Georgia CCDR Microsimulation

Renato Vargas

## 1 Introduction

In this calculation file, we “age” the Georgian household survey according to demographic projections and different macroeconomic scenarios to explore the impact of climate-related risks and policy measures on the consumption expenditure distribution. It is part of a larger project with all background contributions to Georgia’s CCDR, [available in this repository](https://rvconsulting.github.io/geo-ccdr24/supporting-materials/geo-microsimulation.html).

# Georgia administrative level 1 shapefile  
adm1 <- sf::read\_sf("data/gis/geo-adm1.shp") |>   
 dplyr::select(ADM1\_PCODE, ADM1\_EN, ADM1\_KA, geometry) |>   
 dplyr::arrange(ADM1\_PCODE)  
  
tmap::tm\_shape(adm1)+  
 tmap::tm\_fill("ADM1\_EN", legend.show = FALSE, palette = "Set1") +  
 tmap::tm\_text("ADM1\_EN", size = 0.65, auto.placement = T, col = "black")+  
 tmap::tm\_layout(frame = F)

|  |
| --- |
| Figure 1: Map of Georgia at administrative level 1 |

As a convention, code is presented in the following format in this guide:

# Some comment that is not evaluated by R  
some\_variable <- some\_function(some\_object, some\_parameter = TRUE)

We assume that the reader has created an Rstudio project and is familiar with basic R functions. Within that project we recommend the following file structure:

#| eval: false  
root/  
├── supporting-materials  
│ ├── my\_script.R  
| └── my\_script.qmd  
| └── my\_script.do  
├── data/  
| ├── my\_data.sav  
| ├── my\_data.dta  
| └── my\_data.csv  
└── output  
 ├── my\_output1.csv  
 └── my\_output2.xlsx

Using RStudio project makes it possible to not use setwd() to establish the root directory and refer to subdirectories in a relative manner, making interoperability easier within teams and not hard coding a particular computer’s file structure into the code. If you are not using RStudio, just add setwd(r'(C:\My\path\to\project\root)') at the beginning of your coding session.

## 2 Preamble

We start with a clean environment, making sure that any objects from a previous session are not present. We take this opportunity to keep our country ISO code in a variable iso in case we need it later.

# Clean workspace  
rm(list = ls())  
  
# Georgia country ISO code  
iso <- "GEO"  
  
# Survey year  
survey\_year <- 2023  
  
# Exchange rate USD per GEL  
er <- 0.37  
  
# Years of interest for our macroeconomic scenario analysis  
analysis\_years <- c(2030, 2050)

We call the appropriate libraries.

Rather than calling our libraries as we go, we will make sure we have everything we need from the beginning.

library(tidyverse) # includes dplyr, ggplot2, purr...  
library(haven) # to read SPSS and Stata datasets  
library(readxl) # to read from MS-Excel  
library(openxlsx) # to write to MS-Excel.  
library(gt) # pretty tables  
library(car) # companion to applied regression  
library(modelr) # regression models  
#library(anesrake)   
# Raking reweighting but we don't load it, because   
# it changes the meaning of summarize from dplyr,   
# so we use the form anesrake::anesrake() when using it.  
#library(ebal) # Entropy reweighting (not used)  
library(janitor) # pretty subtotals  
library(broom) # More regressions  
library(zoo) # Calculate moving window average and max value  
# library(ineq) # Inequality measures  
# library(acid)  
  
# Geopackages  
library(sf) # to read and write shapefile maps  
library(terra) # to perform geocalculations  
library(tmap) # for static and interactive maps

## 3 Datasets

We then load the datasets that we need for this study. These are based on Georgia’s Integrated Living Conditions Survey 2022 (GEOSTAT, 2023). We take this oportunity to standardize the household identification variable to household\_id.

# Household size (includes no. of family members and weights)  
hh\_size <- read\_sav(  
 "data/ilcs\_2023/familysize.sav") %>%   
 rename(household\_id = UID)  
  
# Processed income at household level  
hh\_income <- read\_sav(  
 "data/ilcs\_2023/tblincomes.sav") %>%   
 rename(household\_id = UID)  
  
# Consumption aggregate at household level   
hh\_expenditure <- read\_sav(  
 "data/ilcs\_2023/tblexpenditures.sav") %>%   
 rename(household\_id = UID,  
 # rename total expenditure variables  
 total\_expenditure = MTlianixarjebi\_,  
 total\_expenditure\_aeq06 = MTlianimoxmareba\_EqAdScale,  
 total\_expenditure\_aeq08 = Mtlianimoxmareba\_EqAdScale\_08)  
  
# Characteristics of the dwelling  
hh\_chars <- read\_sav(  
 "data/ilcs\_2023/tblshinda01.sav") %>%   
 rename(household\_id = UID)  
  
# Household location  
hh\_location <- read\_sav(  
 "data/ilcs\_2023/sysschedule.sav") %>%   
 rename(household\_id = UID)  
  
# Persons (pp)  
pp <- read\_sav(  
 "data/ilcs\_2023/tblshinda02.sav") %>%   
 rename(household\_id = UID)  
  
# Food diary  
food\_q <- read\_sav(  
 "data/ilcs\_2023/tblconsumption.sav") %>%   
 rename(household\_id = UID)  
  
food\_price <- read\_sav(   
 "data/ilcs\_2023/tblavgprices.sav")

We also have Continuous Labor Survey data at the individual level, which will come in handy if we do not get access to the labor part of the ILCS. See data folder for documents describing the datasets.

# Labor Force Survey  
lfs\_2023 <- read\_sav(  
 "data/lfs\_2023/LFS\_ECSTAT\_ENG\_2023.sav") %>%   
 rename(household\_id = UID)  
  
# Labor Force Survey Demographic Characteristics  
lfs\_2023\_dem <- read\_sav(  
 "data/lfs\_2023/LFS\_Demographic\_ENG\_2023.sav") %>%   
 rename(household\_id = UID)

We will work non-destructively, meaning we will not rewrite these data sets and we will only create intermediate data frame objects from them to perform transformations, selections and other data management tasks. For example, we will keep household assignment to poverty status and consumption deciles handy by creating a subset of our hh\_expenditure data with only our household identifiers, deciles, and poverty if available.

# We will estimate deciles from consumption  
deciles <- hh\_expenditure %>%   
 select(   
 # Keep household id and expenditure variables  
 household\_id,   
 total\_expenditure,  
 total\_expenditure\_aeq06, # Adult equivalent \* 0.6  
 total\_expenditure\_aeq08) # Adult equivalent \* 0.8

Our population data comes from UN’s projections.

population\_projections <- read\_dta("data/population/UN2022\_population.dta") %>%   
 filter(country == iso) # we filter for Georgia

The macro scenario dataset is an input provided by the Macroeconomic CGE simulation team, with yearly information on GDP, working age population, employment by economic activity (for an aggregation of three sectors: agriculture, manufacturing, and services), wages by economic activity, value added by economic activity, remittances, consumer price index, food price index and energy price index (for a bundle of gas, oil, coal, electricity) by decile (10 representative households in the macro model), and carbon tax revenue transfers to household deciles.

scenario\_file <- "data/ARM-Microsimulation/GEO\_MacroScenarioInformation.xlsx"  
# scenario\_varlist <- read\_xlsx(  
# "data/ARM-Microsimulation/ARM\_Macro\_varlist.xlsx")  
# prices\_2030 <-   
# read.csv("data/ARM-Microsimulation/prices2030.csv")

Economic Activities in the Survey is in Georgian. The following dataset is a lookup table with the English names.

# Pending  
sectors <- read\_xlsx("data/ARM-HH-survey/economic\_activity\_codes.xlsx")

We also have geographical information for level 1 in Shapefile format, which we import with the sf package. We rename the column with the name of the administrative region to match our household survey data set conventions to ease mergers. The dplyr package from the tidyverse meta package allows us to “pipe” or link processing steps using the %>% pipe. Although there is no geoprocessing in this analysis, this will come in handy for graphical presentations.

# Georgia administrative level 1 shapefile  
adm1 <- sf::read\_sf("data/gis/geo-adm1.shp") |>   
 dplyr::select(ADM1\_PCODE, ADM1\_EN, ADM1\_KA, geometry) |>   
 dplyr::arrange(ADM1\_PCODE)

And we plot it for reference (see [Figure 2](#fig-map-adm1-2)). This is done with the tmap R package and the code shown below.

tmap::tm\_shape(adm1)+  
 tmap::tm\_fill("ADM1\_EN", legend.show = FALSE, palette = "Set1") +  
 tmap::tm\_text("ADM1\_EN", size = 0.65, auto.placement = T, col = "black")+  
 tmap::tm\_layout(frame = F)

|  |
| --- |
| Figure 2: Map of Georgia at administrative level 1 (ADM1) |

## 4 Data preparation income outliers and missings

We start with various renames for standardization. Naming conventions in the guidance code use traditional abbreviations like nli for non-lablor income. We are opting for more descriptive variable names like non\_labor\_income, labor\_income, etc. to have more easily readable code. We make an exception for total consumption (tc), because it’s a variable that we use in every scenario and it supersedes lenght limits when adding scenario identifiers.

# Uncomment the correct total expenditure variable below  
ex <- hh\_expenditure %>%   
 rename(  
 tc =  
 total\_expenditure  
 #total\_expenditure\_aeq06 # Adult equivalent \* 0.6  
 #total\_expenditure\_aeq08 # Adult equivalent \* 0.8  
 )

### 4.1 Demographic characteristics, education, labor force

Here the original code calls for Zone data, which is not present in our dataset, due to the different administrative structure of Georgia. However, we use hh\_01\_code (settlement) for this purpose. In the end, this variable was never used.

Demographic data, merge with zone data Note that ed\_03 (educy) below is not years of education, but education level (primary, general, secondary, etc.) However, it is ordered in a way that higher levels imply more years of education. We perform several steps within the first pipe call. The variable lstatus (Labor Force Status) here is very important for the reweigthing of the dataset later on. Note that from here onwards we will be creating \_microsim versions of our datasets with the transformations needed for calculations. That way we avoid changing our original data and can refer to it later without fearing we’ve left things behind.

pp\_microsim <- pp %>%  
 mutate(  
 # Demographic characteristics  
 # Unique person id  
 person\_id = paste0(household\_id, "-", str\_pad(MemberNo, 2, pad = "0")),  
 head = ifelse(Relations == 1, 1, 0),  
 # Education level  
 educy = ifelse(is.na(Education), 0, Education),  
 # Labor Force Status  
 lstatus = case\_when(  
 # 1. Employed  
 est\_03 == 1 | est\_04 == 1 | est\_05 == 1 |  
 est\_06 == 1 | est\_08 == 1 ~ 1L,  
 # 2. Unemployed (available, and searching)  
 est\_10 == 1 ~ 2L,  
 # 3. Inactive (available, not searching)  
 est\_10 == 2 ~ 3L,  
 # Out of the labor force  
 .default = 4L # Default to OLF  
 ),  
 employed = (lstatus == 1),  
 # Salaried status (1. paid employee; 2 self-employed)  
 salaried = ifelse(  
 !is.na(emp\_11a),  
 1L,  
 ifelse(is.na(emp\_11a) &  
 employed == TRUE, 0L, NA\_integer\_)  
 )  
 ) %>%  
 rename(rel = mem\_03, # relationship to HH head  
 gender = mem\_02,  
 age = mem\_05)

Later, when we conduct the reweighting of the dataset, we need to summarize into three levels of education.

pp\_microsim <- pp\_microsim %>%  
 mutate(calif = case\_when(  
 educy >= 0 & educy <= 4 ~ "None - General",  
 educy > 3 & educy <= 9 ~ "Secondary - Vocational",  
 educy > 7 & educy <= 13 ~ "Higher +",  
 TRUE ~ NA\_character\_ # Values outside the specified ranges  
 ))

Count the number of employed persons by household. Note that it is necessary to explicitly tell R to ignore missing values(NA). This is different from Stata where 1 + .= 1 (where . is “missing”). In R 1 + NA = NA (where NA means “not available”). Not adding na.rm = TRUE to aggregation functions such as sum() in [Listing 1](#lst-employed-hh) below will not throw an error and only provide a column with NA for households where at least one individidual has an employed status of NA.

|  |
| --- |
| Listing 1: Employed in household  # Pending data from pp\_ecstat hh\_labor <- pp\_ecstat %>%   mutate(employed = (Status == 1)) %>%   group\_by(household\_id) %>%   # Count within each household  mutate(employed\_hh = sum(employed, na.rm = TRUE)) %>%   ungroup() |

Here the original Stata code calculates income variables and aggregates them by household. We skip that because the dataset ic already has these elements calculated by the WB poverty team. We’ll add them later as we need them.

However, as we’ll see later labor income information is heavily non-reported in the dataset. Labor income is a crucial step in merging the dataset with macroeconomic information and so we will predict income for those that do not report it below. These variables are related to labor income, amount and frequency, which we have to standardized to a monthly or yearly value.

**Primary and Secondary Job income:**

* **emp\_11** How much was %rostertitle%’s payment for wages/salary/income for last month?
* **emp\_12** What period of time was the wage/income for?
* **emp\_25** How much was %rostertitle%’s payment for wages/salary/income for last month?
* **emp\_26** What period of time was the wage/income for?

Bonus, In-Kind, and food from job was not asked in Georgia, If it were, you should add a mutate() statement like the ones below for each subcategory in [Listing 2](#lst-annualized-labor-income). We use coalesce(colname, 0) when adding the annual\_labor\_total again to prevent sums of NA’s. This function replaces a value with 0 within the calculation if it’s missing, but doesn’t change its value permanently.

|  |
| --- |
| Listing 2: Annualized labor income  pp\_microsim <- pp\_microsim %>%   # Labor income primary job  mutate(annual\_labor\_income\_primary = case\_when(  emp\_12 == 1 ~ emp\_11 \* 365,  emp\_12 == 2 ~ (emp\_11/7) \* 365, # Assuming weekly rate   emp\_12 == 3 ~ (emp\_11/14) \* 365,  emp\_12 == 4 ~ emp\_11 \* 12,  emp\_12 == 5 ~ emp\_11 \* 2,  emp\_12 == 6 ~ emp\_11,  emp\_12 == 7 ~ NA  )) %>%   # Labor income secondary job  mutate(annual\_labor\_income\_secondary = case\_when(  emp\_26 == 1 ~ emp\_25 \* 365,  emp\_26 == 2 ~ (emp\_25/7) \* 365, # Assuming weekly rate   emp\_26 == 3 ~ (emp\_25/14) \* 365,  emp\_26 == 4 ~ emp\_25 \* 12,  emp\_26 == 5 ~ emp\_25 \* 2,  emp\_26 == 6 ~ emp\_25,  emp\_26 == 7 ~ NA  )) %>%   # Annual labor total in thousands of dram  mutate(annual\_labor\_total =   (coalesce(annual\_labor\_income\_primary, 0) +   coalesce(annual\_labor\_income\_secondary, 0))/1000)  # Restore annual\_labor\_total to NA if both NA pp\_microsim <- pp\_microsim %>%   mutate(annual\_labor\_total =  if\_else(  is.na(annual\_labor\_income\_primary)  & is.na(annual\_labor\_income\_secondary),  NA,   annual\_labor\_total)) |

Now we need to check the share of individuals that are employed, but did not report income. This is done in [Listing 3](#lst-employed-no-income) below.

|  |
| --- |
| Listing 3: Employed with no income reported  total\_employed\_no\_income <- pp\_microsim %>%  filter(employed == TRUE & is.na(annual\_labor\_total)) %>%   nrow()  total\_employed <- pp\_microsim %>%  filter(employed == TRUE) %>%  nrow()  percent\_employed\_no\_income <-   (total\_employed\_no\_income / total\_employed) \* 100  print(  paste0(  "There is ",  format(  percent\_employed\_no\_income,digits = 2, nsmall=2  ),  "% of the employed population that reports no income.")  ) |

We also need to mark income outliers as those with incomes outside 5 standard deviations.

pp\_microsim <- pp\_microsim %>%   
 mutate(  
 # Calculate standard deviation  
 sd = sd(annual\_labor\_total, na.rm = TRUE),   
 d = annual\_labor\_total / sd,   
 # Combined outlier condition  
 outlier = (d > 5) | (employed == TRUE & annual\_labor\_total == 0),   
 # Mark potential missings  
 missings = if\_else(employed == TRUE, is.na(annual\_labor\_total), NA)   
 )

Economic sector. The economic sectors dataset contains a lookup table for sector aggregation which we add to the pp\_microsim database in [Listing 4](#lst-sector-aggregation).

|  |
| --- |
| Listing 4: Sector aggregation  pp\_microsim <- pp\_microsim %>%  mutate(emp\_04 = as.integer(emp\_04)) %>%   left\_join(sectors, join\_by("emp\_04" == "economic\_activity\_code") ) %>%   rename(sector = ea\_shortcode) |

Some individuals report no sector for either their primary or secondary job. In [Listing 5](#lst-assign-sector) we find out the sector of other family members in their home and assign the sector of whoever is closest using fill( other\_sector, .direction = "downup").

|  |
| --- |
| Listing 5: Assign sector to those who don’t report one  pp\_microsim <- pp\_microsim %>%  group\_by(household\_id) %>%  mutate(  # Create a temporary variable 'other\_sector'  # which captures the sector of any employed   # individual in the household  other\_sector =   if\_else(employed == TRUE & !is.na(sector), sector, NA\_real\_)  ) %>%  # Use 'fill' to propagate 'other\_sector' values within the household  fill(other\_sector, .direction = "downup") %>%  mutate(  # Impute missing 'sector' values based on the 'other\_sector'  sector =   if\_else(is.na(sector) & employed == TRUE, other\_sector, sector)  ) %>%  # Drop the temporary 'other\_sector' variable  select(-other\_sector) %>%  ungroup() |

We then assign a specific value for missing sectors for those employed with no one else in the hh to assign value. We select services as it’s the heaviest sector in the dataset (we do it like this, instead of say, any matching technique, because it’s only 2 observations).

pp\_microsim <- pp\_microsim %>%  
 mutate(sector = if\_else(is.na(sector) & employed == TRUE, 3, sector))

We provide value labels for sector factors.

pp\_microsim <- pp\_microsim %>%  
 mutate(sector\_name = factor(sector, levels = c(1, 2, 3),  
 labels = c("Agriculture",   
 "Manufacturing",   
 "Services")))

We make sure that those outside the labor force (OLF) do not report a sector, which we replace with NA for those who meet the condition.

|  |
| --- |
| Listing 6: No sector for OLF  pp\_microsim <- pp\_microsim %>%  mutate(lstatus = as.numeric(lstatus),  sector =   if\_else(lstatus == 4,   as.character(NA),   as.character(sector)),  industry = as.factor(sector)) %>%  # We need this for reweighting and   # not messing up the regression below.  mutate(sector\_w = sector) |

### 4.2 The regression

Since labor income was a key variable, which we needed to match with the future wage bill by economic activity, we first checked for missing values among employed individuals. We found that almost a third of respondents (28.6%) did not report income for either their primary or secondary job. To overcome this limitation, we used the available information from the remaining respondents to estimate an extended Mincer equation, as shown in [Equation 1](#eq-labor-income-regression), and implemented in [Listing 7](#lst-regression-model). For the respondents with available information, we also identified outliers as those outside of five standard deviations from the mean labor income.

$$
\ln(lab\_i) = \beta\_0 + \beta\_1 \text{age}\_i + \\
\beta\_2 \text{gender}\_i + \beta\_3 \text{educy}\_i + \\
\beta\_4 \text{age}^2\_i + \beta\_5 \text{marz}\_i + \\
\beta\_6 \text{sector}\_i + \epsilon\_i
\qquad(1)$$

Where:

* is the natural logarithm of labor income for individual .
* is the intercept term.
* are the coefficients for the respective independent variables.
* is the age of individual .
* is a binary variable indicating the gender of individual (1 for male, 2 for female).
* represents the level of education for individual (ordered: 1) None to General, 2) Secondary to Vocational, 3) Higher education).
* is the square of the age of individual , included to capture non-linear effects of age on labor income.
* represents the region where individual resides.
* represents the sector of employment for individual (i.e., agriculture, manufacturing or services).
* is the error term for individual .

We first prepare our variables for the regression.

pp\_microsim <- pp\_microsim %>%  
 mutate(  
 educy2 = educy^2,  
 age2 = age^2,  
 male = case\_when(  
 gender == 1 ~ 1,  
 gender == 2 ~ 0  
 ),  
 lnlab = log(annual\_labor\_total),  
 simuli = NA\_real\_ # Initialize simuli  
 )

Filter the data for regression conditions.

regression\_data <- pp\_microsim %>%  
 filter(employed == TRUE & outlier == FALSE & missings == FALSE)

Regression model.

|  |
| --- |
| Listing 7: Income regression model  model <- lm(lnlab ~ age + gender + educy + age2 + marz + sector,   data = regression\_data) |

Predict for specific conditions

pp\_microsim <- pp\_microsim %>%  
 mutate(  
 condition = (lstatus == 1 & (outlier == TRUE | missings == TRUE))  
 )

Applying predictions.

Note: The ‘predict’ function in R does not directly support conditions within the function call, so we handle this by filtering or subsetting the data as needed.

temp2 equivalent - Note: ‘type = “response”’ might be needed depending on model type.

pp\_microsim$simuli[pp\_microsim$condition==TRUE] <- exp(  
 predict(model, pp\_microsim[pp\_microsim$condition==TRUE, ], type = "response"))

Handling negative values in ‘simuli’.

pp\_microsim <- pp\_microsim %>%  
 mutate(  
 simuli = if\_else(simuli < 0, 0, simuli)  
 )

There were 8 observations that met the criteria:

We will replace annual\_labor\_total with this value for those observations.

pp\_microsim <- pp\_microsim %>%  
 mutate(annual\_labor\_total = if\_else(  
 employed == TRUE & (outlier == TRUE | missings == TRUE),  
 simuli, annual\_labor\_total))  
  
# And get monthly incomes for everyone  
pp\_microsim <- pp\_microsim %>%   
 mutate(monthly\_labor\_income = annual\_labor\_total / 12)

Merging datasets.

pp\_microsim <- pp\_microsim %>%  
 left\_join(poverty\_designations, by = "household\_id")

### 4.3 Total income and shares

Total labor income at HH level.

pp\_microsim <- pp\_microsim %>%  
 group\_by(household\_id) %>%  
 mutate(lab\_hh = sum(annual\_labor\_total, na.rm = TRUE)) %>%  
 ungroup()

Monthly incomes come from the ic data set.

incomes <- ic %>%   
 select(household\_id, inc1, inc2, inc3, inc4, inc5, inc6, inc7, inc8)

Total income at HH level (the commented out portion was a less efficient way of accomplishing the same result of coalescing NAs to 0 so that the sum can be performed). Note that here we need to use the magittr pipe %>% instead of the newer Native Pipe %>% , because we need to reference the correct scope with the dot ..

pp\_microsim <- pp\_microsim %>%  
 left\_join(incomes, by = c("household\_id" = "household\_id")) %>%  
 mutate(across(inc5:inc8, ~replace\_na(., 0))) %>%  
 mutate(nli\_hh = 12 \* rowSums(select(., inc5:inc8), na.rm = TRUE)) %>%  
 mutate(income\_hh = lab\_hh + nli\_hh)  
  
# pp\_microsim <- pp\_microsim %>%  
# left\_join(incomes, join\_by(household\_id == household\_id)) %>%   
# mutate(nli\_hh = ( coalesce(inc5) +   
# coalesce(inc6) +  
# coalesce(inc7) +  
# coalesce(inc8)) \* 12) %>%   
# mutate(income\_hh = lab\_hh + nli\_hh)

Final subset of data.

pp\_microsim <- pp\_microsim %>%  
 select(household\_id, person\_id, industry, salaried,  
 rural\_dummy, hhsize,hhsize\_R, marz\_no, aepc, weight,   
 Foodpovln2022, Lpovln2022, Upovln2022, Avpovln2022,   
 poor\_Foodpovln2022, poor\_Lpovln2022, poor\_Upovln2022,   
 poor\_Avpovln2022, decile, settlement, urban\_rural,  
 gender, age, head, rel, educy, calif, sector, sector\_name,  
 annual\_labor\_total,annual\_labor\_income\_primary,  
 annual\_labor\_income\_secondary,monthly\_labor\_income,  
 lstatus, sector\_w, marz.x ) %>%  
 rename(marz = marz.x)  
  
# Exporting to Stata (might be necessary for reweigthing with wentropy)  
# write\_dta(pp\_microsim, path = "outputs/pp\_microsim.dta", version = 10)

## 5 UN Population Projections

Now we are ready to move to our demographic projections and macroeconomic model information.

First, filtering based on country (our iso variable).

population\_projections <- population\_projections %>%   
 filter(country == iso)

Collapsing data by summing up variables starting with “yf” and “ym” and reshaping data to long format.

population\_projections <- population\_projections %>%  
 group\_by(Variant, country, cohort) %>%  
 summarize(across(starts\_with(c("yf", "ym")), sum)) %>%  
 ungroup()  
  
population\_projections <- pivot\_longer(population\_projections,  
 cols = starts\_with(c("yf", "ym")),  
 names\_to = c(".value", "year"),  
 names\_pattern = "(yf|ym)(.\*)")

Creating new variable total\_population as the sum of yf and ym. Dropping country variables.

population\_projections <- population\_projections %>%  
 mutate(total\_population = yf + ym) %>%  
 select( -country) %>%   
 mutate(year = as.numeric(year))

Summarizing the year to find the range.

minyear <- survey\_year # Make sure `survey\_year` is correctly defined  
maxyear <- max(as.numeric(population\_projections$year))

We have that the “Min Year” is minyear and the “Max Year” is maxyear. Now we create a population growth rate by demographic variant dataset. We initialize an empty list to store our data by variant and we loop over variants to create this list.

# With minyear and maxyear defined above  
# Initialize a list to store growth data  
pop\_growth <- list()  
  
# Loop over variants  
variants <- unique(population\_projections$Variant)  
for (variant in variants) {  
 for (t in minyear:maxyear) {  
   
 # Calculate population for year t  
 pop\_t <- population\_projections %>%  
 filter(year == t, Variant == variant) %>%  
 summarize(sum\_pop = sum(total\_population)) %>%  
 pull(sum\_pop)  
   
 # Calculate population for base year  
 pop\_base <- population\_projections %>%  
 filter(year == minyear, Variant == variant) %>%  
 summarize(sum\_pop = sum(total\_population)) %>%  
 pull(sum\_pop)  
   
 # Calculate growth rate and store in list with dynamic naming  
 growth\_rate <- pop\_t / pop\_base  
 pop\_growth[[paste0(t, "\_", variant)]] <- list(  
 growth\_rate = growth\_rate, pop\_t = pop\_t  
 )  
 }  
}

With the list ready, we convert back to dataframe by stitching the list elements one on top of the other.

# Convert list to dataframe  
pop\_growth <- do.call(rbind, lapply(names(pop\_growth), function(x) {  
 # Extract year and variant from the name  
 parts <- unlist(strsplit(x, "\_"))  
 year <- as.integer(parts[1])  
 variant <- parts[2]  
   
 # Create a tibble for each entry  
 tibble(year = year,   
 variant = variant,   
 total\_population = pop\_growth[[x]]$pop\_t,  
 pop\_growth\_rate = pop\_growth[[x]]$growth\_rate)  
}))  
  
# Arrange the dataframe for better readability  
pop\_growth <- arrange(pop\_growth, variant, year)  
  
# Display the first few rows of the dataframe  
pop\_growth[c(1:09),]

GEOSTAT. (2023). *Integrated Living Conditions Survey 2023*.