Replication: The Structure of Inequality and the Politics of Redistribution

Filippo Teoldi, Zara Riaz and Julian Gerez

October 23rd, 2018

First we open the dataset with the `haven` package, which allows us to open .dta files.

```
library('haven')
directory <- "/Users/juliangerez/Google Drive/Semester_Fall_2018/Political Economy of Development/Lupu-
data <- read_dta(paste0(directory, "LupPon_APSR.dta"))
```

Data cleaning

First, the authors redefine inverse disproportionality measures, `disp_gall` as such.

```
data$disp_gall <- data$disp_gall*-1
```

Then the variables female participation, `fempar`, and annual net union density, `union` are multiplied by 100 so that they are rescaled.

```
data$fempar <- data$fempar*100
data$union <- data$union*100
```

The variables `pjoint` and `disp_gall`, are partisanship and disproportionality, respectively. These are standardized from [0,1]. To do so, we are defining a function, `range01`, which standardizes the range of a variable such that it takes on values from 0 to 1.

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

data$stdpjoint <- range01(data$pjoint)
data$stdpdisp_gall <- range01(data$disp_gall)
```

Next, we interpolate missing values by first defining all the variables that need to be interpolated: `pratio9050`, `pratio5010`, `pratio9050s`, `pratio5010s`, `pforeign`, and `pvoc`. To interpolate missing values for each country, rather than for the dataset as a whole, we write a loop to define the object `data_countries` as a list of the data (with these aforementioned new variables) subsetted by each country.

```
data$pratio9050 <- NA
data$pratio5010 <- NA
data$pratio9050s <- NA
data$pratio5010s <- NA
data$pforeign <- NA
data$pvoc <- NA

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)
```

At this point, we can interpolate missing values for each variable. The `zoo` package allows use to use the function `na.approx` to linearly interpolate missing values. We use a set of loops that interpolates missing values indexed for each country, `i`, in our list of `data.frames`, `data_countries`, *for each variable.*[1] Finally, we can use `rbind` to bind this new list into a single `data.frame`, and remove our list of `data.frames`.

```
library('zoo')

# Interpolate pratio9050 (data_countries[[i]][,24]) using ratio9050 (data_countries[[i]][,5])

for (i in 1:length(data_countries)){
data_countries[[i]][,24] <- na.approx(data_countries[[i]][,5], x = index(data_countries[[i]][,3], data_c
}

# Interpolate pratio5010 (data_countries[[i]][,25]) using ratio5010 (data_countries[[i]][,6])

for (i in 1:length(data_countries)){
data_countries[[i]][,25] <- na.approx(data_countries[[i]][,6], x = index(data_countries[[i]][,3], data_c
}

# Interpolate pratio9050s (data_countries[[i]][,26]) using ratio9050s (data_countries[[i]][,7])

for (i in 1:length(data_countries)){
data_countries[[i]][,26] <- na.approx(data_countries[[i]][,7], x = index(data_countries[[i]][,3], data_c
}

# Interpolate pratio5010s (data_countries[[i]][,27]) using ratio9050 (data_countries[[i]][,8])

for (i in 1:length(data_countries)){
data_countries[[i]][,27] <- na.approx(data_countries[[i]][,8], x = index(data_countries[[i]][,3], data_c
}

# Interpolate pforeign (data_countries[[i]][,28]) using foreign (data_countries[[i]][,16])

for (i in 1:length(data_countries)){
data_countries[[i]][,28] <- na.approx(data_countries[[i]][,16], x = index(data_countries[[i]][,3], data_
}

# Interpolate pvoc (data_countries[[i]][,29]) using ratio9050 (data_countries[[i]][,19])

for (i in 1:length(data_countries)){
data_countries[[i]][,29] <- na.approx(data_countries[[i]][,19], x = index(data_countries[[i]][,3], data_
}

data <- do.call("rbind", data_countries)
rm(data_countries)
```

We generate an immigration measure, `fpop` which reflects the percentage of the population that is foreign-born by using our interpolated measure `pforeign`, multiplying it by 1000, and dividing this result by `pop`, which is total population.

```
data$pforeign <- data$pforeign*1000
data$fpop <- data$pforeign/data$pop
```

[1] This is what `data_countries[[i]][,y>23]` refers to, where `i` is each country and `y` represents the new variables. The 24th column is `pratio9050`, the 25th column `pratio5010`, and so on. Each of these are interpolated using the original variables, which are represented in `data_countries[[i]][,z>5]`, where `z` represents the original variables corresponding the new variables (i.e. `pratio9050` is interpolated using `ratio9050`, which is in the 5th column, and so on). Note that the index along which the function is operating is by year (`data_contries[[i]][,3]`) for *every* variable. In other words, we are replacing the variables of interest in each country for missing years.

Our last data cleaning step before moving on to generating the averages for the redistribution models is to generate additional measures of inequality as defined by manipulations to our existing measures of inequality: `ratio9010`, `ratio9010s`, `skew`, and `skews`.

```
data$ratio9010 <- data$pratio9050*data$pratio5010
data$ratio9010s <- data$pratio9050s*data$pratio5010s #not extrapolated
data$skew <- data$pratio9050/data$pratio5010
data$skews <- data$pratio9050s/data$pratio5010s #not extrapolated
```

Because data on redistribution are unequally spaced for the period of the study, the authors use a time series cross sectional model where the indepdent variables are averaged across the period since the last redistribution observation.

We generate moving averages for the redistribution models by using a series of loops. First we generate the `since` variable, which represents the years since the last redistribution, `redist`, for each country. We remake our list of the subset of countries as before and define `since` (`data_countries[[i]][35]`) accordingly by creating a new logical vector, `nona`, that tells us when the `redist` variable is and is not defined for each country.

```
data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  data_countries[[i]] <- cbind(data_countries[[i]], NA)
  nona <- !is.na(data_countries[[i]][,4])
  data_countries[[i]][,35][nona] <- c(NA, diff(data_countries[[i]][,3][nona]))
}

data <- do.call("rbind", data_countries)
names(data)[35] <- "since"
rm(data_countries)
```

Now we can calculate the moving averages:

```
library('dplyr')

# Calculate moving average for ratio9010 (var31)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+j] <- lag(rollapply(data_countries[[i]][,31], j, FUN = mean, fill = NA, ali
  }
}

data <- do.call("rbind", data_countries)

data[,46] <- NA

data[,46] <- case_when(
   data[,35] ==  1 ~ data[,36],
   data[,35] ==  2 ~ data[,37],
```

```r
   data[,35] ==  3 ~ data[,38],
   data[,35] ==  4 ~ data[,39],
   data[,35] ==  5 ~ data[,40],
   data[,35] ==  6 ~ data[,41],
   data[,35] ==  7 ~ data[,42],
   data[,35] ==  8 ~ data[,43],
   data[,35] ==  9 ~ data[,44],
   data[,35] ==  10 ~ data[,45]
)

# Calculate moving average for pratio9050 (var24)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+11+j] <- lag(rollapply(data_countries[[i]][,24], j, FUN = mean, fill = NA, a
  }
}

data <- do.call("rbind", data_countries)

data[,46+11] <- NA

data[,46+11] <- case_when(
   data[,35] ==  1 ~ data[,36+11],
   data[,35] ==  2 ~ data[,37+11],
   data[,35] ==  3 ~ data[,38+11],
   data[,35] ==  4 ~ data[,39+11],
   data[,35] ==  5 ~ data[,40+11],
   data[,35] ==  6 ~ data[,41+11],
   data[,35] ==  7 ~ data[,42+11],
   data[,35] ==  8 ~ data[,43+11],
   data[,35] ==  9 ~ data[,44+11],
   data[,35] ==  10 ~ data[,45+11]
)

# Calculate moving average for pratio5010 (var25)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+22+j] <- lag(rollapply(data_countries[[i]][,25], j, FUN = mean, fill = NA, a
  }
}

data <- do.call("rbind", data_countries)
```

```r
data[,46+22] <- NA

data[,46+22] <- case_when(
   data[,35] ==  1 ~ data[,36+22],
   data[,35] ==  2 ~ data[,37+22],
   data[,35] ==  3 ~ data[,38+22],
   data[,35] ==  4 ~ data[,39+22],
   data[,35] ==  5 ~ data[,40+22],
   data[,35] ==  6 ~ data[,41+22],
   data[,35] ==  7 ~ data[,42+22],
   data[,35] ==  8 ~ data[,43+22],
   data[,35] ==  9 ~ data[,44+22],
   data[,35] ==  10 ~ data[,45+22]
)

# Calculate moving average for stdpjoint (var22)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+33+j] <- lag(rollapply(data_countries[[i]][,22], j, FUN = mean, fill = NA, a
  }
}

data <- do.call("rbind", data_countries)

data[,46+33] <- NA

data[,46+33] <- case_when(
   data[,35] ==  1 ~ data[,36+33],
   data[,35] ==  2 ~ data[,37+33],
   data[,35] ==  3 ~ data[,38+33],
   data[,35] ==  4 ~ data[,39+33],
   data[,35] ==  5 ~ data[,40+33],
   data[,35] ==  6 ~ data[,41+33],
   data[,35] ==  7 ~ data[,42+33],
   data[,35] ==  8 ~ data[,43+33],
   data[,35] ==  9 ~ data[,44+33],
   data[,35] ==  10 ~ data[,45+33]
)

# Calculate moving average for skew (var33)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+44+j] <- lag(rollapply(data_countries[[i]][,33], j, FUN = mean, fill = NA, a
```

```
  }
}

data <- do.call("rbind", data_countries)

data[,46+44] <- NA

data[,46+44] <- case_when(
   data[,35] ==  1 ~ data[,36+44],
   data[,35] ==  2 ~ data[,37+44],
   data[,35] ==  3 ~ data[,38+44],
   data[,35] ==  4 ~ data[,39+44],
   data[,35] ==  5 ~ data[,40+44],
   data[,35] ==  6 ~ data[,41+44],
   data[,35] ==  7 ~ data[,42+44],
   data[,35] ==  8 ~ data[,43+44],
   data[,35] ==  9 ~ data[,44+44],
   data[,35] ==  10 ~ data[,45+44]
)

# Calculate moving average for stddisp_gall (var23)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+55+j] <- lag(rollapply(data_countries[[i]][,23], j, FUN = mean, fill = NA, a
  }
}

data <- do.call("rbind", data_countries)

data[,46+55] <- NA

data[,46+55] <- case_when(
   data[,35] ==  1 ~ data[,36+55],
   data[,35] ==  2 ~ data[,37+55],
   data[,35] ==  3 ~ data[,38+55],
   data[,35] ==  4 ~ data[,39+55],
   data[,35] ==  5 ~ data[,40+55],
   data[,35] ==  6 ~ data[,41+55],
   data[,35] ==  7 ~ data[,42+55],
   data[,35] ==  8 ~ data[,43+55],
   data[,35] ==  9 ~ data[,44+55],
   data[,35] ==  10 ~ data[,45+55]
)

# Calculate moving average for pvoc (var29)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
```

```
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+66+j] <- lag(rollapply(data_countries[[i]][,29], j, FUN = mean, fill = NA, a
  }
}

data <- do.call("rbind", data_countries)

data[,46+66] <- NA

data[,46+66] <- case_when(
   data[,35] ==  1 ~ data[,36+66],
   data[,35] ==  2 ~ data[,37+66],
   data[,35] ==  3 ~ data[,38+66],
   data[,35] ==  4 ~ data[,39+66],
   data[,35] ==  5 ~ data[,40+66],
   data[,35] ==  6 ~ data[,41+66],
   data[,35] ==  7 ~ data[,42+66],
   data[,35] ==  8 ~ data[,43+66],
   data[,35] ==  9 ~ data[,44+66],
   data[,35] ==  10 ~ data[,45+66]
)

# Calculate moving average for union (var12)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+77+j] <- lag(rollapply(data_countries[[i]][,12], j, FUN = mean, fill = NA, a
  }
}

data <- do.call("rbind", data_countries)

data[,46+77] <- NA

data[,46+77] <- case_when(
   data[,35] ==  1 ~ data[,36+77],
   data[,35] ==  2 ~ data[,37+77],
   data[,35] ==  3 ~ data[,38+77],
   data[,35] ==  4 ~ data[,39+77],
   data[,35] ==  5 ~ data[,40+77],
   data[,35] ==  6 ~ data[,41+77],
   data[,35] ==  7 ~ data[,42+77],
   data[,35] ==  8 ~ data[,43+77],
   data[,35] ==  9 ~ data[,44+77],
   data[,35] ==  10 ~ data[,45+77]
)
```

```r
# Calculate moving average for fpop (var30)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+88+j] <- lag(rollapply(data_countries[[i]][,30], j, FUN = mean, fill = NA, a
  }
}

data <- do.call("rbind", data_countries)

data[,46+88] <- NA

data[,46+88] <- case_when(
   data[,35] ==  1 ~ data[,36+88],
   data[,35] ==  2 ~ data[,37+88],
   data[,35] ==  3 ~ data[,38+88],
   data[,35] ==  4 ~ data[,39+88],
   data[,35] ==  5 ~ data[,40+88],
   data[,35] ==  6 ~ data[,41+88],
   data[,35] ==  7 ~ data[,42+88],
   data[,35] ==  8 ~ data[,43+88],
   data[,35] ==  9 ~ data[,44+88],
   data[,35] ==  10 ~ data[,45+88]
)

# Calculate moving average for fempar (var10)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+99+j] <- lag(rollapply(data_countries[[i]][,10], j, FUN = mean, fill = NA, a
  }
}

data <- do.call("rbind", data_countries)

data[,46+99] <- NA

data[,46+99] <- case_when(
   data[,35] ==  1 ~ data[,36+99],
   data[,35] ==  2 ~ data[,37+99],
   data[,35] ==  3 ~ data[,38+99],
   data[,35] ==  4 ~ data[,39+99],
   data[,35] ==  5 ~ data[,40+99],
   data[,35] ==  6 ~ data[,41+99],
   data[,35] ==  7 ~ data[,42+99],
```

```r
   data[,35] ==  8 ~ data[,43+99],
   data[,35] ==  9 ~ data[,44+99],
   data[,35] ==  10 ~ data[,45+99]
)

# Calculate moving average for unempl (var11)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+110+j] <- lag(rollapply(data_countries[[i]][,11], j, FUN = mean, fill = NA,
  }
}

data <- do.call("rbind", data_countries)

data[,46+110] <- NA

data[,46+110] <- case_when(
   data[,35] ==  1 ~ data[,36+110],
   data[,35] ==  2 ~ data[,37+110],
   data[,35] ==  3 ~ data[,38+110],
   data[,35] ==  4 ~ data[,39+110],
   data[,35] ==  5 ~ data[,40+110],
   data[,35] ==  6 ~ data[,41+110],
   data[,35] ==  7 ~ data[,42+110],
   data[,35] ==  8 ~ data[,43+110],
   data[,35] ==  9 ~ data[,44+110],
   data[,35] ==  10 ~ data[,45+100]
)

# Calculate moving average for turnout (var13)

data_countries <- lapply(unique(data$country), function(x)
  subset(data, data$country==x)
)

for (i in 1:length(data_countries)){
  for (j in 1:10) {
    data_countries[[i]][,35+121+j] <- lag(rollapply(data_countries[[i]][,13], j, FUN = mean, fill = NA,
  }
}

data <- do.call("rbind", data_countries)

data[,46+121] <- NA

data[,46+121] <- case_when(
   data[,35] ==  1 ~ data[,36+121],
   data[,35] ==  2 ~ data[,37+121],
```

```r
    data[,35] ==  3 ~ data[,38+121],
    data[,35] ==  4 ~ data[,39+121],
    data[,35] ==  5 ~ data[,40+121],
    data[,35] ==  6 ~ data[,41+121],
    data[,35] ==  7 ~ data[,42+121],
    data[,35] ==  8 ~ data[,43+121],
    data[,35] ==  9 ~ data[,44+121],
    data[,35] ==  10 ~ data[,45+121]
)
```

Now, we match these moving averages to redistribution observations by creating a new set of independent variables with values that correspond to the correct moving average based on the period of redistribution. There are three possible scenarios here: 1) A redistribution observation is observed 1 year after the previous: the independent variable takes on its 1-year lagged value. 2) A redistribution observation is observed n years ago, where n is [2,10]: the independent variable takes on its nth year moving average value. 3) A redstribution observation is the first observation for the country: the indpeendnet variable takes on its 10th year moving average value.

Social Spending: To estimate the model using the 2nd dependent variable (socspend), we create five-year moving averages for this variable and all independent variables to represent a slow-moving causal process.

Replication

Redistribution models

Replicating results from Table 2:

```r
library('panelAR')

##Subsetting data and defining time series sequence:

redistsample<- data[!is.na(data$redist),]
redistsample$time<- unlist(by(redistsample,redistsample$id,function(x) seq(1:nrow(x))))
```

Specification 1:

```r
out1 <- panelAR(redist ~ redist.lag + dvratio9050 + dvratio5010 + dvturnout + dvfempar + dvpropind + dv
print(summary(out1))
```

Specification 2 (remove outliers)

```r
#defining outliers
mod1.resid <- out1$residuals
index <- which(abs((mod1.resid-mean(mod1.resid))/sd(mod1.resid)) <= 1.5)
#creating a new subset without these observations
redistsample_noout<- out1$model[index,]

#running same model as spec1 with new subset
```

```
out2 <- panelAR(redist ~ redist.lag + dvratio9050 + dvratio5010 + dvturnout + dvfempar + dvpropind + dvp
print(summary(out2))
```

Specification 3 (no controls)

```
out3 <- panelAR(redist ~ ratio9050 + ratio5010 + as.factor(id), data=redistsample, panelVar='id', timeVa
print(summary(out3))
```

Specification 4 (no controls, no outliers)

```
#defining outliers
mod3.resid <- out3$residuals
index <- which(abs((mod3.resid-mean(mod3.resid))/sd(mod3.resid)) <= 1.5)
#creating a new subset without these observations
redistsample_noout<- out3$model[index,]
#running same model as spec3 with new subset
out4 <- panelAR(redist ~ ratio9050 + ratio5010 + as.factor(id), data=redistsample_noout, panelVar='id',
print(summary(out4))
```

Specification 5 (Using skew as main inequality measure)

```
out5<- panelAR(redist ~ redist.lag + dvratio9010 + dvskew + dvturnout + dvfempar + dvpropind + dvpvoc +
print(summary(out5))
```

Specification 6 (Skew as main measure, no outliers)

```
mod5.resid <- out5$residuals
index <- which(abs((mod5.resid-mean(mod5.resid))/sd(mod5.resid)) <= 1.5)
#creating a new subset without these observations
redistsample_noout<- out5$model[index,]
#running same model as spec5 with new subset
out6<- panelAR(redist ~ redist.lag + dvratio9010 + dvskew + dvturnout + dvfempar + dvpropind + dvpvoc +
print(summary(out6))
```

Specification 7 (Skew as main measure, no controls, country fixed effects)

```
out7 <- panelAR(redist ~ dvratio9010 + dvskew + as.factor(id), data=redistsample, panelVar='id', timeVar
print(summary(out7))
```

Specification 8 (Skew as main measure, no controls, fixed effects without outliers)

```
mod7.resid <- out7$residuals
index <- which(abs((mod7.resid-mean(mod7.resid))/sd(mod7.resid)) <= 1.5)
#creating a new subset without these observations
redistsample_noout<- out7$model[index,]
#running same model as spec7 with new subset
out8 <- panelAR(redist ~ dvratio9010 + dvskew + as.factor(id), data=redistsample_noout, panelVar='id', t
print(summary(out8))
```

Social spending models

For the next table, we use the same 8 specifications but replace our dependent variable with social spending (socspend) and the 5-year moving averages of the independent variable names. We use the full data set for these specifications, except when we drop the outliers.

```
#Creating the time series indicator for the main data set
data$time<- unlist(by(data,data$id,function(x) seq(1:nrow(x))))
```

Specification 9:

```
out9 <- panelAR(socspend ~ socspend.lag + maratio9050 + maratio5010 + maturnout + mafempar + mapropind +
print(summary(out9))
```

Specification 10 (remove outliers)

```
#defining outliers
mod9.resid <- out9$residuals
index <- which(abs((mod9.resid-mean(mod9.resid))/sd(mod9.resid)) <= 1.5)
#creating a new subset without these observations
data_noout<- out9$model[index,]

#running same model as spec9 with new subset
out10 <- panelAR(socspend ~ socspend.lag + maratio9050 + maratio5010 + maturnout + mafempar + mapropind
print(summary(out10))
```

Specification 11 (no controls)

```
out11 <- panelAR(socspend ~ maratio9050 + maratio5010 + as.factor(id), data=data, panelVar='id', timeVa
print(summary(out11))
```

Specification 12 (no controls, no outliers)

```
#defining outliers
mod11.resid <- out11$residuals
index <- which(abs((mod11.resid-mean(mod11.resid))/sd(mod11.resid)) <= 1.5)
#creating a new subset without these observations
data_noout<- out11$model[index,]
#running same model as spec11 with new subset
out12 <- panelAR(socspend ~ maratio9050 + maratio5010 + as.factor(id), data=data_noout, panelVar='id', t
print(summary(out12))
```

Specification 13 (Using skew as main inequality measure)

```
out13<- panelAR(socspend ~ socspend.lag + maratio9010 + maskew + maturnout + mafempar + mapropind + map
print(summary(out13))
```

Specification 14 (Skew as main measure, no outliers)

```
mod13.resid <- out13$residuals
index <- which(abs((mod13.resid-mean(mod13.resid))/sd(mod13.resid)) <= 1.5)
#creating a new subset without these observations
data_nooutt<- out13$model[index,]
#running same model as spec13 with new subset
out14<- panelAR(socspend ~ socspend.lag + maratio9010 + maskew + maturnout + mafempar + mapropind + map
print(summary(out14))
```

Specification 15 (Skew as main measure, no controls, country fixed effects)

```
out15 <- panelAR(socspend ~ maratio9010 + maskew + as.factor(id), data=data, panelVar='id', timeVar='ti
print(summary(out15))
```

Specification 16 (Skew as main measure, no controls, fixed effects without outliers)

```
mod15.resid <- out15$residuals
index <- which(abs((mod15.resid-mean(mod15.resid))/sd(mod15.resid)) <= 1.5)
#creating a new subset without these observations
data_noout<- out15$model[index,]
#running same model as spec15 with new subset
out16 <- panelAR(socspend ~ maratio9010 + maskew + as.factor(id), data=data_noout, panelVar='id', timeVa
print(summary(out16))
```

Immigration

Partisanship

Redistribution and social spending with partisanship

Robustness checks via design modification

Extension

Design declaration

We start by loading in the DeclareDesign package and defining the elements of the design.

- declare_population refers to the sample size of the study. The study concerns country-year units. In this case, there are 858 observations.
- declare_potential_oucomes refers to

```
library('DeclareDesign')

# X: take some parameters based on a simple model of X on Y

modX <- lm(data$redist ~ data$skew)
a_X <- summary(modX)$coefficients["(Intercept)","Estimate"]
b_X <- summary(modX)$coefficients["data$skew","Estimate"]
sd_X <- 1

rho_XY <- -.5 # Confounding
sd_X_type <- .1 # sd on effect heterogeneity
sd_Y_type <- .005 # sd on compliance heterogeneity
rho_XY_type <- 0 # Possible correlation between compliance and effects
```

```
population <- declare_population(
  N = 858,
  redist = sample(data$redist, N, replace = TRUE),
  u_X = rnorm(N, sd = sd_X),
  u_X_type = rnorm(N, df = sd_X_type)
)
```

```
fx <- function(a_X, b_X, u_X_type, u_X)
a_X + (b_X + u_X_type) + u_X

potentials <- declare_step(handler = fabricate,
  redist = fx(skew, a_X, b_X, u_X_type, u_X))
```

```
estimand <- declare_estimand(
  ols = mean((fx(max(skew), a_X, b_X, u_X_type, u_X) - fx(min(skew), a_X, b_X, u_X_type, u_X))/(max(ske
```

```
estimator_1 <- declare_estimator(redist ~ skew, estimand = "ols",
model = lm_robust, label = "lm")
```

```
lupu_pontusson_2011_design <- population + potentials + estimand + estimator_1
```