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",
               "ggridges", "broom", "naniar", "cowplot",
               "tidyr", "benchmarkme"))

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

dir.create(".checkpoint")

library("checkpoint")

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

```r
# 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 coordinates into 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.

```r
# 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)
}
```

```r
# 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.

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

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

# UNIT IS KM3/YEAR
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 ----------------------------
#UNIT IS 10^9 m3/year = km3/year
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 ---------------------------------------

# Read in list of countries in UN
DT <- fread("UN_countries2.csv", select = "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"))]
```

```
## Warning in countrycode(DT[, Country], origin = "country.name", destination = "un"): Some val

## Warning in countrycode(DT[, Country], origin = "country.name", destination = "continent"): S
```

```
# Manually add Micronesia
DT <- DT[, Continent:= ifelse(Country %in% "Micronesia", "Oceania", Continent)]
```

```
# 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 Somalia  706    Africa      0.19308032
## 3:    PCR-GLOBWB  Cyprus  196      Asia      0.02436695
## 4:    PCR-GLOBWB Somalia  706    Africa      0.20744558
## 5:           H08  Cyprus  196      Asia      0.01944480
## 6:           H08 Somalia  706    Africa      0.20371014
## 7:      WaterGap  Cyprus  196      Asia      0.01020237
## 8:      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 by 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"))

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

# Vector with the countries to drop
countries.drop <- tmp.dt[Water.Withdrawn == 0 & is.na(Irrigated.Area) == TRUE] %>%
  .[, unique(Country)]

# Drop the countries
full.dt <- tmp.dt[!Country %in% countries.drop]
```

```
# EXPORT IRRIGATED AREA DT -------------------------------------------------

fwrite(irrigated.dt, "irrigated.dt.csv")
```

```
# 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",
        strip.text = element_text(size = 7))
```

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

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

```
## Warning: Removed 993 rows containing missing values (geom_point).
```

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

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

```
# EXPORT FULL DATASET WITH MISSING VALUES ---------------------------------------

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

```
# SCATTERPLOT SHOWING MISSING VALUES ---------------------------------------

scatter.na <- copy(full.dt)
cols_transform <- c("Irrigated.Area", "Water.Withdrawn")
scatter.na[, (cols_transform):= lapply(.SD, function(x) 10 ^ x), .SDcols = cols_transform]
tmp <- split(scatter.na, scatter.na$Continent)

gg <- list()
for(i in names(tmp)) {
  gg[[i]] <- ggplot(tmp[[i]], aes(Irrigated.Area, Water.Withdrawn)) +
    geom_miss_point() +
    facet_grid(Water.Dataset ~ Area.Dataset) +
    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 ^ (2 * x)),
                    labels = trans_format("log10", math_format(10 ^ .x))) +
    labs(x = "Irrigated area (Mha)",
         y = expression(paste("Water withdrawal", "", 10^9, m^3/year))) +
    theme_AP() +
    theme(legend.position = "top",
          strip.text = element_text(size = 7)) +
    ggtitle(names(tmp[i]))
}

gg
```

```
## $Africa
```

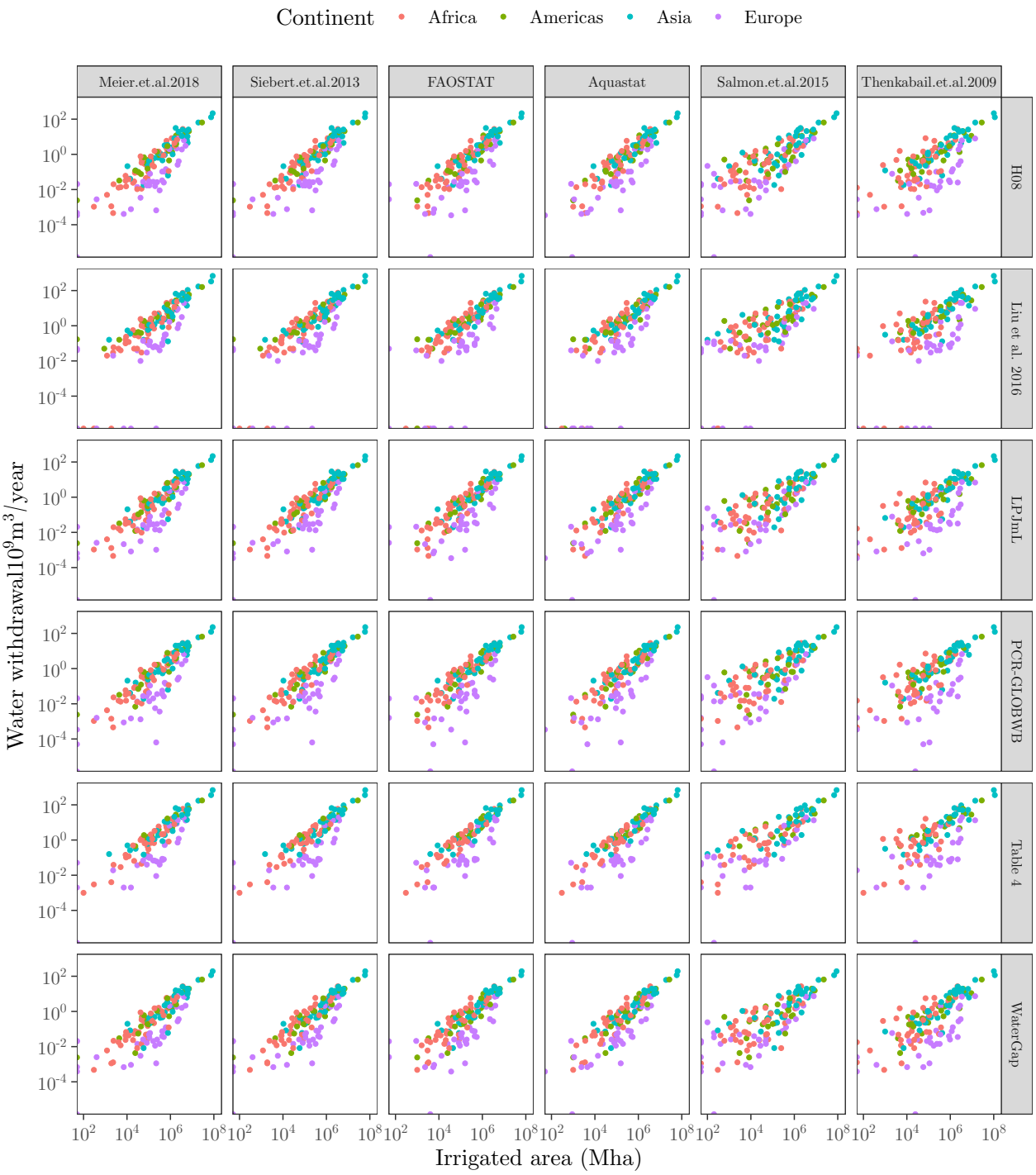
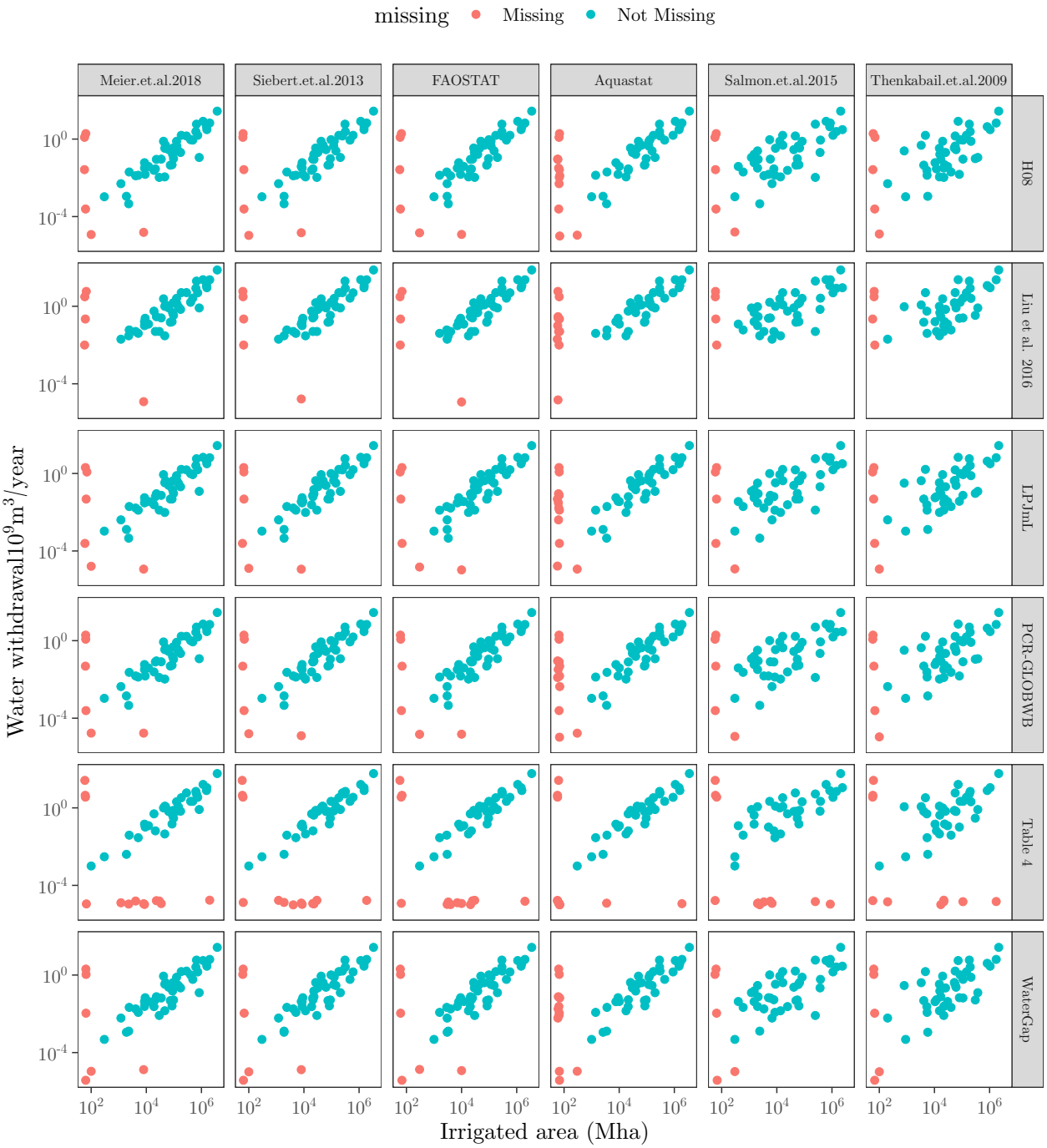


Figure 1: Scatter plots of irrigated areas (x-axis) against irrigation water withdrawal (y-axis). All the possible combinations of datasets are shown.

```
## Warning: Transformation introduced infinite values in continuous x-axis
## Warning: Transformation introduced infinite values in continuous y-axis
## Warning: Removed 47 rows containing non-finite values (stat_miss_point).
```

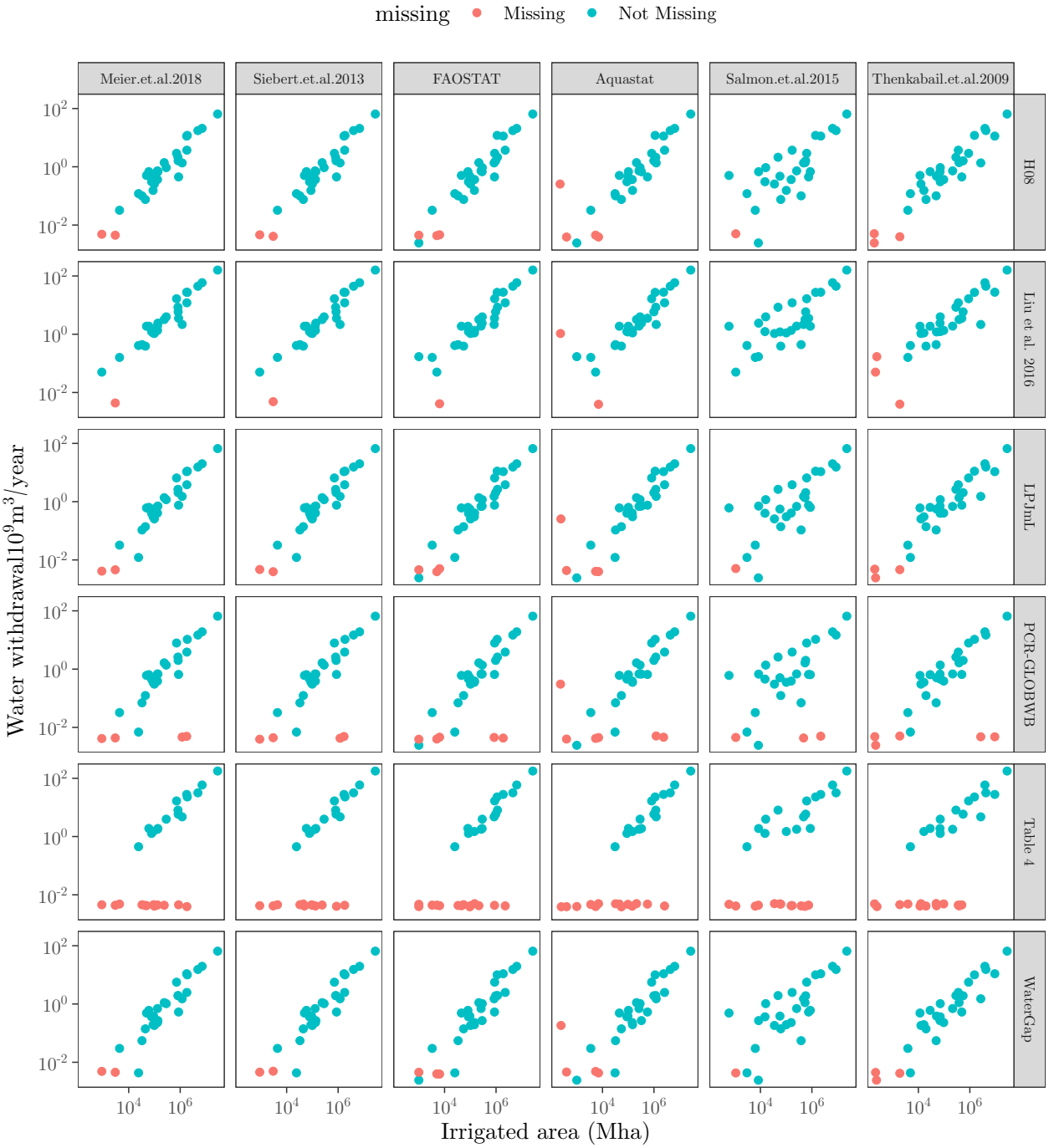
Africa




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

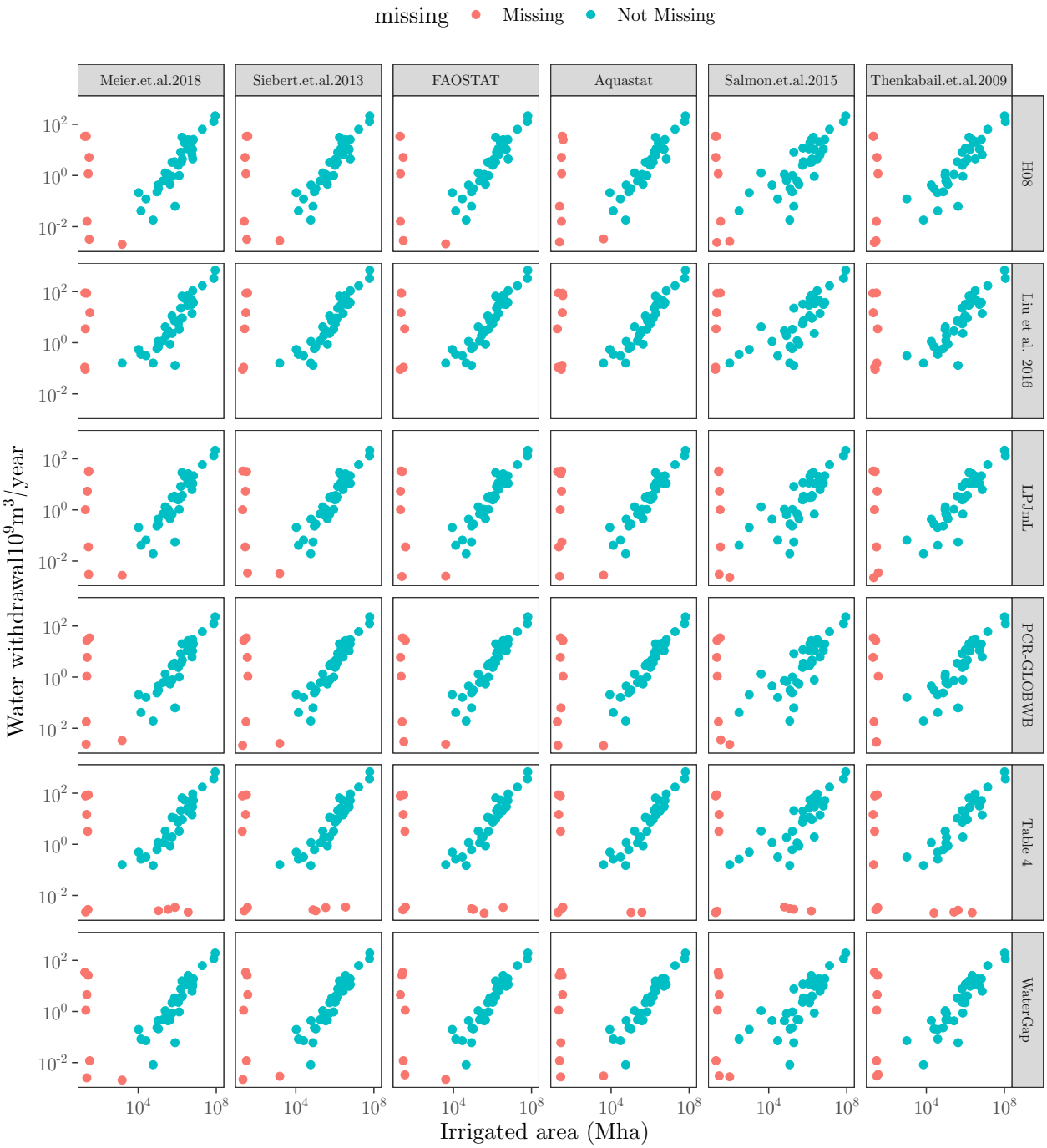
```
## Warning: Transformation introduced infinite values in continuous x-axis
## Warning: Transformation introduced infinite values in continuous y-axis
## Warning: Removed 44 rows containing non-finite values (stat_miss_point).
```

Americas



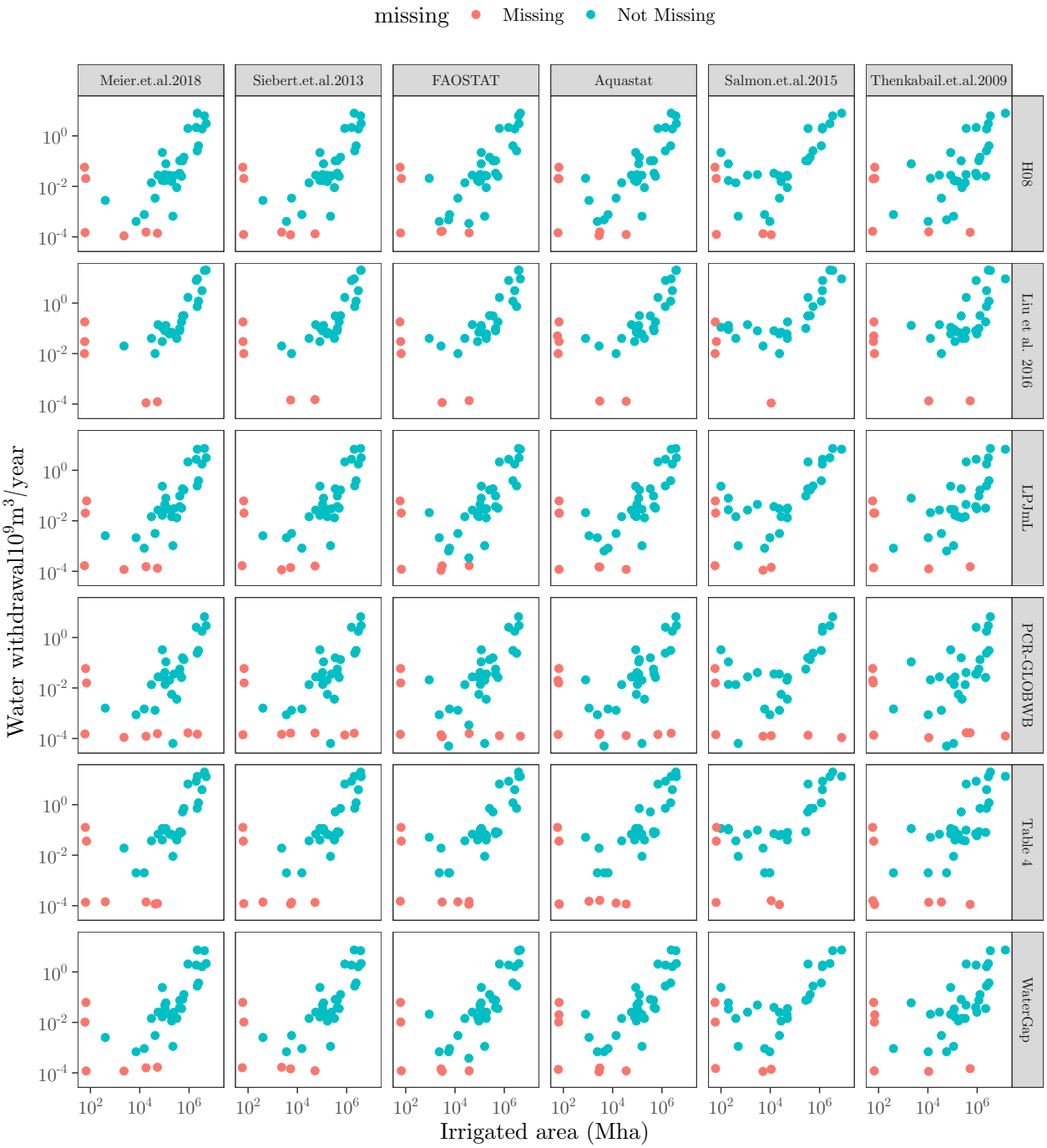

```
##
## $Asia
```

Asia




```
##
## $Europe
## Warning: Transformation introduced infinite values in continuous x-axis
## Warning: Transformation introduced infinite values in continuous y-axis
## Warning: Removed 187 rows containing non-finite values (stat_miss_point).
```

Europe



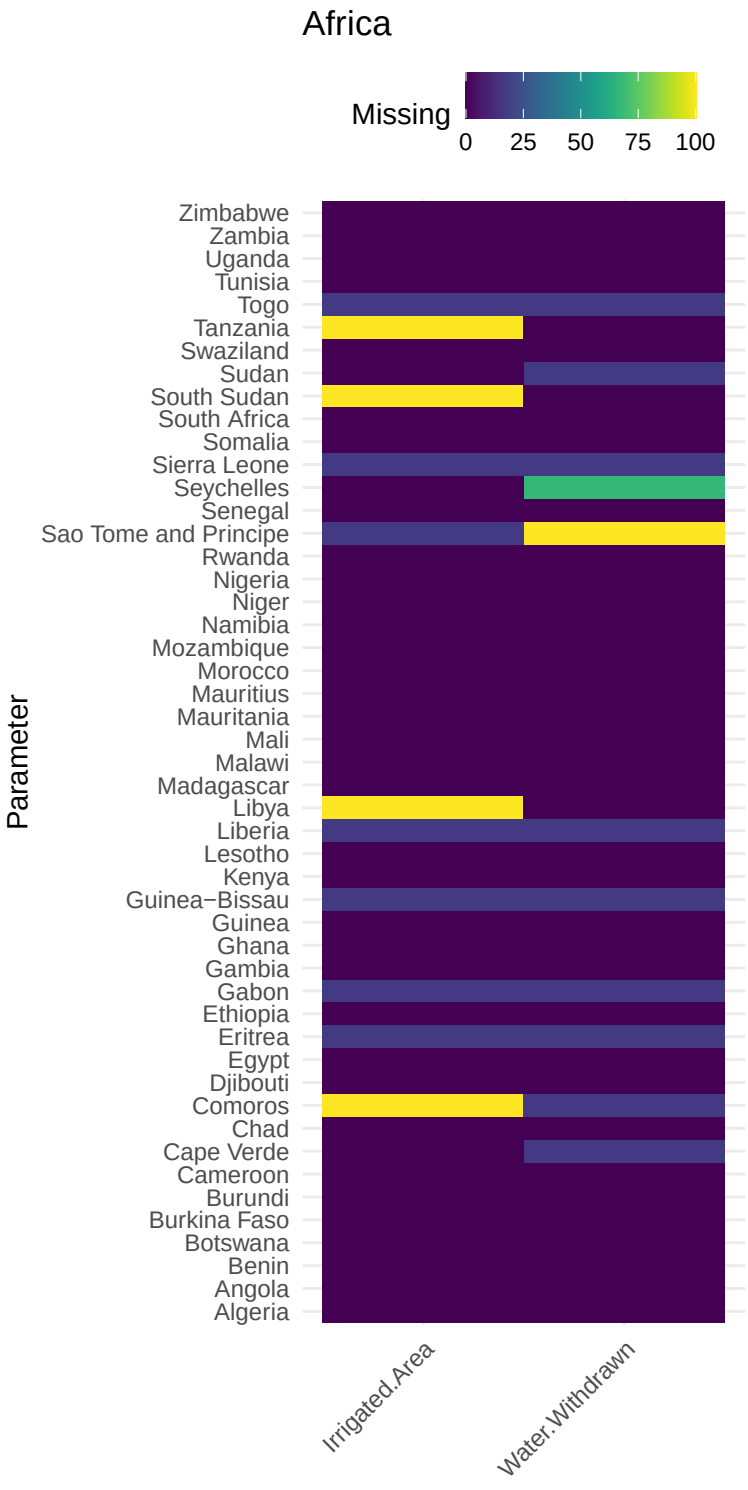

```
# PLOT PERCENTAGE OF MISSING ----------------------------------------------------

na.dt <- full.dt[, .(Continent, Country, Irrigated.Area, Water.Withdrawn)]
tmp <- split(na.dt[, !"Continent", with = FALSE], na.dt$Continent)

gg <- list()
for(i in names(tmp)) {
```

```r
  gg[[i]] <- gg_miss_fct(x = tmp[[i]], fct = Country) +
    coord_flip() +
    labs(x = "Parameter",
         y = "") +
    viridis::scale_fill_viridis(name = "Missing") +
    theme(legend.position = "top") +
    ggtitle(names(tmp[i]))
}
```

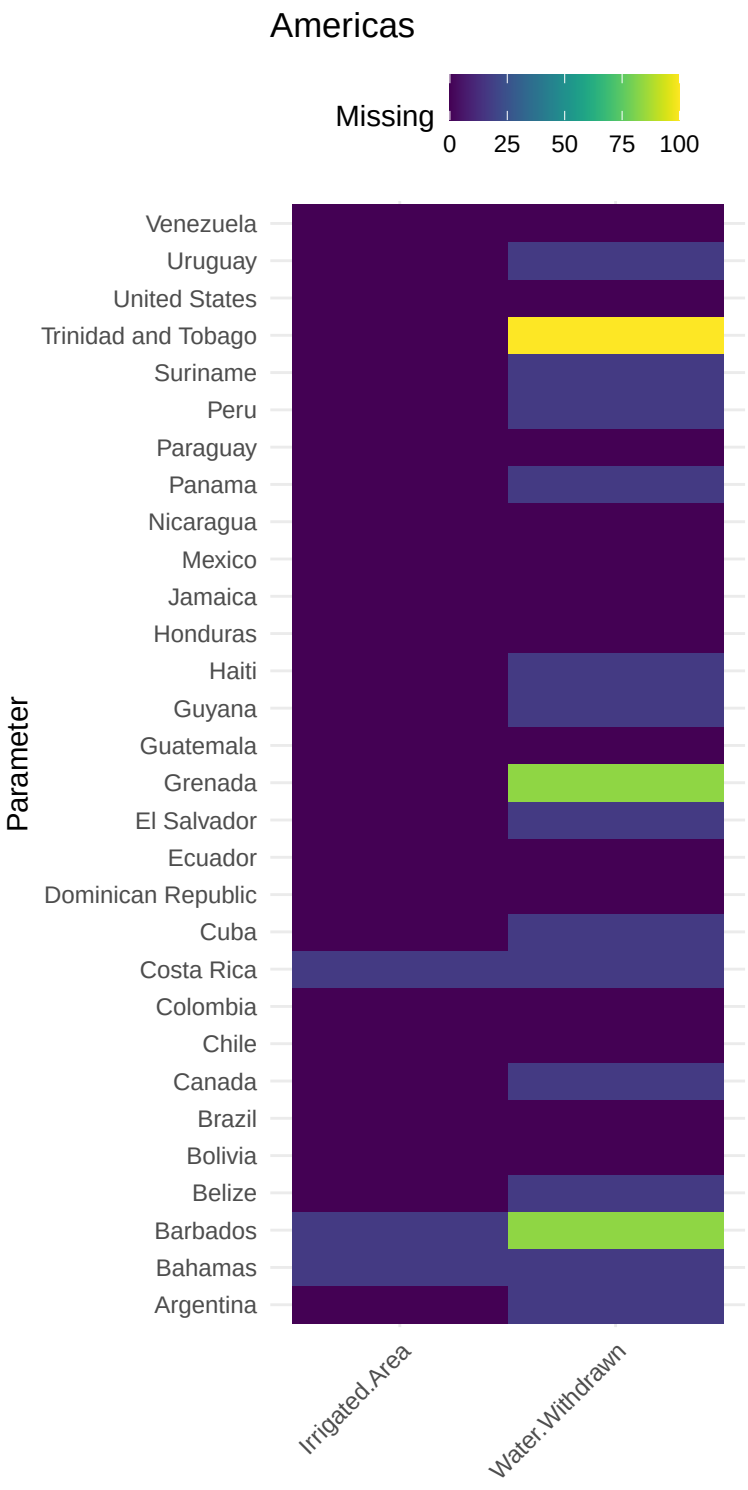
```
## Warning: `cols` is now required.
## Please use `cols = c(data)`

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

## Warning: `cols` is now required.
## Please use `cols = c(data)`

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

## Warning: `cols` is now required.
## Please use `cols = c(data)`

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

## Warning: `cols` is now required.
## Please use `cols = c(data)`

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

```r
gg
```

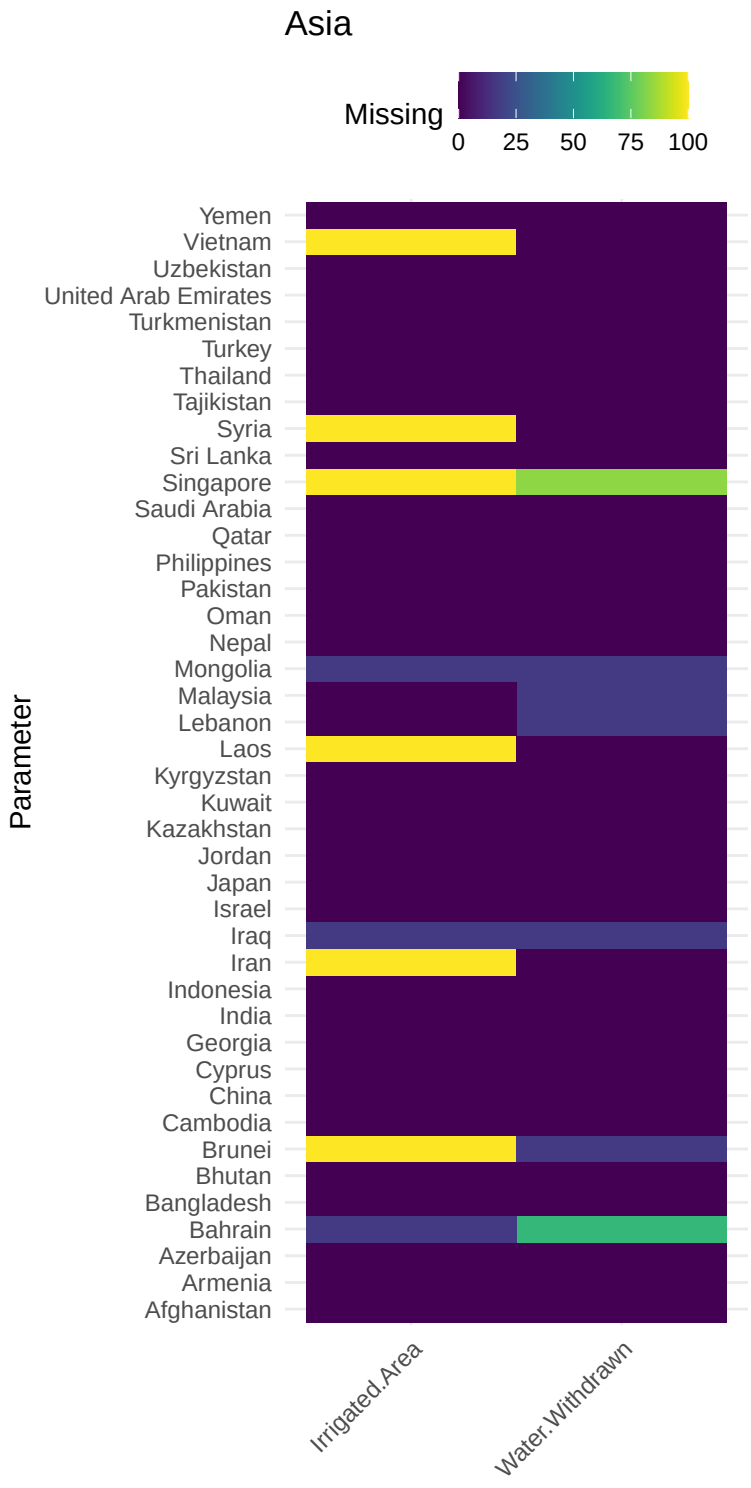
```
## $Africa
```


Africa
Missing
0
25
50
75
100
Parameter
Zimbabwe
Zambia
Uganda
Tunisia
Togo
Tanzania
Swaziland
Sudan
South Sudan
South Africa
Somalia
Sierra Leone
Seychelles
Senegal
Sao Tome and Principe
Rwanda
Nigeria
Niger
Namibia
Mozambique
Morocco
Mauritius
Mauritania
Mali
Malawi
Madagascar
Libya
Liberia
Lesotho
Kenya
Guinea−Bissau
Guinea
Ghana
Gambia
Gabon
Ethiopia
Eritrea
Egypt
Djibouti
Comoros
Chad
Cape Verde
Cameroon
Burundi
Burkina Faso
Botswana
Benin
Angola
Algeria
Irrigated.Area
Water.Withdrawn

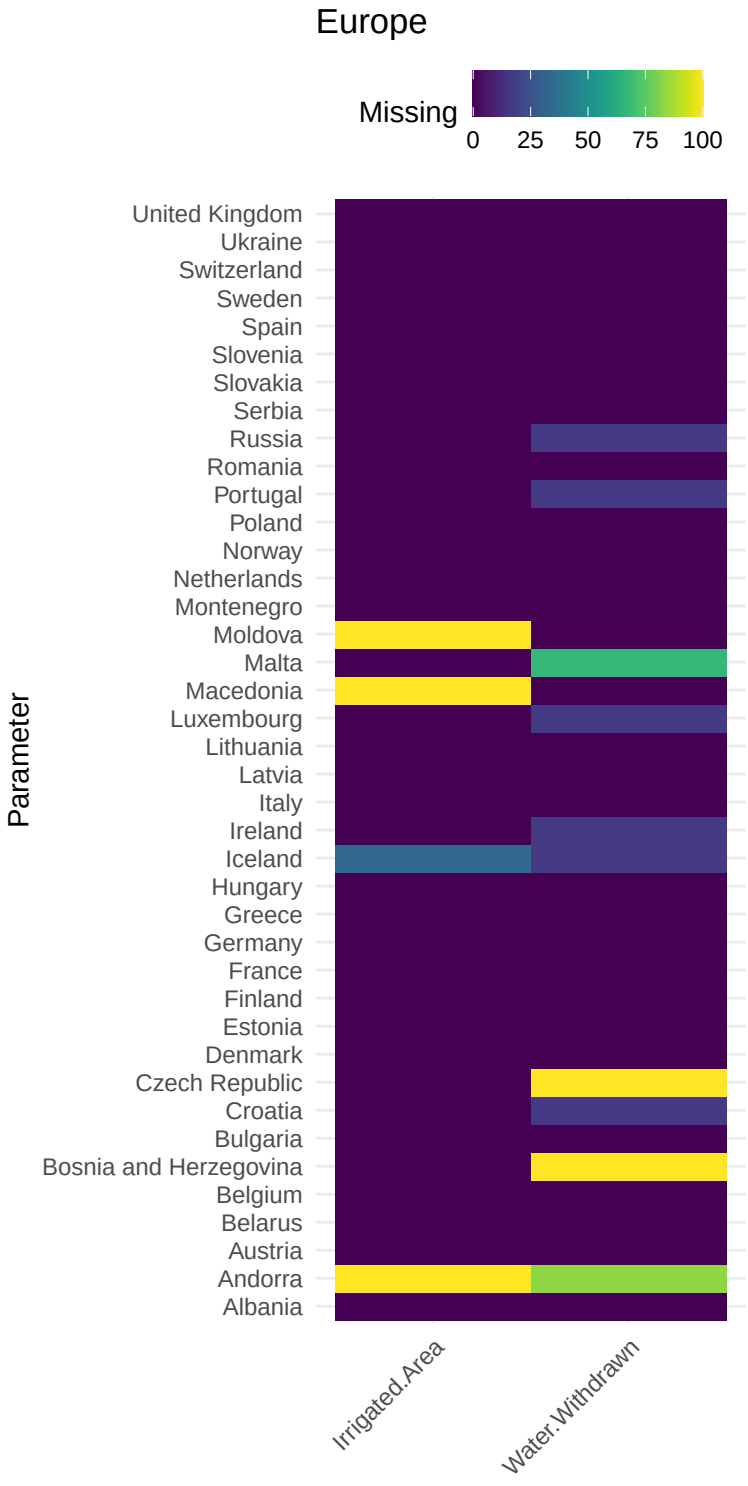
```
##
## $Americas
```


Americas
Missing
0
25
50
75
100
Parameter
Venezuela
Uruguay
United States
Trinidad and Tobago
Suriname
Peru
Paraguay
Panama
Nicaragua
Mexico
Jamaica
Honduras
Haiti
Guyana
Guatemala
Grenada
El Salvador
Ecuador
Dominican Republic
Cuba
Costa Rica
Colombia
Chile
Canada
Brazil
Bolivia
Belize
Barbados
Bahamas
Argentina
Irrigated.Area
Water.Withdrawn

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


Asia
Missing
0
25
50
75
100
Parameter
Yemen
Vietnam
Uzbekistan
United Arab Emirates
Turkmenistan
Turkey
Thailand
Tajikistan
Syria
Sri Lanka
Singapore
Saudi Arabia
Qatar
Philippines
Pakistan
Oman
Nepal
Mongolia
Malaysia
Lebanon
Laos
Kyrgyzstan
Kuwait
Kazakhstan
Jordan
Japan
Israel
Iraq
Iran
Indonesia
India
Georgia
Cyprus
China
Cambodia
Brunei
Bhutan
Bangladesh
Bahrain
Azerbaijan
Armenia
Afghanistan
Irrigated.Area
Water.Withdrawn

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



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",
        strip.text = element_text(size = 7))
```

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.

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

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

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

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

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

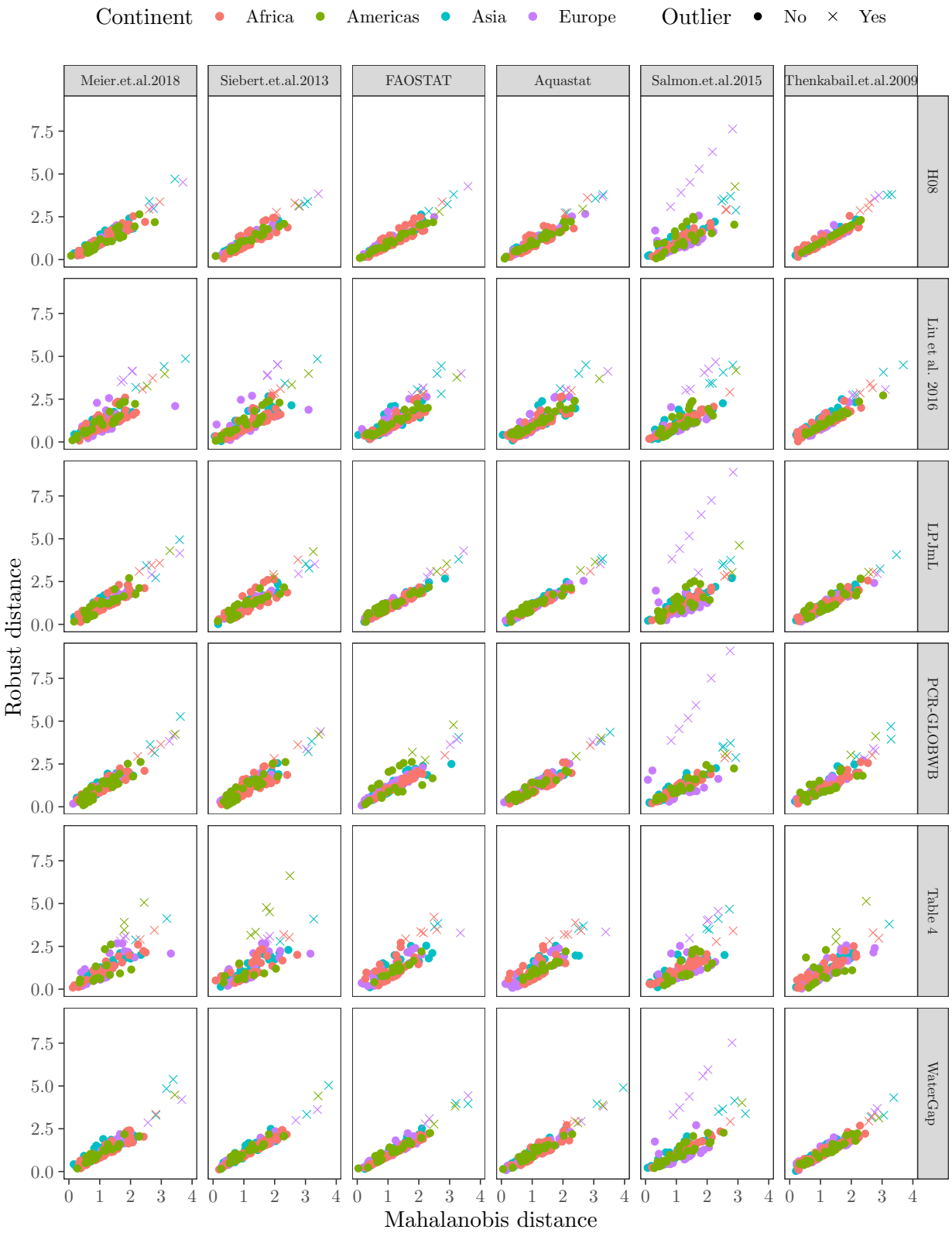


Figure 2: Bivariate outliers.

```
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.

3 Linear regressions and residuals

```
# LINEAR REGRESSIONS AND PULL RESIDUALS -------------------------------------

# Set grouping vectors
cols_group <- c("Continent", "Area.Dataset", "Water.Dataset")
all.cols <- c(cols_group, "Imputation.Method", "Iteration")

dt <- full.imput[, .(fit = list(list(lm(Water.Withdrawn ~ Irrigated.Area)))), all.cols] %>%
  .[, lapply(fit, function(x) lapply(x, function(y) augment(y))), all.cols]
```

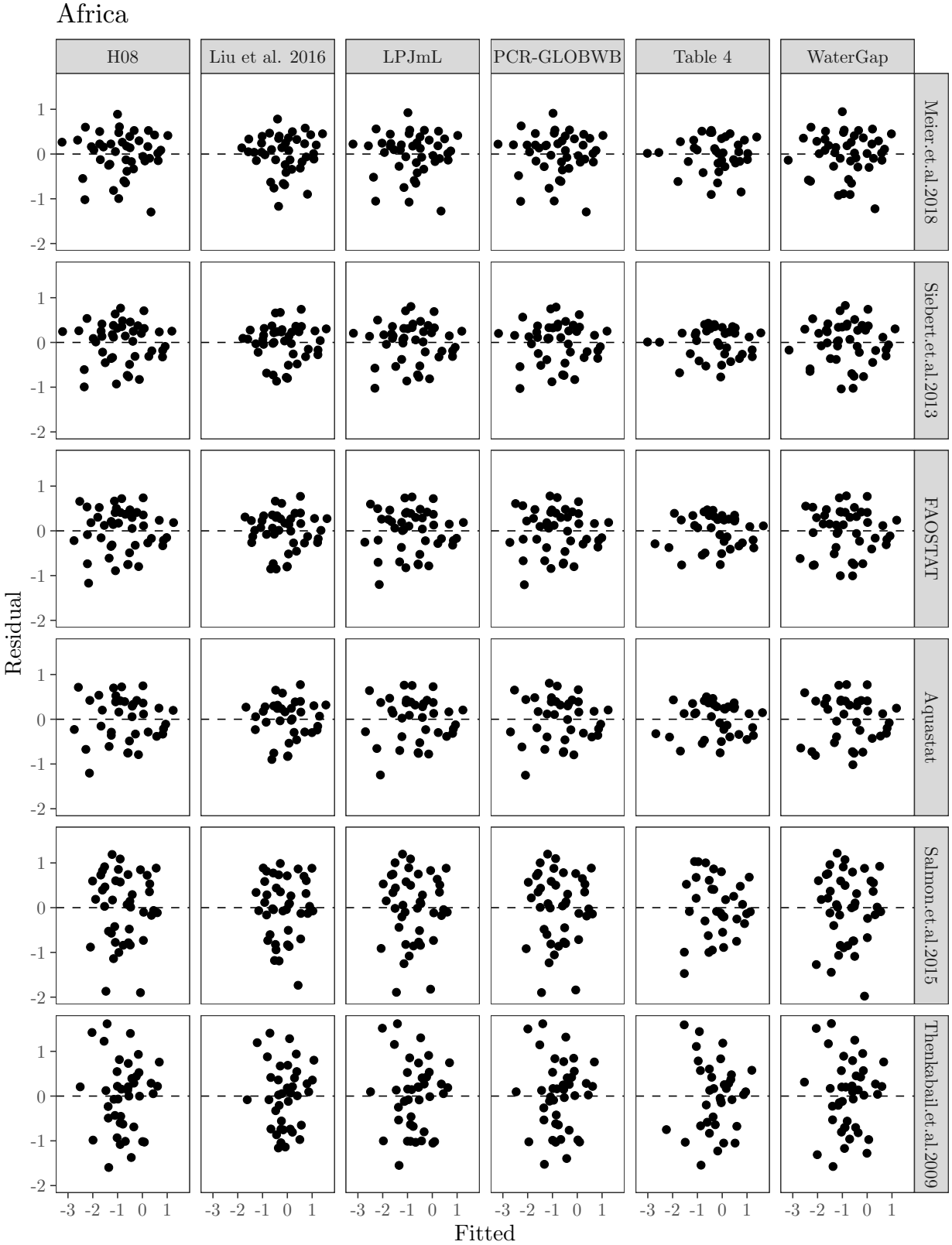


Figure 3: Residual versus fitted values.

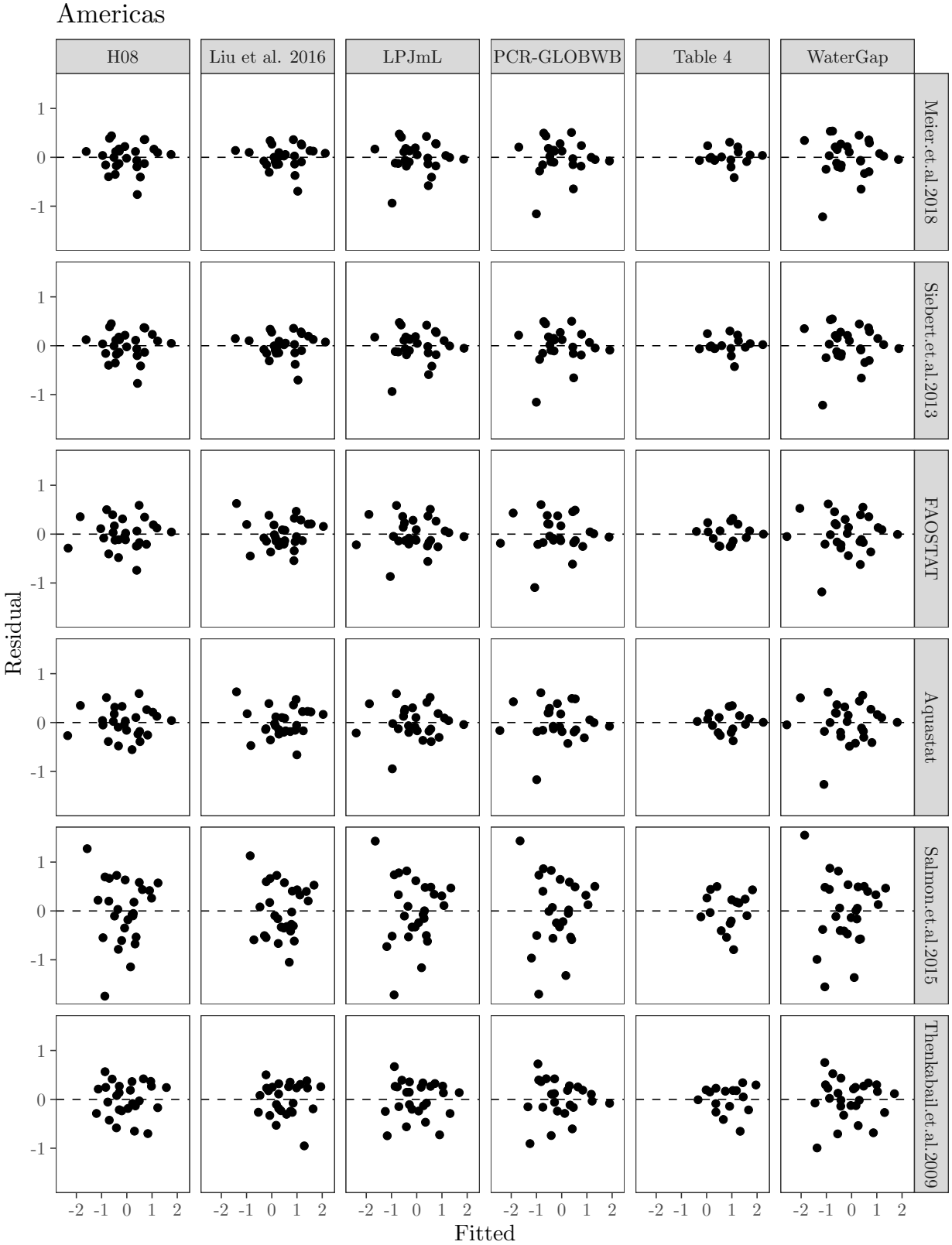


Figure 4: Residual versus fitted values.

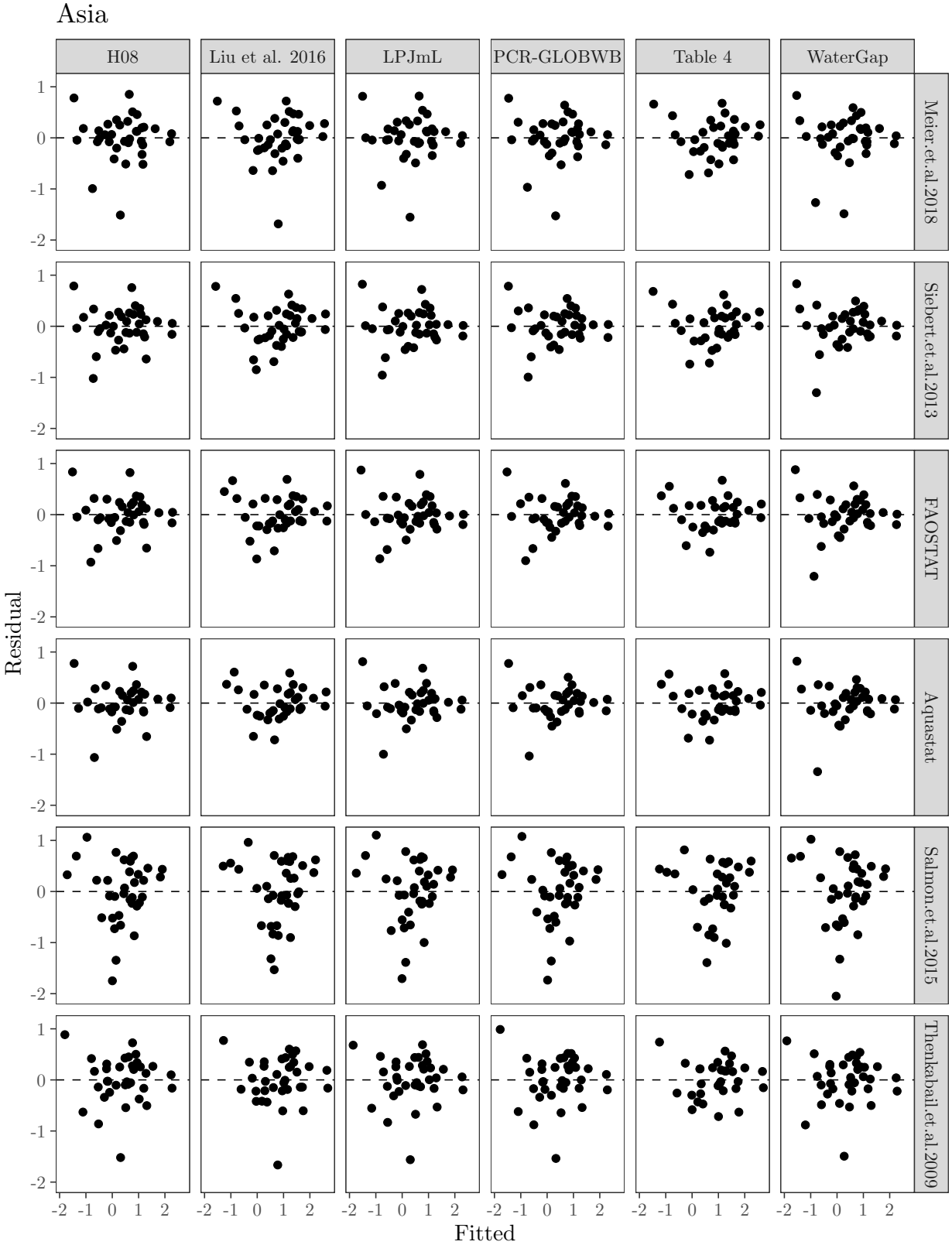


Figure 5: Residual versus fitted values.

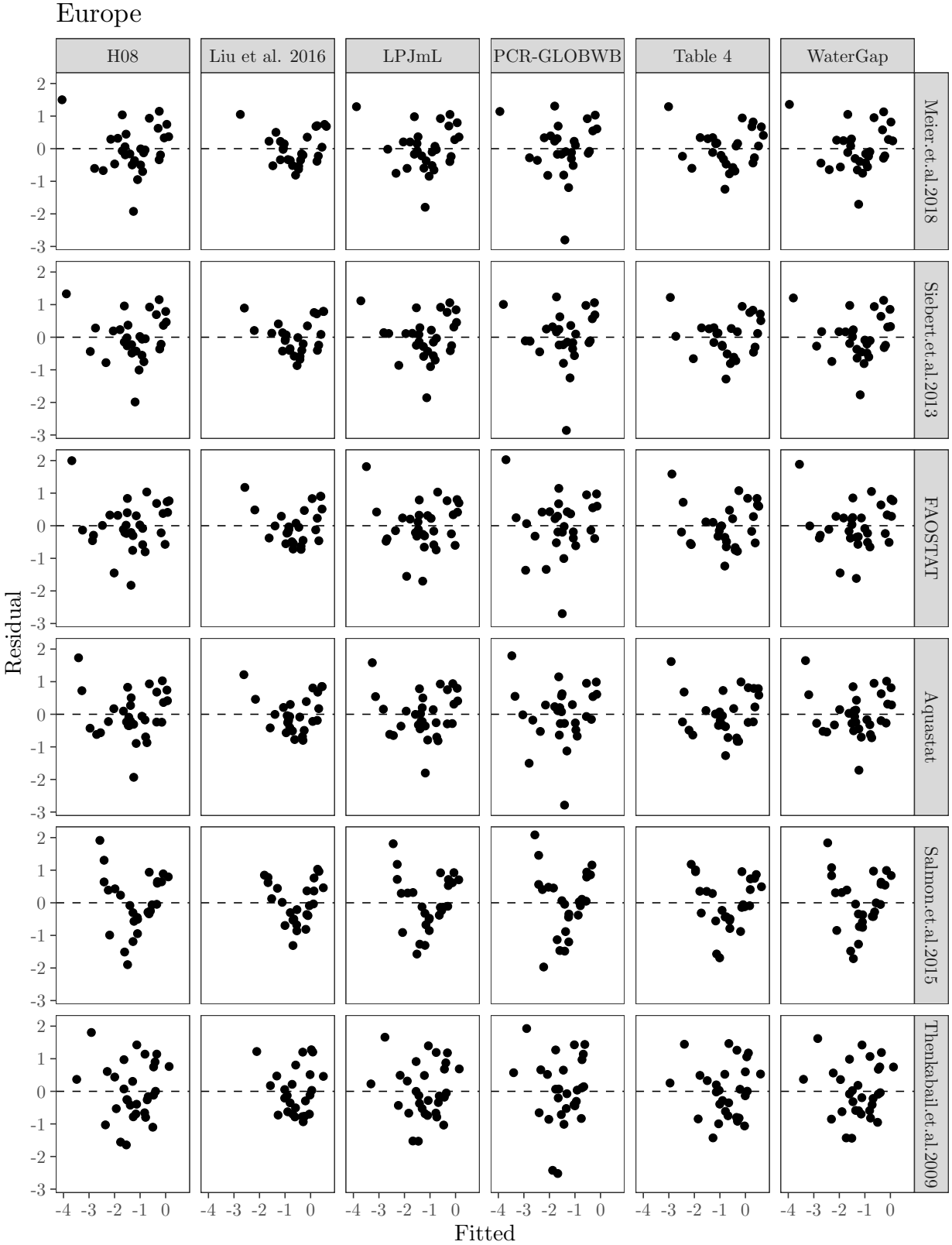


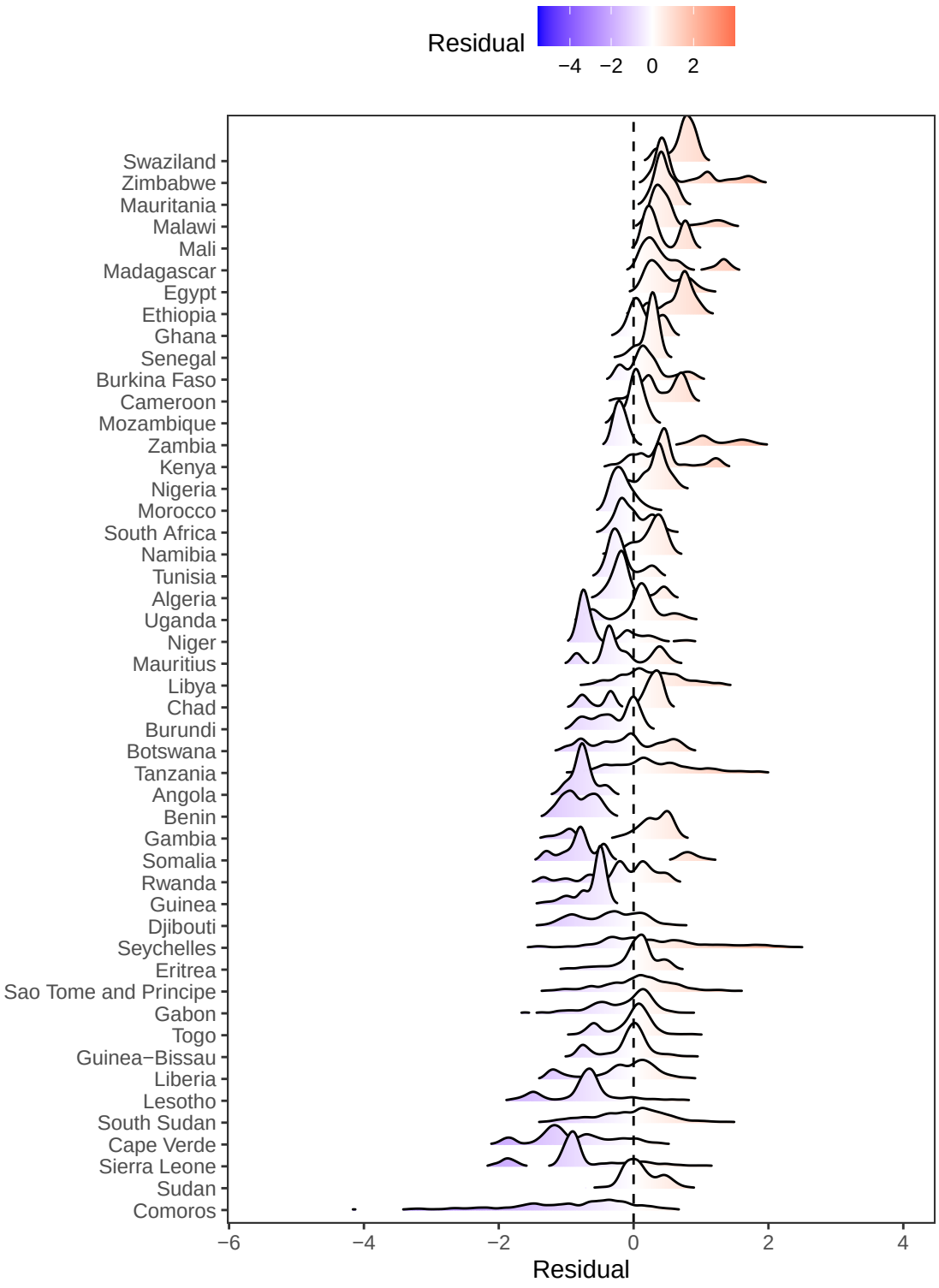
Figure 6: Residual versus fitted values.

```r
resid.dt <- dt[, lapply(V1, function(x) unlist(x$.resid)), all.cols] %>%
  .[, Country:= full.imput$Country] %>%
  .[, Method:= "Normal"]

dtR <- full.imput[, .(fit = list(list(rlm(Water.Withdrawn ~ Irrigated.Area,
                                          maxit = 60)))), all.cols] %>%
  .[, lapply(fit, function(x) lapply(x, function(y) augment(y))), all.cols]


resid.dtR <- dtR[, lapply(V1, function(x) unlist(x$.resid)), all.cols] %>%
  .[, Country:= full.imput$Country] %>%
  .[, Method:= "Robust"]

final.resid <- rbind(resid.dt, resid.dtR)
```

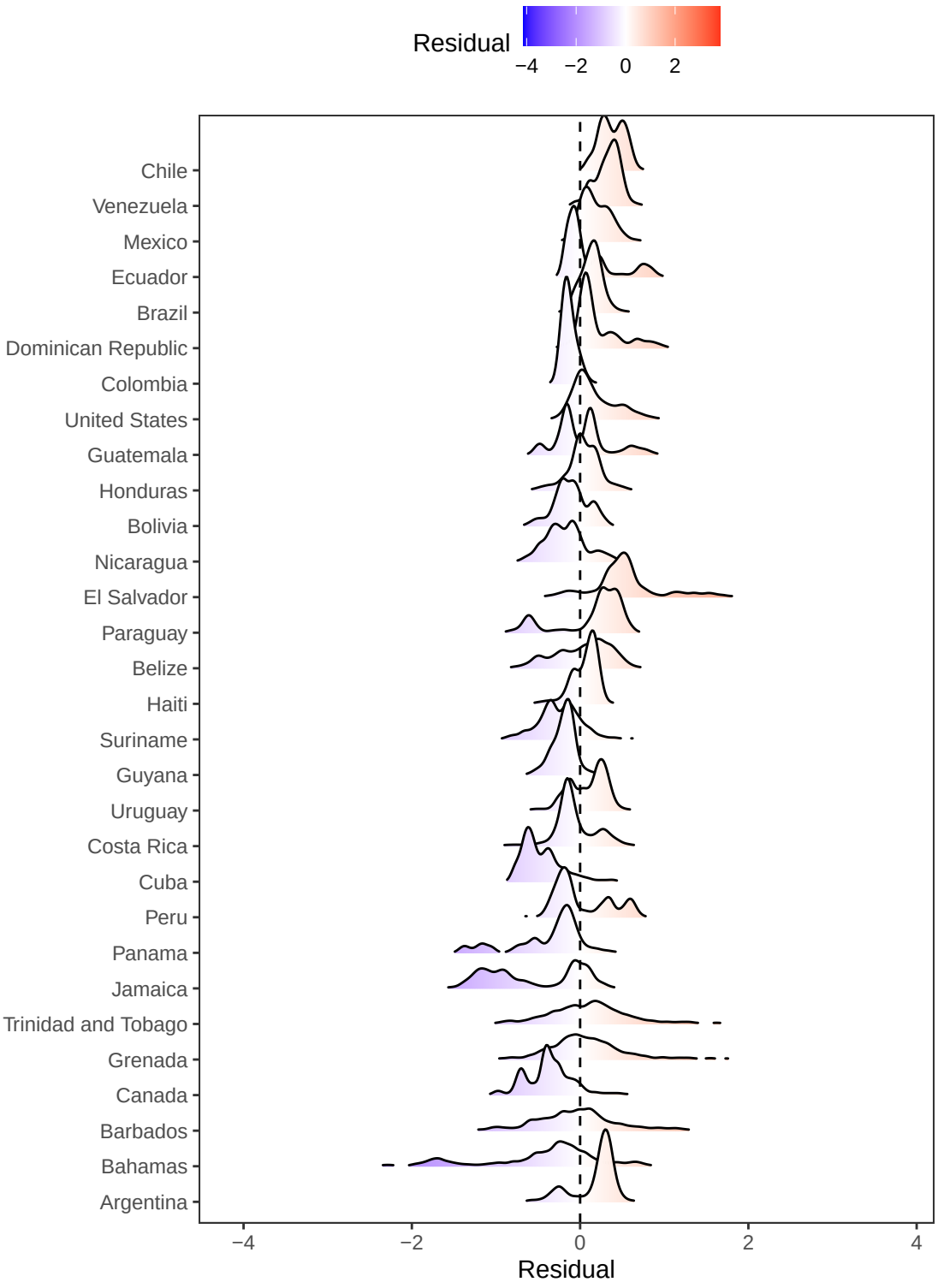
```r
 # PLOT RESIDUALS ------------------------------------------------------------

# function to plot
gg_bar <- gg_ridge <- list()
for(i in c("Africa", "Americas", "Asia", "Europe")) {
  country.vector <- final.resid[Continent == i] %>%
    .[order(V1)] %>%
    .$Country %>%
    unique(.)
  gg_ridge[[i]] <- final.resid[Continent == i] %>%
    .[, Country:= factor(Country, levels = country.vector)] %>%
    ggplot(., aes(V1, Country, fill = ..x..)) +
    geom_density_ridges_gradient(scale = 3, rel_min_height = 0.01) +
    geom_vline(xintercept = 0,
               lty = 2) +
    scale_fill_gradient2(name = "Residual",
                         low = "blue", mid = "white",
                         high = "red", midpoint = 0) +
    labs(x = "Residual",
         y = "") +
    theme_AP() +
    theme(legend.position = "top")
  gg_bar[[i]] <- final.resid[Continent == i] %>%
    .[, sum(V1 > 0) / .N, Country] %>%
    ggplot(., aes(reorder(Country, V1), V1,
                  fill = ..y..)) +
    geom_bar(stat = "identity") +
    coord_flip() +
    labs(y = "Residuals > 0 (%)",
         x = "") +
    theme_AP() +
    theme(legend.position = "top")
```

```
}

gg_ridge
```

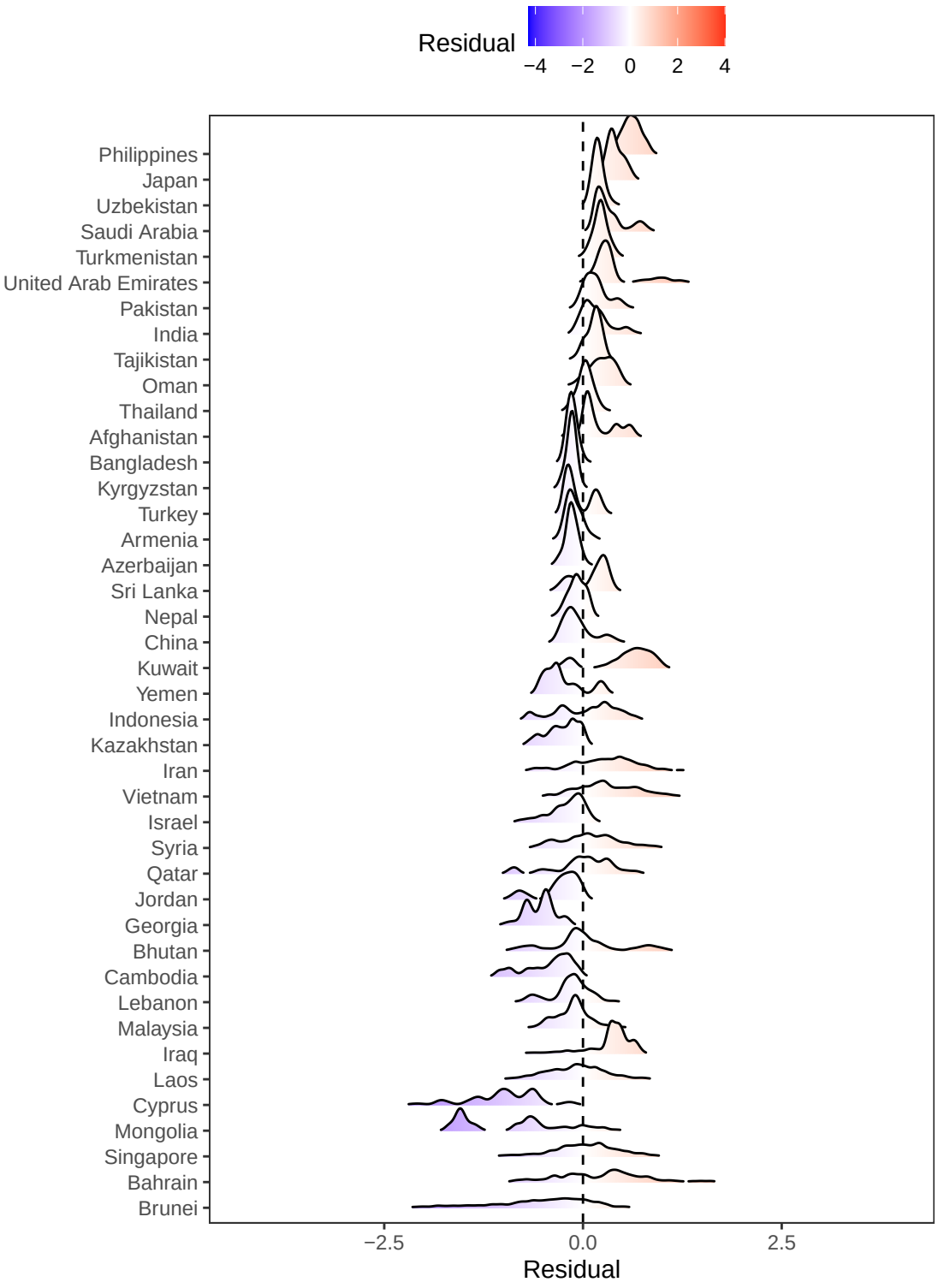
```
## $Africa

## Picking joint bandwidth of 0.0565
```



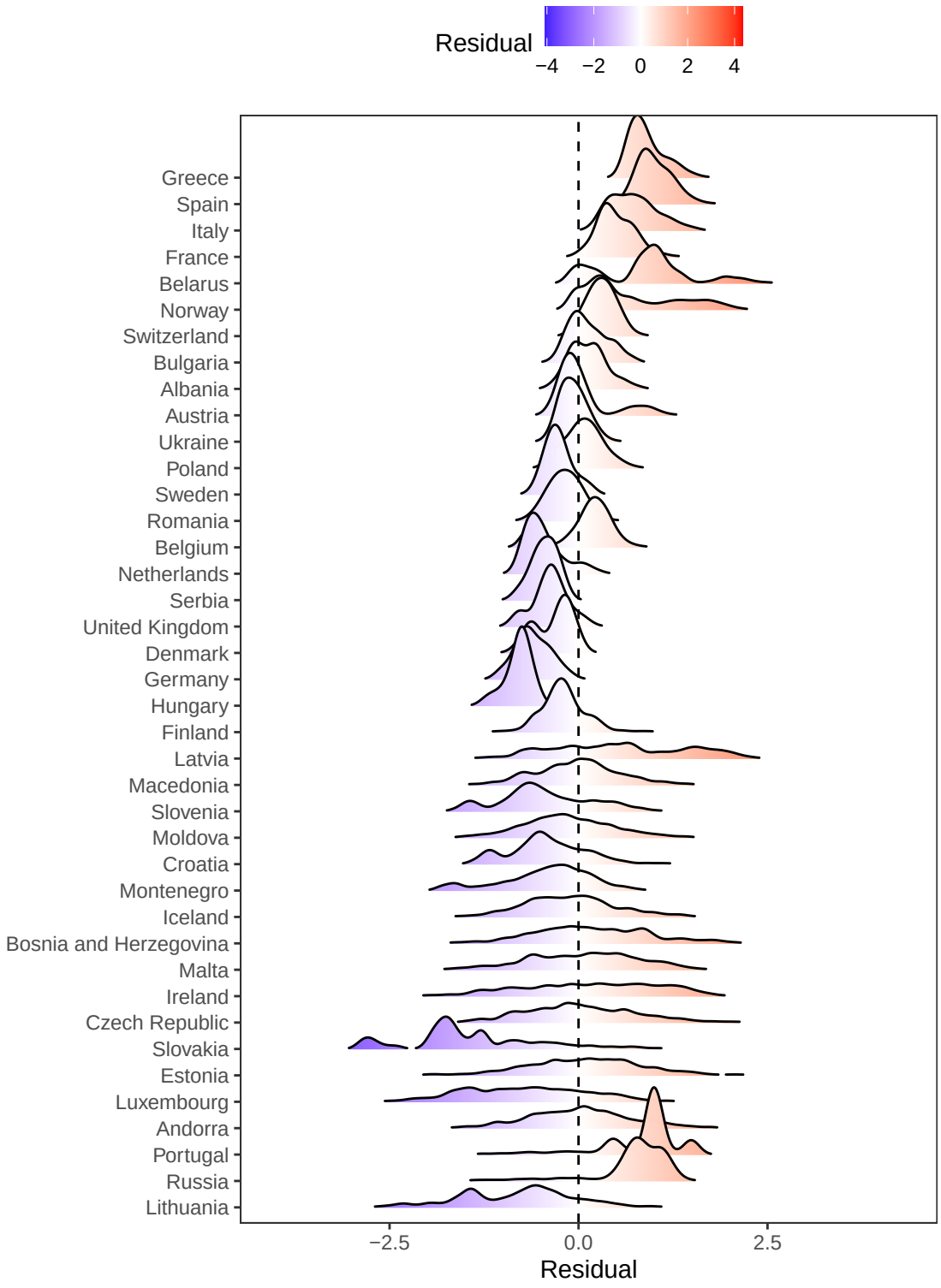

```
##
## $Americas

## Picking joint bandwidth of 0.0428
```




```
##
## $Asia

## Picking joint bandwidth of 0.039
```




```
##
## $Europe

## Picking joint bandwidth of 0.0737
```




```
# EXPORT FULL DATASET WITHOUT MISSING VALUES ----------------------------------------

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

4 Compute all possible beta values

```
# COMPUTE BETA AND R^2 ------------------------------------------------------

regressions <- full.imput[, .(Normal = coef(lm(Water.Withdrawn ~ Irrigated.Area)),
                              Robust = coef(rlm(Water.Withdrawn ~ Irrigated.Area, maxit = 60))
                              r.squared = summary(lm(Water.Withdrawn ~ Irrigated.Area))$r.squa
                      all.cols]

results <- regressions[, Type:= rep(c("Intercept", "Beta"), times = nrow(.SD) / 2)] %>%
  melt(., measure.vars = c("Normal", "Robust"), variable.name = "Regression") %>%
  tidyr::spread(., Type, value) %>%
  .[, index:= paste(Continent, Area.Dataset, Water.Dataset, Regression,
                    Imputation.Method, Iteration, sep = "_")]
```

```
## Warning in `[.data.table`(., , `:=`(index, paste(Continent, Area.Dataset, :
## Invalid .internal.selfref detected and fixed by taking a (shallow) copy of the
## data.table so that := can add this new column by reference. At an earlier point,
## this data.table has been copied by R (or was created manually using structure()
## or similar). Avoid names<- and attr<- which in R currently (and oddly) may
## copy the whole data.table. Use set* syntax instead to avoid copying: ?set, ?
## setnames and ?setattr. If this message doesn't help, please report your use case
## to the data.table issue tracker so the root cause can be fixed or this message
## improved.
```

```
# EXPORT BETA AND R^2 RESULTS -----------------------------------------------

results <- regressions[, Type:= rep(c("Intercept", "Beta"), times = nrow(.SD) / 2)] %>%
  melt(., measure.vars = c("Normal", "Robust"), variable.name = "Regression") %>%
  spread(., Type, value) %>%
  .[, index:= paste(Continent, Area.Dataset, Water.Dataset, Regression,
                    Imputation.Method, Iteration, sep = "_")]
```

```
## Warning in `[.data.table`(., , `:=`(index, paste(Continent, Area.Dataset, :
## Invalid .internal.selfref detected and fixed by taking a (shallow) copy of the
## data.table so that := can add this new column by reference. At an earlier point,
## this data.table has been copied by R (or was created manually using structure()
## or similar). Avoid names<- and attr<- which in R currently (and oddly) may
## copy the whole data.table. Use set* syntax instead to avoid copying: ?set, ?
## setnames and ?setattr. If this message doesn't help, please report your use case
## to the data.table issue tracker so the root cause can be fixed or this message
## improved.
```

```
fwrite(results, "results.csv")
```

4.1 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).

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

lookup <- setkey(results, index)

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

5 The sample matrix

This section designs the sample matrix that will be used to compute the model output, i.e. the β value defined by 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 (i.e. the trigger to select which irrigated area data set to use) and X_2 (i.e. the trigger to select which irrigation water withdrawal data set to use). We also define the initial sample size of the sample matrix ($N = 2^{13}$), for which we use Sobol' Quasi Random Numbers, and create a function to transform its columns to their appropriate distributions.

```r
# 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", 1:5, sep = "")

# Select sample size
n <- 2 ^ 13

# Define order
order <- "third"
```

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

# Create an A, B and AB matrices for each continent
sample.matrix <- lapply(Continents, function(Continents)
  sobol_matrices(N = n,
                 params = parameters,
                 order = order) %>%
    data.table())

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

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

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

```r
# 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 * (2 - 1 + 1)) + 1] %>%
    .[, X3:= ifelse(X3==1, "Normal", "Robust")] %>%
    .[, 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]
}

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

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

We also print the sample matrix to better inform the reader about our analysis:

- X1: Trigger that decides which irrigated area data set to use.
- X2: Trigger that decides which irrigation water withdrawal data set to use.
- X3: Trigger that decides whether to select robust, non-robust or heteroskedasticity-corrected methods.
- X4: Trigger that decides which imputation method should be used for missing values.
- X5: Trigger that decides which iteration of a given imputation method to use.
- X6: Trigger that decides which β value to extract from the pool of bootstrap replicas, with 1 and 500 being respectively the β value with the hlowest and highest values.

```r
# PRINT SAMPLE MATRIX ------------------------------------------------------

print(sample.matrix.dt)
```

```
##         Continent                    X1              X2     X3        X4 X5
##      1:    Africa      Meier.et.al.2018        WaterGap Robust norm.boot 11
```

```
##       2:    Africa     Salmon.et.al.2015             H08 Robust        rf 16
##       3:    Africa                FAOSTAT         Table 4 Normal  norm.nob  6
##       4:    Africa    Siebert.et.al.2013      PCR-GLOBWB Robust        rf 18
##       5:    Africa Thenkabail.et.al.2009 Liu et al. 2016 Normal norm.boot  8
##      ---
## 884732:    Europe                FAOSTAT             H08 Robust  norm.nob 11
## 884733:    Europe     Salmon.et.al.2015         Table 4 Normal norm.boot  1
## 884734:    Europe      Meier.et.al.2018           LPJmL Robust norm.boot  6
## 884735:    Europe              Aquastat        WaterGap Normal        rf 16
## 884736:    Europe              Aquastat             H08 Robust norm.boot  3
```

6 The model

The model is simply a one-line function that runs in the sample matrix as follows: at the v-th row, it creates a vector with all the conditions set by $X_1^{(v)}, ..., X_6^{(v)}$, and uses this vector to extract from the lookup table the $\beta^{(v)}$ value according to these conditions.

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

model <- function(X) lookup[.(paste0(X[, 1:6], collapse = "_"))][, c(Intercept, Beta, r.squared
```

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

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

# 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 <- cbind(sample.matrix.dt, data.table(do.call(rbind, Y))) %>%
  setnames(., c("V1", "V2", "V3"), c("Intercept", "Beta", "r.squared"))

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

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

7 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.

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

# Plot r2
a <- ggplot(AB.dt, aes(r.squared)) +
  geom_histogram(color = "black", fill = "white") +
  theme_AP() +
  labs(x = expression(r ^ 2),
       y = "Density") +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  facet_wrap(~Continent, ncol = 4) +
  theme(panel.spacing.x = unit(4, "mm"))

# Plot beta
b <- ggplot(AB.dt, aes(Beta)) +
  geom_histogram(color = "black", fill = "white") +
  theme_AP() +
  geom_vline(xintercept = 1,
             lty = 2,
             color = "red") +
  labs(x = "$\\beta$",
       y = "") +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  facet_wrap(~Continent, ncol = 4) +
  theme(panel.spacing.x = unit(4, "mm"))


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

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

```r
# CHECK THE COMBINATIONS LEADING TO HIGHEST R2 VALUES (first 200) -------------
```

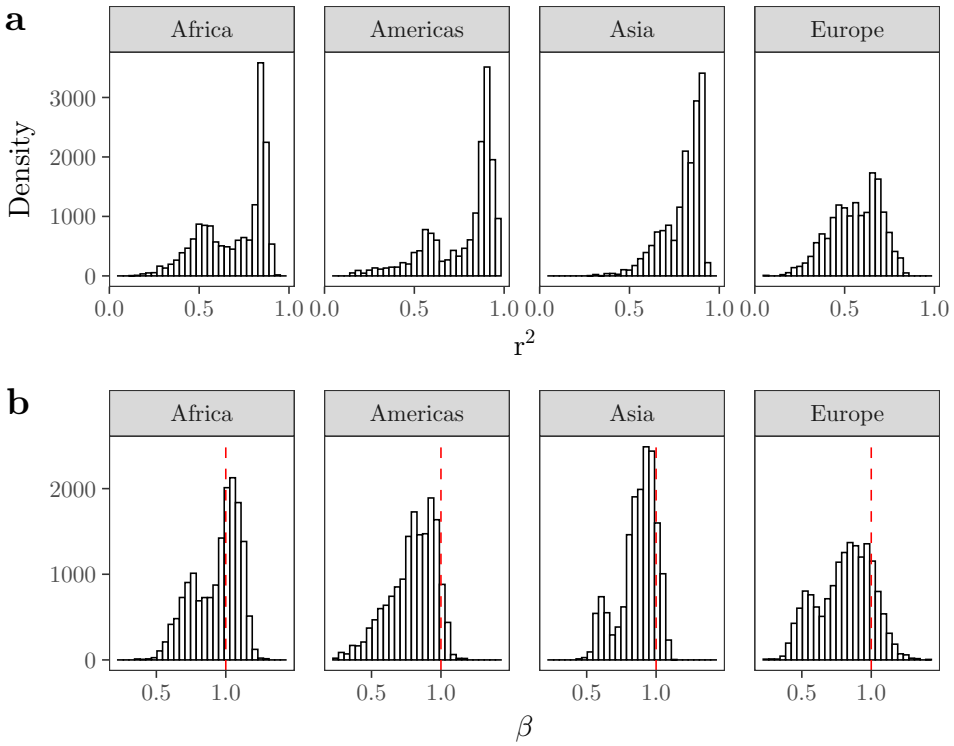


Figure 7: Uncertainty in the model output. a) Uncertainty in the model goodness of fit. All sources of uncertainty have been taken into account except the selection between OLS and OLS robust (trigger X2). Robust OLS does not allow to compute r^2. b) Uncertainty in the slope.

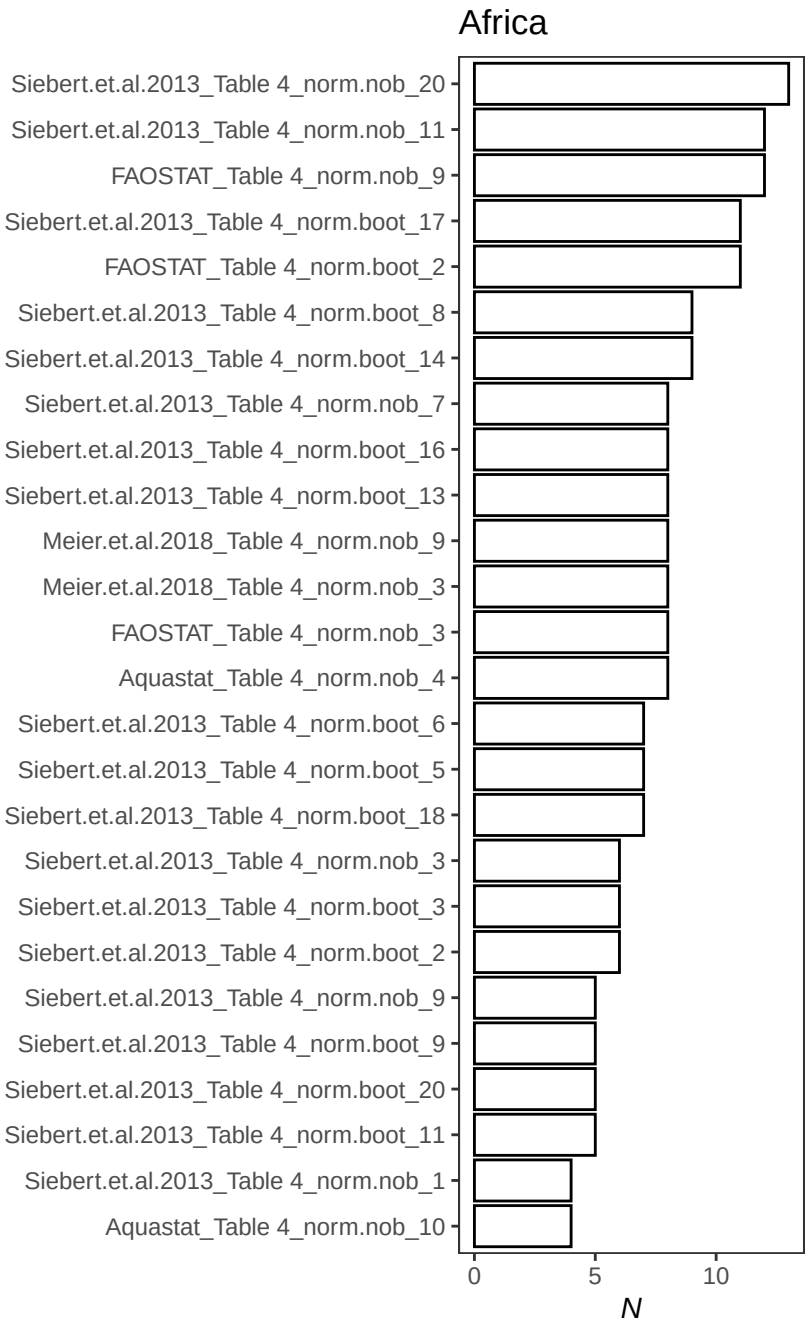
```
# Check min and max of first 200 r.squared values
AB.dt[order(-r.squared), head(.SD, 200), Continent][
  , .(min = min(r.squared), max = max(r.squared)), Continent]
```

```
##    Continent       min       max
## 1:  Americas 0.9672518 0.9766420
## 2:      Asia 0.9208109 0.9358844
## 3:    Africa 0.9031553 0.9235314
## 4:    Europe 0.8048955 0.8489623
```

```
# Create plot
dd <- AB.dt[order(-r.squared), head(.SD, 200), Continent] %>%
  .[, ID:= paste(X1, X2, X4, X5, sep = "_")] %>%
  .[, .N, .(Continent, ID)]

tmp <- split(dd, dd$Continent)
gg <- list()
for(i in names(tmp)) {
  gg[[i]] <- ggplot(tmp[[i]], aes(reorder(ID, N), N)) +
    geom_bar(stat = "identity",
             color = "black",
             fill = "white") +
    coord_flip() +
```

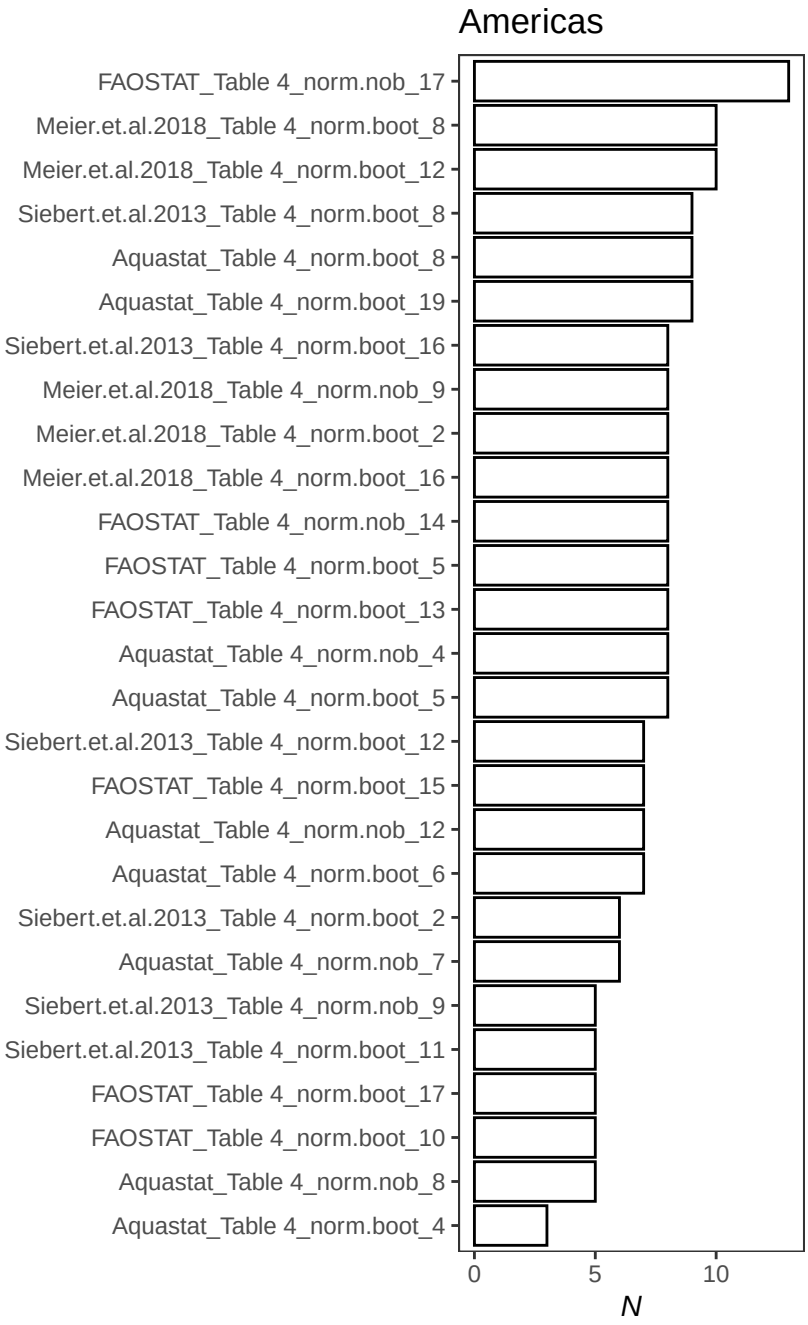
```
    labs(y = expression(italic(N)),
         x = "") +
    theme_AP() +
    ggtitle(names(tmp[i]))
}

gg
```

```
## $Africa
```

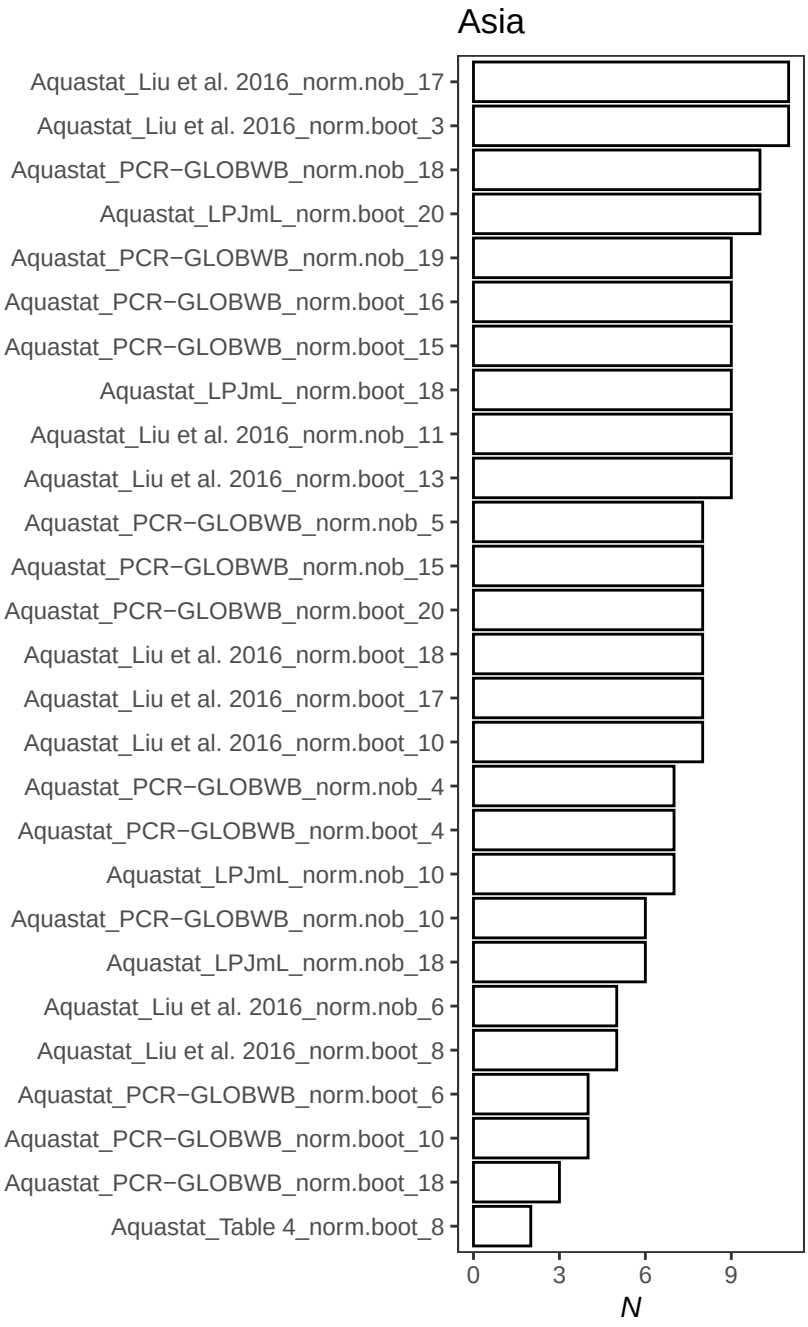



```
##
```

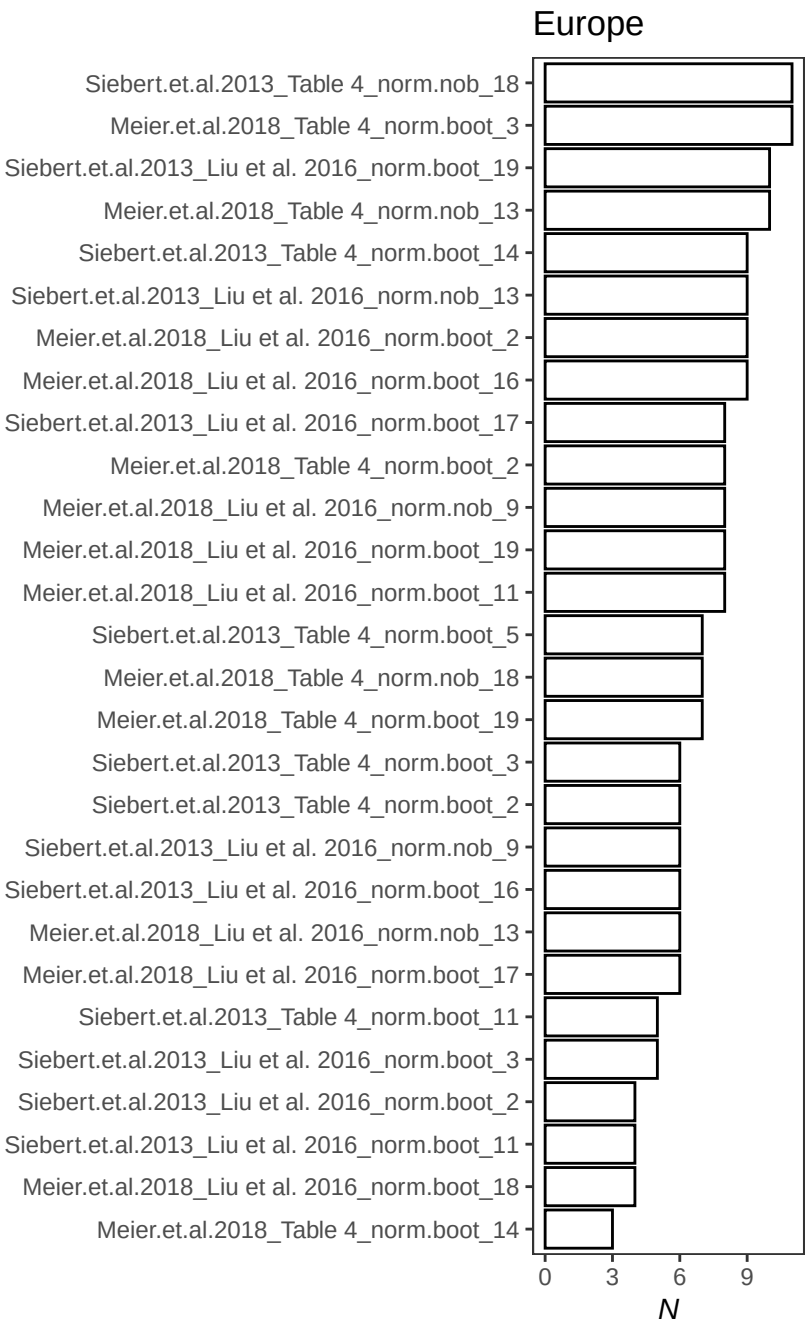
```
## $Americas
```




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


Asia
Aquastat_Liu et al. 2016_norm.nob_17
Aquastat_Liu et al. 2016_norm.boot_3
Aquastat_PCR−GLOBWB_norm.nob_18
Aquastat_LPJmL_norm.boot_20
Aquastat_PCR−GLOBWB_norm.nob_19
Aquastat_PCR−GLOBWB_norm.boot_16
Aquastat_PCR−GLOBWB_norm.boot_15
Aquastat_LPJmL_norm.boot_18
Aquastat_Liu et al. 2016_norm.nob_11
Aquastat_Liu et al. 2016_norm.boot_13
Aquastat_PCR−GLOBWB_norm.nob_5
Aquastat_PCR−GLOBWB_norm.nob_15
Aquastat_PCR−GLOBWB_norm.boot_20
Aquastat_Liu et al. 2016_norm.boot_18
Aquastat_Liu et al. 2016_norm.boot_17
Aquastat_Liu et al. 2016_norm.boot_10
Aquastat_PCR−GLOBWB_norm.nob_4
Aquastat_PCR−GLOBWB_norm.boot_4
Aquastat_LPJmL_norm.nob_10
Aquastat_PCR−GLOBWB_norm.nob_10
Aquastat_LPJmL_norm.nob_18
Aquastat_Liu et al. 2016_norm.nob_6
Aquastat_Liu et al. 2016_norm.boot_8
Aquastat_PCR−GLOBWB_norm.boot_6
Aquastat_PCR−GLOBWB_norm.boot_10
Aquastat_PCR−GLOBWB_norm.boot_18
Aquastat_Table 4_norm.boot_8
0
3
6
9
N

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




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

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

```
##    Continent sublinear superlinear
## 1:    Africa 0.5439453  0.45605469
## 2:  Americas 0.9361572  0.06384277
## 3:      Asia 0.8648071  0.13519287
## 4:    Europe 0.8267822  0.17321777
```

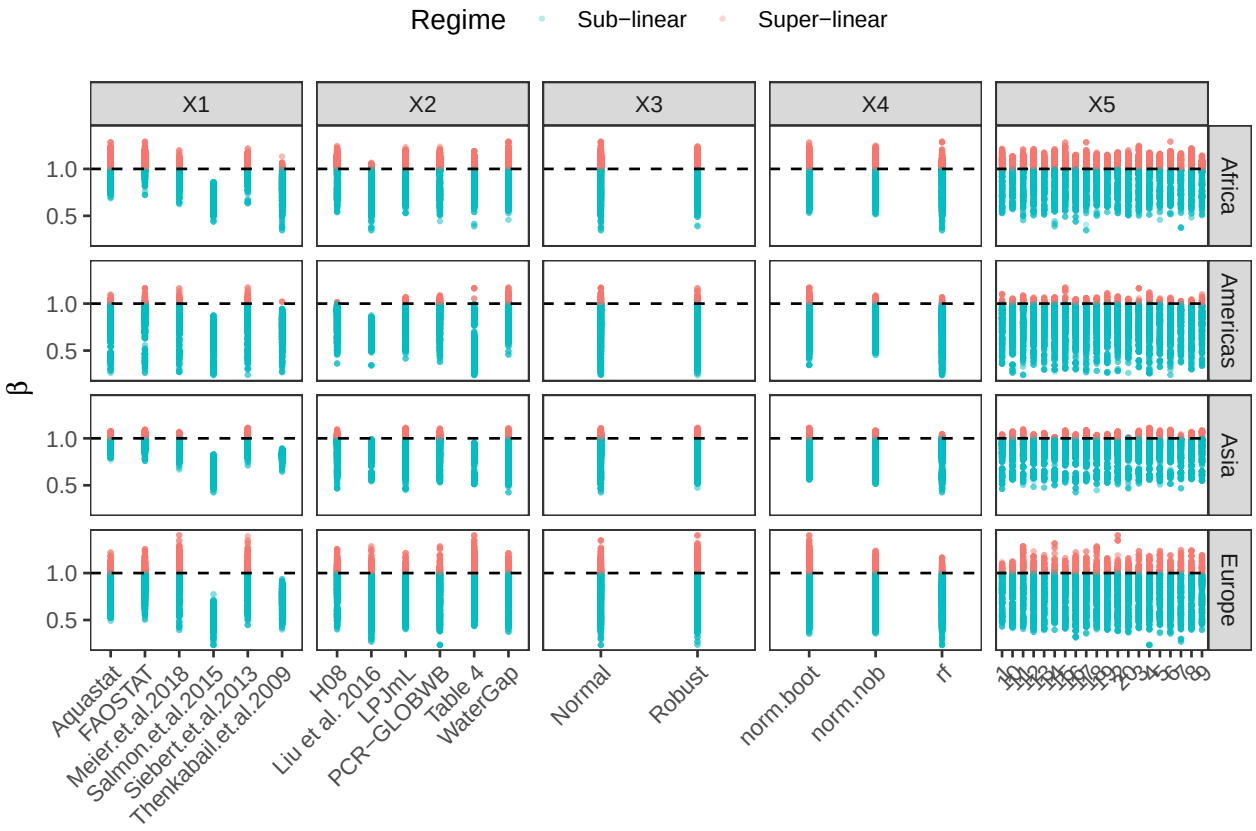
```
AB.dt[, .(">90" = sum(r.squared > 0.9) / .N,
            "75-90" = sum(r.squared > 0.75 & r.squared < 0.9) / .N,
            "50-75" = sum(r.squared > 0.50 & r.squared < 0.75) / .N,
            "25-50" = sum(r.squared > 0.25 & r.squared < 0.5) / .N), Continent]
```

```
##    Continent        >90      75-90     50-75      25-50
## 1:    Africa 0.01513672 0.50408936 0.3066406 0.16699219
## 2:  Americas 0.32006836 0.34979248 0.2324829 0.08227539
## 3:      Asia 0.13842773 0.59545898 0.2493286 0.01678467
## 4:    Europe 0.00000000 0.06994629 0.5859375 0.32513428
```

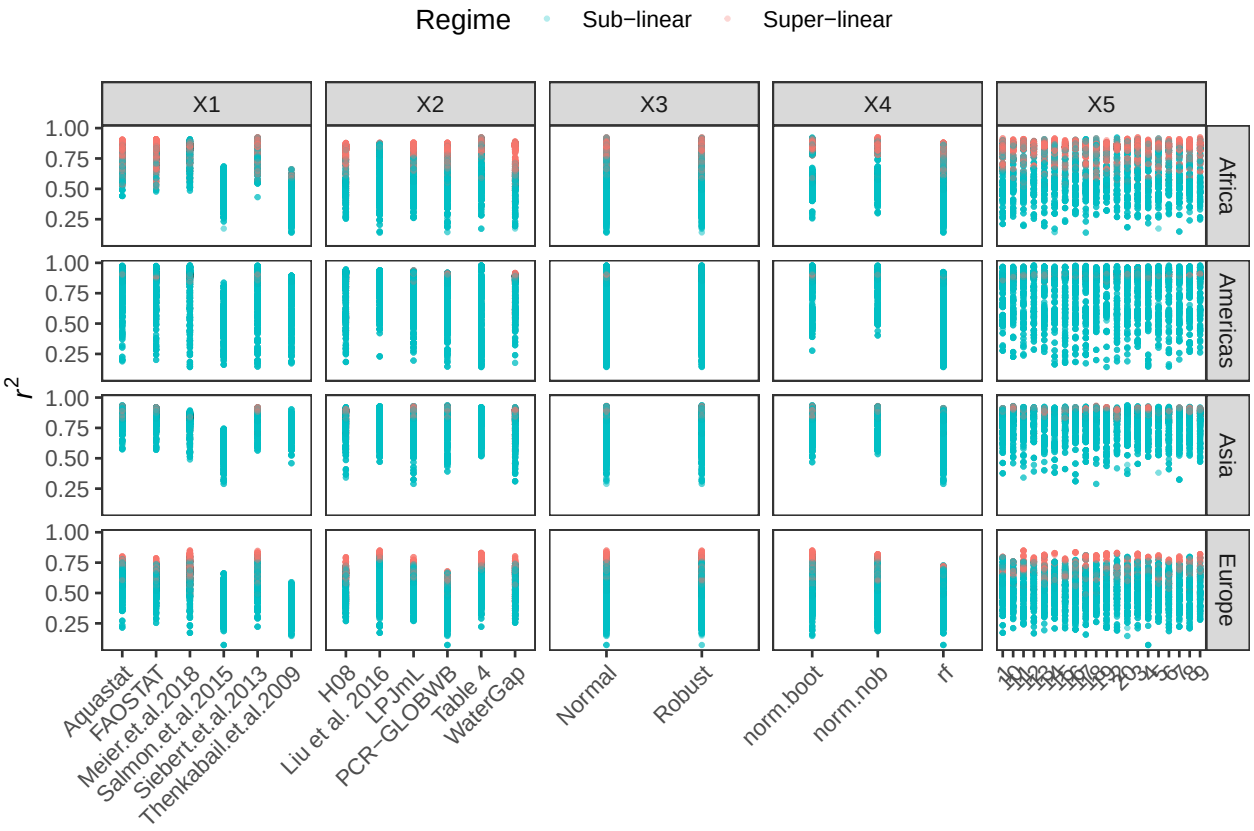
8 Sensitivity analysis

Here we conduct the sensitivity analysis to appraise how much uncertainty in β is conveyed by each of the triggers. We use the Saltelli et al. (2010) and the Jansen (1999) estimators to compute first-order effects (S_i) and total-order effects (S_{Ti}) respectively, as per the established best practices.Figure 8 plots the results. It seems that the trigger conveying the most uncertainty in the value of β is X1, which reflects the uncertainty in the current extension of irrigation, followed by X6, which represents the uncertainty in the true value of β. X3 (i.e. the trigger that decides whether to correct the regressions for outliers/heteroskedasticity) does not have any effect. It is worth mentioning the presence of significant interactions between the triggers: below we show that they are responsible for 10–25% of the uncertainty in β. Figures 9 and 10 indicate the presence of relevant second and third-order interactions in the case of the Americas and Europe.

```
# PLOT SCATTERPLOTS OF PARAMETERS VS MODEL OUTPUT -----------------------------

AB.dt <- AB.dt[, X5:= factor(X5, levels = as.factor(1:m.iterations))]

scatter.dt <- melt(AB.dt, measure.vars = paste("X", 1:5, sep = "")) %>%
  .[, Regime:= ifelse(Beta > 1, "Super-linear", "Sub-linear")]

# Beta
ggplot(scatter.dt, aes(value, Beta, color = Regime)) +
  geom_point(alpha = 0.3, size = 0.5) +
  facet_grid(Continent ~ variable,
             scales = "free_x") +
  geom_hline(yintercept = 1,
             lty = 2) +
  scale_color_manual(values = c("#00BFC4", "#F8766D")) +
  theme_AP() +
  labs(x = "", y = expression(beta)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "top")
```




```
# R squared
ggplot(scatter.dt, aes(value, r.squared, color = Regime)) +
  geom_point(alpha = 0.3, size = 0.5) +
  facet_grid(Continent ~ variable,
             scales = "free_x") +
  labs(x = "",
       y = expression(italic(r)^2)) +
  scale_color_manual(values = c("#00BFC4", "#F8766D")) +
  theme_AP() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "top")
```




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

# Number of bootstrap replicas
R <- 1000
```

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

# Beta
indices <- sample.matrix.dt[, sobol_indices(Y = Beta,
                                            N = n,
                                            params = parameters,
                                            first = "jansen",
                                            R = R,
                                            boot = TRUE,
                                            parallel = "multicore",
                                            ncpus = n_cores,
                                            order = "third"),
                               Continent]

# r squared
indicesR <- sample.matrix.dt[, sobol_indices(Y = r.squared,
                                             N = n,
                                             params = parameters,
                                             first = "jansen",
                                             R = R,
```

```
                                                boot = TRUE,
                                                parallel = "multicore",
                                                ncpus = n_cores,
                                                order = "third"),
                                  Continent]

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

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

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

a <- ggplot(indices[sensitivity %in% c("Si", "Ti")], aes(parameters, original, fill = sensitivi
  geom_bar(stat = "identity",
           position = position_dodge(0.6),
           color = "black") +
  geom_errorbar(aes(ymin = low.ci,
                    ymax = high.ci),
                position = position_dodge(0.6)) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  facet_wrap(~Continent,
             ncol = 4) +
  labs(x = "",
       y = "Sobol' index") +
  scale_fill_discrete(name = "Sobol' indices",
                      labels = c(expression(S[italic(i)]),
                                 expression(T[italic(i)]))) +
  theme_AP() +
  theme(legend.position = "none")
```

```
b <- ggplot(indicesR[sensitivity %in% c("Si", "Ti")], aes(parameters, original, fill = sensitiv
  geom_bar(stat = "identity",
           position = position_dodge(0.6),
           color = "black") +
  geom_errorbar(aes(ymin = low.ci,
                    ymax = high.ci),
                position = position_dodge(0.6)) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  facet_wrap(~Continent,
             ncol = 4) +
  labs(x = "",
       y = "Sobol' index") +
  scale_fill_discrete(name = "Sobol' indices",
                      labels = c(expression(S[italic(i)]),
                                 expression(T[italic(i)]))) +
  theme_AP() +
  theme(legend.position = "none")

legend <- get_legend(a + theme(legend.position = "top"))

all <- plot_grid(a, b, ncol = 1, align = "hv", labels = "auto")

plot_grid(legend, all, ncol = 1,  rel_heights = c(0.1, 1))
```

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

lapply(list(indices, indicesR), function(x)
  x[sensitivity == "Si"] %>%
      .[, sum(original), Continent])
```

```
## [[1]]
##    Continent        V1
## 1:    Africa 0.8634830
## 2:  Americas 0.7679276
## 3:      Asia 0.9028628
## 4:    Europe 0.8029202
##
## [[2]]
##    Continent        V1
## 1:    Africa 0.8834140
## 2:  Americas 0.6773428
## 3:      Asia 0.7624898
## 4:    Europe 0.7807006
```

```
# PLOT SOBOL' INDICES (SECOND AND THIRD ORDER) -------------------------------

lapply(c("Sij", "Sijk"), function(x)
  ggplot(indices[sensitivity == x], aes(reorder(parameters, original), original)) +
```

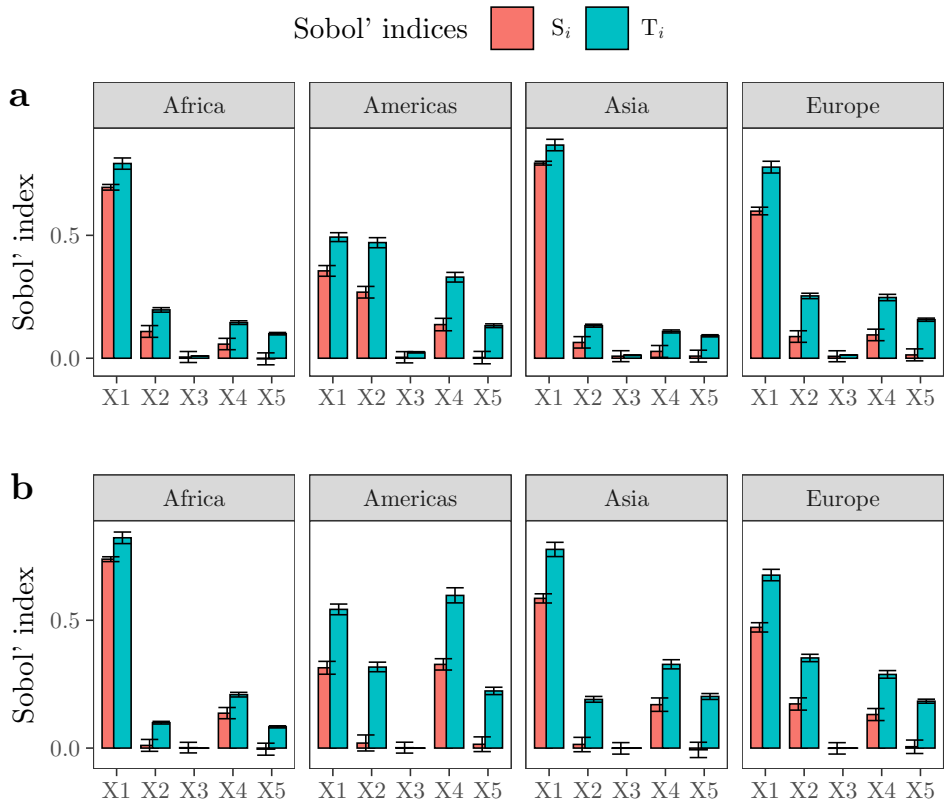


Figure 8: Sobol' indices.

```
  geom_point() +
  geom_errorbar(aes(ymin = low.ci,
                    ymax = high.ci)) +
  facet_wrap(~Continent) +
  theme_bw() +
  labs(x = "", y = "Sobol' index") +
  geom_hline(yintercept = 0, lty = 2, color = "red") +
  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),
        axis.text.x = element_text(angle = 45,
                                   hjust = 1)))
```

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

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

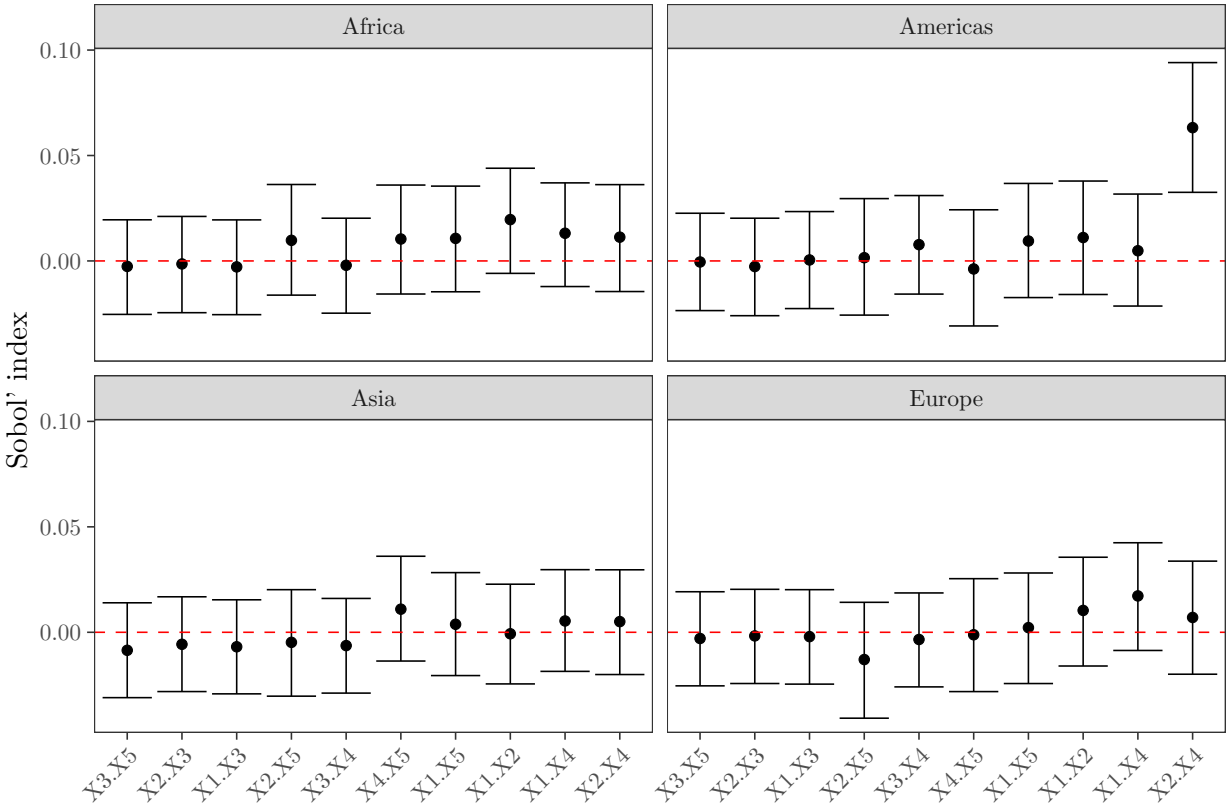


Figure 9: High-order interactions between the triggers.

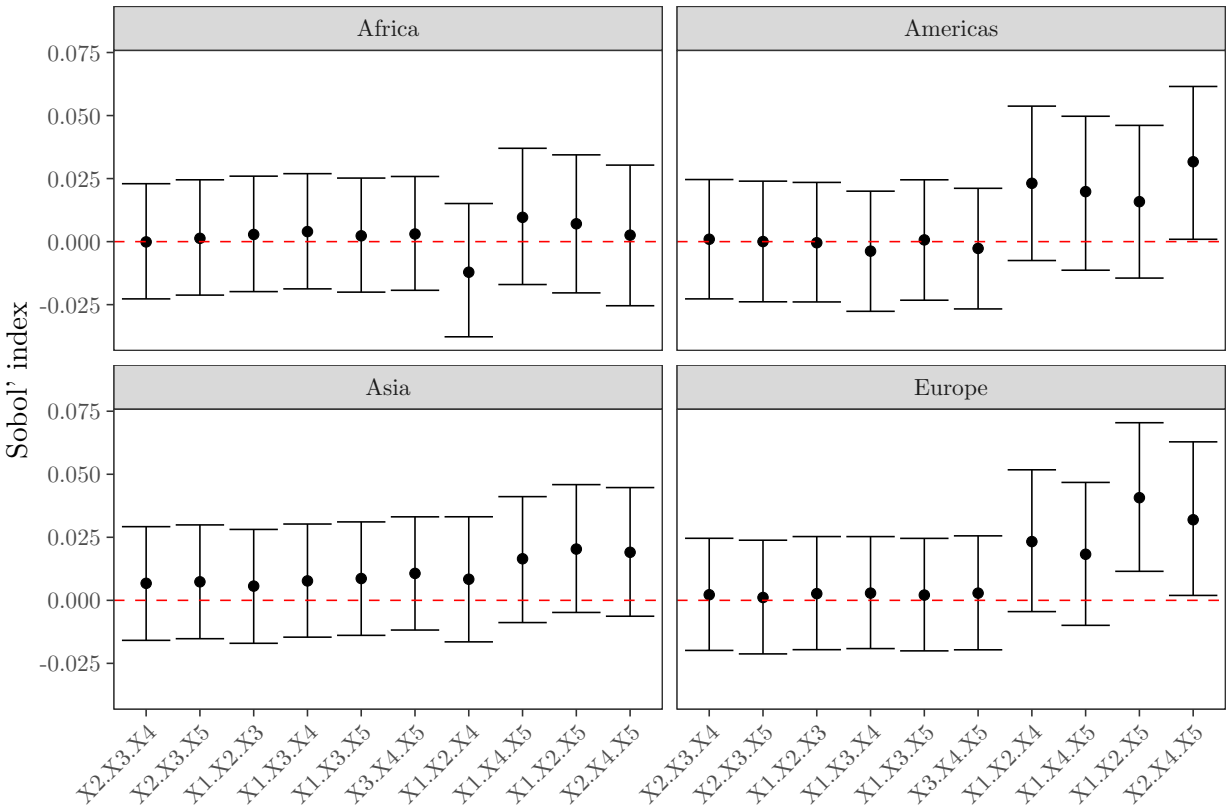


Figure 10: High-order interactions between the triggers.

9 Session information

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

sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.4
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] checkpoint_0.4.8  benchmarkme_1.0.3 tidyr_1.0.0       cowplot_1.0.0
##  [5] naniar_0.4.2      broom_0.5.3       ggridges_0.5.2    mice_3.7.0
##  [9] lattice_0.20-38   lmtest_0.9-37     zoo_1.8-7         sandwich_2.5-1
## [13] mvoutlier_2.0.9   sgeostat_1.0-27   complmrob_0.7.0   doParallel_1.0.15
## [17] iterators_1.0.12  foreach_1.4.8     MASS_7.3-51.5     boot_1.3-24
## [21] IDPmisc_1.1.20    dplyr_0.8.3       countrycode_1.1.0 rworldmap_1.3-6
## [25] sp_1.3-2          ncdf4_1.17        scales_1.1.0      sensobol_0.3
## [29] ggplot2_3.3.0     data.table_1.12.8
##
## 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            visdat_0.5.3
##   [7] mclust_5.4.5          pls_2.7-2             cvTools_0.3.2
##  [10] flexmix_2.3-15        mvtnorm_1.0-12        ranger_0.12.1
##  [13] codetools_0.2-16      splines_3.6.1         sROC_0.1-2
##  [16] robustbase_0.93-5     knitr_1.27            spam_2.5-1
##  [19] nloptr_1.2.1          robCompositions_2.2.0 cluster_2.1.0
##  [22] kernlab_0.9-29        httr_1.4.1            rrcov_1.5-2
##  [25] compiler_3.6.1        backports_1.1.5       assertthat_0.2.1
##  [28] Matrix_1.2-18         htmltools_0.4.0       tools_3.6.1
##  [31] dotCall64_1.0-0       gtable_0.3.0          glue_1.3.2
##  [34] maps_3.3.0            Rcpp_1.0.4            carData_3.0-3
##  [37] cellranger_1.1.0      vctrs_0.2.4           zCompositions_1.3.3-1
##  [40] nlme_3.1-143          fpc_2.2-4             gbRd_0.4-11
##  [43] xfun_0.12             laeken_0.5.0          stringr_1.4.0
##  [46] lme4_1.1-21           openxlsx_4.1.4        lifecycle_0.2.0
```

```
##  [49] pan_1.6               DEoptimR_1.0-8        VIM_4.8.0
##  [52] hms_0.5.3             RColorBrewer_1.1-2    fields_10.0
##  [55] yaml_2.2.0            curl_4.3              NADA_1.6-1
##  [58] rpart_4.1-15          reshape_0.8.8         stringi_1.4.6
##  [61] maptools_0.9-9        pcaPP_1.9-73          e1071_1.7-3
##  [64] zip_2.0.4             bibtex_0.4.2.2        benchmarkmeData_1.0.3
##  [67] truncnorm_1.0-8       Rdpack_0.11-1         rlang_0.4.5
##  [70] pkgconfig_2.0.3       prabclus_2.3-2        evaluate_0.14
##  [73] purrr_0.3.3           tidyselect_0.2.5      GGally_1.4.0
##  [76] plyr_1.8.5            magrittr_1.5          R6_2.4.1
##  [79] generics_0.0.2        mitml_0.3-7           pillar_1.4.3
##  [82] haven_2.2.0           foreign_0.8-75        withr_2.1.2
##  [85] survival_3.1-8        abind_1.4-5           nnet_7.3-12
##  [88] tibble_2.1.3          crayon_1.3.4          car_3.0-6
##  [91] jomo_2.6-10           rmarkdown_2.1         grid_3.6.1
##  [94] readxl_1.3.1          forcats_0.4.0         vcd_1.4-5
##  [97] digest_0.6.25         diptest_0.75-7        stats4_3.6.1
## [100] munsell_0.5.0
```

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

```
## Machine:
```

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

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

```
## Num cores:
```

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

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

```
## Num threads:
```

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

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

```
## RAM:
```

```
## 34.4 GB
```

References

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.

Jansen, M. 1999. "Analysis of variance designs for model output." *Computer Physics Communications* 117 (1-2): 35–43. https://doi.org/10.1016/S0010-4655(98)00154-4.

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.

Saltelli, A., P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola. 2010. "Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index." *Computer Physics Communications* 181 (2). Elsevier B.V.: 259–70. https://doi.org/10.1016/j.cpc.2009.09.018.

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.