Updating Data to be used in SRA

Julia Levine

10 August, 2020

Contents

Load in old data

The following code reads in the old data. It then takes the most recent data for each country and replicates it for the years that we are updating – this let's us easily keep the time-invariant values, like region. I then set aside the old data, and update the data that will be appended.

```
# read in data from the previous years , called dat

  load("input/prepared2018predictors_2_11_20.RData")

# carry forward all data from the most recent year

  carry_from <- max(dat$year)

  # this mainly applies to updating multiple years simultaneously
  # it carries forward the most recent data for each of the years to be updated
  dat_new <- rbindlist(rep(list(dat[year == carry_from]), times = length(update_years)))

  # relabel the years

  dat_new$year <- rep(update_years, each = nrow(dat_new))


# rename old data and leave it alone until you've updated all
# variables except for the lag/leads

  dat_old <- dat
  rm(dat)
```

Update

Country age

This chunk updates country age by adding the difference between the updated years and the year the data was carried from to the country age variable.

```
# update country age

  diff <- update_years - carry_from

  for(i in 1:length(update_years)){
    dat_new$countryage[dat_new$year == update_years[i]] <-
    dat_new$countryage[dat_new$year == update_years[i]] + diff[i]
  }

# update logged country age variable

  dat_new$countryage.ln <- log(dat_new$countryage + 1)
```

merge_dat function

The following is a function that removes the variables that were carried forward from the base data, and then merges in the updated variables from the appropriate year.

```
merge_dat <- function(new_data, base_data = dat_new,
                      merge_new_by = c("sftgcode", "year"),
                      merge_base_by = c("sftgcode", "year")){
  new_data_vars <- colnames(new_data)[!colnames(new_data) %in% merge_new_by]
  dat1 <- subset(base_data,
                 select = colnames(base_data)[!(colnames(base_data) %in%
                                                  new_data_vars)])
  dat1 <- merge(dat1, new_data,
                by.x = merge_base_by,
                by.y = merge_new_by, all.x = TRUE)
  dat1
}
```

VDEM

The following code reads in the most recent V-DEM data, selecting the relevant variables and renaming them.

```
# read in data

  vdem <- fread("input/Country_Year_V-Dem_Full+others_CSV_v10/V-Dem-CY-Full+Others-v10.csv")

# subset to just use these variables

  vdem_vars <- c("v2cldmovem_ord",
                 "v2cldmovew_ord",
                 "v2clkill_ord",
                 "v2clsocgrp_ord",
                 "v2clrgunev_ord",
                 "v2csreprss_ord",
                 "v2pepwrsoc_ord",
                 "v2elrstrct",
                 "v2psparban_ord",
                 "v2psoppaut_ord",
                 "v2jureform_ord",
                 "v2clrelig_ord",
                 "v2xcl_disc",
                 "v2pepwrses_ord",
                 "v2pepwrses",
                 "e_migdpgro",
                 "e_mipopula",
                 "e_cow_exports",
                 "e_cow_imports",
                 "e_migdppc"
                 )
  keep <- c("COWcode", "country_name", "year", vdem_vars)

# function that renames and formats variables,
```

```r
# takes dataset and vector of variables to keep as inputs

  source("helper scripts/format_vdem.R")

# function that maps COW codes onto PITF country codes,
# contained in format_vdem

  source("helper scripts/cowtopitf2018.R")

# format and save vdem

  vdem <- format_vdem(dat = vdem, keep = keep)
  save(vdem, file = "input/temp/vdem.Rdata")
```

This chunk removes the old year's V-DEM variables, summarizes missingness, and merges in the new ones.

```r
# read in formatted VDEM variables for the update years

  load("input/temp/vdem.Rdata")

# limit to the update years

  vdem <- vdem[year %in% update_years]

# count missing values
# check which variables are missing values

  missing <- apply(vdem, 2, function(x) sum(is.na(x)))
  kable(missing, col.names = "NA count")
```

	NA count
COWcode	4
country_name	0
year	0
candidaterestriction	1
partyban	0
barrierstoparties	5
judicialreform	0
religiousfreedom	0
freediscussion	0
ses_power_dist	0
e_migdpgro	179
e_mipopula	179
e_cow_exports	179
e_cow_imports	179
e_migdppc	179
sftgcode	0
gdppcgrowth	179
popsize	179
popsize.ln	179
tradeshare	179
pol_killing_approved	0

	NA count
freemove_men4	0
freemove_women4	0
social_inequality	0
even_civilrights	0
repress_civilsoc	0
social_power_dist	0
minorityrule	0

```
# check which variables are missing for all countries, indicating that they
# have not been updated yet, and are potentially discontinued

  all.missing <- names(missing)[missing == nrow(vdem)]
  cat(paste0("The following variables have not been updated as of ",
             Sys.Date(),":\n", paste(all.missing, collapse = "\n")))
```

```
## The following variables have not been updated as of 2020-08-10:
## e_migdpgro
## e_mipopula
## e_cow_exports
## e_cow_imports
## e_migdppc
## gdppcgrowth
## popsize
## popsize.ln
## tradeshare
```

```
# this ensures that we dont replace old country names
# VDEM sometimes changes them

  vdem$country_name <- NULL

# update V-DEM data

  dat_new <- merge_dat(new_data = vdem)
  dat_new[, tradeshare.ln := log(tradeshare)]
```

Polity

We've decided to drop the polity variables, but I'm just updating the durable and pol.durable variables here the same way I updated country age.

```
# update durable the same way we update country age
# Note: pol.durable and durable are the same thing

  diff <- update_years - carry_from

  for(i in 1:length(update_years)){
    dat_new$durable[dat_new$year == update_years[i]] <-
    dat_new$durable[dat_new$year == update_years[i]] + diff[i]
```

```
    dat_new$pol.durable[dat_new$year == update_years[i]] <-
    dat_new$pol.durable[dat_new$year == update_years[i]] + diff[i]
  }

# update the logged variables

  dat_new$pol.durable.ln <- log(dat_new$pol.durable + 1)
  dat_new$durable.ln <- dat_new$pol.durable.ln
```

WDI

This chunk pulls select variables from the WDI API.

```
# In June 2020 WDI decommissioned V1 of the API
# so you may need to re-install the WDI package.

  # devtools::install_github("vincentarelbundock/WDI")
  library(WDI)

# list indicators we want to pull

  wdilist <- c("NE.TRD.GNFS.ZS",     # Trade (% of GDP)
               "NY.GDP.PCAP.PP.KD",  # GDP per capita, PPP (constant 2005 intl $)
               "NY.GDP.PCAP.KD",     # GDP per capita (constant 2000 US$)
               "NY.GDP.MKTP.KD.ZG",  # GDP growth (annual %)
               "FP.CPI.TOTL.ZG",     # Inflation, consumer prices (annual %)
               "SP.POP.TOTL",        # Population, total
               "SP.DYN.IMRT.IN"      # Infant mortality rate
               #"NE.EXP.GNFS.ZS",     # exports (% of GDP)
               #"NE.IMP.GNFS.ZS"      # imports % of GDP

      )

  # confirm that all of the old indicator names are in use

    (check.names <- rbindlist(lapply(wdilist, function(x) {
      # search the indicator list for each of the variables in wdilist
       res <- WDIsearch(x, field = "indicator")
       # take the first match (drops those with extra letters on the end)
       if(length(res) > 2){
         res <- res[1, ]
       }
       data.table("indicator" = res[1], "name" = res[2])})))

# Extract latest version of desired variables from WDI
# also pull 5 years from before so that we can carry forward
# more recent values for variables like infant mortality that are slow to update

  wdi <- WDI(country="all", indicator=wdilist, extra=FALSE,
             start=(min(update_years) - 5))

# Add PITF country codes for merging
```

```
  source("helper scripts/f.pitfcodeit.R")
  wdi <- pitfcodeit(wdi, "country")
  wdi$country <- as.character(wdi$country)

# Subset to drop cases with missing PITF codes, cut extra id vars

  wdi <- subset(wdi, !is.na(sftgcode), select=-c(1, 2))

# Reorder for easier review

  wdi <- wdi[order(wdi$sftgcode, wdi$year),]

# Rename variables-- add a "new" to indicate these are newly brought in from wdi
# to avoid reusing names already in the old EWP data

  names(wdi) <- c("year",
                  "wdi.trade.new",
                  "wdi.gdppcppp.new",
                  "wdi.gdppc.new",
                  "wdi.gdppcgrow.new",
                  "wdi.inflation.new",
                  "wdi.popsize.new",
                  "wdi.imrate.new",
                  #"wdi.expshare.new",
                  #"wdi.impshare.new",
                  "sftgcode"
                  )

# save to the input folder

  fwrite(wdi, paste0("input/temp/wdi/pulled-", Sys.Date(), ".csv"))
```

```
# function to carry forward values of a variable for countries where it's NA

  carry_forward <- function(variable, carry_to = 2019, data = dat, carry_from = 2018){
    na_var <- data[year == carry_to & is.na(get(variable))]$sftgcode

    cat(paste0("Missing values for ", length(na_var), " countries."))

    carry_var <- subset(data[year == carry_from & sftgcode %in% na_var],
                        select = c("sftgcode", variable))
    cat(paste0("\nFilling in values for ",
               nrow(carry_var[!is.na(get(variable))]), " countries."))

    for(i in 1:length(na_var)){
      data[year == carry_to & sftgcode == na_var[i],
           (variable) := carry_var[sftgcode == na_var[i], get(variable)]]
    }
  }
```

This chunk reads in the latest pull from the WDI API, counts the missing values, and merges it into the main dataset.

```r
# list all files in the wdi directory and read in the latest pull

  wdi.files <- list.files("input/temp/wdi/")
  latest.pull <- which.max(lapply(wdi.files,
                                  function(x) {
                                    as.Date(gsub("pulled-|.csv", "", x))}))
  wdi <- fread(paste0("input/temp/wdi/", wdi.files[latest.pull]))
  new.wdi <- wdi[year %in% update_years]

# check which variables are missing values

  missing <- apply(subset(new.wdi,
                          select = -c(1, ncol(new.wdi))), 2,
                   function(x) sum(is.na(x)))
  kable(missing, col.names = "NA count")
```

	NA count
wdi.trade.new	36
wdi.gdppcppp.new	10
wdi.gdppc.new	11
wdi.gdppcgrow.new	10
wdi.inflation.new	30
wdi.popsize.new	1
wdi.imrate.new	170

```r
# check which variables are missing for all countries, indicating that they
# have not been updated yet, but potentially will be in the future

  all.missing <- names(missing)[missing == length(unique(wdi$sftgcode))]
  cat(paste(all.missing, "not updated as of", Sys.Date()))
```

```
## wdi.imrate.new not updated as of 2020-08-10
```

```r
  # we have been carrying forward a 2015 version of imrate
  # update the value to be carried forward to the most recent non-missing value

  old.wdi <- wdi[year < min(update_years) & !is.na(wdi.imrate.new)]
  to.carry <- old.wdi[, .("wdi.imrate.new" = wdi.imrate.new[which.max(year)]),
                      by = "sftgcode"]
  to.carry[, year := update_years]

  new.wdi <- merge_dat(new_data = to.carry, base_data = new.wdi)

  # do the same thing for missing values of tradeshare and gdp as they were
  # mostly updated in 2018 and the earliest values from CIA factbook are  2017

  wdi <- rbind(new.wdi, old.wdi)

  carry_forward(variable = "wdi.trade.new", carry_from = 2018, data = wdi)
```

```
## Missing values for 36 countries.
```

```
## Filling in values for 20 countries.
```

```
  carry_forward(variable = "wdi.trade.new", carry_from = 2017, data = wdi)
```

```
## Missing values for 16 countries.
## Filling in values for 3 countries.
```

```
  carry_forward(variable = "wdi.trade.new", carry_from = 2016, data = wdi)
```

```
## Missing values for 13 countries.
## Filling in values for 1 countries.
```

```
  carry_forward(variable = "wdi.gdppc.new", carry_from = 2018, data = wdi)
```

```
## Missing values for 11 countries.
## Filling in values for 4 countries.
```

```
  carry_forward(variable = "wdi.gdppc.new", carry_from = 2017, data = wdi)
```

```
## Missing values for 7 countries.
## Filling in values for 1 countries.
```

```
  carry_forward(variable = "wdi.gdppc.new", carry_from = 2016, data = wdi)
```

```
## Missing values for 6 countries.
## Filling in values for 0 countries.
```

```
  missing <- apply(subset(wdi[year %in% update_years],
                          select = -c(1, ncol(wdi))), 2,
                   function(x) sum(is.na(x)))
  kable(missing, col.names = "NA count")
```

	NA count
year	0
wdi.trade.new	12
wdi.gdppcppp.new	10
wdi.gdppc.new	6
wdi.gdppcgrow.new	10
wdi.inflation.new	30
wdi.popsize.new	1

```
# Merge it in to main data

  dat_new <- merge_dat(new_data = wdi[year %in% update_years])
  dat_new[, imr.sqrt := sqrt(wdi.imrate.new)]
  dat_new[, wdi.trade.ln.new := log(wdi.trade.new)]
```

```r
check <- wdi[is.na(wdi.trade.new) & year == 2019]$sftgcode
variable <- "wdi.trade.new"
data <- wdi
wdi[!is.na(wdi.trade.new) & sftgcode %in% check]
```

```
##    sftgcode year wdi.trade.new wdi.gdppcppp.new wdi.gdppc.new
## 1:      SOL 2014     111.97355         2307.501     1446.4362
## 2:      SOL 2015      98.39379         2303.239     1443.7644
## 3:      SSD 2014      64.63452               NA      831.8505
## 4:      SSD 2015      65.55135               NA      730.9320
## 5:      VEN 2014      48.09081               NA    14025.3576
##    wdi.gdppcgrow.new wdi.inflation.new wdi.popsize.new wdi.imrate.new
## 1:          2.250091         5.1659024          587079           19.3
## 2:          2.542240        -0.5744693          603118           18.8
## 3:          3.373648         1.6552236        10554883           63.7
## 4:        -10.793365        52.8146521        10715658           63.7
## 5:         -3.894386        62.1686500        30045134           14.9
```

UCDP battle-related deaths

The following code reads in the UCDP data and changes country names to match. Below, I sum the number of battledeaths for each country in the update years, and merge this onto dat_new.

```r
# read in UCDP data

  ucdp <- fread("input/ucdp-brd-dyadic-201.csv")

# making sure we have the same variables as in 18.1 version.
# colnames for 18.1 are uppercase, use tolower when checking

  colnames(ucdp) <- gsub("_", "", colnames(ucdp))

# limit to the update years and conflict type >= 3

  ucdp <- ucdp[year %in% update_years & typeofconflict >= 3]

# change country names to match

  diff_loc <- setdiff(unique(ucdp$locationinc), unique(dat_new$country_name))

  # drops the locations that involve multiple countries

    diff_loc <- diff_loc[grepl(", ", diff_loc) == F]
    diff_loc

  # change names

    ucdp$locationinc[ucdp$locationinc=="Russia (Soviet Union)"] = "Russia"
    ucdp$locationinc[ucdp$locationinc=="Myanmar (Burma)"] = "Burma/Myanmar"
    ucdp$locationinc[ucdp$locationinc=="Yemen (North Yemen)"] = "Yemen"
    ucdp$locationinc[ucdp$locationinc=="DR Congo (Zaire)"] = "Democratic Republic of Congo"
```

```
    if(length(setdiff(unique(ucdp$locationinc),
                      unique(dat_new$country_name)))>0){
      stop("Different country names")}

# group by year/country and sum battledeaths

  bd <- ucdp[, .("battledeaths" = sum(bdbest)), by = c("locationinc", "year")]

# save battledeath data

  fwrite(bd, "input/temp/battledeaths.csv")
```

I then read in the battledeaths for each country and merge it into the new data, replacing missing values with zeroes and updating the logged version of this variable.

```
# read in battledeaths data

  bd <- fread("input/temp/battledeaths.csv")

# merge battledeaths into main data

  dat_new <- merge_dat(new_dat = bd,
                       merge_new_by = c("locationinc", "year"),
                       merge_base_by = c("country_name", "year"))


# fill in zeroes and update logged variable

  dat_new[, battledeaths := ifelse(is.na(battledeaths), 0, battledeaths)]
  dat_new[, battledeaths.ln := log(battledeaths + 1)]
```

Coup Attempts

This code reads in the most updated version of the Powell and Thyne data which is posted on their website. It selects coups in the years to be updated

```
# pull coup data from powell and thyne websit

  coup_dat <- as.data.table(read.delim("http://www.uky.edu/~clthyn2/coup_data/powell_thyne_coups_final.t

# keep coups in the update years

  new_coup <- coup_dat[year %in% update_years]

# where coup == 2 it was a successful coup
# where coup == 1 it was a failed coup

  new_coup[, cou.s.d := ifelse(coup == 2, 1, 0)]
  new_coup[, cou.f.d := ifelse(coup == 1, 1, 0)]

# save coup data

  fwrite(new_coup, paste0("input/temp/", "coups-pulled-", Sys.Date(), ".csv"))
```

This updates the coup variables in dat_new.

```
# read in the coup data

  temp.files <- list.files("input/temp")
  new_coup <- fread(paste0("input/temp/", temp.files[grep("coups", temp.files)]))

# select variables and print

  new_coup <- subset(new_coup,
                     select = c("country", "year", "cou.s.d", "cou.f.d"))
  kable(new_coup)
```

country	year	cou.s.d	cou.f.d
Gabon	2019	0	1
Sudan	2019	1	0

```
# merge coup data into dat_new

  dat_new <- merge_dat(new_data = new_coup,
                       merge_new_by = c("country", "year"),
                       merge_base_by = c("country_name", "year"))

# fill in missing values with zeroes

  coup_cols <- c("cou.s.d", "cou.f.d")
  dat_new[, (coup_cols) := lapply(.SD, function(x)
    ifelse(is.na(x), 0, x)), .SD = coup_cols]

# update cou.any variable (1 if there was either a failed or successful coup)

  dat_new[, cou.any := ifelse(cou.s.d>0 | cou.f.d>0, 1, 0)]
```

Mass Killing Variables

There were no starts or ends to mass killing events in 2019.

```
# initialize state led mk vars

  dat_new$mkl.end <- 0
  dat_new$mkl.start <- 0

# initialize non-state led mk vars

  dat_new$nonstatemk.end <- 0
  dat_new$nonstatemk.start <- 0
```

Append new data

This appends dat_new to dat_old, creating the full data-set

```
# check to make sure you can append the new data

  if(length(setdiff(colnames(dat_old), colnames(dat_new)))>0){
    stop("different colnames")}
  if(length(setdiff(colnames(dat_new), colnames(dat_old)))>0){
    stop("different colnames")}

# append new data

  setcolorder(dat_new, colnames(dat_old))
  dat <- rbind(dat_old, dat_new)
```

Mass Killing Onsets

I now make changes to the mass killing variables in years prior to the update years.

There is one change to be made: the end of the non-state led mass killing in Thailand in 2016.

```
# change old data

  dat[, nonstatemk.end := ifelse(sftgcode == "THI" & year == 2016, 1, nonstatemk.end)]
  dat[, nonstatemk.ongoing :=
        ifelse(sftgcode == "THI" & year > 2016, 0, nonstatemk.ongoing)]

# combined onset variables

  dat[, anymk.start := ifelse((!is.na(mkl.start) & mkl.start == 1)|
                                nonstatemk.start == 1, 1, 0)]

  dat[, anymk.ongoing := ifelse((!is.na(mkl.ongoing) & mkl.ongoing == 1)|
                                nonstatemk.ongoing == 1, 1, 0)]

  dat[, anymk.ever := ifelse((!is.na(mkl.ever) & mkl.ever == 1)|
                                nonstatemk.ever == 1, 1, 0)]
```

```
# save mass killing variables

  mkl_dat <- subset(dat, select = c("country_name", "sftgcode", "year",
                                    "mkl.start", "mkl.end", "mkl.ongoing",
                                    "mkl.ever", "nonstatemk.start",
                                    "nonstatemk.end", "nonstatemk.ongoing",
                                    "nonstatemk.ever"))

  fwrite(mkl_dat, file = "mkl_data.csv", row.names = F)
```

Leads and Lags

In this section, I update the variables that involve leads and lags

MK lead variables

```
# read in list of the first time Sftgcodes are used in PITF data

  map <- fread("../../../EWP (1)/2019SRA/Make data/sftg_name_map.csv")
  setnames(map, "V1", "min_ewp_year")

  # aggregate over different country names and drop duplicates

    map <- map[country != ""]
    map[, min_ewp_year := min(min_ewp_year), by = "sftgcode"]
    map[, ':=' (country = NULL, country_name = NULL)]
    map <- map[!duplicated(map)]


# merge in first year a sftgcode was used in the PITF data

  dat <- merge(dat, map, by = "sftgcode", all.x = T)

# order by year, country name, and sftgcode

  setkey(dat, year, country_name, sftgcode)

# create anymk lead variable, shifting by sftgcode

  dat[, anymk.start.1 := shift(anymk.start, 1, type = "lead"), by = "sftgcode"]

  # For new states we don't attribute mass killings to their origin country.
  # if the year is the year before the PITF starts first used an sftgcode,
  # then reset to 0

    dat[, anymk.start.1 := ifelse(year == min_ewp_year - 1, 0, anymk.start.1)]


#  create the rest of the lead variables based off of the one-year lead

  dat[, anymk.start.2 := shift(anymk.start.1, type = "lead"), by = "sftgcode"]
  dat[, anymk.start.3 := shift(anymk.start.1, n= 2, type = "lead"), by = "sftgcode"]

  dat[, anymk.start.2window := as.double((anymk.start.1 + anymk.start.2) > 0) ]
  dat[, anymk.start.3window := as.double((anymk.start.2window + anymk.start.3) > 0)]
```

Coup in the last 5 years

Create dataframe of sftgcodes that are created as a result of a coup. We then turn the coup.try.5yr indicator to 1 for the first four years of that sftgcode's existence.

```
# this is a table of sftgcodes created by a coup

  new_sftg <- data.table("sftgcode" = c("GAB", "ETI", "DJI"),
                         "coup.create.yr" = c(1964, 1989, 2000))
```

```
# merge this into main

  dat <- merge(dat, new_sftg, by = "sftgcode", all = T)

# create a variable for the last year when an sftgcode had a coup

  dat[, last_coup_yr := ifelse(cou.any == 1, year, NA)]
  dat[, last_coup_yr := na.locf(last_coup_yr, fromLast = F, na.rm = F),
      by = "sftgcode"]
  dat[, last_coup_yr := ifelse(is.na(last_coup_yr), -Inf, last_coup_yr)]

# for sftgcodes that were initiated due to a coup,
# replace -Inf with the year of the coup

  dat[, last_coup_yr := ifelse(!is.na(coup.create.yr) &
                                 coup.create.yr > last_coup_yr,
                               coup.create.yr, last_coup_yr)]

# create variable for whether there was a coup in the last 5 years

  dat[, coup.try.5yr := as.double((year - last_coup_yr) <= 4)]
```

Data Preparation

Remake some variables

This section fills in some V-DEM variables by creating an adjusted version of the relevant WDI variable. This significantly cuts down on missingness.

```
make.combined <- function(vdem.var, wdi.proxy){
  # Make an adjusted version of the wdi one to fit the tradeshare (VDEM) one:
    # drop update year as we've carried forward some values to this year

    lm.adjust <- lm(as.formula(paste0(vdem.var, "~", wdi.proxy)),
                    data = dat[year < update_years])

  # fill in update years with fitted values from regression on all years

    dat[year %in% update_years, lm.adjust := coef(lm.adjust)[1] +
          coef(lm.adjust)[2]*get(wdi.proxy)]

  # Where vdem var is missing, replace with the adjusted wdi proxy
    combined.var <- paste0(vdem.var, ".combined")
    dat[year %in% update_years,
        (combined.var) :=
          ifelse(!is.na(get(vdem.var)), get(vdem.var), get(wdi.proxy))]

  # summarize differnce in missingness
    na.old <- sum(is.na(subset(dat[year %in% update_years], select = vdem.var)))
    na.new <- sum(is.na(subset(dat[year %in% update_years], select = combined.var)))
```

```
    cat(paste0("In the update years we were missing ", na.old,
           " observations for ", vdem.var))
    cat(paste0("\nUsing the WD indicator, ",  wdi.proxy,
               ", as a proxy we are missing ",  na.new, " observations"))

    dat[, lm.adjust := NULL]
}
```

```
# create combined variable for tradeshare

  make.combined(vdem.var = "tradeshare.ln", wdi.proxy = "wdi.trade.ln.new")
```

```
## In the update years we were missing 163 observations for tradeshare.ln
## Using the WD indicator, wdi.trade.ln.new, as a proxy we are missing 13 observations
```

```
# create combined variable for popsize

  make.combined(vdem.var = "popsize", wdi.proxy = "wdi.popsize.new")
```

```
## In the update years we were missing 163 observations for popsize
## Using the WD indicator, wdi.popsize.new, as a proxy we are missing 4 observations
```

```
  # update logged version

    dat[year %in% update_years, popsize.ln.combined := log(popsize.combined)]

# create combined variable for GDP per capita growth

  make.combined(vdem.var = "gdppcgrowth", wdi.proxy = "wdi.gdppcgrow.new")
```

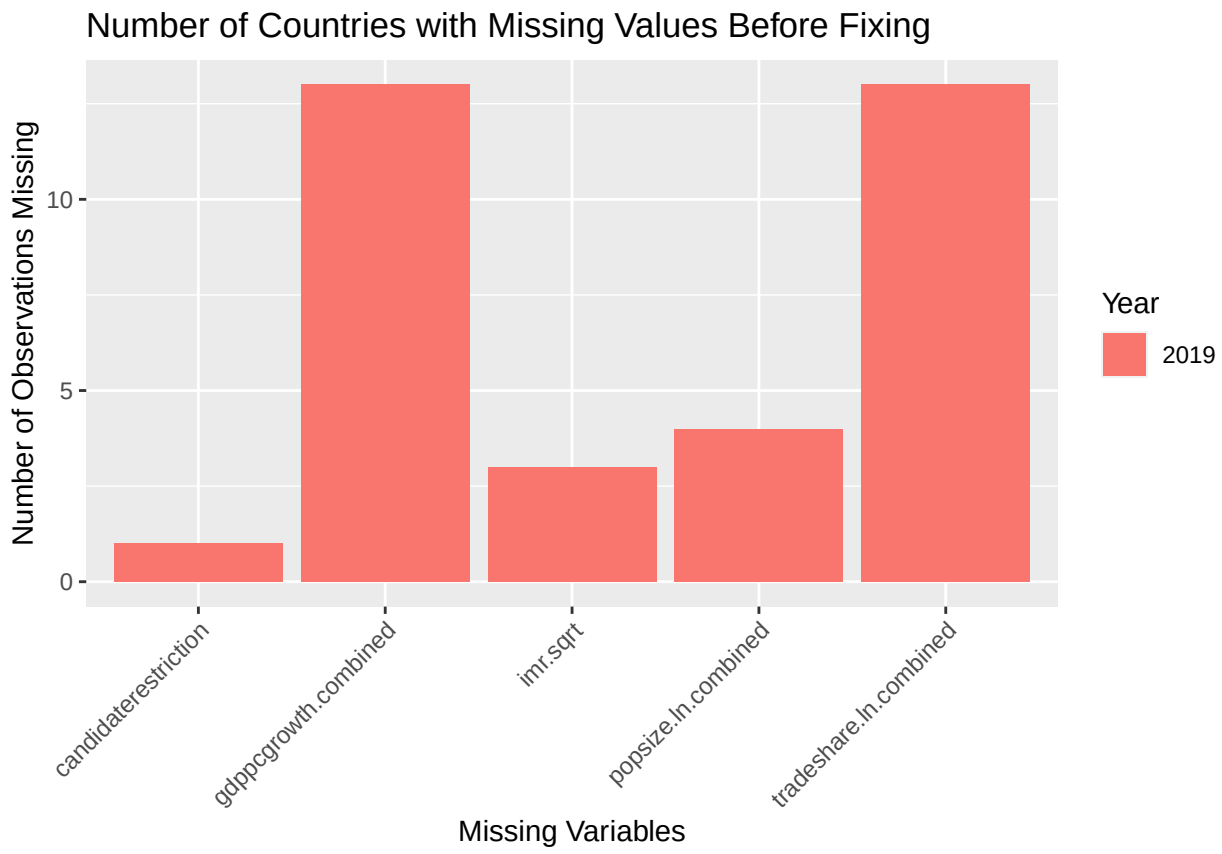
```
## In the update years we were missing 163 observations for gdppcgrowth
## Using the WD indicator, wdi.gdppcgrow.new, as a proxy we are missing 13 observations
```

Missingness

This section looks at remaining missingness and attempts to fill in values.

```
# select variables we would like to check missingness for

  predictornames <- c("anymk.ongoing","anymk.ever",
                      "reg.afr", "reg.eap", "reg.eur", "reg.mna", "reg.sca",
                      "countryage.ln", "popsize.ln.combined", "imr.sqrt",
                      "gdppcgrowth.combined", "ios.iccpr1","includesnonstate",
                      "durable.ln","minorityrule", "elf.ethnic",
                      "battledeaths.ln", "candidaterestriction", "partyban",
                      "judicialreform", "religiousfreedom",
                      "pol_killing_approved", "freemove_men4",
                      "freemove_women4", "freediscussion",
                      "social_inequality","even_civilrights","repress_civilsoc",
                      "social_power_dist", "ses_power_dist",
```

```
                      "tradeshare.ln.combined",
                      "coup.try.5yr")

# look for missingness in the update years for select variables

  dat.check <- subset(dat, year %in% update_years,
                      select = c("country_name","year", predictornames))

# for each of the update years, count the number of NAs for each variable
# select variables with positive NA counts

  comp <- lapply(update_years,
                 function(y) apply(dat.check[year == y],
                                   2, function(x) sum(is.na(x))))
  na.count <- unlist(comp)
  years <- rep(update_years, each = ncol(dat.check))
  check <- data.table(na.count, years, "var"=names(na.count))
  check <- check[na.count > 0]

# visualize missingness

  (missing1 <- ggplot(check) +
      geom_bar(aes(y = na.count, x = var, fill = factor(years)),
               stat = "identity")+
      theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
      labs(x = "Missing Variables", y = "Number of Observations Missing",
           fill = "Year",
           title = "Number of Countries with Missing Values Before Fixing"))
```



Fix missingness

This chunk defines functions to look at patterns in NAs and carry forward values.

```
# function to look at patterns in NAs over time for each country with missing values

  look_na <- function(variable, data = dat){
    missing <- data[is.na(get(variable)) &
                      year %in% update_years, c("year", "country_name")]
    countries <- unique(missing$country_name)
    if(length(countries) == 0){
      look <- "No missing values"
    }else{
      look <-lapply(1:length(countries), function(x){
        out <- subset(data, country_name == countries[x],
                      select = c("year", variable))
        colnames(out) <- c("year", countries[x])
        out})

      # merge all data.tables in the list by year
        look <- Reduce(function(...) merge(..., all = T, by = "year"), look)
        colnames(look) <- c("year", as.character(countries))
        look <- look[order(look$year, decreasing = TRUE), ]
      }
    look
```

```
  }


# create blank spreadsheets to fill out

  fill.tab <- function(var){
    out <- subset(dat[is.na(get(var)) & year %in% update_years],
                  select = c("country_name", var))
    out[, cia.factbook.est := NA]
    out[, est.year := NA]
    fwrite(out, file = paste0("input/temp/missing/", var, ".csv"))
  }


# look at last present value for countries  missing data in update years

  last.present <- function(var){
    look.na.long <- melt(look.na.vars[[var]], id.vars = "year",
                         variable.name = "country_name")
    out <- look.na.long[!is.na(value), .("carried_value" = value[which.max(year)],
                                  "carry_from" = max(year)), by = "country_name"]
    out
  }
```

```
# for each of the variables that we have countries with missing values
# look at the values in the last few years for that country

  look.na.vars <- lapply(c(check$var, "wdi.gdppc.new"), look_na)
  names(look.na.vars) <- c(check$var, "wdi.gdppc.new")


  if(first.pass == T){
    lapply(check$var, fill.tab)
    fill.tab("wdi.gdppc.new")
  }
```

Candidate Restriction

```
# candidate restriction

  head(look.na.vars$candidaterestriction)
```

```
##    year Jamaica
## 1: 2019      NA
## 2: 2018       1
## 3: 2017       1
## 4: 2016       1
## 5: 2015       1
## 6: 2014       1
```

```
# just carry forward this value

  carry_forward("candidaterestriction")
```

```
## Missing values for 1 countries.
## Filling in values for 1 countries.
```

For the variables addressed below, I read in values from the CIA Factbook, and carry those values forward if they're more recent than the last present variable, with several exceptions. Infant mortality rate and population size variable show no change despite the estimates being from more recent years. Recall that I carried forward 2018 infant mortality rates from the WDI already.

GDP per capita growth and tradeshare are carried forward from the CIA Factbook if the estimate is from a year after 2015, as we had been carrying forward these values from 2015 previously.

In the tables presented for each variable, I show the value that is filled in for that country in column 2. This is either the "carried_value", carried from the year listed under "carry_from", or it is a transformed version of the value under "cia.factbook.est".

Infant Mortality

```
  # head(look.na.vars$imr.sqrt)

# read in CIA Factbook estimates

  cia.factbook <- fread("input/temp/missing/imr.sqrt.csv")

  comp <- merge(cia.factbook, last.present("imr.sqrt"),
                by = "country_name")
  comp[, imr.sqrt := ifelse(carry_from > est.year,
                                       carried_value, sqrt(cia.factbook.est))]
  kable(comp)
```

country_name	imr.sqrt	cia.factbook.est	est.year	carried_value	carry_from
Macedonia	2.720294	7.4	2020	2.792848	2018
North Korea	4.472136	20.0	2020	4.690416	2018

```
# replace missing

  for(i in 1:nrow(comp)){
    country.fill <- comp[i, ]
    dat[country_name ==  country.fill$country_name & year == update_years,
        imr.sqrt := country.fill$imr.sqrt]
  }
```

Population size

```
  # head(look.na.vars$popsize.ln.combined)


# read in CIA Factbook estimates

  cia.factbook <- fread("input/temp/missing/popsize.ln.combined.csv")

  comp <- merge(cia.factbook, last.present("popsize.ln.combined"),
                by = "country_name")
  comp[, popsize.ln.combined := ifelse(carry_from > est.year,
                                       carried_value, log(cia.factbook.est))]
  comp[, popsize.combined := ifelse(carry_from > est.year,
                                       carried_value, cia.factbook.est)]
  kable(comp)
```

country_name	popsize.ln.combined	cia.factbook.est	est.year	carried_value	carry_from	popsize.combined
Eritrea	15.62071	6081196	2020	15.60237	2018	6081196
Macedonia	14.56974	2125971	2020	14.64728	2018	2125971
North Korea	17.05980	25643466	2020	17.04908	2018	25643466

```
# replace missing

  for(i in 1:nrow(comp)){
    country.fill <- comp[i, ]
    dat[country_name ==  country.fill$country_name & year == update_years,
        popsize.ln.combined := country.fill$popsize.ln.combined]
  }

  for(i in 1:nrow(comp)){
    country.fill <- comp[i, ]
    dat[country_name ==  country.fill$country_name & year == update_years,
        popsize.combined := country.fill$popsize.combined]
  }
```

GDP Per capita growth

```
# read in CIA Factbook estimates

  cia.factbook <- fread("input/temp/missing/gdppcgrowth.combined.csv")
  comp <- merge(cia.factbook, last.present("gdppcgrowth.combined"),
                by = "country_name")
  # these values were carried to 2018 from 2016 or 2017 so let's overwrite them
  comp[, gdppcgrowth.combined := ifelse(est.year < 2016,
                                       carried_value, cia.factbook.est/100)]
  kable(comp, digits = 3)
```

country_name	gdppcgrowth.combined	cia.factbook.est	est.year	carried_value	carry_from
Bhutan	0.074	7.4	2017	0.012	2018

country_name	gdppcgrowth.combined	cia.factbook.est	est.year	carried_value	carry_from
Cuba	0.016	1.6	2017	-0.009	2018
Eritrea	0.050	5.0	2017	0.000	2018
Iran	0.037	3.7	2017	0.029	2018
Macedonia	0.000	0.0	2017	-0.007	2018
North Korea	-0.017	-1.1	2015	-0.017	2018
Somalia	0.023	2.3	2017	0.024	2018
South Sudan	-0.052	-5.2	2017	-0.006	2018
Syria	-0.138	-36.5	2014	-0.138	2018
Taiwan	0.029	2.9	2017	0.012	2016
Turkmenistan	0.065	6.5	2017	0.044	2018
Venezuela	-0.140	-14.0	2017	-0.133	2018
Yemen	-0.059	-5.9	2017	-0.029	2018

```
# replace missing

  for(i in 1:nrow(comp)){
    country.fill <- comp[i, ]
    dat[country_name ==  country.fill$country_name & year == update_years,
        gdppcgrowth.combined := country.fill$gdppcgrowth.combined]
  }
```

tradeshare.ln.combined

First have to fill in GDP per capita

```
  cia.factbook <- fread("input/temp/missing/wdi.gdppc.new.csv")
  comp <- merge(cia.factbook, last.present("wdi.gdppc.new"),
                by = "country_name", all.x = T)
  comp[, wdi.gdppc.new := ifelse((est.year < carry_from & !is.na(carry_from))|
                                   is.na(cia.factbook.est), carried_value,
                                 cia.factbook.est)]
  kable(comp, digits = 3)
```

country_name	wdi.gdppc.new	cia.factbook.est	est.year	carried_value	carry_from
Bhutan	9000	9000	2017	NA	NA
Cuba	12300	12300	2016	NA	NA
Djibouti	3600	3600	2017	1325.991	2010
Eritrea	1600	1600	2017	514.180	2011
Iran	20100	20100	2017	NA	NA
Macedonia	14900	14900	2017	5245.360	2017
North Korea	1700	1700	2015	NA	NA
Somalia	NA	NA	2017	NA	NA
South Sudan	1600	1600	2017	NA	NA
Syria	2900	2900	2015	NA	NA
Taiwan	50500	50500	2017	NA	NA
Turkmenistan	18200	18200	2017	NA	NA
Venezuela	12500	12500	2017	13709.043	2014
Yemen	2500	2500	2017	NA	NA

```
  # replace missing

  for(i in 1:nrow(comp)){
    country.fill <- comp[i, ]
    dat[country_name ==  country.fill$country_name & year == update_years,
        wdi.gdppc.new := country.fill$wdi.gdppc.new]
  }
```

```
# read in CIA Factbook estimates

  cia.factbook <- fread("input/temp/missing/tradeshare.ln.combined.csv")
  comp <- merge(cia.factbook, last.present("tradeshare.ln.combined"),
                by = "country_name")
  other.vars <- dat[year == update_years,
                    c("wdi.gdppc.new", "popsize.combined", "country_name")]
  comp <- merge(comp, other.vars, by= "country_name", all.x = T)

  # these values were carried to 2018 from 2016 or 2017 so let's overwrite them
  # as long as factbook estimates are at least from 2016

  # The VDEM imports and exports are in millions of USDs, then x10^6.
  # tradeshare = (imports + exports (in trillions))/(gdppcgrowth + popsize)
  # The CIA factbook data is in billions, so needs to be multiplied by 1000.

    comp[, tradeshare.ln.combined :=
           ifelse(est.year < 2016, carried_value,
                  log(1000*cia.factbook.est/((wdi.gdppc.new/1e+6)*popsize.combined)))]
    kable(subset(comp, select = c("country_name", "tradeshare.ln.combined",
                                  "cia.factbook.est", "est.year",
                                  "carried_value", "carry_from")))
```

country_name	tradeshare.ln.combined	cia.factbook.est	est.year	carried_value	carry_from
Eritrea	-1.7148466	1.7513	2017	-1.2330414	2018
Fiji	-0.4178544	2.8192	2017	-1.1186755	2018
Macedonia	-1.0369120	11.2310	2017	-1.0162767	2018
North Korea	-2.8419659	2.5420	2018	-1.4241446	2018
Papua New Guinea	-0.7487609	10.3980	2017	-0.6482966	2018
Solomon Islands	-0.0658079	0.9307	2017	-0.4159678	2018
Somalia	NA	95.2490	2018	-1.4535997	2018
South Sudan	-1.2792054	4.9250	2016	-1.5706660	2018
Syria	-1.8066033	8.1290	2017	-1.2272775	2018
Trinidad and Tobago	-0.2733220	16.0320	2017	-0.0214098	2018
Venezuela	-2.1135935	43.0600	2017	-1.5136212	2018
Yemen	-2.7932213	4.4635	2017	-2.8031193	2018

```
# replace missing

  for(i in 1:nrow(comp)){
    country.fill <- comp[i, ]
    dat[country_name ==  country.fill$country_name & year == update_years,
        tradeshare.ln.combined := country.fill$tradeshare.ln.combined]
```

```
  }

# carry forward for Somalia
  carry_forward("tradeshare.ln.combined")
```

```
## Missing values for 2 countries.
## Filling in values for 1 countries.
```

Check that all is filled in

```
# repeat to make sure there are no more missing

  look.na.vars <- lapply(check$var, function(x)
    look_na(x, dat = dat[country_name!="Taiwan"]))
  names(look.na.vars) <- check$var
  look.na.vars
```

```
## $popsize.ln.combined
## [1] "No missing values"
##
## $imr.sqrt
## [1] "No missing values"
##
## $gdppcgrowth.combined
## [1] "No missing values"
##
## $candidaterestriction
## [1] "No missing values"
##
## $tradeshare.ln.combined
## [1] "No missing values"
```

```
fwrite(dat, file = paste0("output/prepared2019predictors-", Sys.Date(), ".csv"))
```

Checking the append

```
# function to check all.equal for each var

  compare_var <- function(var){
    vars <- paste(var, c("x", "y"), sep = ".")

    check <- subset(compare, select = c("year", "sftgcode", vars[1], vars[2]) )
    colnames(check) <- c("year", "sftgcode", "var.x", "var.y")

    all.equal(check$var.x, check$var.y)
  }
```

```
# look into differences where all.equal == F for each var

  check_diffs <- function(var){
    vars <- paste(var, c("x", "y"), sep = ".")

    check <- subset(compare, select = c("year", "sftgcode", vars[1], vars[2]) )

    check[get(vars[1]) != get(vars[2])|
            is.na(get(vars[1])) & !is.na(get(vars[2]))|
            is.na(get(vars[2])) & !is.na(get(vars[1]))]
  }
```

```
# read in newly updated data and just focus on the last year in old

  dat <- fread("output/prepared2019predictors-2020-08-10.csv")
  new_dat <- dat[year <= 2018]

# read in original base data

  load("input/prepared2018predictors_2_11_20.RData")
  old_dat <- dat
  rm(dat)

# check row count

  if(nrow(old_dat) != nrow(new_dat)){stop("Different observation count")}

# merge new and old data to compare

  compare <- merge(new_dat, old_dat, by = c("sftgcode", "year"))

# make sure they match on all variables in the old data

  check.vars <- colnames(old_dat)[!colnames(old_dat) %in% c("sftgcode", "year")]

  comp <- data.table("all.equal" = sapply(check.vars, compare_var),
                     "var" = check.vars)

# look into the differences where all.equal == F
# and where the issue isn't an NA value mismatch -- come back to this later

  differences <- comp[all.equal!="TRUE"][!grep("is.NA", all.equal)]
  diff.look <- lapply(differences$var, check_diffs)
  names(diff.look) <- differences$var

  cat(paste0("There are ", length(diff.look),
             " variables where the two datasets disagree on non-missing values."))
```

```
## There are 3 variables where the two datasets disagree on non-missing values.
```

```
  for(i in 1:length(diff.look)){
    print(diff.look[[i]])
  }
```

```
##    year sftgcode nonstatemk.ongoing.x nonstatemk.ongoing.y
## 1: 2017      THI                    0                    1
## 2: 2018      THI                    0                    1
##    year sftgcode nonstatemk.end.x nonstatemk.end.y
## 1: 2016      THI                1                0
##    year sftgcode anymk.ongoing.x anymk.ongoing.y
## 1: 2017      THI               0               1
## 2: 2018      THI               0               1
```

```
# now let's look at differences in missing values

  na.diffs <- comp[all.equal!="TRUE"][grep("is.NA", all.equal)]
  na.diff.look <- lapply(na.diffs$var, check_diffs)
  names(na.diff.look) <- na.diffs$var

  # for each variable, count the instances of differences in NAs by year
  count.by.yr <- rbindlist(lapply(na.diff.look,
                                  function(x) x[, .N, by = "year"]))
  count.by.yr[, var := na.diffs$var]
  # We can see that the only differences are the filling of NAs
  # for the appropriate lead variables
  count.by.yr
```

```
##    year   N                 var
## 1: 2018 163       anymk.start.1
## 2: 2017 163 anymk.start.2window
## 3: 2016 163 anymk.start.3window
## 4: 2017 163       anymk.start.2
## 5: 2016 163       anymk.start.3
```