What is the relation between irrigated areas and irrigation water withdrawal?

Arnald Puy et al.

Contents

1 Introduction

Current figures suggest that irrigated agriculture consumes c. 70% of all freshwater resources. The total amount of water used by irrigation agriculture is determined by factors such as soil hydraulic parameters, crop types, the weather or the irrigated area (Wisser et al. 2008), with the latter being especially influential (Puy, Muneepeerakul, and Balbo 2017). It is thus relevant to empirically describe the relationship between the irrigated area and the volume of water required for irrigation: what happens when we increase the total extension of irrigation? does the demand for irrigation water increase proportionally, or in a non-linear fashion? Are small irrigated areas more water efficient, or do they disproportionally require more water than large ones?

2 Materials and methods

Here we aim at tackling these questions. Let us first create a wrapper function that allows to load all the required `R` libraries in one go. We then load the package **`checkpoint`** (Microsoft Corporation 2018), which installs in a local directory the same package versions used in the study. This allows anyone that runs our code to fully reproduce our results anytime. Finally, we cast a function to define the theme of the figures we will plot to present our results.

```
# LOAD PACKAGES ------------------------------------------------------------

# Function to read in all required packages in one go:
loadPackages <- function(x) {
  for(i in x) {
    if(!require(i, character.only = TRUE)) {
      install.packages(i, dependencies = TRUE)
      library(i, character.only = TRUE)
    }
  }
}

loadPackages(c("data.table", "ggplot2", "sensobol", "scales",
               "ncdf4", "rworldmap", "sp", "countrycode",
               "dplyr", "IDPmisc", "boot", "parallel",
               "MASS", "doParallel", "complmrob",
               "mvoutlier", "sandwich", "lmtest", "mice"))

# SET CHECKPOINT -----------------------------------------------------------

dir.create(".checkpoint")

library("checkpoint")

checkpoint("2019-09-10",
           R.version ="3.6.1",
           checkpointLocation = getwd())

# CUSTOM FUNCTION TO DEFINE THE PLOT THEMES --------------------------------
```

```
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))
}
```

Since we will need to read in several global datasets and clean the data before conducting any test, we define here several functions to that aim. Firstly, the function `country_code` will ensure that all countries receive the same name across datasets. Secondly, the `coords2country` function will translate coordinatesinto the country name. Finally, the `get_nc` function will allow to read in `.nc` files, a format widely used to store spatially explicit data on irrigation water withdrawal.

```
# CREATE FUNCTIONS ------------------------------------------------------------

# Function to obtain UN code, Continent and Country names
country_code <- function(dt) {
  dt[, `:=` (Code = countrycode(dt[, Country],
                                origin = "country.name",
                                destination = "un"),
             Continent = countrycode(dt[, Country],
                                     origin = "country.name",
                                     destination = "continent"))]
  dt[, Country:= countrycode(dt[, Code],
                             origin = "un",
                             destination = "country.name")]
  setcolorder(dt, c("Country", "Continent", "Code", "Water.Withdrawn"))
  return(dt)
}

## Function to transform longitude and latitude to country.
# It is borrowed from Andy:
# https://stackoverflow.com/questions/14334970/convert-latitude-and-longitude-coordinates-to-c
coords2country = function(points) {
  countriesSP <- rworldmap::getMap(resolution = 'low')
  pointsSP = sp::SpatialPoints(points, proj4string=CRS(proj4string(countriesSP)))
  indices = sp::over(pointsSP, countriesSP)
  indices$ADMIN
  #indices$ISO3 # returns the ISO3 code
  #indices$continent   # returns the continent (6 continent model)
  #indices$REGION   # returns the continent (7 continent model)
}

# Function to load and extract data from .nc files
```

```
get_nc_data <- function(nc_file) {
  nc <- nc_open(nc_file)
  ww <- ncvar_get(nc, "withd_irr")
  lon <- ncvar_get(nc, "lon")
  lat <- ncvar_get(nc, "lat")
  water <- rowSums(ww[, 469:ncol(ww)]) # Obtain year values for 2010 only
  ww.df <- data.frame(cbind(lon, lat, water))
  countries <- coords2country(ww.df[1:nrow(ww.df), 1:2])
  df <- cbind(countries, ww.df)
  setDT(df)
  final <- df[, .(Water.Withdrawn = sum(water)), countries]
  setnames(final, "countries", "Country")
  country_code(final)
  out <- na.omit(final[order(Continent)])
  out[, Water.Withdrawn:= Water.Withdrawn / 1000] # From mm to m
  return(out)
}
```

2.1 Irrigation water withdrawal datasets

There are several datasets and Global Hydrological Models (GHM) providing information on irrigation water withdrawal at the country level, with significant differences on the values reported. Here we consider this source of uncertainty through the following datasets:

1. The WaterGap GHM [@Doll2002].
2. The LPJmL GHM.
3. The H08 GHM.
4. The PCR-GLOBWB GHM.
5. FAOSTAT irrigation water withdrawal (Table 4).
6. @Liu2016a dataset (Aquastat dataset with all the missing values filled).

In the next code chunk we read in all these datasets, clean them and merge them to obtain a final dataset with all the data on irrigation water withdrawal merged.

```
# READ IN DATASETS ON IRRIGATION WATER WITHDRAWAL ----------------------------

# FAO data (Table 4) ---------------------------
# http://www.fao.org/nr/water/aquastat/water_use_agr/IrrigationWaterUse.pdf

table4.tmp <- fread("table_4.csv", skip = 3, nrows = 167) %>%
  .[, .(Country, Year, Water.withdrawal)] %>%
  setnames(., "Water.withdrawal", "Water.Withdrawn")

# Extract the selected years
table4.dt <- country_code(table4.tmp[Year %in% 1999:2012])[
  , Water.Dataset:= "Table 4"][
```

```
    , Year:= NULL]

# Liu et al. dataset ----------------------------
liu.dt <- fread("liu.csv")[, .(country, irr)] %>%
  setnames(., c("country", "irr"), c("Country", "Water.Withdrawn")) %>%
  country_code(.) %>%
  .[, Water.Dataset:= "Liu et al. 2016"]

# Huang et al datasets --------------------------
names_nc_files <- c("withd_irr_lpjml.nc", "withd_irr_pcrglobwb.nc",
                    "withd_irr_h08.nc", "withd_irr_watergap.nc")
out.nc <- lapply(names_nc_files, function(x) get_nc_data(x))
names(out.nc) <- c("LPJmL", "PCR-GLOBWB", "H08", "WaterGap")

GHM.dt <- rbindlist(out.nc, idcol = "Water.Dataset") %>%
  .[order(Country)]
```

Not all irrigation water withdrawal datasets provide measurements for the same countries. This means that any computation conducted using a given dataset risks being biased by the specific countries included in it. In order to tackle this source of uncertainty, all datasets should include the same countries, regardless of whether there is a measurement available for the country in question. Later on the uncertainties related with the imputation of missing values will be dealt with. For now, we extract a vector with the name of all the different countries mentioned by any of the irrigated water withdrawal datasets, and merge it with each single water withdrawal dataset - missing values will be for the moment treated as `NA`.

```
# ARRANGE THE TOTAL NUMBER OF COUNTRIES ---------------------------------------

# Check how many different countries there are in the water datasets
DT <- data.table(unique(c(liu.dt[, Country], GHM.dt[, Country], table4.dt[, Country])))
setnames(DT, "V1", "Country")

# Give standard country names, UN codes and link with Continent
DT <- DT[, `:=` (Code = countrycode(DT[, Country],
                                    origin = "country.name",
                                    destination = "un"),
                 Continent = countrycode(DT[, Country],
                                         origin = "country.name",
                                         destination ="continent"))]

DT <- DT[, Country:= countrycode(DT[, Code],
                                 origin = "un",
                                 destination = "country.name")]
```

```
# CREATE THE FINAL IRRIGATION WATER WITHDRAWAL DATASET ------------------------

# Merge with the Country vector
tmp <- GHM.dt[, merge(.SD, DT, by = c("Country", "Code", "Continent"),
```

```
                      all.y = TRUE), by = Water.Dataset]

# Check whether there are duplicated Countries
tmp[tmp[, duplicated(Country), Water.Dataset][, V1]]
```

```
##     Water.Dataset                 Country Code Continent Water.Withdrawn
##  1:         LPJmL                  Cyprus  196      Asia      0.02269412
##  2:         LPJmL Palestinian Territories  275      Asia      0.10018817
##  3:         LPJmL                 Somalia  706    Africa      0.19308032
##  4:    PCR-GLOBWB                  Cyprus  196      Asia      0.02436695
##  5:    PCR-GLOBWB Palestinian Territories  275      Asia      0.10580178
##  6:    PCR-GLOBWB                 Somalia  706    Africa      0.20744558
##  7:           H08                  Cyprus  196      Asia      0.01944480
##  8:           H08 Palestinian Territories  275      Asia      0.06340638
##  9:           H08                 Somalia  706    Africa      0.20371014
## 10:      WaterGap                  Cyprus  196      Asia      0.01020237
## 11:      WaterGap Palestinian Territories  275      Asia      0.08438534
## 12:      WaterGap                 Somalia  706    Africa      0.18559815
```

```
# Get mean values for the duplicated Countries
GHM.dt.full <- tmp[, .(Water.Withdrawn = mean(Water.Withdrawn)),
                   .(Water.Dataset, Country, Code, Continent)]

# Arrange Liu data set
liu.dt.full <- merge(DT, liu.dt,
                     by = c("Country", "Code", "Continent"),
                     all.x = TRUE) %>%
  .[, Water.Dataset:= ifelse(is.na(Water.Dataset),
                             "Liu et al. 2016",
                             Water.Dataset)]

# Arrange Table 4 dataset
table4.dt.full <- merge(DT, table4.dt,
                        by = c("Country", "Code", "Continent"),
                        all.x = TRUE) %>%
  .[, Water.Dataset:= ifelse(is.na(Water.Dataset),
                             "Table 4",
                             Water.Dataset)]

# Obtain final irrigation water withdrawal dataset
water.dt <- rbind(liu.dt.full, table4.dt.full, GHM.dt.full)
```

2.2 Irrigated area datasets

There are also several datasets informing on the extension of irrigation at the country level, with large uncertainties (sometimes the values differ for more than one order of magnitude). Here we use the data compiled by Meier, Zabel, and Mauser (2018), which also includes the datasets by Aquastat (FAO 2016), FAOSTAT (FAO 2017), Siebert et al. (2013), Salmon et al. (2015) and

Thenkabail et al. (2009).

```
# READ IN IRRIGATED AREA DATASETS ------------------------------------------

meier.dt <- fread("meier.csv") %>%
  setnames(., "Codes", "Code")
```

Finally, we bind the datasets on irrigation water withdrawal and irrigated area, and obtain the full dataset we will use in our analysis. We also plot the data, which evidences that irrigation water withdrawal and irrigated area are indeed related (Fig. 1). The last code chunk log-transforms these parameters to ease the interpretation of the results and prepare the dataset for the upcoming analysis.

```
# MERGE DATASETS -----------------------------------------------------------

irrigated.area.datasets <- colnames(meier.dt)[-c(1:3)]

irrigated.dt <- melt(meier.dt, measure.vars = irrigated.area.datasets) %>%
  .[, merge(.SD, DT, by = c("Country", "Code", "Continent"),
            all.y = TRUE), by = variable] %>%
  setnames(., c("variable", "value"), c("Area.Dataset", "Irrigated.Area"))

full.dt <- merge(irrigated.dt, water.dt, on = c("Continent", "Country", "Code"),
                 allow.cartesian = TRUE) %>%
  .[!Continent == "Oceania"] # Drop Oceania
```

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

full.dt %>%
  ggplot(., aes(Irrigated.Area, Water.Withdrawn,
                color = Continent)) +
  geom_point(size = 0.8) +
  scale_x_log10(breaks = trans_breaks("log10", function(x) 10 ^ (2 * x)),
                labels = trans_format("log10", math_format(10 ^ .x))) +
  scale_y_log10(breaks = trans_breaks("log10", function(x) 10 ^ x),
                labels = trans_format("log10", math_format(10 ^ .x))) +
  labs(x = "Irrigated area (Mha)",
       y = expression(paste("Water withdrawal", "", 10^9, m^3/year))) +
  facet_grid(Water.Dataset ~ Area.Dataset) +
  theme_AP() +
  theme(legend.position = "top")
```

```
## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 1735 rows containing missing values (geom_point).
```

```
# TRANSFORM DATASET --------------------------------------------------------

cols <- c("Water.Withdrawn", "Irrigated.Area")
```

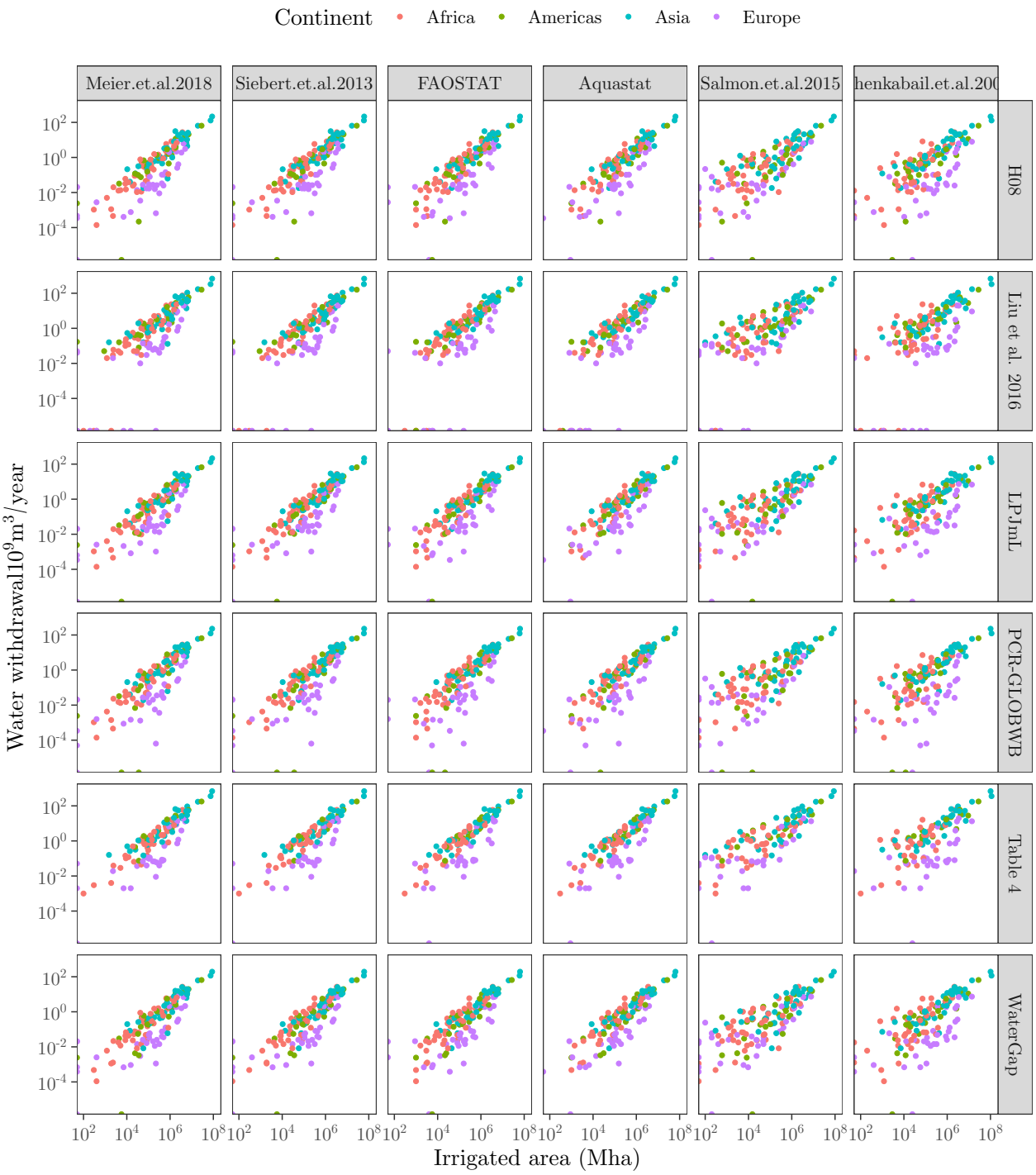


Figure 1: Scatter plots of irrigated areas (x-axis) against irrigation water withdrawal (y-axis). All the possible combinations of datasets are shown.

```
col_names <- c("Continent", "Water.Dataset", "Area.Dataset", "Regression",
               "Imputation.Method", "Iteration")
cols_group <- c("Continent", "Area.Dataset", "Water.Dataset")
full.dt <- full.dt[, (cols):= lapply(.SD, log10), .SDcols = (cols)]
```

2.3 Assesment of uncertainties in the model structure

As shown in Fig. 1, there is a clear relation between irrigated areas and irrigation water withdrawal, more or less sharp depending on which combination of datasets are used. This relationship is prone to be described by OLS regressions, and the resulting slope be used to ascertain whether irrigation water withdrawal tends to scale superlinearly ($\beta > 1$) or sublinearly ($\beta < 1$) for every increase in the area under irrigation. However, besides the uncertainty derived from the different measurements of irrigated area and irrigation water withdrawal available, there is uncertainty with regards to the selected model structure for OLS: robust/non-robust approaches to correct for outliers, whether a correction for heteroskedasticity is needed, which methodology is used to imput missing values, and uncertainty with regards to the true value of β.

Hereafter we explore the first three sources of uncertainty just mentioned.

2.3.1 Multivariate outliers

We first check the existence of bivariate outliers using Mahalanobis classic and robust distances (Filzmoser, Garrett, and Reimann 2005). The results, which are presented in Fig. 2, confirm the presence of extreme values and justify the use of a trigger to decide whether to use robust or non-robust regression methods.

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

ggplot(tmp, aes(md.cla, md.rob,
                shape = outliers,
                color = Continent)) +
  geom_point() +
  scale_shape_manual(name = "Outlier",
                     labels = c("No", "Yes"),
                     values = c(16, 4)) +
  facet_grid(Water.Dataset ~ Area.Dataset) +
  labs(x = "Mahalanobis distance",
       y = "Robust distance") +
  theme_AP() +
  theme(legend.position = "top")
```

2.3.2 Heteroskedasticity

In order to see whether the data exhibits heteroskedasticity, we compute OLS regressions for each combination of irrigated area and water withdrawal dataset, and plot the residual against the fitted values. The results are plotted in Figs 3-6. As can be seen, the variance is not constant for some specific combinations of datasets. This justifies including the correction for heteroskedasticity as a source of structural uncertainty in the computation of the regression equations.

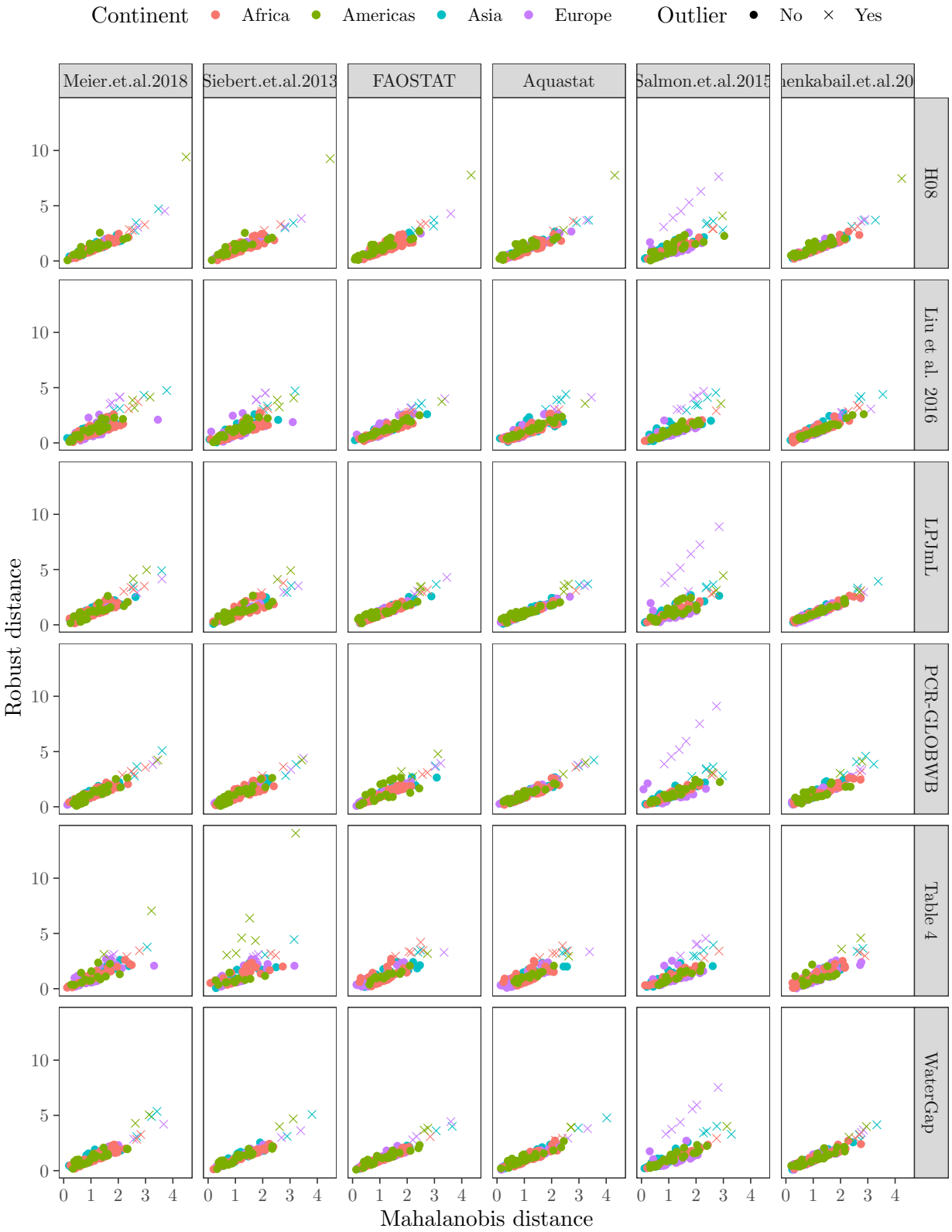


Figure 2: Bivariate outliers.

```r
# CHECK HETEROSKEDASTICITY ----------------------------------------------------

hetero <- NaRV.omit(full.dt)[, .(model = .(lm(Water.Withdrawn ~ Irrigated.Area))), cols_group]
```

```r
# PLOT HETEROSKEDASTICITY -----------------------------------------------------

tmp <- hetero[, c("resid" = lapply(model, residuals),
                  "pred" = lapply(model, fitted)), cols_group]

hetero.plot <- split(tmp, tmp$Continent)

gg <- list()
for(i in names(hetero.plot)) {
  gg[[i]] <- ggplot(hetero.plot[[i]], aes(pred, resid)) +
    geom_point() +
    facet_grid(Area.Dataset ~ Water.Dataset) +
    theme_AP() +
    labs(x = "Fitted",
         y = "Residual") +
    geom_hline(yintercept = 0,
               lty = 2) +
    ggtitle(names(hetero.plot[i]))
}

gg
```

```
## $Africa
```

```
##
## $Americas
```

```
##
## $Asia
```

```
##
## $Europe
```

2.3.3 Missing values

There are several methods to impute missing values, each with its specificities. The selection of the imputation method might therefore influence the results of the model. In this study we will take into account this source of structural uncertainty by analysing the effects of three different imputation methods: linear regression (`norm.predict`), stochastic regression (`norm.nob`) and random forest (`rf`). Since these imputation models are based on random sampling, we should also study the influence of randomness in the obtention of the final imputed missing value. Here we check that source of uncertainty by limiting the random sampling to 5 different imputed data sets for each imputation method.

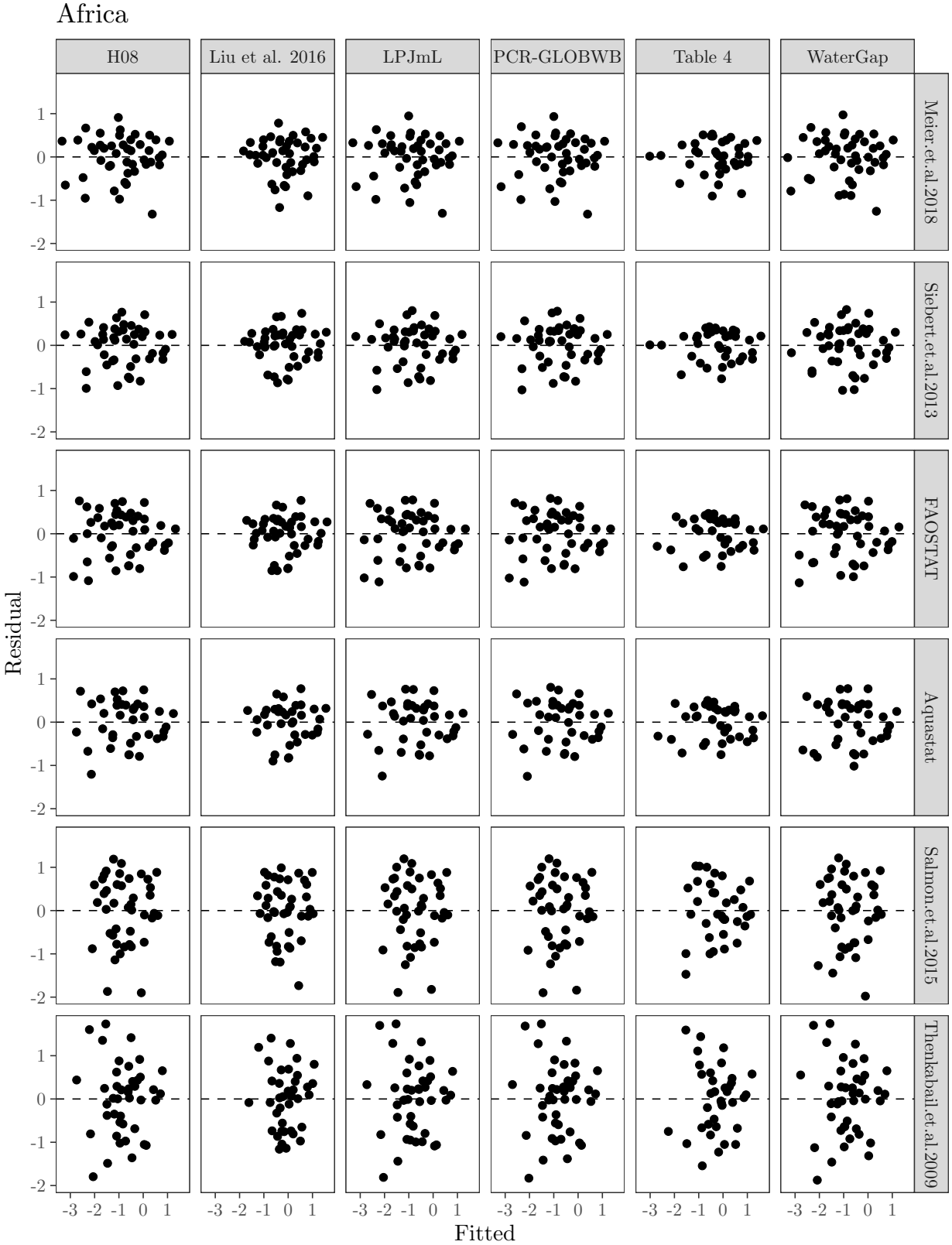


Figure 3: Residual versus fitted values.

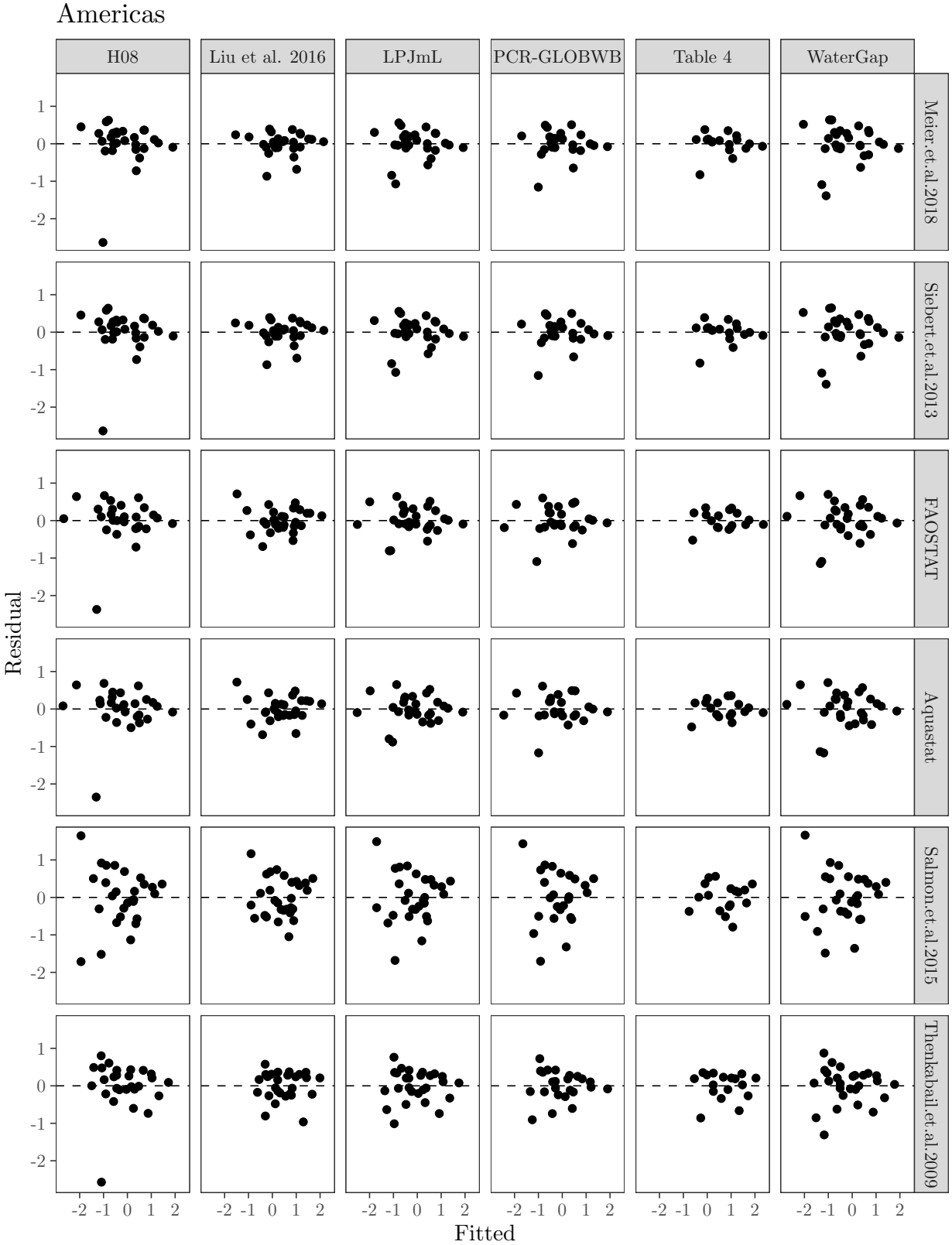


Figure 4: Residual versus fitted values.

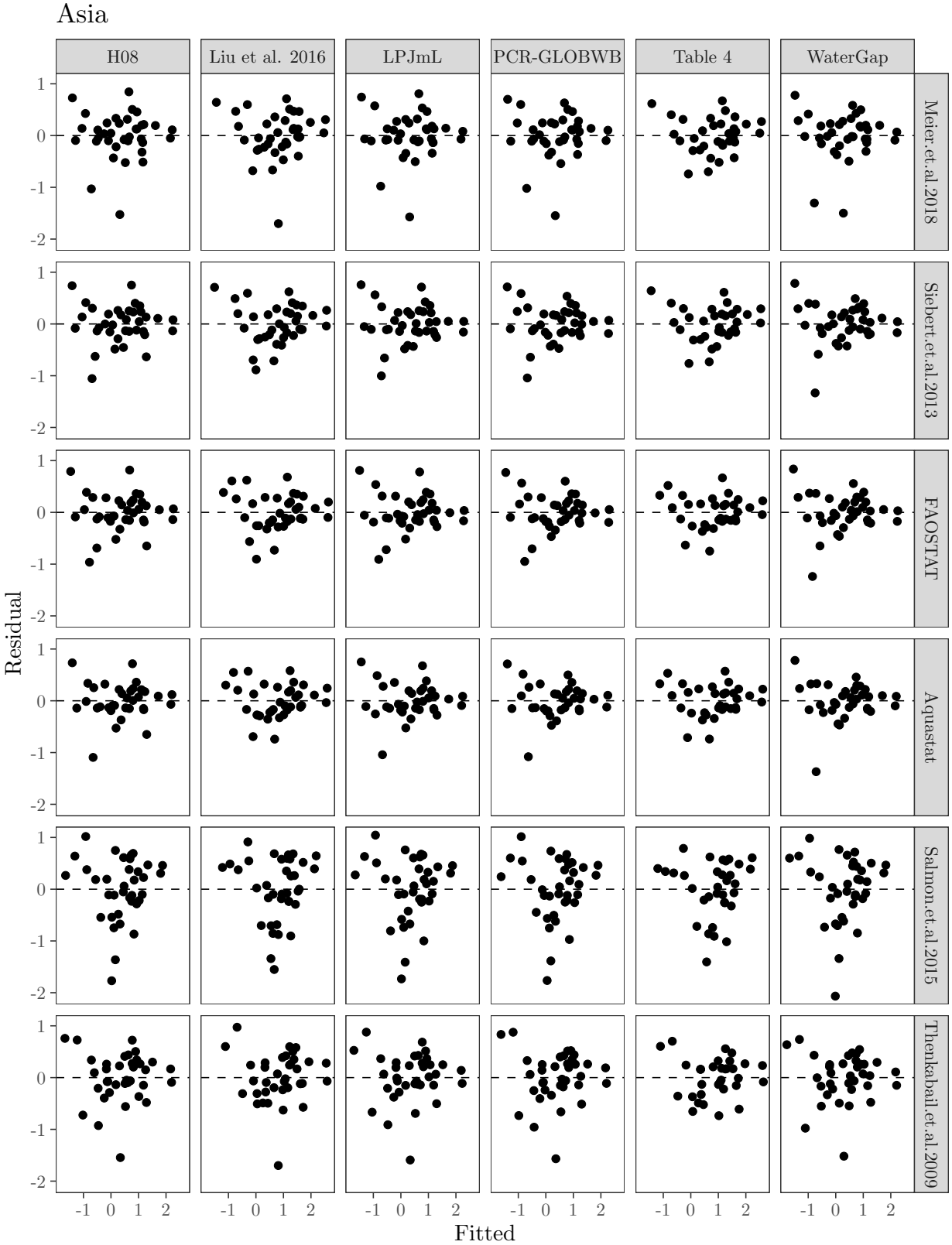


Figure 5: Residual versus fitted values.

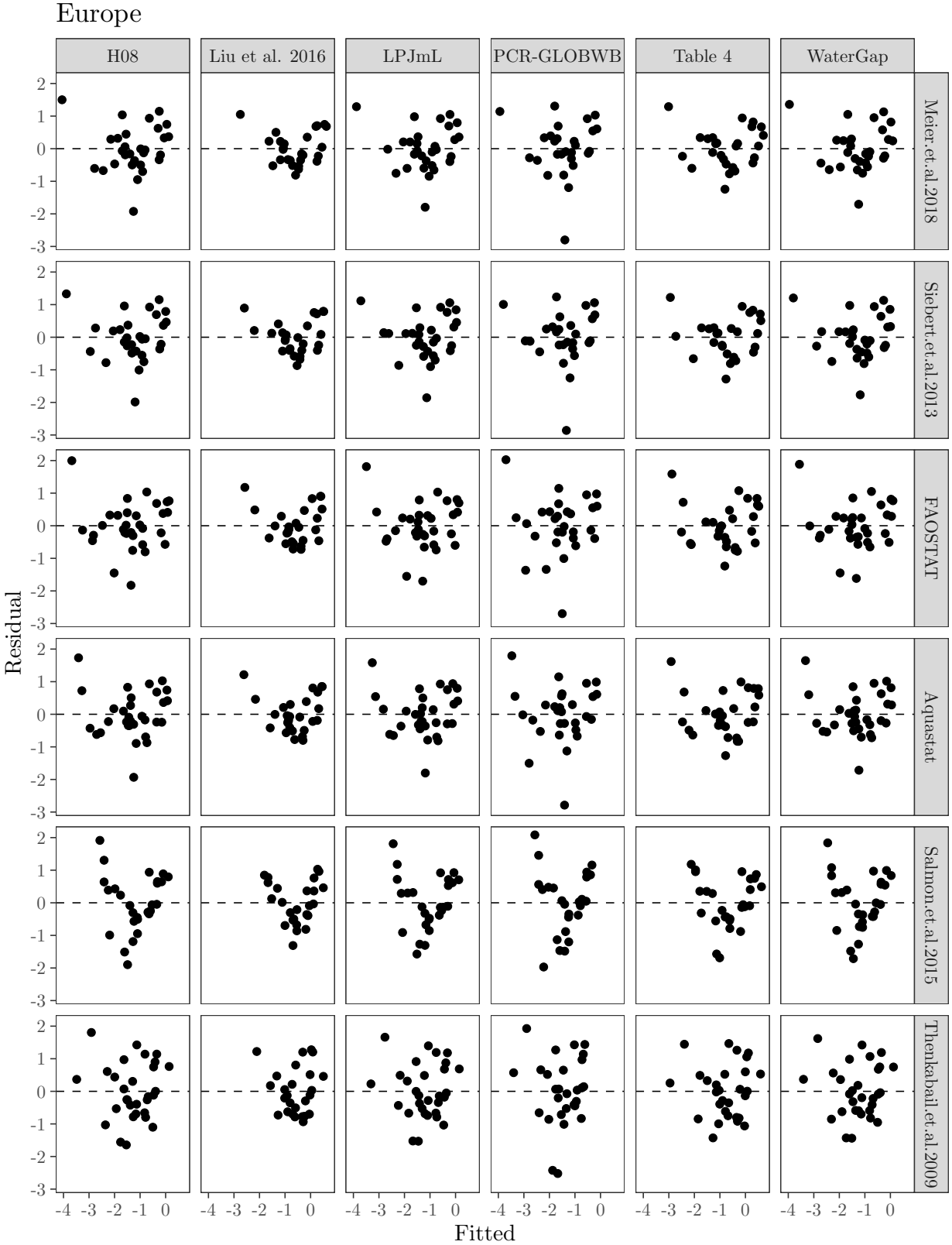


Figure 6: Residual versus fitted values.

3 Bootstrap regressions

In this section we bootstrap ($R = 500$) the regression models in each combination of irrigated area dataset (trigger X1), irrigation water withdrawal dataset (trigger X2), linear regression method (robust/non-robust/heteroskedasticity-corrected; trigger X3), imputation method (trigger X4) and iteration selected (trigger X5). Bootstrap allows to account for an extra source of uncertainty: the true value of β (trigger X6).

```
# PREPARE THE BOOTSTRAP ----------------------------------------------------

# Define number of replicas
R <- 500

# Set number of cores (75 %)
n_cores <- floor(detectCores() * 0.75)

# Set grouping vector
all.cols <- c(cols_group, "Imputation.Method", "Iteration")
```

3.1 Regular bootstrap

We set the formula to bootstrap the classic OLS regression, and launch the computation. We finally extract the results.

```
# REGULAR BOOTSTRAP --------------------------------------------------------

# Function to bootstrap OLS coefficients
boot.ols <- function(formula, x, i) {
  d <- x[i, ]
  out <- coef(lm(formula, data = d))
  return(out)
}

# Bootstrap
ols <- full.imput[, list(list(boot(.SD,
                                   statistic = boot.ols,
                                   R = R,
                                   formula = Water.Withdrawn ~ Irrigated.Area,
                                   parallel = "multicore",
                                   ncpus = n_cores))),
                  all.cols]
```

```
# EXTRACT RESULTS ----------------------------------------------------------

ols.dt <- ols[, `:=`("All", list(lapply(V1, function(x) x["t"])))] %>%
  .[, list(Alpha = lapply(All, function(x) lapply(x, function(y) y[, 1])),
           Beta = lapply(All, function(x) lapply(x, function(y) y[, 2]))),
    all.cols] %>%
  .[, lapply(.SD, unlist), .SDcols = c("Alpha", "Beta"),
```

```
    all.cols] %>%
  .[, Regression:= "Normal"]

ols.dt <- ols.dt[, Alpha:= NULL]
```

3.2 Robust bootstrap

We set the formula to bootstrap robust OLS regression (M estimator), and launch the computation. We then extract the results.

```
# ROBUST BOOTSTRAP ------------------------------------------------------------

# Function to bootstrap robust coefficients
boot.olsR <- function(formula, x, i) {
  d <- x[i, ]
  out <- coef(rlm(formula, data = d))
  return(out)
}

# Bootstrap
olsR <- full.imput[, list(list(boot(.SD,
                                    statistic = boot.olsR,
                                    R = R,
                                    formula = Water.Withdrawn ~ Irrigated.Area,
                                    parallel = "multicore",
                                    ncpus = n_cores))),
                   all.cols]
```

```
## Warning in rlm.default(x, y, weights, method = method, wt.method =
## wt.method, : 'rlm' failed to converge in 20 steps
```

```
# EXTRACT RESULTS -------------------------------------------------------------

olsR.dt <- olsR[, `:=`("All", list(lapply(V1, function(x) x["t"])))] %>%
  .[, list(Alpha = lapply(All, function(x) lapply(x, function(y) y[, 1])),
           Beta = lapply(All, function(x) lapply(x, function(y) y[, 2]))),
    all.cols] %>%
  .[, lapply(.SD, unlist), .SDcols = c("Alpha", "Beta"),
    all.cols] %>%
  .[, Regression:= "Robust"]

olsR.dt <- olsR.dt[, Alpha:= NULL]
```

3.3 Heteroskedastic Bootstrap

We finally perform an heteroskedasticity-corrected bootstrap as follows: for each linear regression, we calculate the covariance matrix of the coefficients using an heteroskedasticity-consistent estimator (White's estimator). We then extract the upper and lower 95% confidence intervals, and use them to compute the mean and the standard deviation (assuming an underlying normal distribution). Finally,

we use these values to randomly generate normal distributions with size R for each combination of triggers.

```
# HETEROSKEDASTIC BOOTSTRAP ------------------------------------------------

hetero2 <- full.imput[, .(model = .(lm(Water.Withdrawn ~ Irrigated.Area))),
                            all.cols]

hetero.dist <- hetero2[, .(cis = .(lapply(model, function(x)
  coefci(x, level = 0.95, vcov. = vcovHC(x, type = "HC1"))))), all.cols]

hetero.dist <- hetero.dist[, ci.min:= lapply(cis, function(x)
  lapply(x, function(y) y[2])), all.cols]

hetero.dist <- hetero.dist[, ci.max:= lapply(cis, function(x)
  lapply(x, function(y) y[4])), all.cols]

hetero.dist <- hetero.dist[, c("ci.min", "ci.max"):= lapply(.SD, as.numeric),
                           .SDcols = c("ci.min", "ci.max")]

hetero.dist <- hetero.dist[, `:=` (mean = (ci.min + ci.max) / 2,
                                   sd = (ci.max - ci.min) / 4)]

# Set seed
set.seed(10)

# Sample
ols.dt.het <- hetero.dist[, .(Beta = rnorm(n = R, mean = mean, sd = sd)), all.cols] %>%
  .[, Regression:= "Hetero"]
```

3.4 Merge results

We bind the results of each bootstrapped regression, and obtained a full dataset with all the replicas.

```
# MERGE REGULAR, ROBUST AND HETEROSKEDASTIC BOOTSTRAP ------------------------

replicas <- rbind(ols.dt, olsR.dt, ols.dt.het)
```

3.5 Create lookup table

Here we define the lookup table that we will use to select the β value according to the conditions set by the triggers in the sample matrix (see below).

```
# CREATE LOOKUP TABLE ------------------------------------------------------

col_names <- c("Continent", "Water.Dataset", "Area.Dataset",
               "Regression", "Imputation.Method", "Iteration")

lookup <- replicas[order(Beta), .SD, col_names] %>%
```

```
  .[, ID:= 1:.N, col_names] %>%
  .[, index:= paste(Continent, Area.Dataset, Water.Dataset,
                    Regression, Imputation.Method, Iteration,
                    ID, sep = "_")]

lookup <- setkey(lookup, index)

# Export lookup
fwrite(lookup, "lookup.csv")
```

4 The sample matrix

This section designs the sample matrix that will be used to select the β value according to the conditions set by the triggers. Firstly, we pool all irrigated area and water withdrawal datasets and calculate the minimum and maximum values for each of these parameters. This is needed to bound the uniform distributions with which we will characterize X_1 and X_2 respectively. We also define the size of the sample matrix ($N = 2^{14}$), and create a function to transform its columns to their appropriate distributions.

```
# DEFINE DISTRIBUTIONS ----------------------------------------------------

# Vector with the name of the irrigated area datasets
irrigated.datasets <- colnames(meier.dt)[4:9]

# Irrigated Area
irrigated.area <- melt(meier.dt, measure.vars = irrigated.datasets) %>%
  .[, value:= value / 10 ^ 6] %>%
  .[, .(Total = sum(value, na.rm = TRUE)), .(Continent, variable)] %>%
  .[, .(min = min(Total), max = max(Total)), Continent]

# Irrigation water withdrawn
water.withdrawn <- water.dt[, .(Total = sum(Water.Withdrawn, na.rm = TRUE)),
                            .(Continent, Water.Dataset)] %>%
  .[, .(min = min(Total), max = max(Total)), Continent]
```

```
# DEFINE THE SETTINGS OF THE SAMPLE MATRIX --------------------------------

Continents <- c("Africa", "Americas", "Asia", "Europe")

# Create a vector with the name of the columns
parameters <- paste("X", 1:6, sep = "")

# Obtain number of parameters
k <- length(parameters)

# Select sample size
n <- 2 ^ 14
```

```
# Check number of bootstrap samples
N.boot <- lookup[, .(N = .N), .(Continent, Area.Dataset, Water.Dataset,
                                Regression, Imputation.Method, Iteration)] %>%
  .[, N] %>%
  .[1]
```

```
# CREATE THE SAMPLE MATRIX ----------------------------------------------------

# Create an A, B and AB matrices for each continent
sample.matrix <- lapply(Continents, function(Continents)
  sobol_matrices(n = n,
                 k = k,
                 second = TRUE,
                 third = TRUE) %>%
    data.table())

# Name the slots, each is a continent
names(sample.matrix) <- Continents

# Name the columns
sample.matrix <- lapply(sample.matrix, setnames, parameters)
```

```
# TRANSFORM THE SAMPLE MATRIX -------------------------------------------------

# Transform the sample matrix
transform_sample_matrix <- function(dt) {
  dt[, X1:= floor(X1 * (6 - 1 + 1)) + 1] %>%
    .[, X1:= ifelse(X1 == 1, "Aquastat",
                    ifelse(X1 == 2, "FAOSTAT",
                           ifelse(X1 == 3, "Siebert.et.al.2013",
                                  ifelse(X1 == 4, "Meier.et.al.2018",
                                         ifelse(X1 == 5, "Salmon.et.al.2015",
                                                "Thenkabail.et.al.2009")))))] %>%
    .[, X2:= floor(X2 * (6 - 1 + 1)) + 1] %>%
    .[, X2:= ifelse(X2 == 1, "LPJmL",
                    ifelse(X2 == 2, "H08",
                           ifelse(X2 == 3, "PCR-GLOBWB",
                                  ifelse(X2 == 4, "WaterGap",
                                         ifelse(X2 == 5, "Table 4",
                                                "Liu et al. 2016")))))] %>%
    .[, X3:= floor(X3 * (3 - 1 + 1)) + 1] %>%
    .[, X3:= ifelse(X3==1, "Normal",
                    ifelse(X3 == 2, "Robust", "Hetero"))] %>%
    .[, X4:= floor(X4 * (length(imputation.methods) - 1 + 1)) + 1] %>%
    .[, X4:= ifelse(X4 == 1, imputation.methods[1],
                    ifelse(X4 == 2, imputation.methods[2], imputation.methods[3]))] %>%
    .[, X5:= floor(X5 * (m.iterations - 1 + 1)) + 1] %>%
```

```
    .[, X6:= floor(X6 * (N.boot - 1)) + 1]
}

sample.matrix <- lapply(sample.matrix, transform_sample_matrix)
sample.matrix.dt <- rbindlist(sample.matrix, idcol = "Continent")

fwrite(sample.matrix.dt, "sample.matrix.dt.csv")
```

5 The model

The model is simply a one-line function that searches in the lookup table the β value defined by the triggers of the sample matrix. In order to speed up the computation, we use parallel computing.

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

model <- function(X) lookup[.(paste0(X[, 1:7], collapse = "_"))][, Beta]
```

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

# Create cluster
cl <- makeCluster(n_cores)
registerDoParallel(cl)

# Run model in parallel
Y <- foreach(i=1:nrow(sample.matrix.dt),
             .packages = "data.table") %dopar%
  {
    model(sample.matrix.dt[i])
  }

# Stop parallel cluster
stopCluster(cl)
```

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

sample.matrix.dt <- sample.matrix.dt[, Y:= as.numeric(cbind(Y))]

# Select the A and B matrix only (for uncertainty analysis)
AB.dt <- sample.matrix.dt[, .SD[1:(n * 2)], Continent]

# Export results
fwrite(sample.matrix.dt, "sample.matrix.dt.csv")
fwrite(AB.dt, "AB.dt.csv")
```

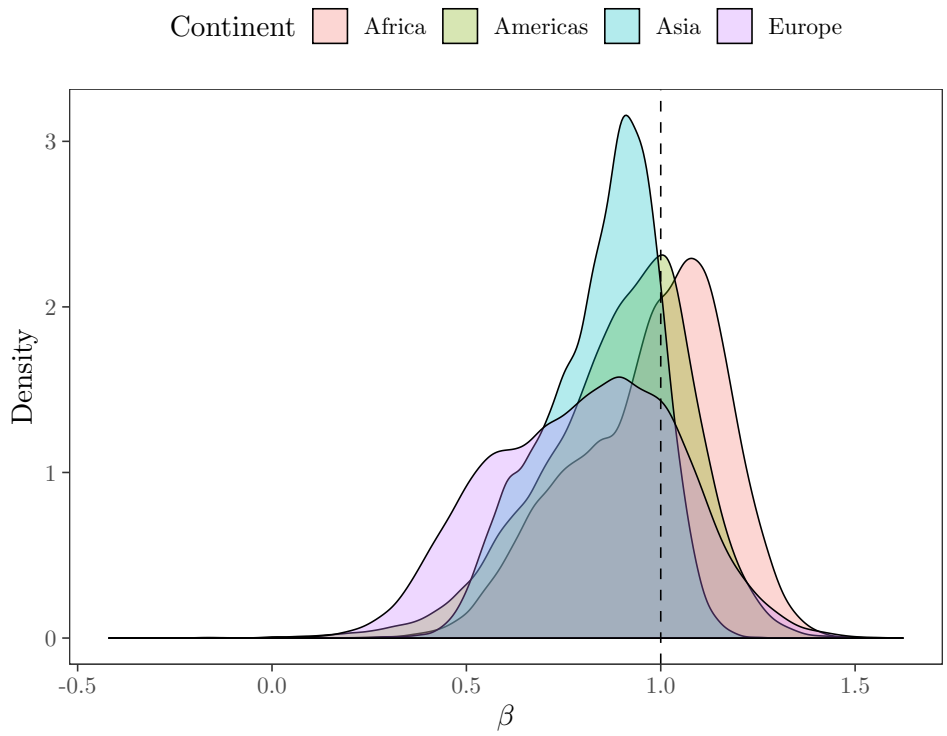


Figure 7: Uncertainty in the model output.

6 Uncertainty analysis

Here we plot the distribution of β for each continent, and calculate the proportion of model runs with $\beta < 1$ (sublinear regime; larger irrigated areas are more water efficient) and $\beta > 1$ (superlinear regime, larger irrigated areas are less water efficient). The results suggest that there are almost 9/10 chances and 3/4 chances of larger irrigated areas in Asia and Europe being more water efficient than smaller ones. In the case of the Americas and Africa, the chances are 3.5/5 and 1/2 respectively.

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

ggplot(AB.dt, aes(Y, fill = Continent)) +
  geom_density(alpha = 0.3) +
  theme_AP() +
  geom_vline(xintercept = 1,
             lty = 2) +
  labs(x = "$\\beta$",
       y = "Density") +
  theme(legend.position = "top")
```

```
# SUPERLINEAR OR SUBLINEAR REGIME? --------------------------------------------

AB.dt[, .(sublinear = sum(Y < 1, na.rm = TRUE) / .N,
          superlinear = sum(Y > 1, na.rm = TRUE) / .N),
      Continent]
```

```
##    Continent sublinear superlinear
```

```
## 1:    Africa 0.5018311   0.4981689
## 2:  Americas 0.6906738   0.3093262
## 3:      Asia 0.8778687   0.1221313
## 4:    Europe 0.7772217   0.2227783
```

7 Sensitivity analysis

```
# SENSITIVITY SETTINGS ------------------------------------------------------

# Number of bootstrap replicas
R <- 1000

# Sensitivity estimator
estimator <- "jansen"

# Method to calculate ci
type <- "norm"

# Confidence interval
conf <- 0.95
```

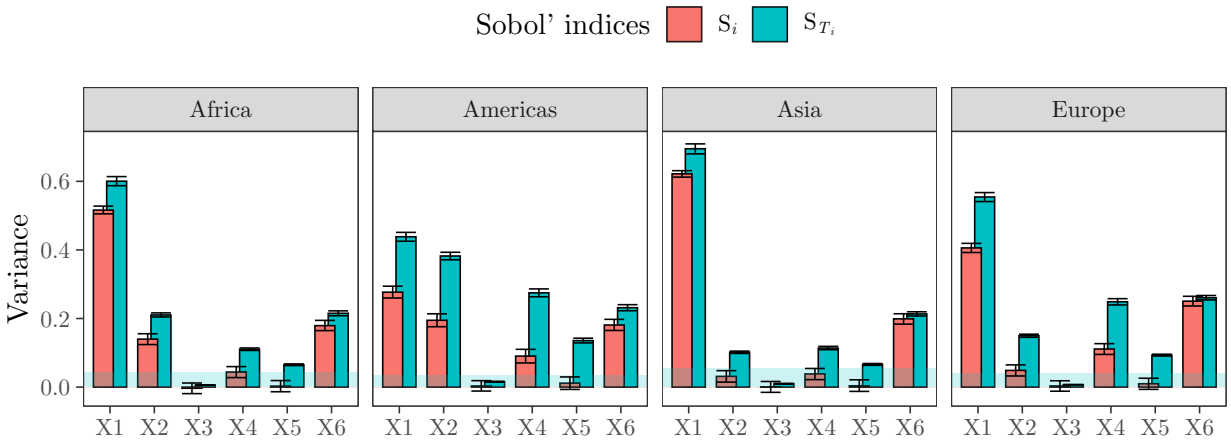
```
# SENSITIVITY ANALYSIS ------------------------------------------------------

# Compute Sobol' indices up to the third order
indices <- sample.matrix.dt[, sobol_indices(Y,
                                            params = parameters,
                                            R = R,
                                            n = n,
                                            type = estimator,
                                            parallel = "multicore",
                                            ncpus = n_cores,
                                            second = TRUE,
                                            third = TRUE),
                            Continent]

# Compute Sobol' indices for the dummy parameter
indices.dummy <- sample.matrix.dt[, sobol_dummy(Y,
                                                params = parameters,
                                                R = R,
                                                n = n,
                                                parallel = "multicore",
                                                ncpus = n_cores),
                                  Continent]
```

```
# COMPUTE CONFIDENCE INTERVALS ----------------------------------------------

# Compute ci for the model parameters
tmp <- split(indices, indices$Continent)
```

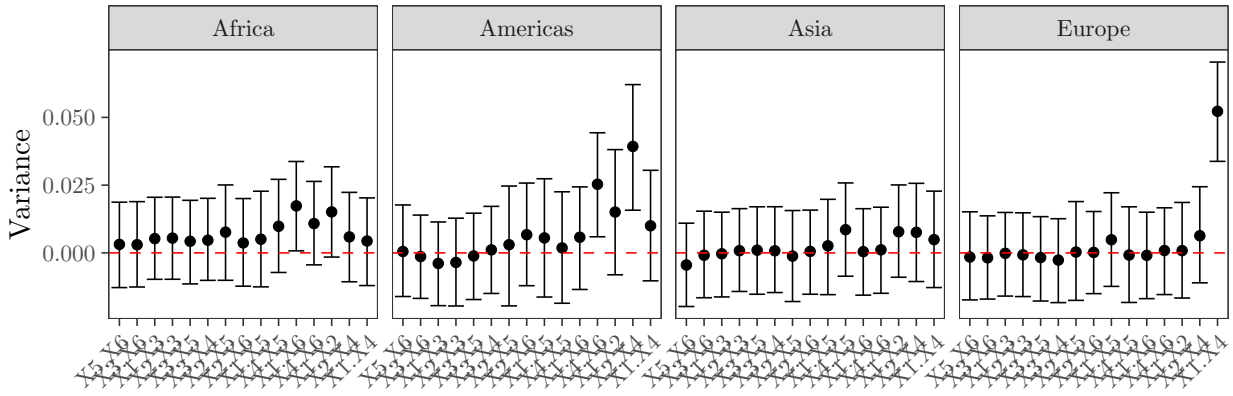
```r
ci <- list()
for(i in names(tmp)) {
  ci[[i]] <- sobol_ci(tmp[[i]],
                      params = parameters,
                      type = type,
                      conf = conf,
                      second = TRUE,
                      third = TRUE)
}

ci <- rbindlist(ci, idcol = "Continent")

# Compute ci for the dummy parameter
tmp <- split(indices.dummy, indices.dummy$Continent)
ci.dummy <- list()
for(i in names(tmp)) {
  ci.dummy[[i]] <- sobol_ci_dummy(tmp[[i]],
                                  type = type,
                                  conf = conf)
}

ci.dummy <- rbindlist(ci.dummy, idcol = "Continent")
```

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

lapply(1:3, function(x) plot_sobol(ci, type = x, dummy = ci.dummy) +
         facet_wrap(~Continent, ncol = 4) +
         theme(aspect.ratio = 1))
```

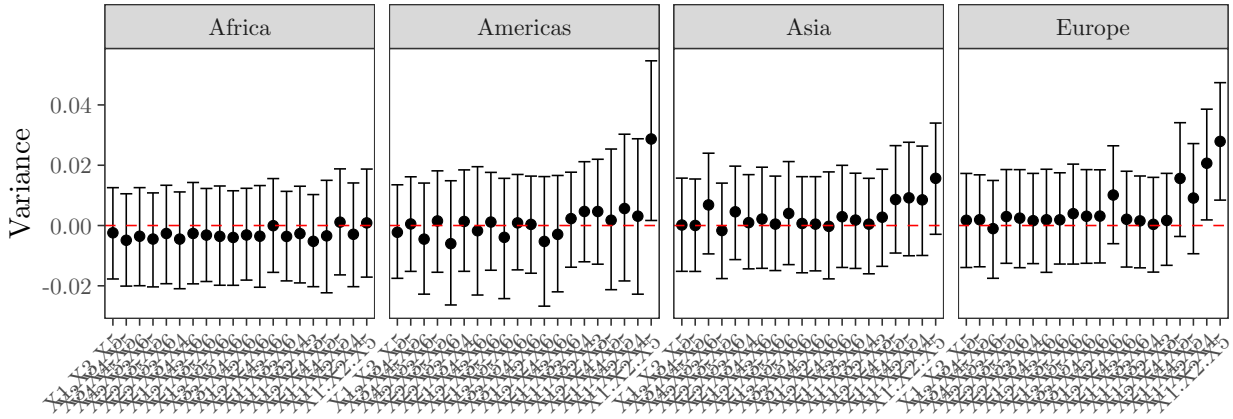
```
## [[1]]
```




```
##
## [[2]]
```




```
##
## [[3]]
```




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

ci[sensitivity == "Si"] %>%
  .[, sum(original), Continent]
```

```
##    Continent        V1
## 1:    Africa 0.8790844
## 2:  Americas 0.7581420
## 3:      Asia 0.8954424
## 4:    Europe 0.8297298
```

8 Session information

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

sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
```

```
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] checkpoint_0.4.7  mice_3.6.0        lattice_0.20-38
##  [4] lmtest_0.9-37     zoo_1.8-6         sandwich_2.5-1
##  [7] mvoutlier_2.0.9   sgeostat_1.0-27   complmrob_0.7.0
## [10] doParallel_1.0.15 iterators_1.0.12  foreach_1.4.7
## [13] MASS_7.3-51.4     boot_1.3-23       IDPmisc_1.1.19
## [16] dplyr_0.8.3       countrycode_1.1.0 rworldmap_1.3-6
## [19] sp_1.3-1          ncdf4_1.16.1      scales_1.0.0
## [22] sensobol_0.2.1    ggplot2_3.2.1     data.table_1.12.2
##
## loaded via a namespace (and not attached):
##   [1] minqa_1.2.4           colorspace_1.4-1      class_7.3-15
##   [4] modeltools_0.2-22     rio_0.5.16            mclust_5.4.5
##   [7] pls_2.7-1             cvTools_0.3.2         flexmix_2.3-15
##  [10] mvtnorm_1.0-11        ranger_0.11.2         codetools_0.2-16
##  [13] splines_3.6.1         sROC_0.1-2            robustbase_0.93-5
##  [16] knitr_1.25            zeallot_0.1.0         spam_2.3-0
##  [19] nloptr_1.2.1          robCompositions_2.1.0 broom_0.5.2
##  [22] cluster_2.1.0         kernlab_0.9-27        rrcov_1.4-7
##  [25] compiler_3.6.1        backports_1.1.4       assertthat_0.2.1
##  [28] Matrix_1.2-17         lazyeval_0.2.2        htmltools_0.3.6
##  [31] tools_3.6.1           dotCall64_1.0-0       gtable_0.3.0
##  [34] glue_1.3.1            maps_3.3.0            Rcpp_1.0.2
##  [37] carData_3.0-2         cellranger_1.1.0      vctrs_0.2.0
##  [40] zCompositions_1.3.2-1 nlme_3.1-141          fpc_2.2-3
##  [43] gbRd_0.4-11           xfun_0.9              laeken_0.5.0
##  [46] stringr_1.4.0         lme4_1.1-21           openxlsx_4.1.0.1
##  [49] lifecycle_0.1.0       pan_1.6               DEoptimR_1.0-8
##  [52] VIM_4.8.0             hms_0.5.1             RColorBrewer_1.1-2
##  [55] fields_9.8-6          yaml_2.2.0            curl_4.1
##  [58] NADA_1.6-1            rpart_4.1-15          reshape_0.8.8
##  [61] stringi_1.4.3         maptools_0.9-5        pcaPP_1.9-73
##  [64] e1071_1.7-2           zip_2.0.4             bibtex_0.4.2
##  [67] truncnorm_1.0-8       Rdpack_0.11-0         rlang_0.4.0
##  [70] pkgconfig_2.0.3       prabclus_2.3-1        evaluate_0.14
##  [73] purrr_0.3.2           tidyselect_0.2.5      GGally_1.4.0
##  [76] plyr_1.8.4            magrittr_1.5          R6_2.4.0
##  [79] generics_0.0.2        mitml_0.3-7           pillar_1.4.2
```

```
##  [82] haven_2.1.1          foreign_0.8-72       withr_2.1.2
##  [85] survival_2.44-1.1    abind_1.4-5          nnet_7.3-12
##  [88] tibble_2.1.3         crayon_1.3.4         car_3.0-3
##  [91] jomo_2.6-9           rmarkdown_1.15       grid_3.6.1
##  [94] readxl_1.3.1         forcats_0.4.0        vcd_1.4-4
##  [97] digest_0.6.21        diptest_0.75-7       tidyr_1.0.0
## [100] stats4_3.6.1         munsell_0.5.0
```

References

FAO. 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/.

Filzmoser, P., R. G. Garrett, and C. Reimann. 2005. "Multivariate outlier detection in exploration geochemistry." *Computers and Geosciences* 31 (5): 579–87. https://doi.org/10.1016/j.cageo.2004.11.013.

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.

Microsoft Corporation. 2018. "checkpoint: Install Packages from Snapshots on the Checkpoint Server for Reproducibility." https://cran.r-project.org/package=checkpoint.

Puy, A., R. Muneepeerakul, and A. L. Balbo. 2017. "Size and stochasticity in irrigated social-ecological systems." *Scientific Reports* 7 (March): 43943. https://doi.org/10.1038/srep43943.

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. Elsevier B.V.: 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." Rome: Rheinische Friedrich-Wilhelms-University; Food; Agriculture Organization of the United Nations.

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.

Wisser, D., S. Frolking, E. M. Douglas, B. M. Fekete, C. J. Vörösmarty, and A. H. Schumann. 2008. "Global irrigation water demand: Variability and uncertainties arising from agricultural and climate data sets." *Geophysical Research Letters* 35 (24): 1–5. https://doi.org/10.1029/2008GL035296.