Armenia CCDR Microsimulation

Renato Vargas

Julie Rozenberg

Colin Lenoble

Natsuko Kiso Nozaki

Thomas Farole

Table of contents

[1 Introduction 1](#_Toc168064786)

[2 Preamble 2](#_Toc168064787)

[3 Datasets 3](#_Toc168064788)

[4 Data preparation income outliers and missings 6](#_Toc168064789)

[4.1 Household consumption aggregates and characteristics 6](#_Toc168064790)

[4.2 Demographic characteristics, education, Labor Force 6](#_Toc168064791)

[4.3 The regression 11](#_Toc168064792)

[4.4 Total income and shares 12](#_Toc168064793)

[5 UN Population Projections 14](#_Toc168064794)

[6 Macro Scenarios 16](#_Toc168064795)

[7 Reweighting of the dataset 19](#_Toc168064796)

[7.1 Aggregation of population data 19](#_Toc168064797)

[7.2 Reweigting 22](#_Toc168064798)

[8 Microsimulation 29](#_Toc168064799)

[8.1 Baseline 29](#_Toc168064800)

[8.2 Climate change 30](#_Toc168064801)

[8.3 Food prices 34](#_Toc168064802)

[9 End 43](#_Toc168064803)

## 1 Introduction

In this calculation file, we “age” the 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.

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:

root/  
├── scripts  
│ └── my\_script.R  
├── 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())  
  
# Armenia country ISO code  
iso <- "ARM"  
  
# Survey year  
surveyyear <- 2022  
  
# Exchange rate USD per dram  
er <- 0.002310

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.

# Load packages  
library(tidyverse) # includes dplyr, ggplot2 and others  
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(ebal) # Entropy reweighting  
library(anesrake) # Raking reweighting  
#library(weights) # Weigthed survey statistics  
library(janitor) # pretty subtotals  
library(broom) # More regressions  
library(purrr) # map vectors (aggregation)  
library(zoo) # Calculate moving window average and max value  
  
  
# Geopackages  
library(sf) # to read and write shapefile maps  
library(terra) # to perform geocalculations  
library(tmap) # for static and interactive maps

Stata integration

library(RStata) # stata integration for wentropy function  
options("RStata.StataPath" = "\"C:\\Program Files (x86)\\Stata11\\StataMP\"")  
options("RStata.StataVersion" = 11)

## 3 Datasets

We then load the datasets that we need for this study. The World Bank has processed some of these already for poverty analysis and so we have the original SPSS datasets with all variables for Households hh and for Individuals pp, as well as a consumption aggregate ca and a household income ic dataset, which are Stata datasets. This is for the year 2022. These are imported using the haven package. These are based on Armenia Integrated Living Conditions Survey 2022 (ARMSTAT, 2023).

# Original SPSS datasets  
# Households (hh)  
hh <- read\_sav(  
 "data/ARM-HH-survey/original-spss-files/ILCS-ARM-2022-Households.sav")  
# Persons (pp)  
pp <- read\_sav(  
 "data/ARM-HH-survey/original-spss-files/ILCS-ARM-2022-Persons.sav")  
  
# Processed WB datasets  
# Consumption aggregate at household level (ca)  
ca <- read\_dta("data/ARM-HH-survey/CONSAGG2022.dta")  
# Processed income at household level (ic)  
ic <- read\_dta("data/ARM-HH-survey/totinc.dta")   
  
# Food diary  
food\_with\_prices <- read\_dta("data/ARM-HH-survey/FOOD\_with\_prices.dta")

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 ca data with only our household identifiers, deciles, and poverty.

# From the WB processed dataset, we extract deciles and poverty  
deciles <- ca %>%   
 select( hhid, decile, poor\_Avpovln2022,   
 poor\_Foodpovln2022, poor\_Lpovln2022, poor\_Upovln2022)

Our population data comes from UN’s projections.

# Population projections UN 2022  
population\_projections <- read\_dta("data/UN2022\_population.dta") %>%   
 filter(country == iso)

Macro scenario dataset

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

Economic sectors.

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, which can be inserted using **Ctrl + m.** Although there is no geoprocessing in this analysis, this will come in handy for graphical presentations. Let’s have a look at it.

# Geodata  
# Armenia marzes or administrative level 1 shapefile  
adm1 <- read\_sf("data/ARM-Geodata/ARM-ADM1.shp") %>%   
 select(NAM\_1, COD\_HH\_SVY, geometry) %>%   
 # Make sure that names match the rest of datasets  
 mutate(NAM\_1 = if\_else(NAM\_1 == "Gergharkunik", "Gegharkunik", NAM\_1))  
names(adm1)[2] <- "hh\_02"  
  
tm\_shape(adm1)+  
 tm\_polygons("NAM\_1", legend.show = FALSE) +  
 tm\_text("NAM\_1", size = 3/4)



Marzes names are more accurate in the shapefile than in the survey. We will use them from here on instead of the survey factor labels.

hh <- hh %>%   
 left\_join(adm1, join\_by(hh\_02 == hh\_02)) %>%   
 select(-geometry)  
  
ic <- ic %>%   
 left\_join(adm1, join\_by(hh\_02 == hh\_02)) %>%   
 select(-geometry)

Labor productivity

file <- r"(data/ARM-Microsimulation/LaborProductivityChanges.xlsx)"  
sheets <- excel\_sheets(file)  
  
# Use lapply to read and process each sheet  
heat\_l\_pdcty <- lapply(sheets, function(sheet) {  
 info <- read\_excel(  
 file,  
 sheet = sheet,  
 col\_names = TRUE,  
 col\_types = c("text", "text", "numeric", "text", "numeric")  
 )  
 info$sector <- sheet  
 return(info)  
})  
  
# Bind all data frames in the list into a single data frame  
heat\_l\_pdcty <- bind\_rows(heat\_l\_pdcty)

Finally, but not least important, we have our vulnerability information.

buildings\_aal <-   
 read\_xlsx("data/ARM-Vulnerability-Analysis/Data\_AAL\_AAE.xlsx",  
 sheet = "Building\_AAL") %>%   
 # Make sure that names match the rest of datasets  
 mutate(NAM\_1 = if\_else(NAM\_1 == "Gergharkunik", "Gegharkunik", NAM\_1))  
buildings\_1in100 <-  
 read\_xlsx("data/ARM-Vulnerability-Analysis/Data\_AAL\_AAE.xlsx",  
 sheet = "Building\_1in100")  
crops\_productivity <-   
 read.csv("data/ARM-Vulnerability-Analysis/ARM\_crops\_combined\_REF\_shock\_admin1.csv") %>%   
 rename(NAM\_1 = Province)  
crops\_aal <-   
 read\_xlsx("data/ARM-Vulnerability-Analysis/Data\_AAL\_AAE.xlsx",  
 sheet = "Agriculture\_AAL")  
crops\_1in100 <-  
 read\_xlsx("data/ARM-Vulnerability-Analysis/Data\_AAL\_AAE.xlsx",  
 sheet = "Agriculture\_1in100")

## 4 Data preparation income outliers and missings

### 4.1 Household consumption aggregates and characteristics

Initial necessary variables.

consumption\_aggregates <- ca %>%   
 mutate(rural = ifelse(urb\_rur == 2, 1, 0), # Create rural indicator  
 yhh = totc, # Total household expenditure  
 wgt\_adj = pweight) %>% # Make a copy of the weight variable   
 select(hhid, rural, hhsize,hhsize\_R, marz, aepc, yhh, wgt\_adj, weight,   
 Foodpovln2022, Lpovln2022, Upovln2022, Avpovln2022,   
 poor\_Foodpovln2022, poor\_Lpovln2022, poor\_Upovln2022,   
 poor\_Avpovln2022, decile ) # Keep only necessary columns

### 4.2 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 Armenia. However, we use hh\_01\_code (settlement) for this purpose.

# Zone data  
zone\_data <- hh %>%   
 select(interview\_\_key, hh\_01\_code, hh\_02, hh\_03, NAM\_1) %>%   
 mutate(  
 hhid = interview\_\_key, # Household id  
 zone = hh\_01\_code, # Settlement  
 marz = hh\_02, # Marz  
 NAM\_1 = NAM\_1, # Marz name  
 urb\_rur = hh\_03 # Urban / rural  
 )

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.

pp\_microsim <- pp %>%  
 rename(hhid = interview\_\_key) %>%   
 left\_join(zone\_data, join\_by( hhid == hhid)) %>%   
 mutate(# Demographic characteristics  
 pid = paste0(interview\_\_key, "-",   
 str\_pad(mem\_001\_\_id, 2, pad = "0")), # Unique person id  
 gender = mem\_02,  
 age = mem\_05,  
 head = ifelse(mem\_03 == 1, 1, 0),  
 # Education level  
 educy = ifelse(is.na(ed\_03) | ed\_03 == 8, 0, ed\_03),  
 # 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) # %>%   
 # select(hhid, pid, gender, age, head, rel, zone, marz, urb\_rur, educy,  
 # lstatus, employed, salaried, )

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

pp\_microsim <- pp\_microsim %>%  
 mutate(calif = case\_when(  
 educy >= 0 & educy <= 2 ~ "None - General",  
 educy > 3 & educy <= 7 ~ "Secondary - Vocational",  
 educy > 7 & educy <= 11 ~ "Higher +",  
 TRUE ~ NA\_character\_ # This handles any values outside the specified ranges  
 ))  
  
# View the first few rows to confirm the recoding  
head(pp\_microsim[,c("calif")])

# A tibble: 6 × 1  
 calif   
 <chr>   
1 Secondary - Vocational  
2 None - General   
3 Secondary - Vocational  
4 Secondary - Vocational  
5 Secondary - Vocational  
6 Secondary - Vocational

Count the number of employed persons by household.

pp\_microsim <- pp\_microsim %>%   
 mutate(employed = (lstatus == 1)) %>%   
 group\_by(hhid) %>%   
 mutate(employed\_hh = sum(employed, na.rm = TRUE)) %>% # Count within each household   
 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.

**Primary and Secondary Job income**

* **emp\_11** 11.How much was %rostertitle%’s payment for wages/salary/income for last month?
* **emp\_12** 12.What period of time was the wage/income for?
* **emp\_25** 25.How much was %rostertitle%’s payment for wages/salary/income for last month?
* **emp\_26** 26.What period of time was the wage/income for?

Bonus, In-Kind, and food from job was not asked in Armenia, If it were, you should add a mutate() statement like the ones below for each subcategory.

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))

26.23% employed with no labor income reported!!! We calculate this way:

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(percent\_employed\_no\_income)

[1] 28.57496

Let’s flag outliers now

pp\_microsim <- pp\_microsim %>%   
 # Filter for employed and positive income   
 #filter(employed == TRUE & annual\_labor\_total > 0) %>%   
 mutate(  
 sd = sd(annual\_labor\_total, na.rm = TRUE), # Calculate standard deviation  
 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

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)

Impute sector for those with missing employed by hh head sector.

Step 1: Impute sector for missing employed by the sector of any other hh member.

pp\_microsim <- pp\_microsim %>%  
 group\_by(hhid) %>%  
 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()

Step 2: 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, matching, because it’s only 2 observations).

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

Step 4: Label the sector variable.

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

Step 5: No sector for OLF and clonevar industry=sector (this from original Stata code).

pp\_microsim <- pp\_microsim %>%  
 mutate(lstatus = as.numeric(lstatus),  
 sector = if\_else(lstatus == 4, as.character(NA), as.character(sector)),  
 industry = as.factor(sector)) %>%   
 mutate(sector\_w = sector) # We need this for reweighting and not messing up   
 # and not mess up the regression below.

### 4.3 The regression

Prepare the data.

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.

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(consumption\_aggregates, by = "hhid")

### 4.4 Total income and shares

Total labor income at HH level.

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

Monthly incomes come from the ic data set.

incomes <- ic %>%   
 select(interview\_\_key, 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("hhid" = "interview\_\_key")) %>%  
 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(hhid == interview\_\_key)) %>%   
# mutate(nli\_hh = ( coalesce(inc5) +   
# coalesce(inc6) +  
# coalesce(inc7) +  
# coalesce(inc8)) \* 12) %>%   
# mutate(income\_hh = lab\_hh + nli\_hh)

Calculating shares:

pp\_microsim <- pp\_microsim %>%  
 mutate(  
 s\_lab = lab\_hh / income\_hh,  
 s\_nli = nli\_hh / income\_hh,  
 lny = log(income\_hh),  
 lnc = log(yhh), # comes from consumption aggregates  
 mpc = yhh / income\_hh  
 )

Shares of labor and non-labor income, and additional calculations.

pp\_microsim <- pp\_microsim %>%  
 mutate(  
 share = if\_else(employed == TRUE, annual\_labor\_total / lab\_hh, NA\_real\_),  
 ylb = yhh \* s\_lab,  
 ynl = yhh \* (1 - s\_lab),  
 ylbi = if\_else(employed == TRUE, ylb \* share, NA\_real\_)  
 )

Final subset of data.

pp\_microsim <- pp\_microsim %>%  
 select(hhid, pid, industry, yhh, ylb, ynl, ylbi, salaried,  
 rural, hhsize,hhsize\_R, marz.x, aepc, yhh, wgt\_adj, weight,   
 Foodpovln2022, Lpovln2022, Upovln2022, Avpovln2022,   
 poor\_Foodpovln2022, poor\_Lpovln2022, poor\_Upovln2022,   
 poor\_Avpovln2022, decile, zone, urb\_rur,  
 gender, age, head, rel, zone, educy, calif, sector, sector\_name,  
 annual\_labor\_total,annual\_labor\_income\_primary,  
 annual\_labor\_income\_secondary,monthly\_labor\_income,  
 lstatus, sector\_w, NAM\_1 ) %>%   
 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) %>%  
 summarise(across(starts\_with(c("yf", "ym")), sum)) %>%  
 ungroup()

`summarise()` has grouped output by 'Variant', 'country'. You can override  
using the `.groups` argument.

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 <- surveyyear # Make sure `surveyyear` is correctly defined  
maxyear <- max(as.numeric(population\_projections$year))  
  
# Print the year range as a check  
print(paste("Min Year:", minyear, "- Max Year:", maxyear))

[1] "Min Year: 2022 - Max Year: 2100"

# 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) %>%  
 summarise(sum\_pop = sum(total\_population)) %>%  
 pull(sum\_pop)  
   
 # Calculate population for base year  
 pop\_base <- population\_projections %>%  
 filter(year == minyear, Variant == variant) %>%  
 summarise(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  
 )  
 }  
}  
  
# 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),]

# A tibble: 9 × 4  
 year variant total\_population pop\_growth\_rate  
 <int> <chr> <dbl> <dbl>  
1 2022 Constant-fertility 2780. 1   
2 2023 Constant-fertility 2778. 0.999  
3 2024 Constant-fertility 2778. 0.999  
4 2025 Constant-fertility 2776. 0.998  
5 2026 Constant-fertility 2774. 0.998  
6 2027 Constant-fertility 2770. 0.996  
7 2028 Constant-fertility 2766. 0.995  
8 2029 Constant-fertility 2761. 0.993  
9 2030 Constant-fertility 2755. 0.991

We load elasticities.

elasticities <- c(0.82, 0.9, 0.79) # Agr, Manuf, Services  
yearsto <- c(2030)

## 6 Macro Scenarios

The following code accomplishes the following:

* Import data from Excel sheets corresponding to each scenario and combine them into one data frame.
* Rename columns, create a ‘scenid’ to identify scenarios, and merge with population projections.
* Calculate real wages and consumption per capita.

# Macro Scenario File imported in "Datasets" section (scenario\_file)   
sheets <- excel\_sheets(scenario\_file)  
  
# Define the names of the scenarios and the variants   
scenarios <- sheets[c(1,2)] # modify list with the tab numbers or names with scenarios  
  
# Create an empty list to store data frames for each scenario  
scen\_data\_list <- list()  
  
# Import data for each scenario and store it in the list  
for (i in seq\_along(scenarios)) {  
 sheet\_data <- read\_excel(scenario\_file,   
 sheet = scenarios[i],   
 range = "B3:AT31",  
 col\_names = FALSE)  
 sheet\_data$scenario\_id <- scenarios[i]  
 colnames(sheet\_data) <- scenario\_varlist$var\_short\_name  
 scen\_data\_list[[i]] <- sheet\_data  
}

New names:  
New names:  
• `` -> `...1`  
• `` -> `...2`  
• `` -> `...3`  
• `` -> `...4`  
• `` -> `...5`  
• `` -> `...6`  
• `` -> `...7`  
• `` -> `...8`  
• `` -> `...9`  
• `` -> `...10`  
• `` -> `...11`  
• `` -> `...12`  
• `` -> `...13`  
• `` -> `...14`  
• `` -> `...15`  
• `` -> `...16`  
• `` -> `...17`  
• `` -> `...18`  
• `` -> `...19`  
• `` -> `...20`  
• `` -> `...21`  
• `` -> `...22`  
• `` -> `...23`  
• `` -> `...24`  
• `` -> `...25`  
• `` -> `...26`  
• `` -> `...27`  
• `` -> `...28`  
• `` -> `...29`  
• `` -> `...30`  
• `` -> `...31`  
• `` -> `...32`  
• `` -> `...33`  
• `` -> `...34`  
• `` -> `...35`  
• `` -> `...36`  
• `` -> `...37`  
• `` -> `...38`  
• `` -> `...39`  
• `` -> `...40`  
• `` -> `...41`  
• `` -> `...42`  
• `` -> `...43`  
• `` -> `...44`  
• `` -> `...45`

# Combine all data frames into one  
combined\_data <- bind\_rows(scen\_data\_list)  
  
# Rename population\_m from the data set because we will use   
# UN pop projections from the other data set.  
combined\_data <- combined\_data %>%   
 rename(population\_m\_macrodata = population\_m)  
  
# Calculate real wages  
combined\_data <- combined\_data %>%  
 mutate(rwage\_agr\_m\_amd = wage\_agr\_m\_amd / cpi,  
 rwage\_man\_m\_amd = wage\_man\_m\_amd / cpi,  
 rwage\_ser\_m\_amd = wage\_ser\_m\_amd / cpi)  
  
pop\_data <- population\_projections %>%   
 group\_by(Variant, year) %>%   
 summarise(female = sum(yf),  
 male = sum(ym),  
 total\_population = sum(total\_population) ) %>%   
 ungroup()

`summarise()` has grouped output by 'Variant'. You can override using the  
`.groups` argument.

# Filter population data to macro model years  
pop\_data <- pop\_data %>%   
 filter(year <= max(combined\_data$year),  
 Variant == variants[7])  
# Merge the combined data with population projections  
macro\_data <- combined\_data %>%  
 left\_join(pop\_data, by = c("year"))  
  
# Calculate consumption per capita and other totals  
macro\_data <- macro\_data %>%  
 mutate(  
 consumption\_pc = consumption\_b\_amd / (total\_population),  
 total\_employment = lab\_agr\_1000p + lab\_man\_1000p + lab\_ser\_1000p,  
 employment\_rate = working\_age\_pop\_m / total\_population  
 )  
  
# Function to add growth rate columns directly in the dataframe  
calculate\_growth <- function(data, value\_column) {  
 growth\_col\_name <- paste0(value\_column, "\_growth") # dynamic name for growth column  
 data %>%  
 arrange(year) %>%  
 group\_by(Variant, scenario\_id) %>%  
 mutate(  
 base\_value = first(!!sym(value\_column)),  
 !!sym(growth\_col\_name) := !!sym(value\_column) / base\_value  
 ) %>%  
 select(-base\_value) %>% # optionally remove base\_value column if not needed  
 ungroup()  
}  
  
# Columns to calculate growth for  
value\_columns <- c(  
 "gdp\_b\_amd", # GDP  
 "consumption\_b\_amd", # Consumption  
 "consumption\_pc", # Consumption PC  
 "remittances\_b\_amd", # Remittances  
 "total\_employment", # Employment  
 "employment\_rate", # Employment rate  
 "working\_age\_pop\_m", # Working age population  
 "va\_agr\_b\_amd", # Value added agriculture  
 "va\_man\_b\_amd", # Value added manufacturing  
 "va\_ser\_b\_amd", # Value added services  
 "wage\_agr\_m\_amd", # Nominal wage agriculture  
 "wage\_man\_m\_amd", # Nominal wage manufacturing  
 "wage\_ser\_m\_amd", # Nominal wage services  
 "rwage\_agr\_m\_amd", # Real wage agriculture  
 "rwage\_man\_m\_amd", # Real wage manufacturing  
 "rwage\_ser\_m\_amd" # Real wage services  
 )  
  
# Applying the growth calculation to the macro\_data for each column  
for (col in value\_columns) {  
 macro\_data <- calculate\_growth(macro\_data, col)  
}  
  
# Now `macro\_data` will have growth rate columns for each of the variables listed  
# We rearrange the dataset for clarity  
macro\_data <- macro\_data %>%   
 relocate(scenario\_id, Variant, .before = year) %>%   
 arrange(scenario\_id, Variant, year)

## 7 Reweighting of the dataset

### 7.1 Aggregation of population data

This is based on a custom command to reweight the survey according to macroeconomic data for every possible combination of variant, year, and country. In the macro data we know they only used the “medium” variant and we only need to reweight for a specific year (2030) for Armenia (ARM), so we will conduct the reweighting directly with these parameters.

population\_projections <- population\_projections %>%  
 # filter(Variant == "Medium") %>%  
 # Recoding cohorts into ordered factors  
 mutate(cohort\_short = factor(case\_when(  
 cohort %in% c("P0004", "P0509","P1014",  
 "P1519","P2024", "P2529") ~ "P0029",  
 cohort %in% c("P3034", "P3539") ~ "P3039",  
 cohort %in% c("P4044", "P4549") ~ "P4049",  
 cohort %in% c("P5054", "P5559") ~ "P5059",  
 cohort %in% c("P6064", "P6569","P7074", "P7579",  
 "P8084", "P8589", "P9094", "P9599",  
 "P100up") ~ "P60up"  
 ), levels = c("P0029", "P3039",  
 "P4049", "P5059", "P60up"))) %>%  
 # mutate(cohort = factor(case\_when(  
 # cohort %in% c("P0004", "P0509") ~ "P0009",  
 # cohort %in% c("P1014", "P1519") ~ "P1019",  
 # cohort %in% c("P2024", "P2529") ~ "P2029",  
 # cohort %in% c("P3034", "P3539") ~ "P3039",  
 # cohort %in% c("P4044", "P4549") ~ "P4049",  
 # cohort %in% c("P5054", "P5559") ~ "P5059",  
 # cohort %in% c("P6064", "P6569") ~ "P6069",  
 # cohort %in% c("P7074", "P7579", "P8084", "P8589", "P9094", "P9599", "P100up") ~ "P70up"  
 # ), levels = c("P0009", "P1019", "P2029", "P3039", "P4049", "P5059", "P6069", "P70up"))) %>%  
 # Convert factor 'cohort' to numeric codes  
 mutate(cohort\_code = as.integer(cohort\_short))  
  
# Checking the resulting dataset  
print(pop\_data)

# A tibble: 60 × 5  
 Variant year female male total\_population  
 <chr> <dbl> <dbl> <dbl> <dbl>  
 1 Medium 1991 1867. 1750. 3618.  
 2 Medium 1992 1850. 1724. 3575.  
 3 Medium 1993 1799. 1658. 3457.  
 4 Medium 1994 1763. 1610. 3374.  
 5 Medium 1995 1741. 1581. 3323.  
 6 Medium 1996 1731. 1568. 3299.  
 7 Medium 1997 1719. 1552. 3271.  
 8 Medium 1998 1705. 1535. 3241.  
 9 Medium 1999 1689. 1517. 3206.  
10 Medium 2000 1672. 1496. 3169.  
# ℹ 50 more rows

Let’s now create cohorts in our pp\_microsim data to match our population projection data.

# Convert 'age' into 'cohort' factor with levels ordered as specified  
pp\_microsim <- pp\_microsim %>%  
 mutate(cohort = factor(case\_when(  
 age >= 0 & age <= 29 ~ "P0029",  
 age >= 30 & age <= 39 ~ "P3039",  
 age >= 40 & age <= 49 ~ "P4049",  
 age >= 50 & age <= 59 ~ "P5059",  
 age >= 60 ~ "P60up"  
 ), levels = c("P0029", "P3039", "P4049", "P5059", "P60up")))  
 # mutate(cohort = factor(case\_when(  
 # age >= 0 & age <= 9 ~ "P0009",  
 # age >= 10 & age <= 19 ~ "P1019",  
 # age >= 20 & age <= 29 ~ "P2029",  
 # age >= 30 & age <= 39 ~ "P3039",  
 # age >= 40 & age <= 49 ~ "P4049",  
 # age >= 50 & age <= 59 ~ "P5059",  
 # age >= 60 & age <= 69 ~ "P6069",  
 # age >= 70 ~ "P70up"  
 # ), levels = c("P0009", "P1019", "P2029", "P3039", "P4049", "P5059", "P6069", "P70up")))  
  
# Convert the 'cohort' and 'gender' factor to numeric codes  
pp\_microsim <- pp\_microsim %>%  
 mutate(cohort\_code = as.integer(cohort)) %>%   
 mutate(gender\_code = as.integer(gender))

We also need demographic targets for 2030

# Ensure pop\_targets\_2030 is correctly prepared  
pop\_targets\_2030 <- population\_projections %>%   
 filter(year == 2030, Variant == variants[7]) %>%   
 group\_by(cohort\_code, cohort\_short) %>%   
 summarise(female = sum(yf),  
 male = sum(ym),   
 total = sum(total\_population),  
 ) %>%  
 ungroup()

`summarise()` has grouped output by 'cohort\_code'. You can override using the  
`.groups` argument.

pop\_total <- sum(pop\_targets\_2030$total)  
  
pop\_targets\_2030 <- pop\_targets\_2030 %>%   
 mutate(pct\_total = total / pop\_total)

And economic targets from our macroeconomic scenario data.

economic\_targets\_2030 <- macro\_data %>%  
 filter(year == 2030, Variant == "Medium", scenario\_id == "baseline") %>%  
 summarise(  
 target\_lab\_agr = sum(lab\_agr\_1000p \* 1000),  
 target\_lab\_man = sum(lab\_man\_1000p \* 1000),  
 target\_lab\_ser = sum(lab\_ser\_1000p \* 1000)  
 )

For a better representation of the labor market, we will take into account the combination between labor status and economic sector of the employed and adjust that combination according to the macrodata so that we can accurately model changes in total employment, sector distribution of the employed and overall population changes.

pp\_microsim <- pp\_microsim %>%   
 mutate(lmarket = case\_when(  
 lstatus == 1 & sector\_w == 1 ~ 1, # Agriculture  
 lstatus == 1 & sector\_w == 2 ~ 2, # Manufactures  
 lstatus == 1 & sector\_w == 3 ~ 3, # Services  
 lstatus == 2 & is.na(sector\_w) ~ 4, # Unemployed  
 lstatus == 3 & is.na(sector\_w) ~ 4, # Unemployed  
 lstatus == 4 & is.na(sector\_w) ~ 5, # OLF  
   
 ))

And we check the values.

write.table(  
 pp\_microsim %>%   
 group\_by(lmarket) %>%   
 summarise(total\_pp = sum(weight, na.rm = TRUE)) %>%   
 ungroup(),   
 "clipboard", sep="\t", row.names=FALSE  
 )

Note that the differences between the totals of the survey and the macro file for the base year are very much different. We’ll adjust the survey only with relative growth instead of total numbers so that labor income doesn’t change completely.

### 7.2 Reweigting

We use anesrake to calculate targets from known future proportions of sex, age, economic sector. We first create a target list.

# Target for each variable  
  
gender\_code <- c(  
 sum(pop\_targets\_2030$male) /   
 (sum(pop\_targets\_2030$male)+ sum(pop\_targets\_2030$female)),   
 sum(pop\_targets\_2030$female) /   
 (sum(pop\_targets\_2030$male)+ sum(pop\_targets\_2030$female)))  
  
cohort\_code <- pop\_targets\_2030$pct\_total  
  
# Four digits are better than two in this case, raking is quite accurate.  
lmarket <- c(0.1341, 0.0494, 0.2611, 0.2472, 0.3082)  
  
# Target list  
  
targets <- list(gender\_code  
 , cohort\_code  
 , lmarket  
 )  
  
names(targets) <- c("gender\_code",   
 "cohort\_code",   
 "lmarket"  
 )  
  
# Since this uses base R, we need to turn the data frame into base R object  
rakedata <- as.data.frame(pp\_microsim)

And now we perform the reweighting, using the original weights.

anesrakefinder(targets, rakedata, choosemethod = "total")

gender\_code cohort\_code lmarket   
 0.03626510 0.09677000 0.07212418

outsave <- anesrake(targets,   
 rakedata,   
 caseid = rakedata$pid,   
 #verbose = FALSE,  
 choosemethod = "total",  
 type = "pctlim",  
 #cap = 100,  
 pctlim = 0.05,  
 nlim = 3,  
 iterate = TRUE,  
 force1 = TRUE,  
 verbose = TRUE,  
 weightvec = rakedata$weight)

[1] "Raking...Iteration 1"  
[1] "Current iteration changed total weights by 2336.85909302283"  
[1] "Raking...Iteration 2"  
[1] "Current iteration changed total weights by 196.013077923158"  
[1] "Raking...Iteration 3"  
[1] "Current iteration changed total weights by 48.9461383276265"  
[1] "Raking...Iteration 4"  
[1] "Current iteration changed total weights by 15.2825935005305"  
[1] "Raking...Iteration 5"  
[1] "Current iteration changed total weights by 4.79740637305395"  
[1] "Raking...Iteration 6"  
[1] "Current iteration changed total weights by 1.50755591186616"  
[1] "Raking...Iteration 7"  
[1] "Current iteration changed total weights by 0.473847760331075"  
[1] "Raking...Iteration 8"  
[1] "Current iteration changed total weights by 0.148945583420334"  
[1] "Raking...Iteration 9"  
[1] "Current iteration changed total weights by 0.0468190426274527"  
[1] "Raking...Iteration 10"  
[1] "Current iteration changed total weights by 0.0147169948904372"  
[1] "Raking...Iteration 11"  
[1] "Current iteration changed total weights by 0.00462611317398909"  
[1] "Raking...Iteration 12"  
[1] "Current iteration changed total weights by 0.00145416443982054"  
[1] "Raking...Iteration 13"  
[1] "Current iteration changed total weights by 0.00045709959257946"  
[1] "Raking...Iteration 14"  
[1] "Current iteration changed total weights by 0.000143683917964627"  
[1] "Raking...Iteration 15"  
[1] "Current iteration changed total weights by 4.51653621963283e-05"  
[1] "Raking...Iteration 16"  
[1] "Current iteration changed total weights by 1.41972053157718e-05"  
[1] "Raking...Iteration 17"  
[1] "Current iteration changed total weights by 4.46272523452196e-06"  
[1] "Raking...Iteration 18"  
[1] "Current iteration changed total weights by 1.40280532930082e-06"  
[1] "Raking...Iteration 19"  
[1] "Current iteration changed total weights by 4.40955060437842e-07"  
[1] "Raking...Iteration 20"  
[1] "Current iteration changed total weights by 1.38610069977396e-07"  
[1] "Raking...Iteration 21"  
[1] "Current iteration changed total weights by 4.35708336493645e-08"  
[1] "Raking...Iteration 22"  
[1] "Current iteration changed total weights by 1.36949823009713e-08"  
[1] "Raking...Iteration 23"  
[1] "Current iteration changed total weights by 4.30483908242518e-09"  
[1] "Raking...Iteration 24"  
[1] "Current iteration changed total weights by 1.35239161536127e-09"  
[1] "Raking...Iteration 25"  
[1] "Current iteration changed total weights by 4.25884119636954e-10"  
[1] "Raking...Iteration 26"  
[1] "Current iteration changed total weights by 1.34918465288791e-10"  
[1] "Raking...Iteration 27"  
[1] "Current iteration changed total weights by 4.08980349586585e-11"  
[1] "Raking...Iteration 28"  
[1] "Current iteration changed total weights by 1.39531441956109e-11"  
[1] "Raking...Iteration 29"  
[1] "Current iteration changed total weights by 4.07238132105192e-12"  
[1] "Raking...Iteration 30"  
[1] "Current iteration changed total weights by 2.60423627107542e-12"  
[1] "Raking...Iteration 31"  
[1] "Current iteration changed total weights by 2.40631126136037e-12"  
[1] "Raking...Iteration 32"  
[1] "Current iteration changed total weights by 0"  
[1] "Raking...Iteration 33"  
[1] "Current iteration changed total weights by 0"  
[1] "Raking converged in 33 iterations"

summary(outsave)

$convergence  
[1] "Complete convergence was achieved after 33 iterations"  
  
$base.weights  
[1] "Using Base Weights Provided"  
  
$raking.variables  
[1] "cohort\_code" "lmarket"   
  
$weight.summary  
 Min. 1st Qu. Median Mean 3rd Qu. Max.   
 0.1103 0.6867 0.9272 1.0000 1.2758 3.2390   
  
$selection.method  
[1] "variable selection conducted using \_pctlim\_ - discrepancies selected using \_total\_."  
  
$general.design.effect  
[1] 1.241567  
  
$gender\_code  
 Target Old Weights N Old Weights % Wtd N Wtd % Change in %  
<NA> 0.4514133 NA NA NA NA NA  
<NA> 0.5485867 NA NA NA NA NA  
Total 1.0000000 0 0 0 0 0  
 Resid. Disc. Orig. Disc.  
<NA> NA NA  
<NA> NA NA  
Total 0 0  
  
$cohort\_code  
 Target Old Weights N Old Weights % Wtd N Wtd % Change in %  
<NA> 0.3638899 NA NA NA NA NA  
<NA> 0.1420277 NA NA NA NA NA  
<NA> 0.1630905 NA NA NA NA NA  
<NA> 0.1042513 NA NA NA NA NA  
<NA> 0.2267405 NA NA NA NA NA  
Total 1.0000000 0 0 0 0 0  
 Resid. Disc. Orig. Disc.  
<NA> NA NA  
<NA> NA NA  
<NA> NA NA  
<NA> NA NA  
<NA> NA NA  
Total 0 0  
  
$lmarket  
 Target Old Weights N Old Weights % Wtd N Wtd % Change in % Resid. Disc.  
<NA> 0.1341 NA NA NA NA NA NA  
<NA> 0.0494 NA NA NA NA NA NA  
<NA> 0.2611 NA NA NA NA NA NA  
<NA> 0.2472 NA NA NA NA NA NA  
<NA> 0.3082 NA NA NA NA NA NA  
Total 1.0000 0 0 0 0 0 0  
 Orig. Disc.  
<NA> NA  
<NA> NA  
<NA> NA  
<NA> NA  
<NA> NA  
Total 0

# add weights to the dataset  
  
rakedata$weightvec <- unlist(outsave[1])  
n <- length(rakedata$sector)  
  
# Calculate the sum of original weights  
original\_weight\_sum <- sum(rakedata$weight)  
  
# # Target scaling for original weights  
# In a first version, we used the population scaling from UN,  
# But to match with the labor market given by the scenario, we take  
# the total population from the macro scenario information.  
# original\_weight\_scaling\_factor <- pop\_data$total\_population[pop\_data$year == 2030] /  
# pop\_data$total\_population[pop\_data$year == 2022]  
  
# For year 2030 (Calculated in Excel, but need a more elegant solution from dataset)  
original\_weight\_scaling\_factor <- 0.992805755  
  
# Scaled original weights  
original\_weight\_sum <- (original\_weight\_sum   
 \* original\_weight\_scaling\_factor)  
  
# Calculate the sum of the new weights  
new\_weight\_sum <- sum(rakedata$weightvec)  
  
# Scale the new weights to match the sum of the original weights  
scaling\_factor <- original\_weight\_sum / new\_weight\_sum  
rakedata$weightvec <- rakedata$weightvec \* scaling\_factor  
  
# Verify the adjustment  
head(rakedata[, c("weight", "weightvec")])

weight weightvec  
1 185.7685 182.0355  
2 185.7685 196.3028  
3 122.7176 107.8986  
4 185.7685 182.0355  
5 326.8796 257.7991  
6 326.8796 320.3109

summary(rakedata$weightvec)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 16.94 105.46 142.38 153.57 195.92 497.42

summary(rakedata$weight)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 21.48 109.82 156.33 154.68 192.16 326.88

rakedata <- rakedata %>%   
 relocate(weightvec, .after = weight) %>%   
 mutate(hh\_weight = weightvec / hhsize)  
  
# We calculate new weights for households in the hh database  
weights\_2030 <- rakedata %>%   
 group\_by(hhid) %>%  
 summarise(hh\_weight\_2030 = sum(hh\_weight, na.rm = TRUE))  
  
pp\_microsim <- tibble(rakedata)  
rm(rakedata)

As a last step, we rescale labor income according to changes in the wage bill in the macro scenario.

# Wage rescale factor by sector from macro (Agriculture, Manufacturing, Services)  
wrf <- c(1.269821454, 1.284834838, 1.328623737)  
  
# We check the wage bill by sector  
wages\_by\_sector <- pp\_microsim %>%   
 group\_by(sector\_w, .drop = TRUE) %>%   
 summarise(  
 wages\_2022 = sum(annual\_labor\_total \* weight, na.rm = TRUE),  
 wages\_2030 = sum(annual\_labor\_total \* weightvec, na.rm = TRUE)  
 )  
  
wages\_by\_sector <- wages\_by\_sector[c(-4),]  
  
# Compare how much it changed with reweighting with how it should have changed  
# Derive coefficients (wtc\_2030) from that  
wages\_by\_sector <- wages\_by\_sector %>%   
 mutate(wages\_target\_2030 = case\_when(  
 sector\_w == 1 ~ wages\_2022 \* wrf[1],  
 sector\_w == 2 ~ wages\_2022 \* wrf[2],  
 sector\_w == 3 ~ wages\_2022 \* wrf[3],  
 .default = NA  
 ),  
 wtc\_2030 = wages\_target\_2030 / wages\_2030)   
  
wages\_by\_sector %>%  
 gt()

| sector\_w | wages\_2022 | wages\_2030 | wages\_target\_2030 | wtc\_2030 |
| --- | --- | --- | --- | --- |
| 1 | 329980600 | 332988623 | 419016446 | 1.258351 |
| 2 | 250074262 | 262381786 | 321304124 | 1.224567 |
| 3 | 1255196730 | 1317702602 | 1667684171 | 1.265600 |

We then add the coefficient to rescale each weight by sector

# Assign rescale the annual and monthly wage depending on the sector  
pp\_microsim <- pp\_microsim %>%   
 rename(monthly\_labor\_income\_old = monthly\_labor\_income,  
 annual\_labor\_total\_old = annual\_labor\_total) %>%   
 mutate(  
 monthly\_labor\_income = case\_when(  
 sector\_w == 1 ~ monthly\_labor\_income\_old \* wages\_by\_sector$wtc\_2030[1],  
 sector\_w == 2 ~ monthly\_labor\_income\_old \* wages\_by\_sector$wtc\_2030[2],  
 sector\_w == 3 ~ monthly\_labor\_income\_old \* wages\_by\_sector$wtc\_2030[3],  
 TRUE ~ NA  
 ),  
 annual\_labor\_total = case\_when(  
 sector\_w == 1 ~ annual\_labor\_total\_old \* wages\_by\_sector$wtc\_2030[1],  
 sector\_w == 2 ~ annual\_labor\_total\_old \* wages\_by\_sector$wtc\_2030[2],  
 sector\_w == 3 ~ annual\_labor\_total\_old \* wages\_by\_sector$wtc\_2030[3],  
 TRUE ~ NA  
 )  
 )  
  
# This takes care of different household members coming from different sectors  
hh\_li\_baseline <- pp\_microsim %>%   
 group\_by(hhid) %>%   
 summarise(mli\_2022 = sum(monthly\_labor\_income\_old),  
 mli\_2030 = sum(monthly\_labor\_income),  
 mli\_coef\_2030 = mli\_2030 / mli\_2022) %>%   
 select(hhid, mli\_coef\_2030)  
   
ic\_microsim <- ic %>%   
 left\_join(hh\_li\_baseline, join\_by(interview\_\_key == hhid)) %>%  
 rename(inc2\_old = inc2,  
 inc3\_old = inc3,  
 totalinc\_old = totalinc) %>%   
 mutate(mli\_coef\_2030 = if\_else(is.na(mli\_coef\_2030), 1,mli\_coef\_2030)) %>%   
 mutate(inc2 = inc2\_old \* mli\_coef\_2030,  
 inc3 = inc3\_old \* mli\_coef\_2030) %>%   
 mutate(totalinc = totalinc\_old - coalesce(inc2\_old,0) - coalesce(inc3\_old,0)  
 + coalesce(inc2,0) + coalesce(inc3,0)) %>%   
 mutate(totinc\_coef\_2030 = if\_else(totalinc\_old == 0, 1, totalinc / totalinc\_old)) %>%   
 mutate(totinc\_coef\_2030 = if\_else(is.na(totinc\_coef\_2030), 1, totinc\_coef\_2030))  
  
ic\_coef\_2030 <- ic\_microsim %>%   
 select(interview\_\_key, totinc\_coef\_2030)

## 8 Microsimulation

We now implement different shocks according to various scenarios.

### 8.1 Baseline

For the baseline we only adjust labor income according to the reweighting procedure and rescaling of the wage bill.

ca\_microsim <- ca %>%   
 left\_join(weights\_2030, join\_by(hhid == hhid)) %>%   
 left\_join(ic\_coef\_2030, join\_by(hhid == interview\_\_key)) %>%   
 # We adjust total consumption by the income coefficient  
 # Make sure to remember that totc is baseline 2030  
 rename(totc\_old = totc,  
 # Make sure to remember that pov2022 is baseline 2030  
 poor\_Avpovln2022\_old = poor\_Avpovln2022) %>%   
 mutate(totc = totc\_old \* totinc\_coef\_2030 )

And recalculate poverty.

ca\_microsim <- ca\_microsim %>%   
 rename(aec\_r\_old = aec\_r,  
 weight\_old = weight,  
 weight = hh\_weight\_2030) %>%   
 mutate(aec\_r = totc / ae\_r / PI) %>%   
 mutate(poor\_Avpovln2022 =   
 if\_else(aec\_r < 52883, 1, 0)  
 )

Test

test <- ca\_microsim %>%  
 rename(poor\_original = poor\_Avpovln2022\_old,  
 poor\_2030 = poor\_Avpovln2022) %>%  
 group\_by(poor\_2030) %>%   
 summarise(no\_hh = round(sum(weight, na.rm = TRUE)),  
 no\_pp = round(sum(weight \* hhsize, na.rm = TRUE))) %>%   
 ungroup()  
  
test <- ca\_microsim %>%  
 rename(poor\_original = poor\_Avpovln2022\_old,  
 poor\_2030 = poor\_Avpovln2022) %>%  
 group\_by(poor\_original) %>%   
 summarise(no\_hh = round(sum(weight, na.rm = TRUE)),  
 no\_pp = round(sum(weight \* hhsize, na.rm = TRUE))) %>%   
 ungroup()  
  
test %>%   
 gt()

| Poor, Avpovln2022 | no\_hh | no\_pp |
| --- | --- | --- |
| 0 | 638528 | 2126723 |
| 1 | 143033 | 707267 |

#write.table(test, "clipboard", sep="\t", row.names=FALSE)

### 8.2 Climate change

In the climate change scenario, we ask ourselves, what would happen if agriculture revenues from crops and livestock are reduced due to losses in productivity due to heat. For this, we use crops data.

We add a moving window average and max value for our labor productivity data.

# First calculate moving window average  
heat\_l\_pdcty <- heat\_l\_pdcty %>%  
 group\_by(ADM1\_EN, clim\_scenario) %>%  
 arrange(year) %>%  
 # Moving window average 5 years before, 5 after  
 mutate(  
 moving\_avg = rollapply(  
 pct\_change\_productivity,   
 width = 11,  
 FUN = mean,  
 partial = TRUE,  
 align = "center",   
 fill = NA,  
 na.rm = TRUE),  
 # Moving window max value 5 years before, 5 after  
 # Since it's expressed in negative values (min) is the maximum  
 moving\_max = rollapply(  
 pct\_change\_productivity,   
 width = 11,  
 FUN = min,  
 partial = TRUE,  
 align = "center",   
 fill = NA,  
 na.rm = TRUE)) %>%   
 ungroup()  
  
# Clim scenarios to select  
cs <- unique(heat\_l\_pdcty$clim\_scenario)  
  
# Moving average for year of interest  
lab\_loss\_avg <- heat\_l\_pdcty %>%   
 filter(clim\_scenario == cs[1], year == yearsto[1] ) %>%  
 select(-pct\_change\_productivity,-ADM1\_PCODE,  
 -year,-clim\_scenario, -moving\_max) %>%   
 pivot\_wider(names\_from = sector,  
 values\_from = moving\_avg) %>%   
 rename(agr\_avg = Agriculture,  
 man\_avg = Manufacturing,  
 ser\_avg = Services)  
  
# Max value for year of interest  
lab\_loss\_max <- heat\_l\_pdcty %>%   
 filter(clim\_scenario == cs[1], year == yearsto[1] ) %>%  
 select(-pct\_change\_productivity, -ADM1\_PCODE,  
 -year,-clim\_scenario,-moving\_avg) %>%   
 pivot\_wider(names\_from = sector,  
 values\_from = moving\_max) %>%   
 rename(agr\_max = Agriculture,  
 man\_max = Manufacturing,  
 ser\_max = Services)

We add a moving window average and max value for our crops and livestock productivity data.

# First calculate moving window average  
crops\_productivity <- crops\_productivity %>%  
 group\_by(NAM\_1, climate\_scenario) %>%  
 arrange(year) %>%  
 # Moving window average   
 mutate(  
 moving\_avg = rollapply(  
 pct\_change\_prod,   
 width = 11, # 5 years before, 5 after + reference year = 11  
 FUN = mean,  
 partial = TRUE,  
 align = "center",   
 fill = NA,  
 na.rm = TRUE),  
 # Moving window max value 5 years before, 5 after  
 # Since it's expressed in negative values (min) is the maximum  
 moving\_max = rollapply(  
 pct\_change\_prod,   
 width = 11,  
 FUN = min,  
 partial = TRUE,  
 align = "center",   
 fill = NA,  
 na.rm = TRUE)) %>%   
 ungroup()  
  
# Clim scenarios to select  
cs <- unique(crops\_productivity$climate\_scenario)  
  
# Moving average for year of interest  
ag\_pdcvty\_loss <- crops\_productivity %>%   
 filter(climate\_scenario == cs[1], year == yearsto[1] ) %>%  
 select(-pct\_change\_prod,-GID\_1, -year,-climate\_scenario) %>%   
 rename(crops\_avg\_loss = moving\_avg,  
 crops\_max\_loss = moving\_max)

And then we introduce these values in our ag income and labor income data. First, we attach the percentage losses to the appropriate data set.

# Persons processed dataset  
pp\_microsim\_cc <- pp\_microsim %>%  
 left\_join(lab\_loss\_avg, join\_by(NAM\_1==ADM1\_EN)) %>%   
 left\_join(lab\_loss\_max, join\_by(NAM\_1==ADM1\_EN))  
  
# Household income processed dataset  
ic\_microsim\_cc <- ic\_microsim %>%  
 left\_join(ag\_pdcvty\_loss, join\_by(NAM\_1==NAM\_1))

And we first shock labor income.

# Labor income according to sector  
pp\_microsim\_cc <- pp\_microsim\_cc %>%   
 mutate(sector = as.numeric(sector)) %>%   
 mutate(mli\_cc\_avg = case\_when(  
 # \* 1000 because its thousands of Dram  
 sector == 1 ~ monthly\_labor\_income \* (1 + agr\_avg)\* 1000,   
 sector == 2 ~ monthly\_labor\_income \* (1 + man\_avg)\* 1000,  
 sector == 3 ~ monthly\_labor\_income \* (1 + ser\_avg)\* 1000,  
 TRUE ~ NA  
 )) %>%   
 mutate(mli\_cc\_max = case\_when(  
 # \* 1000 because its thousands of Dram  
 sector == 1 ~ monthly\_labor\_income \* (1 + agr\_max)\* 1000,   
 sector == 2 ~ monthly\_labor\_income \* (1 + man\_max)\* 1000,  
 sector == 3 ~ monthly\_labor\_income \* (1 + ser\_max)\* 1000,  
 TRUE ~ NA  
 ))

We aggregate at household level and register the percent difference between the two labor incomes, so that we can impact labor income by that amount. We don’t do it with absolute numbers because we don’t know the assumptions made by the poverty team to construct the income variable.

ic\_new\_incomes <- pp\_microsim\_cc %>%   
 group\_by(hhid) %>%   
 summarise(  
 mli\_cc\_avg = sum(mli\_cc\_avg, na.rm = TRUE),  
 mli\_cc\_max = sum(mli\_cc\_max, na.rm = TRUE),  
 mli\_original = sum(monthly\_labor\_income\*1000, na.rm = TRUE)  
 ) %>%   
 mutate(mli\_avg\_coef =   
 if\_else(mli\_original == 0 | is.na(mli\_original), 1,  
 mli\_cc\_avg / mli\_original),  
 mli\_max\_coef =   
 if\_else(mli\_original == 0 | is.na(mli\_original), 1,   
 mli\_cc\_max / mli\_original)  
 ) %>%   
 ungroup()  
  
ic\_microsim\_cc <- ic\_microsim\_cc %>%   
 left\_join(ic\_new\_incomes, join\_by(interview\_\_key == hhid)) %>%   
 mutate(inc2\_cc\_avg = inc2 \* mli\_avg\_coef,  
 inc2\_cc\_max = inc2 \* mli\_max\_coef,  
 inc3\_cc\_avg = inc3 \* mli\_avg\_coef,  
 inc3\_cc\_max = inc3 \* mli\_max\_coef)

And now we impact agricultural income inc4.

ic\_microsim\_cc <- ic\_microsim\_cc %>%   
 mutate(inc4\_cc\_avg = inc4 \* (1 + crops\_avg\_loss),  
 inc4\_cc\_max = inc4 \* (1 + crops\_max\_loss))

And recalculate total income.

ic\_microsim\_cc <- ic\_microsim\_cc %>%   
 mutate(totalinc\_cc\_avg = totalinc - rowSums(select(., c(inc2, inc3, inc4)),   
 na.rm = TRUE) +  
 rowSums(select(., c(inc2\_cc\_avg, inc3\_cc\_avg, inc4\_cc\_avg)),   
 na.rm = TRUE),  
 totalinc\_cc\_max = totalinc - rowSums(select(., c(inc2, inc3, inc4)),   
 na.rm = TRUE) +  
 rowSums(select(., c(inc2\_cc\_max, inc3\_cc\_max, inc4\_cc\_max)),   
 na.rm = TRUE)) %>%  
 mutate(totalinc\_cc\_avg\_coef =   
 if\_else(totalinc == 0, 1,  
 totalinc\_cc\_avg / totalinc),  
 totalinc\_cc\_max\_coef =   
 if\_else(totalinc == 0, 1,   
 totalinc\_cc\_max / totalinc)  
 ) %>%   
 mutate(totalinc\_cc\_avg\_coef = if\_else(is.na(totalinc\_cc\_avg\_coef), 1, totalinc\_cc\_avg\_coef),  
 totalinc\_cc\_max\_coef = if\_else(is.na(totalinc\_cc\_max\_coef), 1, totalinc\_cc\_max\_coef))

We assume that the loss in income translates in a loss of expenditure.

income\_losses <- ic\_microsim\_cc %>%   
 select(interview\_\_key,totalinc\_cc\_avg\_coef, totalinc\_cc\_max\_coef)  
  
  
ca\_microsim\_cc <- ca\_microsim %>%   
 left\_join(income\_losses, join\_by(hhid == interview\_\_key))  
  
# And now reduce total consumption  
  
ca\_microsim\_cc <- ca\_microsim\_cc %>%   
 mutate(totc\_cc\_avg = totc \* totalinc\_cc\_avg\_coef,  
 totc\_cc\_max = totc \* totalinc\_cc\_max\_coef) %>%   
 mutate(aec\_r\_cc\_avg = totc\_cc\_avg / ae\_r / PI,  
 aec\_r\_cc\_max = totc\_cc\_max / ae\_r / PI) %>%   
 mutate(poor\_cc\_avg =   
 if\_else(aec\_r\_cc\_avg < 52883, 1, 0),  
 poor\_cc\_max =   
 if\_else(aec\_r\_cc\_max < 52883, 1, 0))  
  
# We make a table to see who became poor.   
  
test <- ca\_microsim\_cc %>%  
 rename(poor\_original = poor\_Avpovln2022,  
 poor\_cc = poor\_cc\_avg) %>%  
 group\_by(poor\_original, poor\_cc) %>%   
 summarise(no\_hh = round(sum(weight, na.rm = TRUE)),  
 no\_pp = round(sum(weight \* hhsize, na.rm = TRUE))) %>%   
 ungroup()

`summarise()` has grouped output by 'poor\_original'. You can override using the  
`.groups` argument.

#write.table(test, "clipboard", sep="\t", row.names=FALSE)

### 8.3 Food prices

We start by looking at the differences of food prices between scenarios.

scenarios <- unique(macro\_data$scenario\_id)  
  
# We extract and reformat the price data  
price\_data <- macro\_data %>%   
 select(year, scenario\_id, starts\_with( c("fpi" , "epi") )) %>%   
 rename(scenario = scenario\_id) %>%   
 pivot\_longer(starts\_with( c("fpi" , "epi") ),   
 names\_to = "type\_decile",   
 values\_to = "index") %>%  
 mutate(decile = parse\_number(type\_decile)) %>%   
 mutate(commodity\_group =   
 case\_when(  
 str\_starts(type\_decile, "fpi") ~ "food",  
 str\_starts(type\_decile, "epi") ~ "energy",  
 TRUE ~ NA\_character\_  
 )) %>%   
 select(-type\_decile) %>%   
 relocate(index, .after = commodity\_group)  
  
# We take a look at price information in 2030  
price\_data %>%   
 filter(year == 2030) %>%   
 group\_by(commodity\_group, scenario) %>%   
 summarise(index = mean(index, na.rm = TRUE)) %>%   
 gt()

`summarise()` has grouped output by 'commodity\_group'. You can override using  
the `.groups` argument.

| scenario | index |
| --- | --- |
| energy | |
| Dry-hot scenario | 1.124883 |
| baseline | 1.129182 |
| food | |
| Dry-hot scenario | 1.075696 |
| baseline | 1.009986 |

So, we will assign a price index depending on which decile the household belonged to in the base year 2022. We will have a column for each scenario. So we manipulate our price data according to our years of interest (in this case, only 2030).

# Filter `price\_data` for the years of interest  
price\_data\_yearsto <- price\_data %>%  
 filter(year %in% yearsto)  
  
# Create a named vector for scenario indices  
scenario\_indices <- setNames(seq\_along(scenarios), scenarios)  
  
# Create the composite string column  
price\_data\_yearsto <- price\_data\_yearsto %>%  
 mutate(  
 scenario\_index = scenario\_indices[scenario],  
 composite\_column = paste(commodity\_group, scenario\_index, year, sep = "\_")  
 ) %>%   
 select(decile,index,composite\_column)  
  
composite\_column\_names <- unique(price\_data\_yearsto$composite\_column)  
  
price\_data\_yearsto <- price\_data\_yearsto %>%   
 pivot\_wider(names\_from="composite\_column", values\_from = index)

And we join with our household’s dataset.

# PP microsim already has decile information from previous join  
ca\_microsim\_cc <- ca\_microsim\_cc %>%   
 left\_join(price\_data\_yearsto, join\_by(decile==decile))

Since we don’t have quantities for the aggregate food expenditure category or for the aggregate energy bundle, we assume a price of 1 in the survey year.

We will estimate price elasticities for a single “food” commodity from the consumption aggregate FOOD\_with\_prices dataset. We add decile data to the original.

food\_summary <- food\_with\_prices %>%   
 left\_join(deciles, join\_by(hhid))  
  
# Step 1: Summarize the data at the household level  
food\_summary <- food\_summary %>%  
 group\_by(hhid, decile) %>%  
 summarise(  
 total\_quantity = sum(q, na.rm = TRUE),  
 weighted\_price = sum(avrpr\_mean \* q, na.rm = TRUE) / sum(q, na.rm = TRUE),  
 .groups = 'drop'  
 )  
  
# Define a function to fit the model and extract the elasticity  
fit\_model <- function(data) {  
 model <- lm(log(total\_quantity) ~ log(weighted\_price), data = data)  
 coef(model)["log(weighted\_price)"]  
}  
  
# Apply the model fitting function by decile  
decile\_models <- food\_summary %>%  
 group\_by(decile) %>%  
 nest() %>%  
 mutate(price\_elasticity = map\_dbl(data, fit\_model)) %>%  
 select(decile, price\_elasticity) %>%   
 mutate(price\_elasticity = if\_else(price\_elasticity >0,  
 price\_elasticity \*(-1),  
 price\_elasticity))  
  
decile\_models

# A tibble: 10 × 2  
# Groups: decile [10]  
 decile price\_elasticity  
 <dbl> <dbl>  
 1 10 -0.206   
 2 8 -0.00939  
 3 2 -0.360   
 4 3 -0.428   
 5 1 -0.483   
 6 9 -0.118   
 7 6 -0.116   
 8 4 -0.299   
 9 5 -0.253   
10 7 -0.0598

Let’s add back the elasticity data to the analysis dataset.

ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 left\_join(decile\_models, by = "decile")

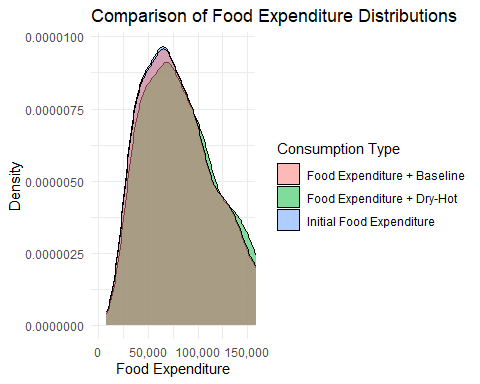
Let’s apply the elasticities to the new data.

# Calculate the implicit price  
# Assuming implicit\_price can be calculated from the expenditure (food1)  
# If we assume baseline quantity consumed is proportional to expenditure/price  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(implicit\_price = food1 / food1, # This is 1 as we don't have baseline price  
 food\_quantity = food1 / implicit\_price)  
  
# Calculate the percentage change in prices for each decile  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(food\_1\_dprice = (food\_1\_2030 - 1),  
 food\_2\_dprice = (food\_2\_2030 - 1))  
  
# Estimate the new food consumption levels  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(  
 food\_q1\_sim = food\_quantity \*   
 (1 + food\_1\_dprice \* price\_elasticity),  
 food\_q2\_sim = food\_quantity \*   
 (1 + food\_2\_dprice \* price\_elasticity))  
  
# Calculate the new expenditure levels  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(food\_exp1\_sim = food\_q1\_sim \* food\_1\_2030,  
 food\_exp2\_sim = food\_q2\_sim \* food\_2\_2030)  
  
# View the results  
print(ca\_microsim\_cc %>% select(decile, food1, food\_1\_2030, food\_1\_dprice, food\_q1\_sim, food\_exp1\_sim, food\_2\_dprice, food\_q2\_sim, food\_exp2\_sim))

# A tibble: 5,184 × 9  
 decile food1 food\_1\_2030 food\_1\_dprice food\_q1\_sim food\_exp1\_sim  
 <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 9 92178. 1.07 0.0749 91365. 98208.  
 2 7 105290. 1.07 0.0743 104822. 112613.  
 3 2 22733. 1.08 0.0761 22111. 23794.  
 4 2 163379. 1.08 0.0761 158905. 170998.  
 5 4 102718. 1.08 0.0774 100342. 108105.  
 6 10 208995. 1.08 0.0777 205650. 221627.  
 7 8 116151. 1.08 0.0759 116068. 124876.  
 8 5 67911. 1.07 0.0742 66634. 71578.  
 9 6 152144. 1.07 0.0749 150818. 162119.  
10 8 80033. 1.08 0.0759 79976. 86045.  
# ℹ 5,174 more rows  
# ℹ 3 more variables: food\_2\_dprice <dbl>, food\_q2\_sim <dbl>,  
# food\_exp2\_sim <dbl>

Let’s plot the distributions to see changes:

# Basic density plot comparing food1 and food\_exp\_sim  
ggplot(ca\_microsim\_cc, aes(x = food1, fill = 'Initial Food Expenditure')) +   
 geom\_density(alpha = 0.5) +   
 geom\_density(  
 data = ca\_microsim\_cc,   
 aes(x = food\_exp1\_sim, fill = 'Food Expenditure + Dry-Hot'),   
 alpha = 0.5) +geom\_density(  
 data = ca\_microsim\_cc,   
 aes(x = food\_exp2\_sim, fill = 'Food Expenditure + Baseline'),   
 alpha = 0.5) +  
 labs(  
 fill = "Consumption Type",   
 title = "Comparison of Food Expenditure Distributions",   
 x = "Food Expenditure",   
 y = "Density") +  
 theme\_minimal() +  
 coord\_cartesian(xlim = c(0, 150000)) + # Adjust the xlim for zoom  
 scale\_x\_continuous(labels = scales::comma) +  
 scale\_y\_continuous(labels = scales::comma)



Calculate losses in consumer surplus and purchasing power loss.

# Calculate the implicit price (assuming baseline implicit price is 1 for simplicity)  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(implicit\_price = 1, # Simplified assumption  
 food\_quantity = food1 / implicit\_price)  
  
# Calculate the percentage change in prices for each decile  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(food\_1\_dprice = (food\_1\_2030 - 1),  
 food\_2\_dprice = (food\_2\_2030 - 1))  
  
# Use the existing new food consumption levels (already calculated)  
# ca\_microsim\_cc <- ca\_microsim\_cc %>%  
# mutate(food\_q1\_sim = food\_quantity \* (1 + food\_1\_dprice \* price\_elasticity),  
# food\_q2\_sim = food\_quantity \* (1 + food\_2\_dprice \* price\_elasticity))  
  
# Calculate Consumer Surplus loss for food1 and food2 scenarios  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(food1\_CSloss = ((food\_quantity \* implicit\_price) / totc) \* food\_1\_dprice \* (1 + (price\_elasticity / 2) \* food\_1\_dprice),  
 food2\_CSloss = ((food\_quantity \* implicit\_price) / totc) \* food\_2\_dprice \* (1 + (price\_elasticity / 2) \* food\_2\_dprice),  
 ttl\_CSloss\_1 = food1\_CSloss,  
 ttl\_CSloss\_2 = food2\_CSloss)  
  
# Calculate Purchasing Power loss for food1 and food2 scenarios  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(food1\_PPloss = (food1 / totc) \* food\_1\_dprice,  
 food2\_PPloss = (food1 / totc) \* food\_2\_dprice,  
 ttl\_PPloss\_1 = food1\_PPloss,  
 ttl\_PPloss\_2 = food2\_PPloss)  
  
# Adjust total expenditure (totc) based on the purchasing power loss  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(totc\_cc\_avg\_food1 = totc\_cc\_avg \* (1 - ttl\_PPloss\_1),  
 totc\_cc\_avg\_food2 = totc\_cc\_avg \* (1 - ttl\_PPloss\_2))  
  
# View the results  
print(ca\_microsim\_cc %>% select(decile, food1, food\_1\_2030, food\_1\_dprice, food\_q1\_sim, food\_exp1\_sim, food\_2\_dprice, food\_q2\_sim, food\_exp2\_sim, food1\_CSloss, food2\_CSloss, ttl\_CSloss\_1, ttl\_CSloss\_2, food1\_PPloss, food2\_PPloss, ttl\_PPloss\_1, ttl\_PPloss\_2, totc\_cc\_avg\_food1, totc\_cc\_avg\_food2))

# A tibble: 5,184 × 19  
 decile food1 food\_1\_2030 food\_1\_dprice food\_q1\_sim food\_exp1\_sim  
 <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 9 92178. 1.07 0.0749 91365. 98208.  
 2 7 105290. 1.07 0.0743 104822. 112613.  
 3 2 22733. 1.08 0.0761 22111. 23794.  
 4 2 163379. 1.08 0.0761 158905. 170998.  
 5 4 102718. 1.08 0.0774 100342. 108105.  
 6 10 208995. 1.08 0.0777 205650. 221627.  
 7 8 116151. 1.08 0.0759 116068. 124876.  
 8 5 67911. 1.07 0.0742 66634. 71578.  
 9 6 152144. 1.07 0.0749 150818. 162119.  
10 8 80033. 1.08 0.0759 79976. 86045.  
# ℹ 5,174 more rows  
# ℹ 13 more variables: food\_2\_dprice <dbl>, food\_q2\_sim <dbl>,  
# food\_exp2\_sim <dbl>, food1\_CSloss <dbl>, food2\_CSloss <dbl>,  
# ttl\_CSloss\_1 <dbl>, ttl\_CSloss\_2 <dbl>, food1\_PPloss <dbl>,  
# food2\_PPloss <dbl>, ttl\_PPloss\_1 <dbl>, ttl\_PPloss\_2 <dbl>,  
# totc\_cc\_avg\_food1 <dbl>, totc\_cc\_avg\_food2 <dbl>

Okay so now we estimate new welfare and poverty.

ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(aec\_r\_cc\_avg\_food1 = totc\_cc\_avg\_food1 / ae\_r / PI,  
 aec\_r\_cc\_avg\_food2 = totc\_cc\_avg\_food2 / ae\_r / PI) %>%  
 mutate(poor\_cc\_avg\_food1 =  
 if\_else(aec\_r\_cc\_avg\_food1 < 52883, 1, 0),  
 poor\_cc\_avg\_food2 =  
 if\_else(aec\_r\_cc\_avg\_food2 < 52883, 1, 0))

And now we see who became poor

# We make a table to see who became poor.   
test <- ca\_microsim\_cc  
  
test <- test%>%  
 rename(poor\_original = poor\_Avpovln2022,  
 poor\_cc = poor\_cc\_avg,  
 poor\_cc\_food1 = poor\_cc\_avg\_food1,  
 poor\_cc\_food2 = poor\_cc\_avg\_food2) %>%  
 group\_by(poor\_original, poor\_cc\_food1) %>%   
 summarise(no\_hh = round(sum(weight, na.rm = TRUE)),  
 no\_pp = round(sum(weight\*hhsize, na.rm = TRUE)))

`summarise()` has grouped output by 'poor\_original'. You can override using the  
`.groups` argument.

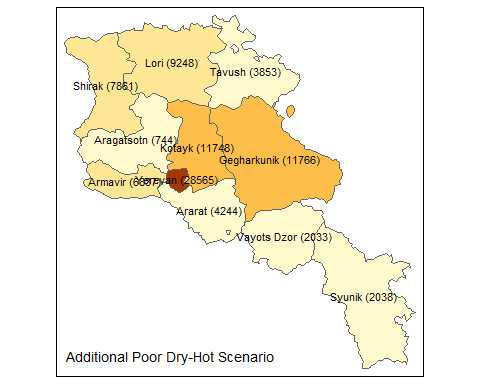
test %>%   
 gt()

| poor\_cc\_food1 | no\_hh | no\_pp |
| --- | --- | --- |
| 0 | | |
| 0 | 615514 | 2013284 |
| 1 | 26154 | 122900 |
| 1 | | |
| 1 | 139892 | 697806 |

#write.table(test, "clipboard", sep="\t", row.names=FALSE)

And we map these results.

foodpoor <- ca\_microsim\_cc %>%  
 mutate(new\_poor\_food\_base = if\_else(  
 poor\_cc\_avg\_food2 == 1 & poor\_cc\_avg == 0, 1, 0),  
 new\_poor\_food\_dryhot = if\_else(  
 poor\_cc\_avg\_food1 == 1 & poor\_cc\_avg == 0, 1, 0),  
 NAM\_1 = as\_factor(marz)) %>%   
 mutate(NAM\_1 = if\_else(NAM\_1 == "VayotsDzor", "Vayots Dzor", NAM\_1)) %>%   
 mutate(NAM\_1 = if\_else(NAM\_1 == "Sjunik", "Syunik", NAM\_1)) %>%   
 select(NAM\_1, poor\_Avpovln2022, poor\_cc\_avg, poor\_cc\_max,  
 poor\_cc\_avg\_food1, poor\_cc\_avg\_food2, new\_poor\_food\_base,  
 new\_poor\_food\_dryhot, weight, hhsize)  
  
fp <-foodpoor %>%   
 group\_by(NAM\_1) %>%   
 summarise(new\_poor = round(sum(new\_poor\_food\_dryhot \* weight\*hhsize, na.rm = TRUE))) %>%   
 mutate(label = paste0(NAM\_1," (", new\_poor, ")"))  
  
  
#write.table(fp, "clipboard", sep="\t", row.names=FALSE)  
fp\_map <- adm1 |>   
 left\_join(fp, join\_by(NAM\_1 == NAM\_1))  
  
fp\_map <-tm\_shape(fp\_map)+  
 tm\_polygons("new\_poor", legend.show = FALSE) +  
 tm\_text("label", size = .7, col = "black")+  
 tm\_layout(legend.position = c("right", "top"),   
 title= "Additional Poor Dry-Hot Scenario",   
 title.position = c('left', 'bottom'),  
 title.size = 0.9)  
fp\_map



# We check that our reweighting was successful  
table <- pp\_microsim\_cc %>%   
 group\_by(lmarket,sector\_w) %>%   
 summarise(total\_pp = sum(weightvec, na.rm = TRUE)) %>%  
 ungroup()

`summarise()` has grouped output by 'lmarket'. You can override using the  
`.groups` argument.

table %>%   
 gt() %>%   
 fmt\_number(columns = total\_pp, decimals = 0)

| lmarket | sector\_w | total\_pp |
| --- | --- | --- |
| 1 | 1 | 380,038 |
| 2 | 2 | 139,999 |
| 3 | 3 | 739,955 |
| 4 | NA | 700,562 |
| 5 | NA | 873,436 |

write.table(  
 table,   
 "clipboard", sep="\t", row.names=FALSE  
 )

## 9 References

ARMSTAT. (2023). *Integrated Living Conditions Survey 2022*.