Appendices

Appendix A

Plots and Quantile Regression Using R and Stata Commands

Introduction

This appendix shows the basic commands to replicate the analysis, the tables and the plots proposed in the previous chapters using two statistical packages: R (R Core Team 2016). and Stata. R and Stata commands are shown in bold font while the comments using regular font and preceded by a "#" the hash character.

R Commands

Preliminary Basics

R is so popular for its vast array of packages available and a standard set of them is included. Packages are collections of R functions, data, and compiled codes in a well-defined format.

For the purpose of this book, we used a set of external packages.

The first step is to download and install a package and then, to use the package, invoke the *library* command to load it into the current session.

A.A. Romano et al., *Climate Finance as an Instrument to Promote the Green Growth in Developing Countries*, SpringerBriefs in Climate Studies,
DOI 10.1007/978-3-319-60711-5

```
#to install package (once time)
install.packages("package")
#to load library of package (in each session)
library(package)
```

Sometime, developers can distribute R packages that are developing on GitHub. Thus, the *devtools* package provides *install_github()* that enables installing packages from GitHub (a development platform where to host code, manage projects, and build software jointly others developers).

```
#to install package from GitHub, first we need to install and
load devtools package
install.packages("devtools")
library(devtools)
#then, to install package from GitHub
devtools::install_github("repository_names/package")
```

Loading Data

The first step, and sometime more difficult, is to prepare and load data for successive operations.

The two more common formats of files are tab delimited text (*txt*) and comma separated values (*csv*), but equally common for statistical analysis are the spreadsheet file (*xls, xlsx*) and Stata dataset (*dta*).

Generally, the format of dataset is a matrix with the cases in row and variables in columns.

We assume that the file of dataset is in the current working directory.

```
#to show the current working directory
getwd()
#to set a different working directory must shall indicate the
full path, for example
setwd ("C:/Users/.../Data")
```

To load data from three different sources we must shall download packages and load libraries.

```
#to import data from a comma separated values
install.packages("readr") #Wickham et al. (2016)
library(readr)

#to import data from a Stata dataset
install.packages("haven") #Wickham and Miller. (2016)
library(haven)

#to import data from a spreadsheet
install.packages("readxl ") #Wickham (2016)
library(readxl)
```

Now we can import data in R environment.

```
#import data from a semi-column separated values (csv)
flow_disb <-
  read_delim(
    "flow_disb_comp.csv",
    ";",
    escape_double = FALSE,
    na = "NA",
    trim_ws = TRUE
  )
#NOTE: the option "escape_double" is to file escape quotes by
doubling them; "na" indicates the character vector of strings
to use for missing values; "trim_ws" indicates if whitespace
be trimmed from each field.

#to import data from a Stata dataset
new <- read_dta("new.dta")

#to import data from a spreadsheet
indic <- read_excel("indic.xlsx",
                    sheet = "indic")
```

It is recommendable to verify the imported data with *head()* and *tail()* function that return the first or last parts, respectively, of imported data.

```
head(new)
tail(new)
```

Sometime we need subset and order the dataset.

```
new_cat1<-subset(new[order(-new$Tot_rec, new$CC),], categ==1)
#NOTE: We created a subset of dataset "new" where the variable categ
is equal to 1 and it is ordered by the variable Tot_rec and CC
```

Exploring Data

We report the codes used to reproduce the figures in the text. The figures was mainly created with "plotly" package (Sievert et al. 2016). Thus, we need install and load the package.

```
install.packages("plotly")
#to install the latest version by github
devtools::install_github("ropensci/plotly") library(plotly)
```

Fig. 3.1 Committed and disbursed funding in terms of financing type

The Fig. 3.1 shows a pie chart with the sum of commitment and disbursement for every type of flow. Before plot the graph, we need to create a suitable dataset. We need a dataset where rows are the flows type to plot and the columns are the sum of commitment and disbursement. To do this we use a function to merge rows grouping by a category (*aggregate*) and rename the columns created (*setNames*). Both functions are included in *stats* package.

```
sum_comm_disb <-
#to insert in variable the new dataset
  setNames(
    aggregate(
      cbind(
#to merge needed columns
        flow_disb$commitment_amount_usd_constant,
```

(continued)

```
        flow_disb$disbursement_amount_usd_constant
      ),
      by = list(crs_flow_name = flow_disb$flow_type),
#the clause condition to group the rows
      FUN = sum,
#the function to apply to grouped rows
      na.rm = TRUE,
      na.action = NULL
#to remove the missing value and do not apply operations to
them
    ),
    c('crs_flow_name', 'comm', 'disb')
#the names to assign the new columns
  )
```

Now, we can proceed to plot the graph. We invite the reader to refer to official documentation of every package.

The figure will show two adjacent pie charts (delimited in the space by *domain* option). The first pie chart shows the amount of commitment grouped by flow type, while the second (added to first with *add_pie* function) shows the grouped disbursed funding. Both graphs are drawn with a hole and they show inside every part of the pie the real value and the percentage on total. The y-x axis are not drawn, such as ticks labels.

The *annotation* function is used to create the caption for every pie chart. To export the graph we use the *export* function of *plotly* package.

```
#to create the variable containing the plot
p1 <-
  plot_ly(
    sum_comm_disb,
    labels = ~ crs_flow_name,
    values = ~ comm,
    type = 'pie',
    textposition = 'inside',
    textinfo = 'percent+value',
    insidetextfont = list(color = '#FFFFFF'),
    marker = list(colors = colors,
                  line = list(color = '#FFFFFF', width = 1)),
    name = "Commitment",
    domain = list(x = c(0, 0.5), y = c(0, 0.9)),
```

(continued)

```
    showlegend = T,
    hole = 0.3,
    sort = F
  ) %>%
  add_pie(
    sum_comm_disb,
    labels = ~ crs_flow_name,
    values = ~ disb,
    type = 'pie',
    textposition = 'inside',
    textinfo = 'percent+value',
    insidetextfont = list(color = '#FFFFFF'),
    marker = list(colors = colors,
                  line = list(color = '#FFFFFF', width = 1)),
    name = "Disbursement",
    domain = list(x = c(0.5, 1), y = c(0, 0.9)),
    showlegend = T,
    hole = 0.3,
    sort = F
  ) %>%
  layout(
    title = 'Flow type in 2010 (% of Commitment & Disburse-
ment)',
    font = list(family = 'sans serif', size = 14),
    legend = list(
      x = 0.5,
      y = -0.10,
      xanchor = "center",
      orientation = 'h',
      font = list(family = 'sans serif', size = 12)
    ),
    showlegend = T,
    xaxis = list(
      showgrid = FALSE,
      zeroline = FALSE,
      showticklabels = FALSE
    ),
yaxis = list(
      showgrid = FALSE,
      zeroline = FALSE,
      showticklabels = FALSE
```

(continued)

```
    ),
    annotations = list(
      list(
        x = 0.15 ,
        y = -0.1,
        text = "Committed Funds",
        showarrow = F,
        xref = 'paper',
        yref = 'paper'
      ),
      list(
        x = 0.85 ,
        y = -0.1,
        text = "Disbursed Funds",
        showarrow = F,
        xref = 'paper',
        yref = 'paper'
      )
    )
  )
#to display and check the output
p1

#to export the created pie chart in the working directory
export(p1, file = "flowtype.png")
```

Fig. 3.2 Committed and disbursed funding following the World Bank's Income classification of recipientcountries

The figure 3.2 shows a grouped bar chart to compare the amount of commitment and disbursement for every level of recipient's income. To have the total amount of both funding for every income's level we can create a suitable dataset as before done, or we can use an internal function of *plotly* package using another function belonging to *dplyr* package (Wickham and Francois 2016). Thus, we before install and load the package.

```
install.packages("dplyr ")
library(dplyr)
```

Now, we can proceed to plot the graph. The first argument is the dataset to use. The function *arrange*, belonging to *dplyr* package, allows to group the rows containing the individual disbursement and commitment (for each flow) by summing them for every income level and order them from high commitment and then

high disbursement. To obtain a grouped by chart we use the function *add_bars*. The order of labels is fixed by two options: *type* and *categoryorder* (in layout section). The former indicates that the labels are passed by category of *x* values and the second indicates that the order is the same of dataset (which we have ordered by *tot_comm* and *tot_disb*).

```
p2 <-
  plot_ly(
    arrange(flow_disb%>%
      group_by(rec_income)%>%
      summarize(
        tot_comm=sum(commitment_amount_usd_constant),
        tot_disb=sum(disbursement_amount_usd_constant)
        ), desc(tot_comm, tot_disb)),
    x = ~ rec_income,
    y = ~ tot_comm,
    type = 'bar',
    name = 'Commitment'
  ) %>%
  add_bars(y = ~ tot_disb,
           name = 'Disbursement') %>%
  layout(
    title = "Amount of Commitment and Disbursement for Income
level of recipient",
    font = list(family = 'sans serif', size = 12),
    xaxis = list(
      title = "Income level of Recipient",
      x = 0 ,
      y = -0.5,
      tickfont = list(family = 'sans serif', size = 10),
      xref = 'paper',
      yref = 'paper',
      type = 'category',
      categoryorder = 'trace'
    ),
    yaxis = list(title = "Amount usd constant"),
    legend = list(
      x = 1,
      y = -0.3,
      xanchor = "right",
      orientation = 'h',
      font = list(family = 'sans serif', size = 10),
```

(continued)

```
      traceorder = "normal"
    )
  )

p2

export(p2, file = "Comm-Disb_Income.png")
```

Fig. 3.3 Incidence of Disbursement on pledges

The figure 3.3 shows a radar chart (a.k.a. spider plot) to represent the incidence of pledges of donors.

To draw it we use the function *radarchart* belonging to *fmsb* package (Nakazawa 2015). The dataset to be use must include maximum values as row 1, minimum values as row 2 and the data to show as row 3 for each variables, and the number of columns (variables) must be more than 2.

Then, to export the image as *ps* file, we use the function *CairoPS* belonging to *Cairo* package (Urbanek and Horner 2015).

```
#for radarchart function
install.packages("fmsb")
library(fmsb)

#for export image
install.packages("Cairo")
library(Cairo)
```

```
CairoPS(
  file = "pledge",
  bg = "white",
  width = 5,
  pointsize = 12
)
#radar chart
radarchart(
  indic,
  maxmin = T,
  axistype = 1,
  #custom the grid
```

(continued)

```
  cglcol = "grey80",
  cglty = 1,
  axislabcol = "grey10",
  cglwd = 0.5,
  #custom labels
  vlcex = 0.8,
  title = "Incidence of Disbursement"
)
dev.off()
```

Fig. 3.4 Geographic distribution of commitment and disbursement

The figure shows a grouped bar chart to compare the amount of commitment and disbursement for every recipient's geographic region. The code are similar to the code used to draw the Fig. 3.2.

```
p4 <-
  plot_ly(
    arrange(flow_disb%>%
              group_by(region)%>%
              summarize(
                tot_comm=sum(commitment_amount_usd_constant),
               tot_disb=sum(disbursement_amount_usd_constant)
              ), desc(tot_comm, tot_disb)),
    x = ~ region,
    y = ~ tot_comm,
    type = 'bar',
    name = 'Commitment'
  ) %>%
  add_bars(y = ~ tot_disb,
           name = 'Disbursement') %>%
  layout(
    title = "Amount of Commitment and Disbursement for Region
of recipient",
    font = list(family = 'sans serif', size = 12),
    xaxis = list(
      title = "Region of Recipient",
      x = 0 ,
      y = -0.5,
      tickfont = list(family = 'sans serif', size = 10),
      xref = 'paper',
```

(continued)

```
      yref = 'paper',
      type = 'category',
      categoryorder = 'trace'
    ),
    yaxis = list(title = "Amount usd constant"),
    legend = list(
      x = 1,
      y = -0.3,
      xanchor = "right",
      orientation = 'h',
      font = list(family = 'sans serif', size = 10),
      traceorder = "normal"
    )
  )
p4

export(p4, file = "Comm-Disb_Region.png")
```

Fig. 3.5 Flow of climate finance for the year 2010

The figure 3.5 shows the map of flow of funding, disbursed in 2010, in the World and in Europe.

First, we need to import data.

```
#Import Data
#dati_na contains the data related to EPI indicator (CC var-
iable) and category of country (cod indicates if it is a
donor, recipient or untreated country)
dati_na <-
  read.csv(
    "dataset_bioen.csv",
    header = T,
    sep = ";",
    dec = ".",
    na.strings = "NULL"
  )

#dati_flow contains the data related to each flow of funding
and includes the ratios of donors (and recipients) on total
amount of disbursed funding, the amount disbursed by each
donor and the latitude and longitude of donors and recipi-
ents.
```

(continued)

```
dati_flow <-
  read.csv(
    "flow_bio_en.csv",
    header = T,
    sep = ";",
    dec = ".",
    na.strings = "NULL"
  )
```

To draw the two map, colored on basis of the level of emissions (summarize by EPI) of each country we use the *rworldmap* package (South 2011), while to represent the flows of funding we use the function *gcIntermediate* belonging to *geosphere* package (Hijmans 2016), used to build another function *clean.Inter* (reported following). The colors of quantiles' distribution of EPI and category of countries (*cod*) come from the *classInt* package (Bivand 2015) (to obtain the interval of values within any quantile) and *RColorBrewer* package (Neuwirth 2014) to get the color palettes.

Then, to export the image as *PDF* file, we use the function *mapDevice* belonging to *rworldmap* package that creates a plot device suited for *rworldmap* plotting functions.

Thus, we start installing and loading the needed packages.

```
#to draw maps
install.packages("rworldmap")
library(rworldmap)

#to draw lines of flows
install.packages("geosphere")
library(geosphere)

#to get classes of quantiles
install.packages("classInt")
library(classInt)

#to get the color palettes
install.packages("RColorBrewer")
library(RColorBrewer)
```

Now, we can create the basis for draw the maps, starting with the building of needed functions.

```
#Function to draw lines of the flows
checkDateLine <- function(l) {
  n <- 0
  k <- length(l)
  k <- k - 1
  for (j in 1:k) {
    n[j] <- l[j + 1] - l[j]
  }
  n <- abs(n)
  m <- max(n, rm.na = TRUE)
  ifelse(m > 30, TRUE, FALSE)
}

clean.Inter <- function(p1, p2, n, addStartEnd) {
   inter <- gcIntermediate(p1, p2, n = n, addStartEnd =
addStartEnd)
    if (checkDateLine(inter[, 1])) {
    m1 <- midPoint(p1, p2)
    m1[, 1] <- (m1[, 1] + 180) %% 360 - 180
    a1 <- antipode(m1)
    l1 <- gcIntermediate(p1, a1, n = n, addStartEnd =
addStartEnd)
    l2 <- gcIntermediate(a1, p2, n = n, addStartEnd =
addStartEnd)
    l3 <- rbind(l1, l2)
    l3
  }
  else{
    inter
  }
}

# functions to add transparence to colors
addalpha <- function(colors, alpha = 1.0) {
  r <- col2rgb(colors, alpha = T)
  # Apply alpha
  r[4,] <- alpha * 255
  r <- r / 255.0
  return(rgb(r[1,], r[2,], r[3,], r[4,]))
}
```

The following text box include the code to join the imported data referenced by country codes (*iso_3* variable) to the internal map of *rworldmap* package.

```
#Create basis for map

set.seed(123)

sPDF <-
  joinCountryData2Map(
    dati_na ,
    joinCode = "ISO3" ,
    nameJoinColumn = "iso_3",
    nameCountryColumn = "country",
    suggestForFailedCodes = FALSE,
    mapResolution = "coarse",
    projection = NA,
    verbose = T
  )
#NOTE: the verbose option enables to check the successful
joins

#to remove the Antarctica
sPDF <- sPDF[-which(row.names(sPDF) == 'Antarctica'),]

#getting class intervals using a 'jenks' classification in
classInt package
classInt <-
  classIntervals(
    sPDF$CC,
    n = 4,
    style = "quantile",
    unique = T,
    dataPrecision = T,
    intervalClosure = c("left", "right")
  )
catMethod = classInt$brks

#getting a color scheme
#for quantiles
colourPalette <- brewer.pal(4, 'YlGn')
colourPalette <- addalpha(colourPalette, 0.5)
#for cod
```

(continued)

```
colourPalette1 <- colorRampPalette(c("grey60", "grey30"), space
= "Lab", bias = 3)(3)
colourPalette1 <- addalpha(colourPalette1, 0.3)
```

The following code shows the procedure to recreate the maps. The map are an overlapping of the two maps. In the former, the countries are hatched using *cod* variable. The second map is overlaid on the first and the countries are colored using the *CC* (EPI) variable.

```
#to open the graphic device
mapDevice(
  device = "pdf",
  file = "flowWorld.pdf",
  useDingbats = T,
  pagecentre = T,
  mai = c(0, 0, 0.2, 0),
  xaxs = "i",
  yaxs = "i"
)

#to set the parameters of first map
mapParams1 <-
  mapCountryData(
    sPDF,
    mapRegion = "world",
    nameColumnToPlot = 'cod',
    catMethod = "categorical",
    colourPalette = colourPalette1,
    addLegend = FALSE,
    borderCol = "black",
    mapTitle = "",
    aspect = 1,
    missingCountryCol = "grey90",
    lwd = 0.5,
    oceanCol = "aliceblue",
    nameColumnToHatch = "cod"
  )
#to set the parameters of second map
mapParams <-
  mapCountryData(
```

(continued)

```
    sPDF,
    mapRegion = "world",
    nameColumnToPlot = 'CC',
    catMethod = catMethod,
    colourPalette = colourPalette,
    addLegend = FALSE,
    borderCol = "black",
    mapTitle = "Flow of Funds in the World",
    aspect = 1,
    missingCountryCol = "grey90",
    add = T,
    lwd = 0.5,
    oceanCol = "aliceblue"
  )

#to add legend on map
do.call(
  addMapLegend
  ,
  c(
    mapParams,
    legendLabels = "all",
    legendWidth = 0.5,
    legendIntervals = "data" ,
    legendMar = 2
  )
)

#to draw lines of flows
#the width of lines are proportional to ratio of amount dis-
bursed by donor to a specific recipient country on total
amount disbursed by same donor to all recipients.

pal <- colorRampPalette(c("#ccffcc", "#006600"))
colors <- pal(100)

for (i in 1:length(dati_flow[, 1]))
{
  gC <-
    clean.Inter(dati_flow[i, c(4, 3)],
                dati_flow[i, c(7, 6)],
```

(continued)

```
                n = 100,
                addStartEnd = TRUE)

  colindex <-
    round((dati_flow[i,]$ratio_t_don) / 100 * length(colors))
  lines(gC, col = colors[colindex], lwd = (colindex / 50))

}

#to draw marker points of countries: red circle for donors
and green triangle point down for recipients
#the expansion of symbols are proportional to ratio of amount
disbursed by donor on total disbursed to all recipient coun-
tries for donor symbol and to ratio of amount received by
recipient on total funding for recipient symbol.

radius_don <- sqrt(dati_flow$ratio_gt_don / pi)
radius_rec <- sqrt(dati_flow$ratio_gt_rec / pi)

points(
  dati_flow$lon_don,
  dati_flow$lat_don,
  pch = 1,
  cex = radius_don ,
  col = "red"
)
points(
  dati_flow$lon_rec,
  dati_flow$lat_rec,
  pch = 6,
  cex = radius_rec ,
  col = "green"
)

legend(
  "bottomleft",
  box.col = "grey85",
  c("D", "T"),
  pch = c(1, 6),
  col = c("red", "green"),
  inset = c(.05, 0.2),
```

(continued)

```
  xjust = 0,
  cex = 0.9
)

dev.off()
```

Fig. 3.6 Flow of climate finance for the year 2010 with focus on European countries

The code included in the following box are similar to the code used to draw the previous map, but focus on Europe region.

```
mapDevice(
  device = "pdf",
  file = "flowEurope.pdf",
  useDingbats = T,
  pagecentre = T,
  mai = c(0, 0, 0.2, 0),
  xaxs = "i",
  yaxs = "i"
)

mapParams1 <-
  mapCountryData(
    sPDF,
    mapRegion = "eurasia",
    nameColumnToPlot = 'cod',
    catMethod = "categorical",
    colourPalette = colourPalette1,
    addLegend = FALSE,
    borderCol = "black",
    mapTitle = "",
    aspect = 1,
    missingCountryCol = "grey90",
    lwd = 0.5,
    oceanCol = "aliceblue",
    nameColumnToHatch = "cod"
  )

mapParams <-
  mapCountryData(
```

(continued)

```
    sPDF,
    mapRegion = "eurasia",
    nameColumnToPlot = 'CC',
    catMethod = catMethod,
    colourPalette = colourPalette,
    addLegend = FALSE,
    borderCol = "black",
    mapTitle = "Flow of Funds in Europe",
    aspect = 1,
    missingCountryCol = "grey90",
    add = T,
    lwd = 0.5,
    oceanCol = "aliceblue"
  )

do.call(
  addMapLegend,
  c(
    mapParams,
    legendLabels = "all",
    legendWidth = 0.5,
    legendIntervals = "data",
    legendMar = 2
  )
)

# flow lines #

pal <- colorRampPalette(c("#ccffcc", "#006600"))
colors <- pal(100)

for (i in 1:length(dati_flow[, 1]))
{
  gC <-
    clean.Inter(dati_flow[i, c(4, 3)],
                dati_flow[i, c(7, 6)],
                n = 100,
                addStartEnd = TRUE)
```

(continued)

```
  colindex <-
    round((dati_flow[i,]$ratio_t_don) / 100 * length(colors))
  lines(gC, col = colors[colindex], lwd = (colindex / 50))

}

radius_don <- sqrt(dati_flow$ratio_gt_don / pi)
radius_rec <- sqrt(dati_flow$ratio_gt_rec / pi)

# marker points #

points(
  dati_flow$lon_don,
  dati_flow$lat_don,
  pch = 1,
  cex = radius_don ,
  col = "red"
)
points(
  dati_flow$lon_rec,
  dati_flow$lat_rec,
  pch = 6,
  cex = radius_rec ,
  col = "green"
)

legend(
  "bottomleft",
  box.col = "grey85",
  c("D", "T"),
  pch = c(1, 6),
  col = c("red", "green"),
  inset = c(.05, 0.2),
  xjust = 0,
  cex = 0.9
)

dev.off()
```

Fig. 3.7 Number of Treated and Untreated Countries on the basis of quartiles of GHG emission distribution

The figure 3.7 shows a stacked horizontal bar chart to compare the number of treated and untreated countries for every quantile of EPI's distribution using the function *count*. The code are similar to the code used to draw the Figs. 3.2, 3.4.

```
p7 <- new %>% count(categ, d_t_u) %>%
  plot_ly(
    x = ~ n,
    y = ~ categ,
    type = "bar",
    orientation = 'h',
    textposition = 'inside',
    text = ~ n ,
    insidetextfont = list(color = '#FFFFFF'),
    color = ~ d_t_u
  ) %>%
  layout(
    barmode = 'stack',
    title = "Total of Treated and Untreated countries",
    font = list(family = 'sans serif', size = 12),
    yaxis = list(
      title = "Quantiles of GHG distribution",
      tickmode = "array",
      tickvals = 1:5,
      ticktext = c("25th", "50th", "75th", "90th", "Over 90th")
  ),
  xaxis = list(title = "Number of Countries"),
  legend = list(
   x = 1,
   y = -0.10,
   xanchor = "right",
   orientation = 'h',
   font = list(family = 'sans serif', size = 10),
   traceorder = "normal"
  )
 )
p7
export(p7, file = "treatUnt.png")
```

Fig. 3.8 Climate funds received by developing countries

The figure 3.8 shows a horizontal bar chart to compare the amount of disbursed funding for every quantile of EPI's distribution. The code are similar to the code used to draw the Fig. 3.7.

```
p8 <- plot_ly(
  new,
  x = ~ Tot_rec,
  y = ~ categ,
  type = "bar",
  orientation = 'h',
  textposition = 'inside',
  insidetextfont = list(color = '#FFFFFF')
) %>%
  layout(
    barmode = 'stack',
    title = "Total of Recipients' received funding",
    font = list(family = 'sans serif', size = 12),
    yaxis = list(
      title = "Quantiles of GHG distribution",
      tickmode = "array",
      tickvals = 1:5,
      ticktext = c("25th", "50th", "75th", "90th", "Over 90th")
  ),
  xaxis = list(title = "$ Disbursed"),
  legend = list(
   x = 1,
   y = -0.10,
   xanchor = "right",
   orientation = 'h',
   font = list(family = 'sans serif', size = 10),
   traceorder = "normal"
  )
 )
p8
export(p81, file = "treat_disb.png")
```

Fig. 3.9 Disbursement and Environmental Performance of recipients

The figure 3.9 shows two-stacked horizontal bar chart with a line plot to compare the amount of funding with level of GHG emissions (EPI) and with share of renewable/fossil source, for the countries included in 25th quantile of EPI's distribution. To build the figure we use the function *subplot* belonging to *plotly* package

whit the option "*shareY*" to align every y-axis and option "*legendgroup*" to group the entry legend for every plot. For the code of bar chart, see Fig. 3.7.

The order of labels on y-axis is indicated by the option "*categoryorder = "array"*" where array include the names of countries ordered by amount of funding, first, and level of GHG emission, after.

```
p9.1 <-
  plot_ly(
    data = new_cat1,
    x = ~ tot_rec_en,
    y = ~ country,
    type = 'bar',
    orientation = 'h',
    name = 'Energy',
    marker = list(color = 'green', line = list(color = 'green',
width = 1)),
  legendgroup = 'group1'
 ) %>%
 add_trace(
  x = ~ tot_rec_bio,
  name = 'Biosphere',
  marker = list(
   color = 'deepskyblue',
   line = list(color = 'deepskyblue', width = 1)
  ),
  legendgroup = 'group1'
 ) %>%
 layout(
  barmode = 'stack',
  yaxis = list(
   type = "category",
   categoryorder = "array",
   categoryarray = new_cat1$country[with(new_cat1, order
(Tot_rec,-CC))],
   showgrid = TRUE,
   showline = FALSE,
   showticklabels = TRUE,
   domain = c(0, 0.9)
  ),
  xaxis = list(
   zeroline = FALSE,
   showline = FALSE,
```

(continued)

```
    showticklabels = TRUE,
    showgrid = TRUE
   )

   )
p9.2 <-
  plot_ly(
    data = new_cat1,
    x =  ~ CC,
    y =  ~ country,
    type = 'scatter',
    name = 'Ghg emissions',
    mode = 'lines+markers',
    line = list(color = 'rgb(128, 0, 128)'),
    legendgroup = 'group2'
  ) %>%
  layout(
    yaxis = list(
      type = "category",
      categoryorder = "array",
      categoryarray = new_cat1$country[with(new_cat1, order
(Tot_rec,-CC))],
      showgrid = TRUE,
      showline = TRUE,
      showticklabels = FALSE,
      linecolor = 'rgba(102, 102, 102, 0.8)',
      linewidth = 2,
      domain = c(0, 0.9)
    ),
    xaxis = list(
      zeroline = FALSE,
      showline = FALSE,
      showticklabels = TRUE,
      showgrid = TRUE
    )
  )
p9.3 <-
  plot_ly(
    data = new_cat1,
    x = ~ sh_foss,
    y = ~ country,
```

(continued)

```
    type = 'bar',
    orientation = 'h',
    name = 'Share of Fossil',
    marker = list(color = 'rgba(204,204,204,1)',  line = list
(color = 'grey', width = 0.5)),
    legendgroup = 'group3'
  ) %>%
  add_trace(
    x = ~ sh_ren,
    type = 'bar',
    orientation = 'h',
    name = 'Share of Renewable',
    marker = list(color = 'rgba(50, 171, 96, 0.7)', line = list
(color = 'green', width = 0.5)),
  legendgroup = 'group3'
 ) %>%
 layout(
  barmode = 'stack',
  yaxis = list(
   type = "category",
   categoryorder = "array",
   categoryarray   =   new_cat1$country[with(new_cat1,   order
(Tot_rec,-CC))],
   showgrid = TRUE,
   showline = FALSE,
   showticklabels = TRUE,
   domain = c(0, 0.9)
  ),
  xaxis = list(
   zeroline = FALSE,
   showline = FALSE,
   showticklabels = TRUE,
   showgrid = TRUE
  )
 )

p9.1_2_3 <- subplot(p9.1, p9.2, p9.3, shareY = TRUE, margin =
0) %>%
  layout(
    autosize = T,
    margin(t = 0),
```

(continued)

```
    title = "Total funds and Environmental Performance",
    font = list(family = 'sans serif', size = 14),
    legend = list(
      xref = 'paper',
      yref = 'paper',
      x = 1,
      y = -0.1,
      xanchor = "right",
      orientation = 'h',
      font = list(family = 'sans serif', size = 10)
    ),
    yaxis = list(
      title = '',
      tickangle = 45,
      tickfont = list(family = 'sans serif', size = 8)
    ),
    xaxis = list(
      title = '',
      tickangle = 45,
      tickfont = list(family = 'sans serif', size = 8)
    ),
    xaxis2 = list(
      title = '',
      tickangle = 45,
      tickfont = list(family = 'sans serif', size = 8)
    ),
    xaxis3 = list(
      title = '',
      tickangle = 45,
      tickfont = list(family = 'sans serif', size = 8)
    )
  ) %>%
  add_annotations(
    xref = 'paper',
    yref = 'paper',
    x = 0,
    y = -0.15,
    text = paste('25% quantile of GHG distribution'),
    font = list(
      family = 'sans serif',
      size = 10,
```

(continued)

```
      color = 'rgb(150,150,150)'
    ),
    showarrow = FALSE
  )

p9.1_2_3

export(p9.1_2_3, file = "tot_funds_and_envir_perfor.png")
```

Fig. 3.10 Disbursement and Development of recipients

The figure 3.10 shows two-stacked horizontal bar chart to compare the amount of funding with level of GDP and the Electricity consumption, for the countries included in 25th quantile of EPI's distribution. The code is similar to the code of Fig. 3.9.

```
p10 <-
  plot_ly(
    data = new_cat1,
    y = ~ country,
    x = ~ lnelcons,
    type = 'bar',
    orientation = 'h',
    name = 'Electricity Consumption (log scale)',
    marker = list(color = 'rgba(222, 237, 255, 0.7)', line =
list(color = 'green', width = 0.5)),
    legendgroup = 'group3'
  ) %>%
  add_trace(
    x = ~ lgdp,
    type = 'bar',
    orientation = 'h',
    name = 'GDP (log scale)',
    marker = list(color = 'rgba(150,150,150,1)', line = list
(color = 'grey', width = 0.5)),
    legendgroup = 'group3'
  ) %>%
  layout(
    barmode = 'group',
    yaxis = list(
      type = "category",
```

(continued)

```
      categoryorder = "array",
      categoryarray = new_cat1$country[with(new_cat1, order
(Tot_rec,-CC))],
      showgrid = TRUE,
      showline = FALSE,
      showticklabels = TRUE,
      domain = c(0, 0.9)
    ),
    xaxis = list(
      zeroline = FALSE,
      showline = FALSE,
      showticklabels = TRUE,
      showgrid = TRUE
    )
  )
p10_9.1_4 <- subplot(p9.1, p10, shareY = TRUE) %>%
  layout(
    title = "Total funds and Development",
    font = list(family = 'sans serif', size = 12),
    legend = list(
      x = 1,
      y = -0.10,
      xanchor = "right",
      orientation = 'h',
      font = list(family = 'sans serif', size = 10),
      traceorder = "grouped+reversed"
    ),
    yaxis = list(
      title = '',
      tickangle = 45,
      tickfont = list(family = 'sans serif', size = 8)
    ),
    xaxis = list(
      title = '',
      tickangle = 45,
      tickfont = list(family = 'sans serif', size = 8)
    ),
    xaxis2 = list(
      title = '',
      tickangle = 45,
      tickfont = list(family = 'sans serif', size = 8)
```

(continued)

```
    )
  ) %>%
  add_annotations(
    xref = 'paper',
    yref = 'paper',
    x = 0,
    y = -0.3,
    text = paste('25% quantile of GHG distribution'),
    font = list(
      family = 'sans serif',
      size = 8,
      color = 'rgb(150,150,150)'
    ),
    showarrow = FALSE
  )

p10_9.1_4

export( p10_9.1_4, file = "tot_funds_and_dev.png")
```

Building the Indicator

To obtain the Composite Indicators of GHG emission with the weighting method based on factor analysis, we used the function *ci_factor* included in *Compind* package (Vidoli et al. 2015). The raw data of GHG are in the dataset *dataCC*.

```
CI<-ci_factor(dataCC,method="CH", dim=2)
# method = "CH" can be choose the number of the component to
take into account while dim=2 indicates the number of chosen
component
```

Now, we can import the Composite indicator estimated values in dataset.

```
data$CI<-CI$ci_factor_est
```

To scale values between [0; 1] corresponding to [min; max], it's straight-forward to create a small function to do this using basic arithmetic:

```
range01 <- function(x){(ma(x)-x)/(max(x)-min(x))}
```

and then we can import in dataset the scaled values to compare the results.

```
dataCC$CI_norm<-range01(CI$ci_factor_est)
```

Lastly, we can export the new dataset in working directory using the function *write.xlsx* included in *xlsx* package (Dragulescu 2014).

```
install.packages("xlsx")
library(xlsx)

#to export the dataset
write.xlsx(dataCC, "dataCC_new.xlsx")
```

STATA Commands

In this book, we estimated a quantile regression using STATA software, because it wasn't implemented in R software a package with the Parente-Santo Silva procedure. Thus, we estimated the parameters of quantile regression using the *qreg2* module (Parente and Santos Silva 2016) for every decile.

```
//to install the qreg2 module
ssc install qreg2
```

```
# Execute quantile regression for every quantile and store
results
# With the option cluster(GroupI) we specify that the stan-
dard errors are computed allowing for intra-cluster correla-
tion as in Parente and Santos Silva (2016) to account the
social-economic structure of countries (GroupI indicates the
income group of WorldBank).
```

(continued)

```
qreg2 CC Tot_rec pop_fem ei oil_sup sh_foss sh_n lgdp
lnelcons acc_el , quantile(.10) c(GroupI)
est store CC_10th_tot
qreg2 CC tot_rec_en pop_fem ei oil_sup sh_foss sh_n lgdp
lnelcons acc_el , quantile(.10) c(GroupI)
est store CC_10th_en
qreg2 CC tot_rec_bio pop_fem ei oil_sup sh_foss sh_n lgdp
lnelcons acc_el , quantile(.10) c(GroupI)
est store CC_10th_bio

// Execute linear regression specifying that the standard
error reported allow for intragroup correlation and then
store results
regress CC Tot_rec pop_fem ei oil_sup sh_foss sh_n lgdp
lnelcons acc_el, vce(cluster GroupInc)
est store CC_OLS_tot
```

We can export the stored results in a rtf file, to manage it and import them in R software.

```
// export results (this sis only for the total disbursed)
esttab CC_OLS_tot CC_10th_tot CC_20th_tot CC_30th_tot
CC_40th_tot CC_50th_tot CC_60th_tot CC_70th_tot CC_80th_tot
CC_90th_tot CC_95th_tot using C:\...\results.rtf, append
title(This is a quantile regression table Total) nonumbers
mtitles("OLS" "10th" "20th" "30th" "40th" "50th" "60th"
"70th" "80th" "90th" "95th")cells(b(star label(Coef.) fmt
(a3)) se(par fmt(a3))) label varlabels(_cons Constant) legend
stats(r2 N pss_p, labels(R-squared "N. of cases" "Parente-
Santos Silva test p-value")) starlevels(* 0.1 ** 0.05 ***
0.01)

// export coefficients and confidence intervals to be used in R
software
estout CC_OLS_tot CC_10th_tot CC_20th_tot CC_30th_tot
CC_40th_tot CC_50th_tot CC_60th_tot CC_70th_tot CC_80th_tot
CC_90th_tot CC_95th_tot using C:\...\results_b.xls, cells(b)
estout CC_OLS_tot CC_10th_tot CC_20th_tot CC_30th_tot
CC_40th_tot CC_50th_tot CC_60th_tot CC_70th_tot CC_80th_tot
CC_90th_tot CC_95th_tot using C:\...\results_ci.xls, cells
(ci_l ci_u)
```

R: Create graph of Quantiles Plot (Fig. 5.3 (a) – (j) Quantile coefficient plots)

To create quantile coefficients plot we can use the STATA command, but it is not implemented to obtain confidence interval for the Parente-Silva procedure. To remedy the lack of a specific procedure to obtain the confidence interval with the standard errors proposed by Parente-Silva, we can import the values of coefficients and the related lower and upper confidence intervals in R, after wwe exported them by STATA and create the plots using a simple loop in R.

```
# Import saved coefficients from Stata
data_coef <-
  data.frame(
    en_Tot_rec = c(..., ..., ... ),
    en_min_Tot_rec = c(..., ..., ...),
    en_max_Tot_rec = c(..., ..., ...),
    en_pop_fem = c( ... ) ...
           )
```

To create the loop we need to create a list of variables' names to pass in loop.

```
# create a list of variables' names
var_q <-
  c(
    "Tot_rec",
    "pop_fem",
    "ei",
    "oil_sup",
    "sh_foss",
    "sh_nonhydro",
    "lgdp",
    "lnelcons",
    "acc_el",
    "Constant"
  )
```

Now, we can construct the loop.

```
# loop for create and save plots

for (i in seq_along(var_q)) {
  CairoPS(
    file = var_q[i],
    bg = "white",
    width = 5,
    pointsize = 12
  )

  plot(
    q,
    data_coef[, paste("en_", var_q[i], sep = "")],
    type = 'n',
    ylim = c(min(data_coef[, paste("en_min_", var_q[i], sep =
"")], data_coef[, paste("bio_min_", var_q[i], sep =
                                    "")]),
        max(data_coef[, paste("en_max_", var_q[i], sep =
                    "")], data_coef[, paste("bio_max_", var_q
[i], sep = "")])),
    xaxt = 'n',
    xlab = "quantile",
    ylab = var_q[i]
  )

  polygon(
    c(q, rev(q)),
    c(data_coef[, paste("en_max_", var_q[i], sep = "")], rev
(data_coef[, paste("en_min_", var_q[i], sep ="")])),
    col = SetAlpha("green4", .5),
    border = NA,
    density = 25,
    angle = 45,
    lty = 2
  )
  polygon(
    c(q, rev(q)),
    c(data_coef[, paste("bio_max_", var_q[i], sep = "")], rev
(data_coef[, paste("bio_min_", var_q[i], sep =
                                          "")])),
    col = SetAlpha("blue", .5),
    border = NA,
    density = 25,
```

(continued)

```
    angle = -45,
    lty = 3
  )

  points(
    q,
    data_coef[, paste("en_", var_q[i], sep = "")],
    type = 'l',
    col = 'green4',
    lwd = 2,
   ylim = c(min(data_coef[, paste("en_min_", var_q[i], sep =
"")]), max(data_coef[, paste("en_max_", var_q[i], sep =
                              "")]))
 )
 points(
  q,
  data_coef[, paste("bio_", var_q[i], sep = "")],
  type = 'l',
  col = 'blue',
  lwd = 2,
   ylim = c(min(data_coef[, paste("bio_min_", var_q[i], sep =
"")]), max(data_coef[, paste("bio_max_", var_q[i], sep =
                             "")]))
 )

  abline(
    h = data_coef[, paste("en_ols_", var_q[i], sep = "")],
    v = 0,
    col = 'green2',
    lty = 3,
    lwd = 1,
     ylim = c(min(data_coef[, paste("en_ols_min_", var_q[i],
sep = "")]), max(data_coef[, paste("en_ols_max_", var_q[i],
sep = "")]))
  )

  abline(
    h = data_coef[, paste("bio_ols_", var_q[i], sep = "")],
    v = 0,
    col = 'deepskyblue',
    lty = 3,
    lwd = 1,
```

(continued)

```
    ylim = c(min(data_coef[, paste("bio_ols_min_", var_q[i],
sep =
                 "")]), max(data_coef[, paste("bio_ols_max_",
var_q[i], sep = "")]))
  )

  abline(v = 0,
         h = 0,
         col = "grey30",
         lty = 6)

  axis(1, at = q)
  legend(
    "bottomright",
    c("Energy", "Bio", "OLS_en", "OLS_bio"),
    col = c("green4", "blue", "green2", "deepskyblue"),
    pch = 20
  )
  dev.off()
}
```

References

Bivand R (2015) classInt: choose univariate class intervals. R package version 0.1–23

Dragulescu AA (2014) xlsx: read, write, format excel 2007 and excel 97/2000/XP/2003 files. R package version 0.5.7

Hijmans RJ (2016) geosphere: spherical trigonometry. R package version 1.5–5

Nakazawa M (2015) fmsb: functions for medical statistics book with some demographic data. R package version 0.5.2

Neuwirth E (2014) RColorBrewer: colorBrewer palettes. R package version 1.1–2

Parente PM, Santos Silva J (2016) Quantile regression with clustered data. J Econ Methods 5 (1):1–15

R Core Team (2016) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna

Sievert C, Parmer C, Hocking T, Chamberlain S, Ram K., Corvellec M. Despouy P (2016) plotly: create interactive web graphics via 'plotly.js'

South A (2011) rworldmap: a new R package for mapping global data. The R J 3/1: 35–43

Urbanek S, Horner J (2015) Cairo: r graphics device using cairo graphics library for creating high-quality bitmap (PNG, JPEG, TIFF), vector (PDF, SVG, PostScript) and display (X11 and Win32) output. R package version 1.5–9

Vidoli F, Fusco E, Mazziotta C (2015) Non-compensability in composite indicators: a robust directional frontier method. Soc Indic Res 122(3):635–652
Wickham H (2016) readxl: Read excel files. R package version 0.1.1
Wickham H, Francois R (2016) dplyr: a grammar of data manipulation. R package version 0.5.0
Wickham H, Miller E (2016) haven: import and export 'SPSS', 'Stata' and 'SAS' files. R package version 1.0.0
Wickham H, Hester J, Francois R (2016) readr: read Tabular Data. R package version 1.0.0

Appendix B

List of Countries included in our study. For each countries is indicated the Iso3 code, its position on Climate Finance (if Donor, Treated or Untreated) and the Environmental Pollution Index (EPI).

Country	Iso 3	Donor/Treated/Untreated	EPI
Afghanistan	AFG	Treated	0.99254
Albania	ALB	Untreated	0.998725
Algeria	DZA	Treated	0.975688
Angola	AGO	Treated	0.962244
Antigua and Barbuda	ATG	Untreated	0.999436
Argentina	ARG	Treated	0.943719
Armenia	ARM	Treated	0.99853
Australia	AUS	Donor	0.889048
Austria	AUT	Donor	0.992426
Azerbaijan	AZE	Treated	0.987959
Bahamas, The	BHS	Untreated	0.999757
Bahrain	BHR	Untreated	0.997795
Bangladesh	BGD	Treated	0.966298
Barbados	BRB	Untreated	0.999367
Belarus	BLR	Treated	0.987373
Belgium	BEL	Donor	0.989001
Belize	BLZ	Untreated	0.997443
Benin	BEN	Treated	0.997723
Bhutan	BTN	Untreated	0.999816
Bolivia	BOL	Treated	0.989949
Bosnia and Herzegovina	BIH	Treated	0.997229
Botswana	BWA	Treated	0.99674
Brazil	BRA	Treated	0.784867
Brunei	BRN	Untreated	0.996453
Bulgaria	BGR	Untreated	0.993349
Burkina Faso	BFA	Untreated	0.993025
Burundi	BDI	Untreated	0.998966
Cambodia	KHM	Treated	0.993006

(continued)

Country	Iso 3	Donor/Treated/Untreated	EPI
Cameroon	CMR	Treated	0.967426
Canada	CAN	Donor	0.914518
Cape Verde	CPV	Treated	0.999902
Central African Republic	CAF	Untreated	0.981787
Chad	TCD	Untreated	0.990779
Chile	CHL	Treated	0.989776
China	CHN	Treated	0
Colombia	COL	Treated	0.970694
Comoros	COM	Untreated	0.999924
Costa Rica	CRI	Treated	0.997526
Cote d'Ivoire	CIV	Untreated	0.99284
Croatia	HRV	Untreated	0.996672
Cuba	CUB	Treated	0.992686
Cyprus	CYP	Untreated	0.999407
Czech Republic	CZE	Donor	0.990146
Denmark	DNK	Donor	0.993328
Djibouti	DJI	Untreated	0.999675
Dominica	DMA	Untreated	0.999965
Dominican Republic	DOM	Treated	0.995425
Ecuador	ECU	Treated	0.992697
Egypt	EGY	Treated	0.969811
El Salvador	SLV	Treated	0.99757
Equatorial Guinea	GNQ	Untreated	0.995627
Eritrea	ERI	Treated	0.997923
Estonia	EST	Untreated	0.998036
Ethiopia	ETH	Treated	0.961461
Fiji	FJI	Untreated	0.999565
Finland	FIN	Donor	0.992837
France	FRA	Donor	0.939076
Gabon	GAB	Untreated	0.999379
Gambia, The	GMB	Untreated	0.998068
Georgia	GEO	Untreated	0.99703
Germany	DEU	Donor	0.92129
Ghana	GHA	Treated	0.994691
Greece	GRC	Donor	0.991673
Grenada	GRD	Untreated	0.999523
Guatemala	GTM	Treated	0.994989
Guinea	GIN	Untreated	0.995367
Guinea-Bissau	GNB	Untreated	0.999355
Guyana	GUY	Untreated	0.999269
Haiti	HTI	Treated	0.997913
Honduras	HND	Treated	0.995995
Hungary	HUN	Untreated	0.991939

(continued)

Country	Iso 3	Donor/Treated/Untreated	EPI
Iceland	ISL	Donor	0.999555
India	IND	Treated	0.560014
Indonesia	IDN	Treated	0.882877
Iran	IRN	Treated	0.943113
Iraq	IRQ	Untreated	0.962057
Ireland	IRL	Donor	0.990769
Israel	ISR	Untreated	0.982969
Italy	ITA	Donor	0.953287
Jamaica	JAM	Untreated	0.998735
Japan	JPN	Donor	0.871739
Jordan	JOR	Treated	0.996996
Kazakhstan	KAZ	Treated	0.975621
Kenya	KEN	Treated	0.982847
Kiribati	KIR	Untreated	0.999993
Korea, Dem. Rep. (North)	PRK	Untreated	0.988797
Korea, Rep. (South)	KOR	Donor	0.929427
Kuwait	KWT	Untreated	0.963969
Kyrgyzstan	KGZ	Treated	0.997889
Laos	LAO	Treated	0.997015
Latvia	LVA	Untreated	0.997346
Lebanon	LBN	Treated	0.997793
Lesotho	LSO	Treated	0.99938
Liberia	LBR	Untreated	0.999569
Libya	LBY	Untreated	0.975507
Lithuania	LTU	Untreated	0.996133
Luxembourg	LUX	Donor	0.999293
Macedonia, FYR	MKD	Untreated	0.997388
Madagascar	MDG	Treated	0.991808
Malawi	MWI	Treated	0.997034
Malaysia	MYS	Treated	0.964568
Maldives	MDV	Treated	0.999943
Mali	MLI	Treated	0.990379
Malta	MLT	Untreated	0.999491
Mauritania	MRT	Untreated	0.997187
Mauritius	MUS	Untreated	0.999213
Mexico	MEX	Treated	0.881368
Moldova	MDA	Untreated	0.998261
Mongolia	MNG	Treated	0.992912
Montenegro	MNE	Treated	0.99908
Morocco	MAR	Treated	0.991569
Mozambique	MOZ	Treated	0.991746
Namibia	NAM	Treated	0.995872
Nepal	NPL	Treated	0.990549

(continued)

Country	Iso 3	Donor/Treated/Untreated	EPI
Netherlands	NLD	Donor	0.982099
New Zealand	NZL	Donor	0.983135
Nicaragua	NIC	Treated	0.996674
Niger	NER	Treated	0.991823
Nigeria	NGA	Treated	0.93274
Norway	NOR	Donor	0.994249
Oman	OMN	Untreated	0.986424
Pakistan	PAK	Tretated	0.938314
Panama	PAN	Treated	0.99757
Papua New Guinea	PNG	Treated	0.996202
Paraguay	PRY	Treated	0.98938
Peru	PER	Treated	0.985482
Philippines	PHL	Treated	0.971461
Poland	POL	Donor	0.96607
Portugal	PRT	Donor	0.992968
Qatar	QAT	Untreated	0.99559
Russian Federation	RUS	Untreated	0.71649
Rwanda	RWA	Treated	0.998085
Saint Kitts and Nevis	KNA	Untreated	0.999955
Saint Lucia	LCA	Untreated	0.999777
Saint Vincent and Grenadines	VCT	Untreated	0.999973
Samoa	WSM	Untreated	0.999911
Sao Tome and Principe	STP	Treated	0.999973
Saudi Arabia	SAU	Untreated	0.96773
Senegal	SEN	Treated	0.994075
Serbia	SRB	Treated	0.994221
Seychelles	SYC	Untreated	0.999947
Sierra Leone	SLE	Untreated	0.998303
Singapore	SGP	Untreated	0.99369
Slovakia	SVK	Donor	0.996441
Slovenia	SVN	Donor	0.998071
Solomon Islands	SLB	Untreated	0.999893
South Africa	ZAF	Treated	0.951351
Spain	ESP	Donor	0.96571
Sri Lanka	LKA	Treated	0.992422
Sudan	SDN	Untreated	0.953046
Suriname	SUR	Untreated	0.999474
Swaziland	SWZ	Untreated	0.999413
Sweden	SWE	Donor	0.993315
Switzerland	CHE	Donor	0.995168
Syria	SYR	Treated	0.98728
Tajikistan	TJK	Treated	0.997561
Tanzania	TZA	Treated	0.978226

(continued)

Country	Iso 3	Donor/Treated/Untreated	EPI
Thailand	THA	Treated	0.954946
Togo	TGO	Untreated	0.998476
Tonga	TON	Treated	0.999926
Trinidad and Tobago	TTO	Untreated	0.998105
Tunisia	TUN	Treated	0.995896
Turkey	TUR	Treated	0.960223
Turkmenistan	TKM	Untreated	0.985629
Uganda	UGA	Treated	0.989767
Ukraine	UKR	Treated	0.957511
United Arab Emirates	ARE	Untreated	0.982603
United Kingdom	GBR	Donor	0.946788
United States	USA	Donor	0
Uruguay	URY	Treated	0.990651
Uzbekistan	UZB	Untreated	0.959177
Vanuatu	VUT	Treated	0.999811
Venezuela	VEN	Treated	0.964524
Vietnam	VNM	Treated	0.96563
Yemen	YEM	Treated	0.995981
Zambia	ZMB	Treated	0.982392
Zimbabwe	ZWE	Untreated	0.993995