Using the SWIID in R

Frederick Solt
Associate Professor of Political Science
University of Iowa
frederick-solt@uiowa.edu

The Standardized World Income Inequality Database (SWIID) takes a Bayesian approach to standardizing observations collected from the OECD Income Distribution Database, the Socio-Economic Database for Latin America and the Caribbean generated by CEDLAS and the World Bank, Eurostat, the World Bank's PovcalNet, the UN Economic Commission for Latin America and the Caribbean, national statistical offices around the world, and many other sources. Luxembourg Income Study data serves as the standard.

As described in Solt (2019), the SWIID maximizes the comparability of available income inequality data for the broadest possible sample of countries and years. But incomparability remains, and it is sometimes substantial. This remaining incomparability is reflected in the standard errors of the SWIID estimates, making it often crucial to take this uncertainty into account when making comparisons across countries or over time (Solt 2009, 238; Solt 2016, 14; Solt 2019). It was once the case that incorporating the standard errors into an analysis required considerable effort. It is now straightforward.

In version 8.1 of the SWIID, the inequality estimates and their associated uncertainty are represented by 100 draws from the posterior distribution: for any given observation, the differences across these imputations capture the uncertainty in the estimate. The `swiid8_1.zip` includes the file `swiid8_1.rda`, which is pre-formatted to facilitate taking this uncertainty into account. The following sections describe how to subset the data, merge in additional variables, and do analyses.

1 Getting Started

Loading the file `swiid8_1.rda` adds `swiid` to the Global Environment, a list of 100 dataframes. Each of these dataframes includes, in addition to country and year identifiers, the following four variables:

- `gini_disp`: Estimate of Gini index of inequality in equivalized (square root scale) household disposable (post-tax, post-transfer) income, using Luxembourg Income Study data as the standard.
- `gini_mkt`: Estimate of Gini index of inequality in equivalized (square root scale) household market (pre-tax, pre-transfer) income, using Luxembourg Income Study data as the standard.
- `abs_red`: Estimated absolute redistribution, the number of Gini-index points market-income inequality is reduced due to taxes and transfers: the difference between the `gini_mkt` and `gini_disp`.

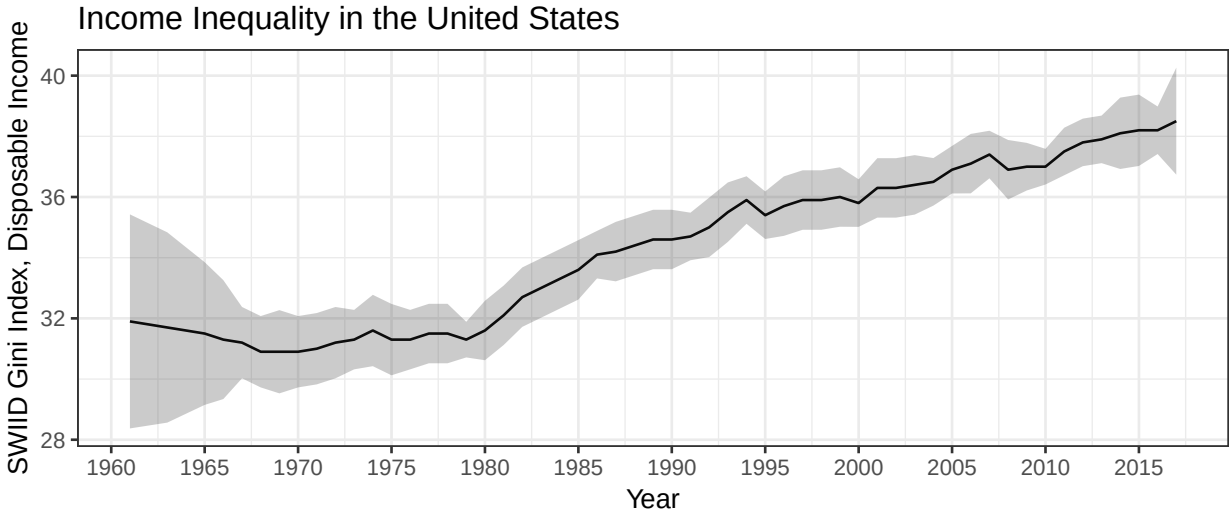
- `rel_red`: Estimated relative redistribution, the percentage reduction in market-income inequality due to taxes and transfers: the difference between the `gini_mkt` and `gini_disp`, divided by `gini_mkt`, multiplied by 100.

The variation in these variables across the 100 dataframes captures the uncertainty in the SWIID estimates. As described below, this format facilitates taking this uncertainty into account when conducting analyses. It does not, however, lend itself easily to tasks such plotting. The mean-plus-standard-error summary format is much better suited to such purposes; for this reason, the object `swiid_summary` is also included in the `swiid8_1.rda`.

```
library(tidyverse)
library(stringr)
library(here)

# Load the SWIID
load("swiid8_1.rda")

# Plot SWIID gini_disp estimates for the United States
swiid_summary %>%
    filter(country == "United States") %>%
    ggplot(aes(x=year, y=gini_disp)) +
    geom_line() +
    geom_ribbon(aes(ymin = gini_disp-1.96*gini_disp_se,
                    ymax = gini_disp+1.96*gini_disp_se,
                    linetype=NA), alpha = .25) +
    scale_x_continuous(breaks=seq(1960, 2015, 5)) +
    theme_bw() +
    labs(x = "Year",
         y = "SWIID Gini Index, Disposable Income",
         title = "Income Inequality in the United States")
```



2 Adding Variables and Subsetting

Adding variables or subsetting the SWIID requires a bit of care, though the `map()` function of the `purrr` package (Wickham 2016) makes the process fairly intuitive. We use `map()` to perform the actions we wish on every element of the SWIID list. Suppose, for example, that we wanted to use the `countrycode` package (Arel-Bundock 2014) to create a `region` variable for each country in the SWIID and then retain only the observations for countries in Latin America and the Caribbean.

```
library(countrycode)

# Generate region variable and subset LAC data
swiid_lac <- swiid %>%
    map(. %>% mutate(region = countrycode(country,
                                          origin = "country.name",
                                          destination = "region"))) %>%
    map(. %>% filter(region == "Caribbean" |
                         region == "Central America" | # WB regions; includes MEX
                         region == "South America"))
```

The result is a new list of 100 data frames, each containing the SWIID data only for the countries of Latin America and the Caribbean.

3 Merging

Merging additional data into the SWIID also needs to be done carefully. Suppose we wanted to do a (simplified) replication of Solt, Habel, and Grant's (2011) analysis of World Values Survey data on religiosity. As our measure of religiosity, we will use the WVS item on respondents' self-report of the importance of God to their lives, which is measured on a ten-point scale. Given secularization theory, we will need to control for GDP per capita, which we will calculate from information from the Penn World Tables (Feenstra, Inklaar, and Timmer 2015). Below we first load the PWT dataset and use it to generate a dataset of GDP per capita (in thousands of dollars). Then we load the WVS data, generate our variables of interest, and merge in our PWT data. Finally, we use `purrr::map()` to merge these data into each of the 100 SWIID dataframes.

```
library(readxl)
library(haven)

# Get GDP per capita data from the Penn World
# Tables, Version 9.0 (Feenstra et al. 2015)
# download.file("http://www.rug.nl/research/ggdc/data/pwt/v90/pwt90.xlsx",
#               "pwt90.xlsx")

pwt90_gdppc <- read_excel("pwt90.xlsx", sheet = "Data") %>%
    transmute(country = country,
              year = year,
              gdppc = rgdpe/pop/1000) %>%
    filter(!is.na(gdppc))

# Get World Values Survey 6-wave data (from
# http://www.worldvaluessurvey.org/WVSDocumentationWVL.jsp),
# generate variables of interest, and merge in the PWT data
wvs <- read_dta("WVS_Longitudinal_1981_2016_stata_v20180912.dta",
                encoding = "latin1") %>%
    transmute(country = countrycode(S003,
                                    origin = "iso3n",
                                    destination = "country.name"),
              year = as.numeric(S020),
              religiosity = ifelse(F063>0, F063, NA),
              male = ifelse(X001>0, as.numeric(X001 == 1), NA),
              educ = ifelse(X025>0, X025, NA),
              age = ifelse(X003>0, X003, NA)) %>%
    filter(complete.cases(.)) %>%
    left_join(pwt90_gdppc, by = c("country", "year"))

# Merge the WVS (and PWT) data into all 100 SWIID dataframes
wvs_swiid <- swiid %>%
    map(. %>% left_join(wvs, by = c("country", "year")))
```

4 Analyzing

Once the desired subset has been extracted and any additional variables created or merged in, we may proceed to analysis. Again, we use `purrr::map()`, this time to estimate our model on each of the 100 different dataframes. Continuing with our example, we estimate a three-level linear mixed-effects model of individual responses nested in country-years nested in countries using `lme4::lmer()` (Bates et al. 2015). Note that performing an analysis 100 times can be time-consuming, so for demonstration purposes, we'll do it on just the first five dataframes here.

```
library(lme4)

# Estimate model on first five dataframes
m1 <- wvs_swiid[1:5] %>% map(~ lmer(religiosity ~ gini_disp + gdppc +
                                    age + educ + male +
                                    (1 | country/year),
                                data = .x))
```

We could now use Rubin's (1987) rules to calculate the mean and standard error for each fixed-effect estimate across these ~~100~~ 5 sets of results (these rules are implemented in the package `mitools` (Lumley 2014) and others), but instead we will use simulation. Using `sim` from the `arm` package (Gelman and Su 2015), we generate the distribution of fixed-effects estimates in the results both for each dataframe and across the ~~100~~ 5 dataframes. Then we will use these distributions to calculate estimates and standard errors for each fixed effect, putting them in a tidy dataframe like that achieved by using `broom::tidy()` (Robinson 2016).

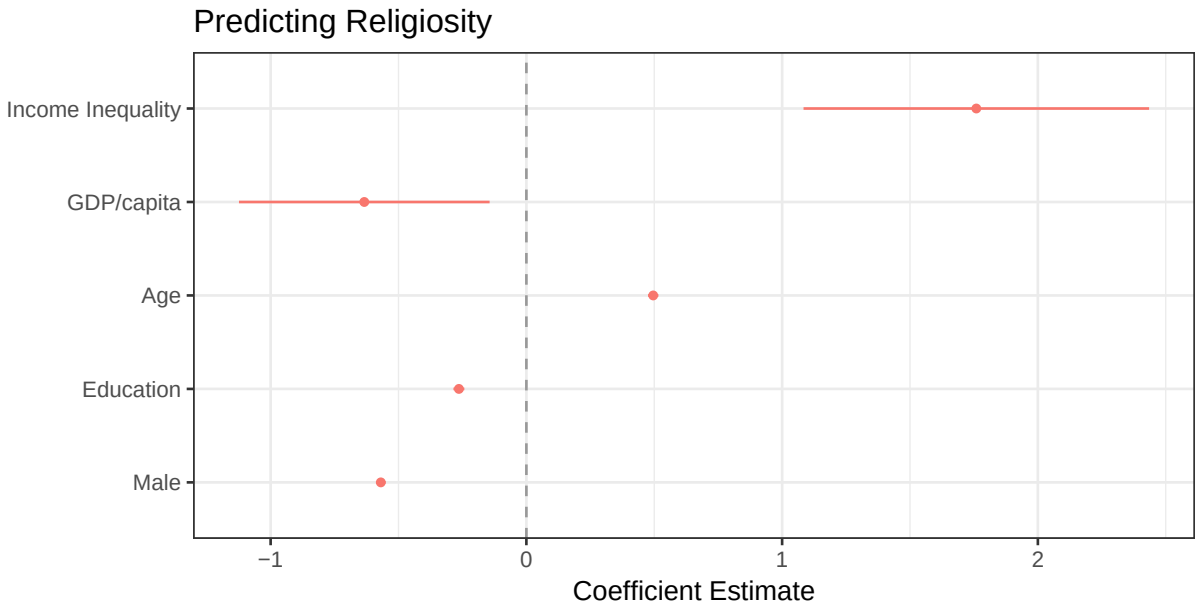
```
# Simulate distribution of estimates
m1_sims <- m1 %>%
    map(. %>% arm::sim(n.sims = 100)) %>%
    map_df(.  %>% slot("fixef") %>% as_tibble())

# Generate a tidy dataframe of results (cf. broom::tidy())
m1_tidy <- data_frame(term = names(m1_sims),
                      estimate = summarise_all(m1_sims, funs(mean)) %>%
                          t() %>%
                          as.vector(),
                      std.error = m1_sims %>%
                          summarise_all(funs(sd)) %>%
                          t() %>%
                          as.vector())
```

Having results in tidy form makes it easy to work with them further. Here we use the `dotwhisker` package (Solt and Hu 2015) to present a dot-and-whisker plot of the results. The dots represent the estimated change on the ten-point religiosity scale for a change of two standard deviations in each independent variable; the whiskers represent the 95% confidence intervals of these estimates.

```r
library(dotwhisker)

# Plot the results
m1_tidy %>%
  by_2sd(wvs_swiid[[1]]) %>%
  relabel_predictors(c(gini_disp = "Income Inequality",
                       gdppc = "GDP/capita",
                       age = "Age",
                       educ = "Education",
                       male = "Male")) %>%
  dwplot() +
  theme_bw() +
  geom_vline(xintercept = 0,
             colour = "grey60",
             linetype = "dashed") +
  theme(legend.position="none") +
  labs(title = "Predicting Religiosity",
       x = "Coefficient Estimate")
```



5 Citing the SWIID

Please cite to the SWIID by referring to its article of record and including the version number and date of release:

Solt, Frederick. 2019. "Measuring Income Inequality Across Countries and Over Time: The Standardized World Income Inequality Database." SWIID Version 8.1, May 2019.

References

Arel-Bundock, Vincent. 2014. "countrycode: Convert Country Names and Country Codes." Available at the Comprehensive R Archive Network (CRAN).

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software* 67(1):1–48.

Feenstra, Robert C., Robert Inklaar, and Marcel P. Timmer. 2015. "The Next Generation of the Penn World Table." *American Economic Review* 105(10):3150–3182.

Gelman, Andrew, and Yu-Sung Su. 2015. "arm: Data Analysis Using Regression and Multilevel/Hierarchical Models." Available at the Comprehensive R Archive Network (CRAN).

Lumley, Thomas. 2014. "mitools: Tools for Multiple Imputation of Missing Data." Available at the Comprehensive R Archive Network (CRAN).

Robinson, David. 2016. "broom: Convert Statistical Analysis Objects into Tidy Data Frames." Available at the Comprehensive R Archive Network (CRAN).

Rubin, Donald B. 1987. *Multiple Imputation for Nonresponse in Surveys.* New York: J. Wiley & Sons.

Solt, Frederick. 2009. "Standardizing the World Income Inequality Database." *Social Science Quarterly* 90(2):231–242.

Solt, Frederick. 2016. "The Standardized World Income Inequality Database." *Social Science Quarterly* 97(5):1267–1281. SWIID Version 8.0, February 2019.

Solt, Frederick. 2019. "Measuring Income Inequality Across Countries and Over Time: The Standardized World Income Inequality Database." SWIID Version 8.0, February 2019.

Solt, Frederick, Philip Habel, and J. Tobin Grant. 2011. "Economic Inequality, Relative Power, and Religiosity." *Social Science Quarterly* 92(2):447–465.

Solt, Frederick, and Yue Hu. 2015. "dotwhisker: Dot-and-Whisker Plots of Regression Results." Available at the Comprehensive R Archive Network (CRAN).

Wickham, Hadley. 2016. "purrr: Functional Programming Tools." Available at the Comprehensive R Archive Network (CRAN).