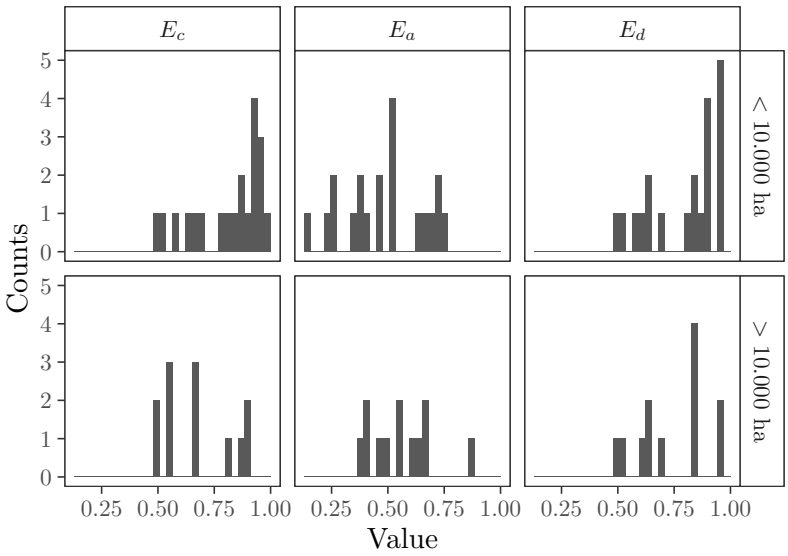


Figure 6: Partial efficiencies as a function of scale according to the data compiled by Bos and Nugteren (1990). E_d stands for distribution efficiency and refers to the water delivery from the source to a specific outlet. E_d is replaced by Rohwer, Gerten, and Lucht (2007) with M_f.

4.1 Rohwer et al. 's factorial design

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

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

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

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

# Truncated Weibull for E_a
```

```r
shape <- 3.502469
scale <- 0.5444373
minimum <- 0.14
maximum <- 0.87
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 <- 0.26
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)

# 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) %>%
  .[, Data:= "Modeled"]

# Filter empirical datasets
dt.ea <- bos.dt[Type == "Surface", `$E_a$`]
dt.ec <- bos.dt[Type == "Surface", `$E_c$`]
dt.mf <- bos.dt[Irrigated_area < 10000, `$E_d$`]

n <- max(length(dt.ea), length(dt.ec), length(dt.mf))

length(dt.ea) <- n
length(dt.ec) <- n
length(dt.mf) <- n
```

```
dt.empirical <- cbind(dt.ea, dt.ec, dt.mf) %>%
  data.table() %>%
  .[, Data:= "Empirical"] %>%
  setnames(., colnames(.), colnames(dt))

rbind(dt.empirical, 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_grid(Data ~ variable, scales = "free_y") +
  theme_AP()
```

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

```
## Warning: Removed 119 rows containing non-finite values (stat_bin).
```

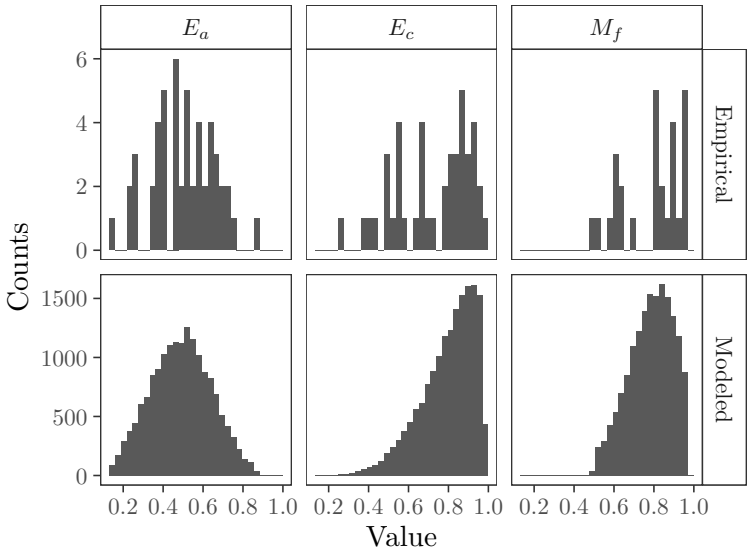


Figure 7: Empirical and modeled distributions for the uncertainty and sensitivity analysis of Equation 5.

```
# PLOT SCATTERPLOTS TO CHECK FOR CORRELATION BETWEEN E_A, E_C, M_F ---------------

out <- t(combn(colnames(dt.empirical)[1:3], 2))
da <- list()

for (i in 1:nrow(out)) {
  cols <- out[i, ]
  da[[i]] <- cbind(dt.empirical[, ..cols], cols[1], cols[2])
  data.table::setnames(da[[i]], colnames(da[[i]]), c("xvar",
                                                     "yvar", "x", "y"))
}

output <- data.table::rbindlist(da)
```

```
ggplot(output, aes(xvar, yvar)) +
  geom_point() +
  facet_wrap(x ~ y) +
  labs(x = "", y = "") +
  theme_AP()
```

```
## Warning: Removed 165 rows containing missing values (geom_point).
```

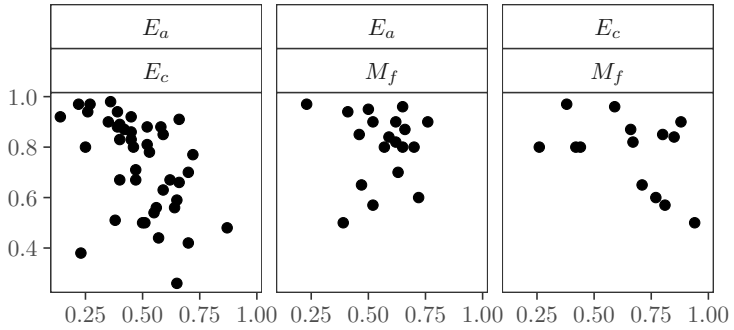


Figure 8: Scatterplots of E_a, E_c, M_f. The topmost and bottomost label facets refer to the x and the y axis respectively.

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

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

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

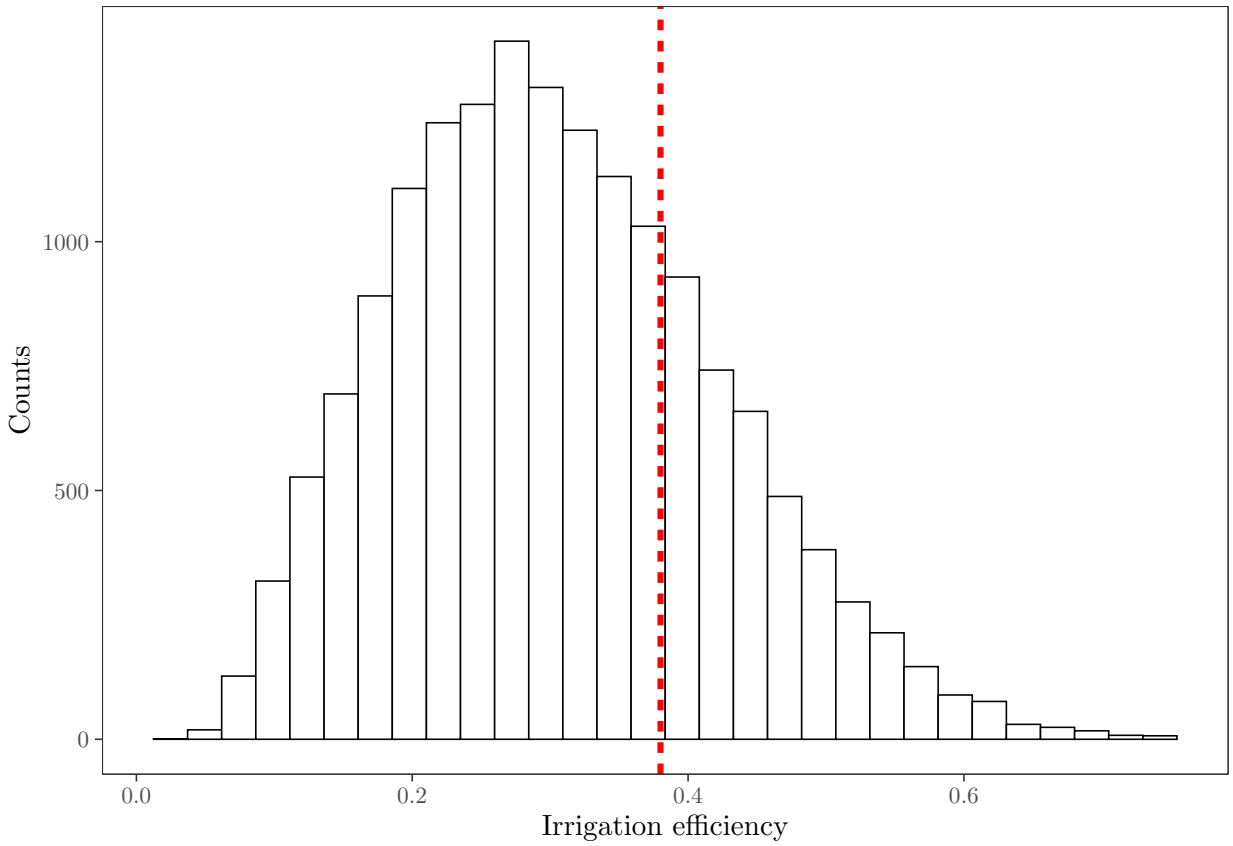
```
##           min       max
## 1: 0.02779241 0.7803253
```

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

```
##      2.5%     97.5%
## 0.1091040 0.5562426
```

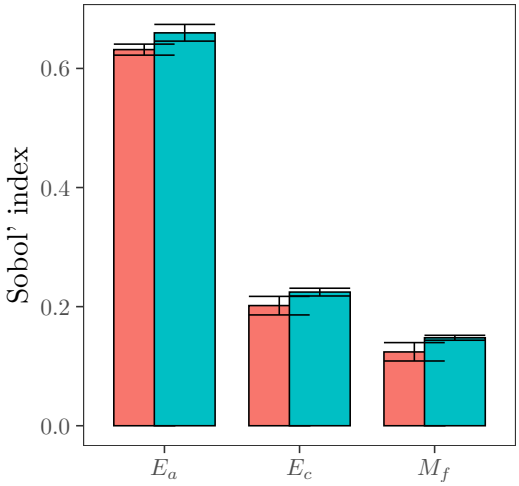
```
# PLOT UNCERTAINTY --------------------------------------------------------------

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

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




```
ep <- a # For later
```

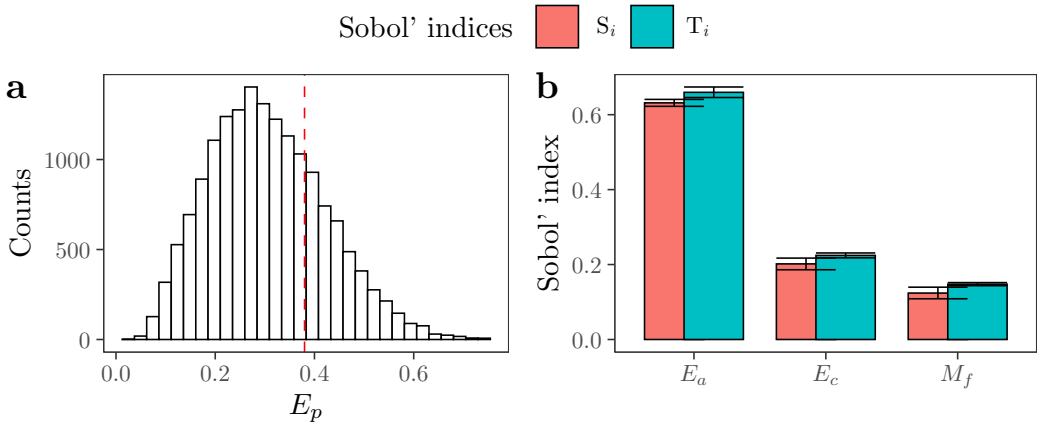
```
# SENSITIVITY ANALYSIS ----------------------------------------------------------

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
```




```
# 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`.
```

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



5 Assessment of uncertainties

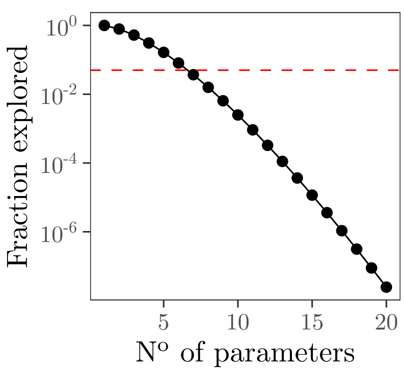
5.1 OAT's perfunctory exploration of the uncertainty space

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

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

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

# Plot
```

```r
oat.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 = "Nº of parameters"
  ) +
  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
    )
  )
oat.plot
```

```
## 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.
```



5.2 Assesment of uncertainties at the grid cell level

5.3 Uncertainty in the extension of irrigated areas in Uvalde, 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
}
```

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

# Define parallel computing
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"
```

```
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)
```

```
# 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
```

```
# 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]
)
```

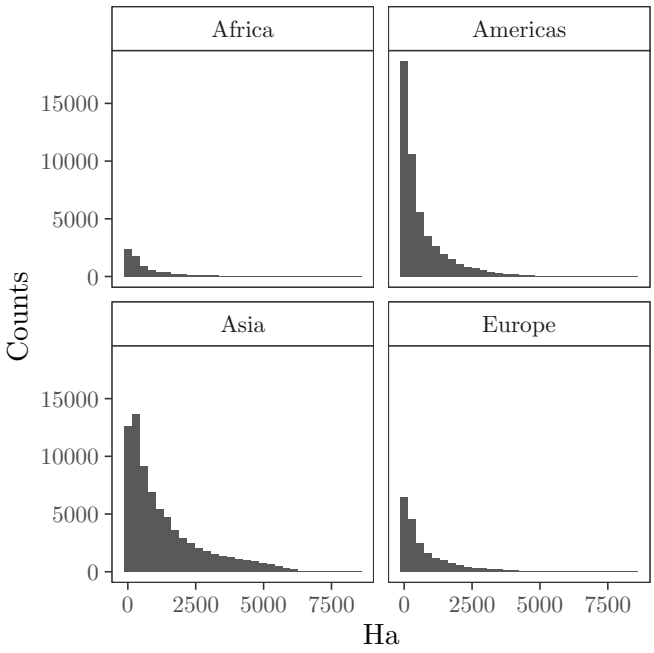
```
# CHECK DIFFERENCES AT THE GRID CELL LEVEL PER CONTINENT ---------------------

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"
```

```
# 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`.
```



5.4 Uncertainty in k_c coefficients for wheat in Texas

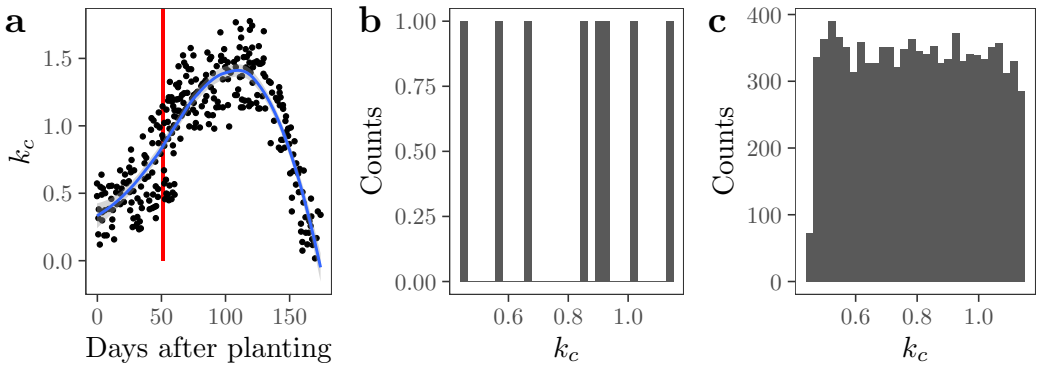
```
# CHECK THE UNCERTAINTY IN KC COEFFICIENTS FOR WHEAT IN TEXAS -----------------

kc_wheat <- fread("kc_wheat_new.csv")

# Define the time frame of the data
min.day <- 50
max.day <- 53
```

```r
# 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))
```

```r
# 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, 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)) +
  geom_histogram() +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  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") +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  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`.
```



5.5 Uncertainty and sensitivity analysis

```
# READ IN CLIMATIC DATA FOR UVALDE FOR JANUARY 2007------------------------------

da <- fread("uvalde_climate_data_january.csv")

# Turn columns into numeric
col_names <- colnames(da)
da[, (col_names):= lapply(.SD, as.numeric), .SDcols = (col_names)]

# Convert Fahrenheit to Celsius
temp_cols <- colnames(da)[da[ , grepl( "Temp", colnames(da))]]
da[, (temp_cols):= lapply(.SD, function(x) (x - 32) / 1.8), .SDcols = (temp_cols)]

# Convert miles per hour to meters per second
wind_cols <- colnames(da)[da[ , grepl( "Wind", colnames(da))]]
da[, (wind_cols):= lapply(.SD, function(x) x / 2.237), .SDcols = (wind_cols)]

# Select days of interest
uvalde_days <- da[Day >= 6 & Day <= 7]

# Mean temperature
T_air <- mean(uvalde_days$`Temp.Avg`)
T_air
```

```
## [1] 11
```

```
# Computation of Vapour pressure (Delta)
# T_air <- 27
e_sat <- 0.6108 * exp((17.27 * T_air) / (T_air + 237.3))
Delta <- (4098 * e_sat) / ((T_air + 237.3) ^ 2)
Delta
```

```
## [1] 0.08725467
```

```
# Computation of vapour deficit (v)
rel_hum <- mean(uvalde_days$`Hum.Avg`) / 100
v <- e_sat - (e_sat * rel_hum)

# Computation of Psychrometric constant (gamma)
z <- 9.1
lambda <- 2.501 - 0.002361 * T_air
P <- 101.3 * ((293 - 0.0065 * z) / 293) ^ 5.256
Gamma <- 0.0016286 * P / lambda
Gamma
```

```
## [1] 0.06658597
```

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

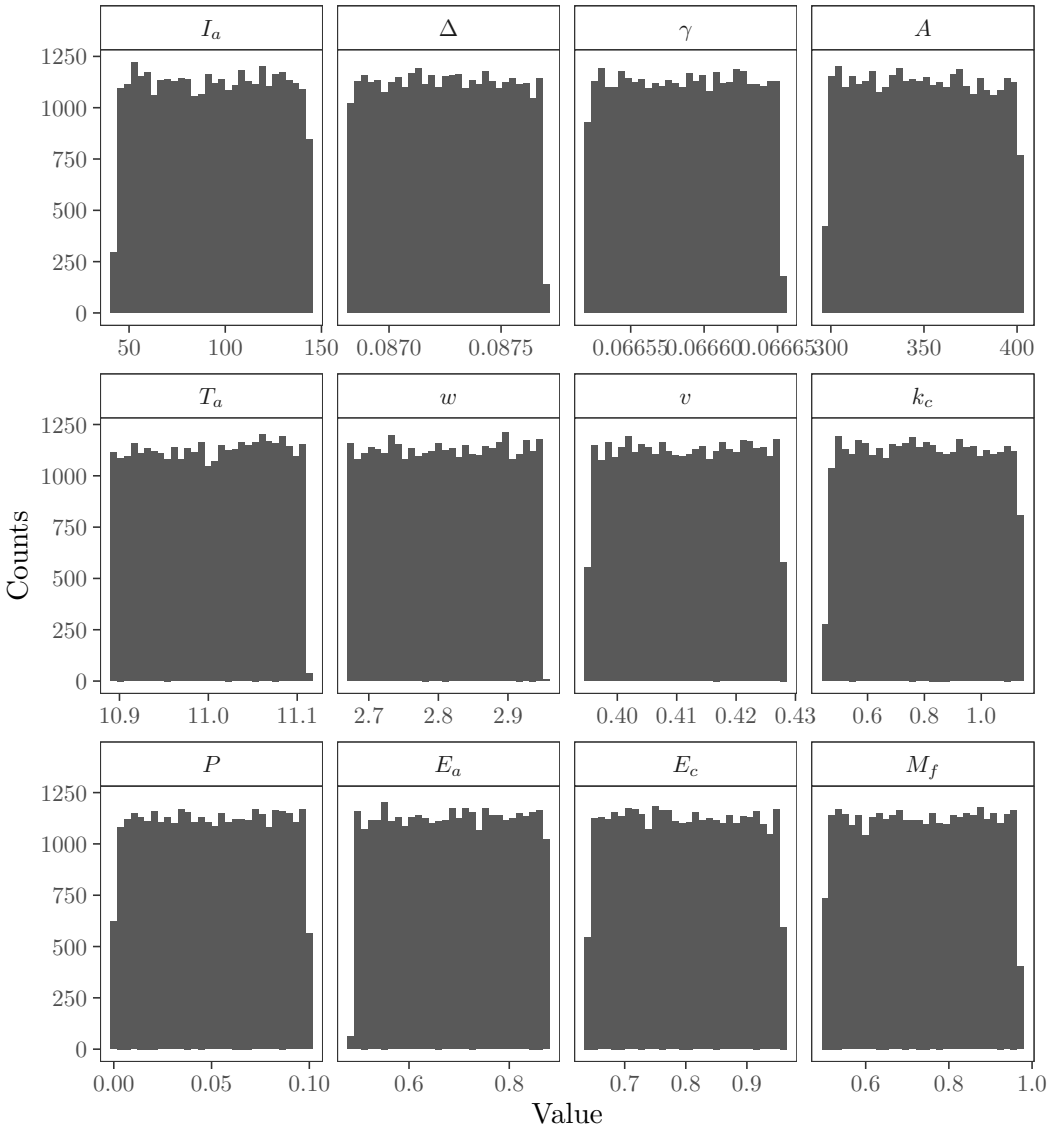
```r
N <- 2^15
R <- 10^3
type <- "R"
order <- "second"
params <- c(
  "I_a", "Delta", "gamma", "A", "T_a", "w", "v",
  "k_c", "P", "E_a", "E_c", "M_f"
)

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

# I_a (ha)
# P (mm)
# 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"], Delta + Delta * -0.005, Delta + Delta * 0.005)
mat[, "gamma"] <- qunif(mat[, "gamma"], Gamma + Gamma * -0.001, Gamma + Gamma * 0.001)
mat[, "A"] <- qunif(mat[, "A"], 350 + 350 * -0.15, 350 + 350 * 0.15)
mat[, "T_a"] <- qunif(mat[, "T_a"], T_air + T_air * -0.01, T_air + T_air * 0.01)
mat[, "w"] <- qunif(mat[, "w"], 2.81 + 2.81 * -0.05, 2.81 + 2.81 * 0.05)
mat[, "v"] <- qunif(mat[, "v"], v + v * -0.04, v + v * 0.04)
mat[, "P"] <- qunif(mat[, "P"], 0, 0.1)
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))
mat[, "E_a"] <- qunif(mat[, "E_a"], min(bos.dt[Type == "Sprinkler", `$E_a$`], na.rm = TRUE),
                      max(bos.dt[Type == "Sprinkler", `$E_a$`], na.rm = TRUE))
mat[, "E_c"] <- qunif(mat[, "E_c"], min(bos.dt[Type == "Sprinkler", `$E_c$`], na.rm = TRUE),
                      max(bos.dt[Type == "Sprinkler", `$E_c$`], na.rm = TRUE))
```

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

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




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

full_model <- function(I_a, Delta, A, gamma, T_a, w, v, k_c, P, E_a, E_c, M_f) {
  ET0 <- ((0.408 * Delta * A + gamma * (900 / (T_a + 273)) * w * v) /
            Delta + gamma * (1 + 0.34 * w))
  ETc <- ET0 * k_c
  Ep <- E_a * E_c * M_f
  y <- ((ETc - P) * I_a) / Ep
  y <- y * 10 #from mm to m3 ha
  return(y)
```

```
}
```

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

y <- full_model(
  I_a = mat[, "I_a"],
  Delta = mat[, "Delta"],
  A = mat[, "A"],
  gamma = mat[, "gamma"],
  T_a = mat[, "T_a"],
  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"]
)
```

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

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

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

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

# Plot
ind
```

```
##
## First-order estimator: saltelli | Total-order estimator: jansen
##
## Total number of model runs: 2621440
##
## Sum of first order indices: 0.9131116
##          original          bias    std.error        low.ci       high.ci
##  1:  3.544176e-01 -7.281756e-04 1.103705e-02  3.335136e-01  3.767780e-01
##  2: -3.614515e-06  3.012475e-08 2.045009e-06 -7.652783e-06  3.635046e-07
##  3:  1.585445e-07 -2.223860e-08 4.134505e-07 -6.295649e-07  9.911311e-07
##  4:  3.313991e-02  1.342477e-04 3.065415e-03  2.699756e-02  3.901376e-02
##  5:  2.154808e-07 -6.401030e-10 1.624909e-07 -1.023555e-07  5.345973e-07
##  6: -2.376462e-05 -1.806058e-07 2.045623e-05 -6.367749e-05  1.650947e-05
##  7: -8.276168e-06  8.668325e-07 1.597253e-05 -4.044858e-05  2.216258e-05
```

```
##  8:  2.220756e-01  2.612966e-04 8.953444e-03  2.042659e-01  2.393627e-01
##  9:  7.569353e-06  2.419895e-07 8.703196e-06 -9.730587e-06  2.438531e-05
## 10:  1.073846e-01 -3.808125e-04 6.642980e-03  9.474537e-02  1.207854e-01
## 11:  4.944203e-02 -3.921652e-05 4.349464e-03  4.095645e-02  5.800604e-02
## 12:  1.466797e-01 -2.401146e-04 7.465691e-03  1.322873e-01  1.615523e-01
## 13:  4.160994e-01 -1.499498e-04 4.503679e-03  4.074223e-01  4.250764e-01
## 14:  1.433228e-08  2.558451e-12 1.627064e-10  1.401082e-08  1.464862e-08
## 15:  6.195055e-10  1.403306e-13 7.520854e-12  6.046245e-10  6.341057e-10
## 16:  3.401125e-02  1.735438e-06 4.115084e-04  3.320297e-02  3.481605e-02
## 17:  8.681151e-11  9.178441e-15 1.034652e-12  8.477445e-11  8.883021e-11
## 18:  1.486186e-06 -5.311242e-10 1.694844e-08  1.453499e-06  1.519935e-06
## 19:  9.273837e-07  4.590520e-10 1.125747e-08  9.048604e-07  9.489889e-07
## 20:  2.689860e-01  1.681913e-04 2.946822e-03  2.630422e-01  2.745935e-01
## 21:  2.716351e-07  1.689191e-11 3.239144e-09  2.652696e-07  2.779668e-07
## 22:  1.315574e-01 -1.267613e-05 1.454505e-03  1.287193e-01  1.344209e-01
## 23:  6.341448e-02  1.034648e-05 7.599366e-04  6.191468e-02  6.489358e-02
## 24:  1.692473e-01 -2.525026e-05 1.878412e-03  1.655909e-01  1.729542e-01
## 25:  1.456109e-06 -8.180165e-08 8.964607e-07 -2.191198e-07  3.294942e-06
## 26:  7.301423e-09 -7.226490e-10 1.827463e-07 -3.501521e-07  3.662003e-07
## 27:  3.259285e-03  1.357750e-06 1.329674e-03  6.518137e-04  5.864041e-03
## 28: -7.715785e-09 -1.900076e-10 6.693328e-08 -1.387126e-07  1.236610e-07
## 29:  5.120918e-06  1.453927e-07 8.841413e-06 -1.235333e-05  2.230438e-05
## 30: -9.000350e-06  1.301121e-07 7.141111e-06 -2.312678e-05  4.865859e-06
## 31:  2.424498e-02 -2.093377e-04 3.785428e-03  1.703502e-02  3.187362e-02
## 32:  2.624273e-06  1.753823e-07 3.750202e-06 -4.901371e-06  9.799153e-06
## 33:  9.549262e-03  4.196903e-05 2.792851e-03  4.033407e-03  1.498118e-02
## 34:  5.608247e-03 -4.265795e-05 1.830098e-03  2.063979e-03  9.237831e-03
## 35:  1.678348e-02  7.910327e-05 3.239378e-03  1.035531e-02  2.305344e-02
## 36: -7.844614e-10 -1.686652e-11 1.646177e-09 -3.994043e-09  2.458853e-09
## 37:  2.713638e-16 -2.103253e-16 1.200449e-16  2.464054e-16  7.169728e-16
## 38:  9.304285e-11  2.028579e-11 6.659584e-10 -1.232497e-09  1.378012e-09
## 39:  1.158694e-07 -1.947743e-09 8.171689e-08 -4.234504e-08  2.779793e-07
## 40:  6.933291e-08 -3.722652e-09 6.689538e-08 -5.805699e-08  2.041681e-07
## 41: -9.806943e-07 -1.504840e-10 7.115567e-07 -2.375169e-06  4.140817e-07
## 42:  1.671273e-18  1.451078e-20 4.310611e-18 -6.791880e-18  1.010540e-17
## 43: -8.655569e-07  7.782165e-11 5.161445e-07 -1.877259e-06  1.459899e-07
## 44: -4.665598e-07 -2.316364e-08 3.357875e-07 -1.101528e-06  2.147354e-07
## 45: -4.198974e-07 -1.669679e-09 5.699167e-07 -1.535244e-06  6.987884e-07
## 46:  1.638423e-16 -1.188698e-16 1.177172e-16  5.199071e-17  5.134336e-16
## 47: -2.868726e-10 -2.365478e-12 1.358967e-10 -5.508597e-10 -1.815452e-11
## 48:  1.579299e-08  3.098371e-10 1.618392e-08 -1.623675e-08  4.720305e-08
## 49:  2.015701e-08 -3.238599e-10 1.358091e-08 -6.137224e-09  4.709897e-08
## 50: -8.530419e-08 -4.562976e-10 1.419296e-07 -3.630249e-07  1.933291e-07
## 51:  7.162600e-18 -1.182077e-19 4.345819e-18 -1.236842e-18  1.579846e-17
## 52: -1.240116e-07 -2.221764e-09 1.024143e-07 -3.225181e-07  7.893839e-08
## 53:  2.229044e-08  3.188571e-09 7.289073e-08 -1.237613e-07  1.619651e-07
## 54: -5.503096e-08  9.667199e-10 1.165053e-07 -2.843438e-07  1.723485e-07
## 55:  2.966412e-16 -2.439266e-16 1.176094e-16  3.100576e-16  7.710781e-16
```

```
## 56:  1.121216e-16 -1.186251e-16 1.178913e-16 -3.159311e-19  4.618093e-16
## 57:  3.047367e-16 -2.730914e-16 1.202753e-16  3.420929e-16  8.135634e-16
## 58:  1.750166e-03 -2.214806e-05 9.917764e-04 -1.715319e-04  3.716160e-03
## 59:  1.256180e-16 -1.417311e-16 1.178335e-16  3.639968e-17  4.982984e-16
## 60:  5.676334e-04 -8.442068e-06 7.671372e-04 -9.274858e-04  2.079637e-03
## 61:  5.062906e-04 -2.336455e-05 5.125165e-04 -4.748588e-04  1.534169e-03
## 62:  1.513909e-03 -1.532364e-05 9.073650e-04 -2.491705e-04  3.307635e-03
## 63: -6.232468e-09  1.875108e-10 6.435796e-09 -1.903391e-08  6.193950e-09
## 64:  8.092168e-09 -1.890155e-10 5.273713e-09 -2.055105e-09  1.861747e-08
## 65:  1.163205e-08  7.139048e-10 5.150772e-08 -9.003512e-08  1.118714e-07
## 66:  3.003792e-18  2.049432e-19 4.377487e-18 -5.780868e-18  1.137857e-17
## 67: -9.445328e-09  1.328459e-09 4.048871e-08 -9.013020e-08  6.858263e-08
## 68: -9.577503e-09  3.601597e-10 2.748756e-08 -6.381229e-08  4.393697e-08
## 69:  6.020935e-08 -2.113359e-09 4.475161e-08 -2.538884e-08  1.500343e-07
## 70:  2.730528e-07  1.158044e-08 6.621227e-07 -1.036264e-06  1.559209e-06
## 71:  3.034902e-06  2.901783e-07 7.017885e-06 -1.101008e-05  1.649952e-05
## 72:  3.208795e-18  2.557369e-19 4.490192e-18 -5.847556e-18  1.175367e-17
## 73: -1.625197e-06  3.167848e-07 5.088281e-06 -1.191483e-05  8.030867e-06
## 74:  5.161398e-07  1.711210e-07 3.385305e-06 -6.290058e-06  6.980095e-06
## 75:  3.366222e-06  2.631217e-07 5.823086e-06 -8.309938e-06  1.451614e-05
## 76: -1.026660e-05 -1.882317e-07 5.735381e-06 -2.131950e-05  1.162777e-06
## 77:  3.253795e-18 -2.783780e-20 4.508109e-18 -5.554098e-18  1.211736e-17
## 78: -6.493624e-06 -9.007550e-09 4.075091e-06 -1.447165e-05  1.502415e-06
## 79: -4.090090e-06 -7.409970e-08 2.772332e-06 -9.449660e-06  1.417680e-06
## 80: -1.423225e-06  3.511761e-08 4.669938e-06 -1.061125e-05  7.694567e-06
## 81:  1.180353e-15 -1.310536e-15 8.048647e-16  9.133832e-16  4.068395e-15
## 82:  5.684312e-03  1.248826e-04 2.178376e-03  1.289890e-03  9.828968e-03
## 83:  3.071927e-03 -1.207678e-04 1.495981e-03  2.606264e-04  6.124762e-03
## 84:  8.043182e-03  5.550981e-05 2.491297e-03  3.104820e-03  1.287052e-02
## 85:  2.537731e-06  1.512434e-07 2.311288e-06 -2.143553e-06  6.916529e-06
## 86: -1.028631e-06 -6.897230e-08 1.545243e-06 -3.988278e-06  2.068962e-06
## 87:  3.727155e-06  5.057716e-08 2.497009e-06 -1.217470e-06  8.570626e-06
## 88:  6.908616e-04  7.310014e-05 1.051053e-03 -1.442265e-03  2.677788e-03
## 89:  4.355550e-03  1.942903e-06 1.803962e-03  8.179071e-04  7.889308e-03
## 90:  1.932074e-03 -1.461825e-05 1.281272e-03 -5.645556e-04  4.457940e-03
##          original          bias    std.error        low.ci       high.ci
##     sensitivity          parameters
##  1:          Si              $I_a$
##  2:          Si           $\\Delta$
##  3:          Si           $\\gamma$
##  4:          Si                 $A$
##  5:          Si              $T_a$
##  6:          Si                $w$
##  7:          Si                $v$
##  8:          Si              $k_c$
##  9:          Si                $P$
## 10:          Si              $E_a$
## 11:          Si              $E_c$
```

```
## 12:          Si               $M_f$
## 13:          Ti               $I_a$
## 14:          Ti           $\\Delta$
## 15:          Ti           $\\gamma$
## 16:          Ti                 $A$
## 17:          Ti               $T_a$
## 18:          Ti                 $w$
## 19:          Ti                 $v$
## 20:          Ti               $k_c$
## 21:          Ti                 $P$
## 22:          Ti               $E_a$
## 23:          Ti               $E_c$
## 24:          Ti               $M_f$
## 25:         Sij     $I_a$.$\\Delta$
## 26:         Sij     $I_a$.$\\gamma$
## 27:         Sij           $I_a$.$A$
## 28:         Sij         $I_a$.$T_a$
## 29:         Sij           $I_a$.$w$
## 30:         Sij           $I_a$.$v$
## 31:         Sij         $I_a$.$k_c$
## 32:         Sij           $I_a$.$P$
## 33:         Sij         $I_a$.$E_a$
## 34:         Sij         $I_a$.$E_c$
## 35:         Sij         $I_a$.$M_f$
## 36:         Sij $\\Delta$.$\\gamma$
## 37:         Sij       $\\Delta$.$A$
## 38:         Sij     $\\Delta$.$T_a$
## 39:         Sij       $\\Delta$.$w$
## 40:         Sij       $\\Delta$.$v$
## 41:         Sij     $\\Delta$.$k_c$
## 42:         Sij       $\\Delta$.$P$
## 43:         Sij     $\\Delta$.$E_a$
## 44:         Sij     $\\Delta$.$E_c$
## 45:         Sij     $\\Delta$.$M_f$
## 46:         Sij       $\\gamma$.$A$
## 47:         Sij     $\\gamma$.$T_a$
## 48:         Sij       $\\gamma$.$w$
## 49:         Sij       $\\gamma$.$v$
## 50:         Sij     $\\gamma$.$k_c$
## 51:         Sij       $\\gamma$.$P$
## 52:         Sij     $\\gamma$.$E_a$
## 53:         Sij     $\\gamma$.$E_c$
## 54:         Sij     $\\gamma$.$M_f$
## 55:         Sij           $A$.$T_a$
## 56:         Sij             $A$.$w$
## 57:         Sij             $A$.$v$
## 58:         Sij           $A$.$k_c$
## 59:         Sij             $A$.$P$
```

```
## 60:         Sij           $A$.$E_a$
## 61:         Sij           $A$.$E_c$
## 62:         Sij           $A$.$M_f$
## 63:         Sij           $T_a$.$w$
## 64:         Sij           $T_a$.$v$
## 65:         Sij         $T_a$.$k_c$
## 66:         Sij           $T_a$.$P$
## 67:         Sij         $T_a$.$E_a$
## 68:         Sij         $T_a$.$E_c$
## 69:         Sij         $T_a$.$M_f$
## 70:         Sij             $w$.$v$
## 71:         Sij           $w$.$k_c$
## 72:         Sij             $w$.$P$
## 73:         Sij           $w$.$E_a$
## 74:         Sij           $w$.$E_c$
## 75:         Sij           $w$.$M_f$
## 76:         Sij           $v$.$k_c$
## 77:         Sij             $v$.$P$
## 78:         Sij           $v$.$E_a$
## 79:         Sij           $v$.$E_c$
## 80:         Sij           $v$.$M_f$
## 81:         Sij           $k_c$.$P$
## 82:         Sij         $k_c$.$E_a$
## 83:         Sij         $k_c$.$E_c$
## 84:         Sij         $k_c$.$M_f$
## 85:         Sij           $P$.$E_a$
## 86:         Sij           $P$.$E_c$
## 87:         Sij           $P$.$M_f$
## 88:         Sij         $E_a$.$E_c$
## 89:         Sij         $E_a$.$M_f$
## 90:         Sij         $E_c$.$M_f$
##     sensitivity          parameters
```

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

```
## [1] 1.000659
```

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

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

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

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

5.6 One-at-a-time (OAT)

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

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

X <- B
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
```

```
# 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_a = mat.oat[, "T_a"],
  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"]
)
```

```
# 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_a = vec_means[["T_a"]],
  w = vec_means[["w"]],
  v = vec_means[["v"]],
  k_c = vec_means[["k_c"]],
  P = vec_means[["P"]],
```

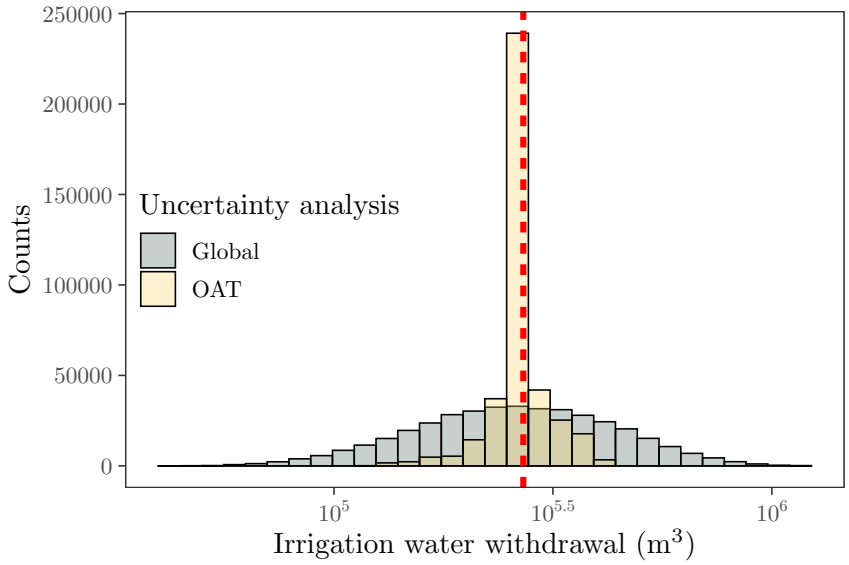
```
  E_a = vec_means[["E_a"]],
  E_c = vec_means[["E_c"]],
  M_f = vec_means[["M_f"]]
)

y.point
```

```
## [1] 270644.4
```

5.7 Compare OAT and global sensitivity analysis

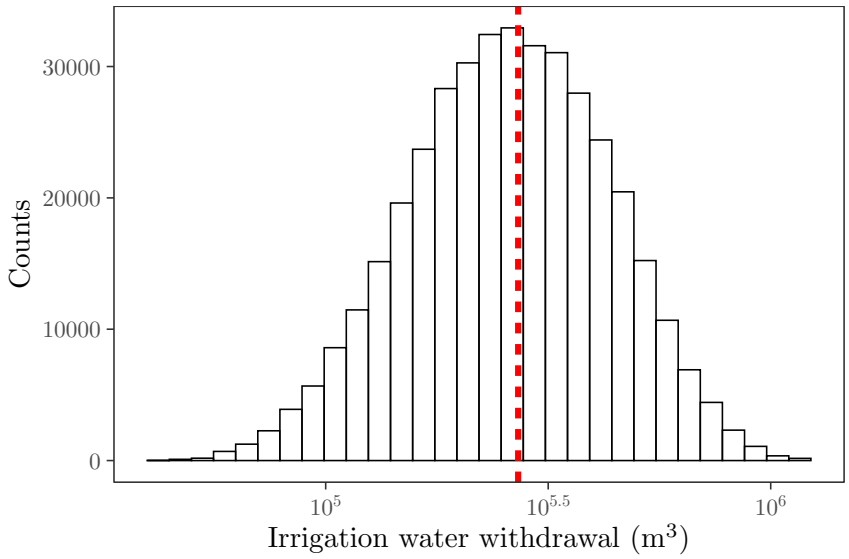
```
# ASSESS UNCERTAINTIES ----------------------------------------------------

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 = "Irrigation water withdrawal (m$^3$)", 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))

a
```

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




```
d <- melt(full.unc, measure.vars = colnames(full.unc)) %>%
  .[variable == "Global"] %>%
  ggplot(., aes(value)) +
  geom_histogram(position = "identity", alpha = 0.3, color = "black", fill = "white") +
  labs(x = "Irrigation water withdrawal (m$^3$)", 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))

d
```

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




```
all.projection <- d # For later
```

```
fwrite(full.unc, "full.unc.csv")
```

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

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

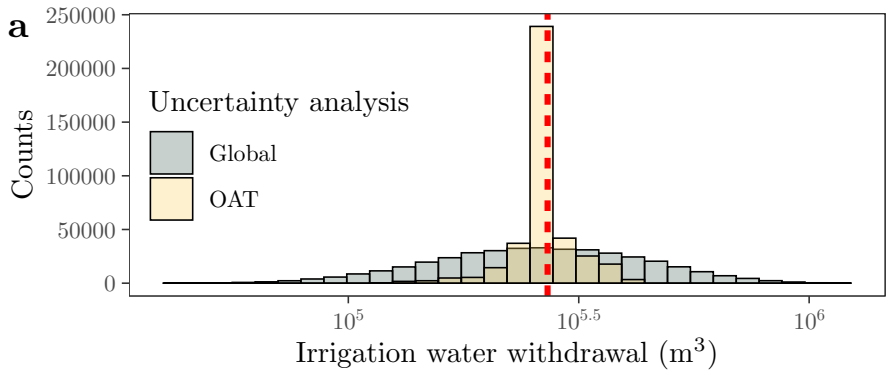
```
##    variable       min       max
## 1:   Global  44008.69 1217141.9
## 2:      OAT 123984.93  417447.5
```

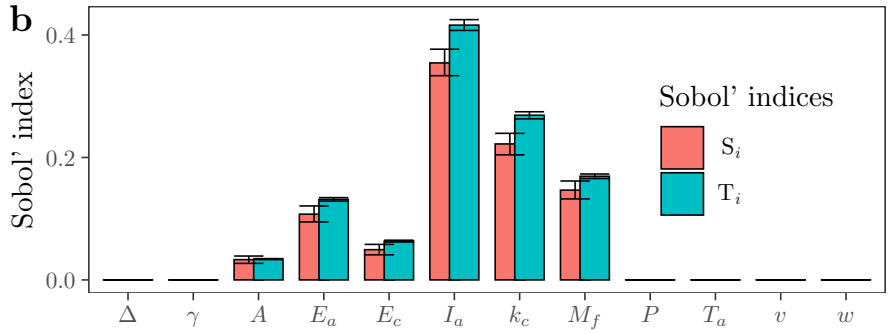
```
# Quantiles
probs.quantile <- c(0, 0.025, 0.1, 0.5, 0.9, 0.975, 1)
stat.full.unc[, .(value = quantile(value, probs = probs.quantile)), variable] %>%
  .[, quantile := rep(probs.quantile, 2)] %>%
  dcast(., variable ~ quantile, value.var = "value") %>%
  print()
```

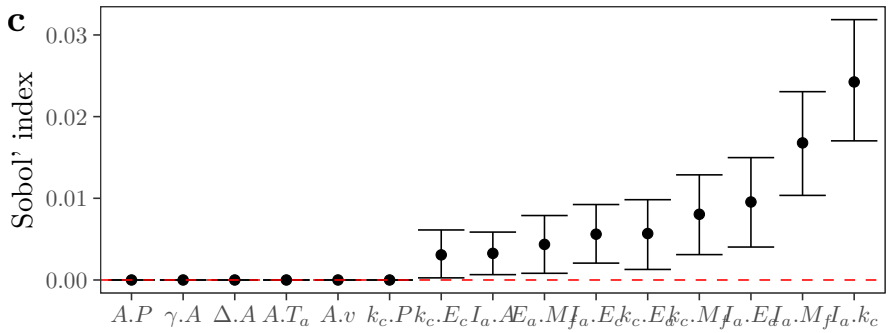
```
##    variable         0     0.025      0.1      0.5      0.9    0.975         1
## 1:   Global  44008.69  91426.65 130191.7 259591.6 498891.2 673199.0 1217141.9
## 2:      OAT 123984.93 179659.20 231011.4 270644.4 320866.5 373187.1  417447.5
```

```
# 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`.
```

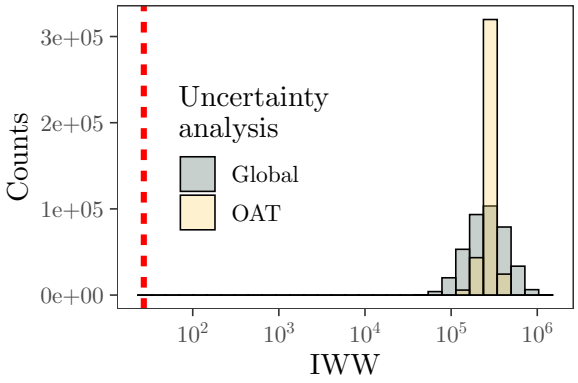







```
melt(full.unc, measure.vars = colnames(full.unc)) %>%
  ggplot(., aes(value, fill = variable)) +
  geom_histogram(position = "identity", alpha = 0.3, color = "black") +
  labs(x = "IWW",
       y = "Counts") +
  scale_x_log10(breaks = trans_breaks("log10",function(x) 10^x),
                labels = trans_format("log10", scales::math_format(10^.x))) +
  geom_vline(xintercept = 27.064, lty = 2, color = "red", size = 2) +
  scale_fill_manual(values = wes_palette(2, name = "Chevalier1"), name = "Uncertainty \n analys
  theme_AP() +
  theme(legend.position = c(0.3, 0.5))
```

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




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

legend <- get_legend(plots.unc[[2]] + theme(legend.position = "top"))
```

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

```
## Warning: Removed 4 rows containing missing values (geom_bar).
```

```
bottom <- plot_grid(etc + theme(legend.position = "none"), ep, labels = "auto")
```

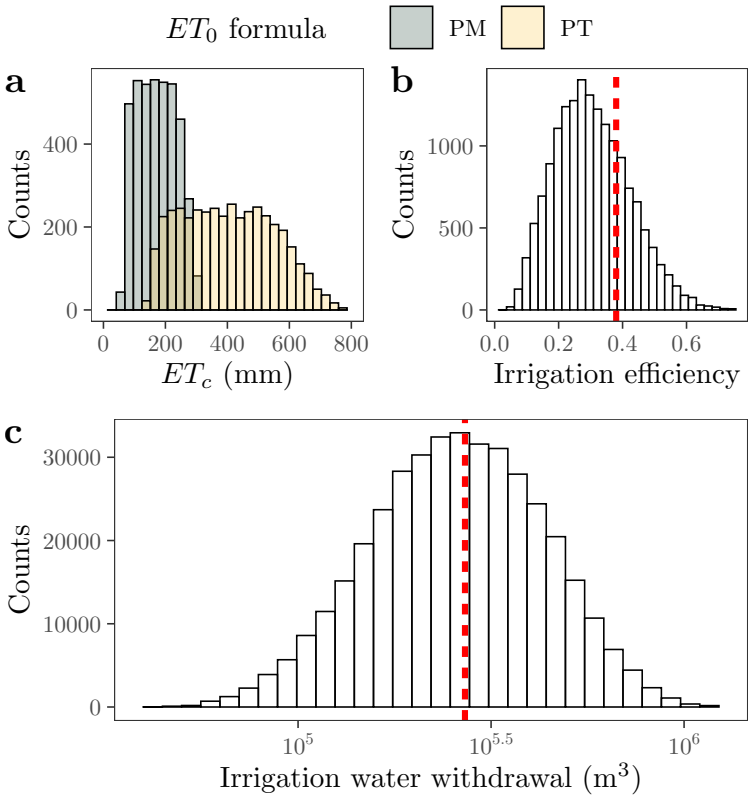
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 4 rows containing missing values (geom_bar).
```

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

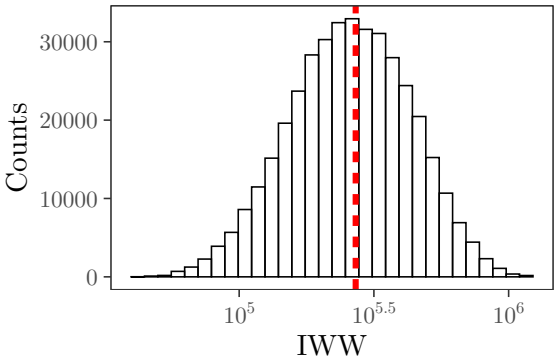
```
top <- plot_grid(legend, bottom, rel_heights = c(0.13, 0.87), ncol = 1)
plot_grid(top, all.projection, labels = c("", "c"), ncol = 1)
```

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




```
d + labs(x = "IWW", y = "Counts")
```

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



References

Allen, Richard G., Luis S. Pereira, D. Raes, and M. Smith. 1998. *Crop evapotranspiration (guidelines for computing crop water requirements)*. Vol. 56. Rome: FAO. https://doi.org/10.1016/j.eja.2010.12.001.

———. 1998. *Crop evapotranspiration (guidelines for computing crop water requirements)*. Vol. 56. Rome: FAO. https://doi.org/10.1016/j.eja.2010.12.001.

Bos, M. G., and J Nugteren. 1990. "On Irrigation Efficiencies." Wageningen: International Institute for Land Recamation; Improvement / ILRI. http://www1.frm.utn.edu.ar/laboratorio_hidraulic

a/Biblioteca_Virtual/Publicaciones%20de%20ILRI/On%20irrigation%20efficiencies.pdf.

Döll, P., and S. Siebert. 2002. "Global modeling of irrigation water requirements." *Water Resources Research* 38 (4): 8-1-8-10. https://doi.org/10.1029/2001WR000355.

———. 2002. "Global modeling of irrigation water requirements." *Water Resources Research* 38 (4): 8-1-8-10. https://doi.org/10.1029/2001WR000355.

FAO. 1997. "Irrigation potential in Africa. A basin approach. FAO Land and Water Bulletin 4." Rome: Food; Agriculture Organization of the United Nations. http://library1.nida.ac.th/termpaper6/sd/2554/19755.pdf.

———. 2016. "AQUASTAT website." Food; Agriculture Organization of the United Nations. http://www.fao.org/nr/water/aquastat/didyouknow/index3.stm.

———. 2017. "FAOSTAT database." Rome. http://www.fao.org/faostat/en/.

Meier, J., F. Zabel, and W. Mauser. 2018. "A global approach to estimate irrigated areas. A comparison between different data and statistics." *Hydrology and Earth System Sciences* 22 (2): 1119–33. https://doi.org/10.5194/hess-22-1119-2018.

Nichols, J., W. Eichinger, D. I. Cooper, J. H. Prueger, L. E. Hipps, C. M. U. Neale, and A. S. Bawazir. 2004. "Comparison of Evaporation Estimation Methods for a Riparian Area. Final report." 436. Vol. 436. Iowa City: IIHR - Hydroscience & Engineering, College of Engineering, University of Iowa.

Rohwer, Janine, Dieter Gerten, and Wolfgang Lucht. 2007. "Development of functional irrigation types for improved global crop modelling." *PIK Report*, no. 104: 1–61.

Salmon, J. M., M. A. Friedl, S. Frolking, D. Wisser, and E. M. Douglas. 2015. "Global rain-fed, irrigated, and paddy croplands: A new high resolution map derived from remote sensing, crop inventories and climate data." *International Journal of Applied Earth Observation and Geoinformation* 38: 321–34. https://doi.org/10.1016/j.jag.2015.01.014.

Siebert, S., V. Henrich, K. Frenken, and J. Burke. 2013. "Update of the digital global map of irrigation areas to version 5." Bonn, Rome: University of Bonn/FAO. https://doi.org/10.13140/2.1.2660.6728.

Solley, Wayne B., Robert R. Pierce, and How Perlman. 1998. "Estimated Use of Water in the United States in 1995." US Geological Survey Circular 1200. http://www.usgs.gov/default.asp.

Thenkabail, P. S., C. M. Biradar, P. Noojipady, V. Dheeravath, Y. Li, M. Velpuri, M. Gumma, et al. 2009. "Global irrigated area map (GIAM), derived from remote sensing, for the end of the last millennium." *International Journal of Remote Sensing* 30 (14): 3679–3733. https://doi.org/10.1080/01431160802698919.