Armenia CCDR Microsimulation

Renato Vargas

Table of contents

## 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. It is part of a larger project with all contributions to Armenia’s CCDR, which can be [downloaded from GitHub](https://github.com/rvconsulting/armenia-ccdr) in the form of an [Rstudio project](https://support.posit.co/hc/en-us/articles/200526207-Using-RStudio-Projects).

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.

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(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(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

## 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).

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

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/ARM\_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 Armenian. The following dataset is a lookup table with the English names.

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.

# 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))  
# We rename with the survey designation  
names(adm1)[2] <- "hh\_02"

And we plot it for reference (see [Figure 1](#fig-map-example)). This is done with the tmap R package and the code shown in [Listing 1](#lst-map-example).

|  |
| --- |
| Listing 1: Plotting a map with with the tmap package  tm\_shape(adm1)+  tm\_polygons("NAM\_1", legend.show = FALSE) +  tm\_text("NAM\_1", size = 3/4) |

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

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.

|  |
| --- |
| Listing 2: Marz name from geodata  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) |

We also have an Excel file with changes to labor productivity due to climate variability. We bind together the datasets found in each Excel sheet.

|  |
| --- |
| Listing 3: Import labor productivity data  file <- r"(data/ARM-Microsimulation/LaborProductivityChanges.xlsx)" sheets <- excel\_sheets(file)  # Use lapply to read and process each sheet labor\_productivity <- 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 labor\_productivity <- bind\_rows(labor\_productivity) |

Finally, we have our climate vulnerability information. For this analysis we only use the crops\_productivity and livestock\_productivity that comes from *Estimating the Economic Impacts of Climate Change in Armenia* (Strzepek, Boehlert, Castillo, & Smet, 2024).

|  |
| --- |
| Listing 4: Import crops and livestock yield loss data  crops\_productivity <-   read.csv("data/ARM-Vulnerability-Analysis/ARM\_crops\_combined\_REF\_shock\_admin1.csv") %>%   rename(NAM\_1 = Province) livestock\_productivity <-  read.csv(  "data/ARM-Vulnerability-Analysis/ARM\_livestock\_REF\_shock\_admin1.csv"  ) %>%   rename(NAM\_1 = Province) |

## 4 Data preparation income outliers and missings

### 4.1 Household consumption aggregates and characteristics

Initial necessary variables. Some of these choices come from conventions used in the guidance code. However, where possible, we’ve tried to continue using the names found in the datasets to avoid introducing more confusion.

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. In the end, this variable was never used.

zone\_data <- hh %>%   
 select(interview\_\_key, hh\_01\_code, hh\_02, hh\_03, NAM\_1) %>%   
 rename(  
 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. 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 %>%  
 rename(hhid = interview\_\_key) %>%  
 left\_join(zone\_data, join\_by(hhid == hhid)) %>%  
 mutate(  
 # Demographic characteristics  
 # Unique person id  
 pid = paste0(hhid, "-", str\_pad(mem\_001\_\_id, 2, pad = "0")),  
 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, # 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 <= 2 ~ "None - General",  
 educy > 3 & educy <= 7 ~ "Secondary - Vocational",  
 educy > 7 & educy <= 11 ~ "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 5](#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 5: Employed in household  pp\_microsim <- pp\_microsim %>%   mutate(employed = (lstatus == 1)) %>%   group\_by(hhid) %>%   # 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 Armenia, If it were, you should add a mutate() statement like the ones below for each subcategory in [Listing 6](#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 6: 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 7](#lst-employed-no-income) below.

|  |
| --- |
| Listing 7: 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.")  ) |

[1] "There is 28.57% 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 8](#lst-sector-aggregation).

|  |
| --- |
| Listing 8: 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 9](#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 9: Assign sector to those who don’t report one  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() |

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 10: 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) |

### 4.3 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 11](#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.

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 11: 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(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) %>%  
 summarize(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) %>%  
 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  
 )  
 }  
}  
  
# 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)  
scenario\_sheets <- sheets[c(1,2,3)]  
  
# Define the names of the scenarios and the variants  
# modify list with the tab numbers in the Excel file  
scenarios <- scenario\_sheets %>%  
 # Convert all text to lowercase  
 str\_to\_lower() %>%   
 # Replace all spaces and hyphens with underscores  
 str\_replace\_all("[ -]", "\_") %>%  
 # Remove the word 'scenario' or 'scenarios'  
 str\_remove\_all("scenario?s?") %>%  
 # Remove leading and trailing underscores  
 str\_replace\_all("^\_+|\_+$", "")   
  
# 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 = scenario\_sheets[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:  
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) %>%   
 summarize(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)  
  
# write.table(macro\_data, "clipboard", sep="\t", row.names=FALSE)

## 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.

# We join several cohorts from 0 to 29 years old and from  
# 60 onwards, because the reweighting procedure works  
# best if each category is at least 5% of the population  
# The solution here works best for Armenia.  
  
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"))) %>%  
  
 # 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")))  
  
# 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  
# We use the "Medium" variant = variants[7]  
pop\_targets\_2030 <- population\_projections %>%   
 filter(year == 2030, Variant == variants[7]) %>%   
 group\_by(cohort\_code, cohort\_short) %>%   
 summarize(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)  
  
#writeClipboard(pop\_targets\_2030)  
# write.table(pop\_targets\_2030, "clipboard", sep="\t", row.names=FALSE)

And economic targets from our macroeconomic scenario data. We deal with this later. Should come back to fix this so we can automate.

# economic\_targets\_2030 <- macro\_data %>%  
# filter(year == 2030, Variant == "Medium", scenario\_id == "baseline") %>%  
# summarize(  
# 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  
   
 ))

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\_baseline <- c(0.1342, 0.0494, 0.2611, 0.2473, 0.3080)  
lmarket\_dry\_hot <- c(0.1369, 0.0489, 0.2593, 0.2473, 0.3076)  
lmarket\_nzs <- c(0.1251, 0.0516, 0.2623, 0.2516, 0.3094)  
# Note how similar the scenarios are  
  
# Target list baseline  
targets\_baseline <- list(gender\_code  
 , cohort\_code  
 , lmarket\_baseline  
 )  
  
names(targets\_baseline) <- c("gender\_code",   
 "cohort\_code",   
 "lmarket"  
 )  
  
# Target list Dry/Hot  
targets\_dry\_hot <- list(gender\_code  
 , cohort\_code  
 , lmarket\_dry\_hot  
 )  
  
names(targets\_dry\_hot) <- c("gender\_code",   
 "cohort\_code",   
 "lmarket"  
 )  
  
# Target list NZS  
targets\_nzs <- list(gender\_code  
 , cohort\_code  
 , lmarket\_nzs  
 )  
  
names(targets\_nzs) <- c("gender\_code",   
 "cohort\_code",   
 "lmarket"  
 )

And now we perform the reweighting, using the original weights. Initially we had used the default option type = “pctlim” combined with pctlim=0.05, because the method recommends that if reweighting changes for one variable according to its target are not of at least 5%, then it’s not worth burdening the procedure with it. It then ignored sex as a reweighting variable, leaving a small percentage difference between the target and the final population. However, we then tried removing this limitation and the procedure reached convergence in 33 iterations very efficiently.

# Since this uses base R, we need to turn the data frame into base R object  
rakedata <- as.data.frame(pp\_microsim)  
  
anesrake::anesrakefinder(targets\_baseline, rakedata, choosemethod = "total")

gender\_code cohort\_code lmarket   
 0.03626510 0.09677000 0.07172418

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

[1] "Raking...Iteration 1"  
[1] "Current iteration changed total weights by 2361.17708851471"  
[1] "Raking...Iteration 2"  
[1] "Current iteration changed total weights by 343.154685460483"  
[1] "Raking...Iteration 3"  
[1] "Current iteration changed total weights by 50.282147123565"  
[1] "Raking...Iteration 4"  
[1] "Current iteration changed total weights by 15.5324150051193"  
[1] "Raking...Iteration 5"  
[1] "Current iteration changed total weights by 4.9125427648284"  
[1] "Raking...Iteration 6"  
[1] "Current iteration changed total weights by 1.55433042098091"  
[1] "Raking...Iteration 7"  
[1] "Current iteration changed total weights by 0.491570475878754"  
[1] "Raking...Iteration 8"  
[1] "Current iteration changed total weights by 0.15548860353216"  
[1] "Raking...Iteration 9"  
[1] "Current iteration changed total weights by 0.0491861537407502"  
[1] "Raking...Iteration 10"  
[1] "Current iteration changed total weights by 0.0155594639669022"  
[1] "Raking...Iteration 11"  
[1] "Current iteration changed total weights by 0.00492206801523239"  
[1] "Raking...Iteration 12"  
[1] "Current iteration changed total weights by 0.0015570434063234"  
[1] "Raking...Iteration 13"  
[1] "Current iteration changed total weights by 0.000492553986909977"  
[1] "Raking...Iteration 14"  
[1] "Current iteration changed total weights by 0.00015581417334512"  
[1] "Raking...Iteration 15"  
[1] "Current iteration changed total weights by 4.92901435997922e-05"  
[1] "Raking...Iteration 16"  
[1] "Current iteration changed total weights by 1.55924099685123e-05"  
[1] "Raking...Iteration 17"  
[1] "Current iteration changed total weights by 4.9324913638793e-06"  
[1] "Raking...Iteration 18"  
[1] "Current iteration changed total weights by 1.56034048766351e-06"  
[1] "Raking...Iteration 19"  
[1] "Current iteration changed total weights by 4.93596604533852e-07"  
[1] "Raking...Iteration 20"  
[1] "Current iteration changed total weights by 1.56144157131832e-07"  
[1] "Raking...Iteration 21"  
[1] "Current iteration changed total weights by 4.93947955038099e-08"  
[1] "Raking...Iteration 22"  
[1] "Current iteration changed total weights by 1.56252071836782e-08"  
[1] "Raking...Iteration 23"  
[1] "Current iteration changed total weights by 4.94344779000677e-09"  
[1] "Raking...Iteration 24"  
[1] "Current iteration changed total weights by 1.56241772297783e-09"  
[1] "Raking...Iteration 25"  
[1] "Current iteration changed total weights by 4.95822036206128e-10"  
[1] "Raking...Iteration 26"  
[1] "Current iteration changed total weights by 1.56211973911802e-10"  
[1] "Raking...Iteration 27"  
[1] "Current iteration changed total weights by 5.09479958221704e-11"  
[1] "Raking...Iteration 28"  
[1] "Current iteration changed total weights by 1.42659634105868e-11"  
[1] "Raking...Iteration 29"  
[1] "Current iteration changed total weights by 6.37030705963326e-12"  
[1] "Raking...Iteration 30"  
[1] "Current iteration changed total weights by 2.28768393117917e-12"  
[1] "Raking...Iteration 31"  
[1] "Current iteration changed total weights by 2.2706836411146e-12"  
[1] "Raking converged in 31 iterations"

summary(outsave)

$convergence  
[1] "Complete convergence was achieved after 31 iterations"  
  
$base.weights  
[1] "Using Base Weights Provided"  
  
$raking.variables  
[1] "gender\_code" "cohort\_code" "lmarket"   
  
$weight.summary  
 Min. 1st Qu. Median Mean 3rd Qu. Max.   
 0.1034 0.6700 0.9288 1.0000 1.2682 3.3858   
  
$selection.method  
[1] "variable selection conducted using \_nolim\_ - discrepancies selected using \_total\_."  
  
$general.design.effect  
[1] 1.244831  
  
$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> 0.1342 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> 0.2473 NA NA NA NA NA NA  
<NA> 0.3080 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$weight\_2030\_baseline <- unlist(outsave[1])  
n <- length(rakedata$sector)  
  
# Calculate the sum of original weights  
original\_weight\_sum <- sum(rakedata$weight)  
  
# # Target scaling for original weights  
  
original\_weight\_scaling\_factor <-  
 pop\_data$total\_population[pop\_data$year == 2030] /  
 pop\_data$total\_population[pop\_data$year == 2022]  
  
# 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$weight\_2030\_baseline)  
  
# Scale the new weights to match the sum of the original weights  
scaling\_factor <- original\_weight\_sum / new\_weight\_sum  
rakedata$weight\_2030\_baseline <- rakedata$weight\_2030\_baseline \* scaling\_factor  
  
# Verify the adjustment  
head(rakedata[, c("weight", "weight\_2030\_baseline")])

weight weight\_2030\_baseline  
1 185.7685 175.0700  
2 185.7685 188.2168  
3 122.7176 101.5832  
4 185.7685 191.8185  
5 326.8796 264.5755  
6 326.8796 337.5253

summary(rakedata$weight\_2030\_baseline)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 15.87 102.86 142.59 153.52 194.69 519.78

summary(rakedata$weight)

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

hh\_size <- rakedata %>%   
 select(hhid, hhsize) %>%   
 mutate(ones = 1,  
 hhsize\_old = hhsize) %>%   
 group\_by(hhid) %>%   
 summarize(hhsize = sum(ones, na.rm = TRUE)) %>%   
 ungroup()  
  
rakedata <- rakedata %>%  
 rename(hhsize\_old = hhsize) %>%   
 left\_join(hh\_size, join\_by(hhid)) %>%   
 relocate(weight, .before = weight\_2030\_baseline) %>%   
 mutate(hh\_weight\_2030\_baseline = weight\_2030\_baseline / hhsize)  
  
pp\_microsim <- tibble(rakedata)  
rm(rakedata)

We now do the Dry/Hot Scenario. The efficient way of doing this is through a loop or sapply, but as we’re strapped for time we will just repeat the code. (Needs rework.)

# Since this uses base R, we need to turn the data frame into base R object  
rakedata <- as.data.frame(pp\_microsim)  
  
anesrake::anesrakefinder(targets\_dry\_hot, rakedata, choosemethod = "total")

gender\_code cohort\_code lmarket   
 0.03626510 0.09677000 0.06632418

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

[1] "Raking...Iteration 1"  
[1] "Current iteration changed total weights by 2346.39562345486"  
[1] "Raking...Iteration 2"  
[1] "Current iteration changed total weights by 325.326676946962"  
[1] "Raking...Iteration 3"  
[1] "Current iteration changed total weights by 47.6038687445994"  
[1] "Raking...Iteration 4"  
[1] "Current iteration changed total weights by 14.7805325726215"  
[1] "Raking...Iteration 5"  
[1] "Current iteration changed total weights by 4.67536358935004"  
[1] "Raking...Iteration 6"  
[1] "Current iteration changed total weights by 1.48066518168051"  
[1] "Raking...Iteration 7"  
[1] "Current iteration changed total weights by 0.468649510562274"  
[1] "Raking...Iteration 8"  
[1] "Current iteration changed total weights by 0.148353295202568"  
[1] "Raking...Iteration 9"  
[1] "Current iteration changed total weights by 0.046965177901018"  
[1] "Raking...Iteration 10"  
[1] "Current iteration changed total weights by 0.0148683286004776"  
[1] "Raking...Iteration 11"  
[1] "Current iteration changed total weights by 0.00470705840641751"  
[1] "Raking...Iteration 12"  
[1] "Current iteration changed total weights by 0.00149017464214521"  
[1] "Raking...Iteration 13"  
[1] "Current iteration changed total weights by 0.000471763960322724"  
[1] "Raking...Iteration 14"  
[1] "Current iteration changed total weights by 0.000149352453236176"  
[1] "Raking...Iteration 15"  
[1] "Current iteration changed total weights by 4.72824469837235e-05"  
[1] "Raking...Iteration 16"  
[1] "Current iteration changed total weights by 1.49688183763291e-05"  
[1] "Raking...Iteration 17"  
[1] "Current iteration changed total weights by 4.7388742776544e-06"  
[1] "Raking...Iteration 18"  
[1] "Current iteration changed total weights by 1.50024780687374e-06"  
[1] "Raking...Iteration 19"  
[1] "Current iteration changed total weights by 4.74952141560347e-07"  
[1] "Raking...Iteration 20"  
[1] "Current iteration changed total weights by 1.50361528755694e-07"  
[1] "Raking...Iteration 21"  
[1] "Current iteration changed total weights by 4.76025212442499e-08"  
[1] "Raking...Iteration 22"  
[1] "Current iteration changed total weights by 1.50707414187101e-08"  
[1] "Raking...Iteration 23"  
[1] "Current iteration changed total weights by 4.77042699786878e-09"  
[1] "Raking...Iteration 24"  
[1] "Current iteration changed total weights by 1.50992897351987e-09"  
[1] "Raking...Iteration 25"  
[1] "Current iteration changed total weights by 4.77459433101401e-10"  
[1] "Raking...Iteration 26"  
[1] "Current iteration changed total weights by 1.5339025882799e-10"  
[1] "Raking...Iteration 27"  
[1] "Current iteration changed total weights by 4.58121596214056e-11"  
[1] "Raking...Iteration 28"  
[1] "Current iteration changed total weights by 1.52340223769087e-11"  
[1] "Raking...Iteration 29"  
[1] "Current iteration changed total weights by 6.39277519809411e-12"  
[1] "Raking...Iteration 30"  
[1] "Current iteration changed total weights by 1.47665213390269e-12"  
[1] "Raking...Iteration 31"  
[1] "Current iteration changed total weights by 4.05270261794044e-12"  
[1] "Raking converged in 31 iterations"

summary(outsave)

$convergence  
[1] "Complete convergence was achieved after 31 iterations"  
  
$base.weights  
[1] "Using Base Weights Provided"  
  
$raking.variables  
[1] "gender\_code" "cohort\_code" "lmarket"   
  
$weight.summary  
 Min. 1st Qu. Median Mean 3rd Qu. Max.   
 0.1032 0.6669 0.9316 1.0000 1.2685 3.3775   
  
$selection.method  
[1] "variable selection conducted using \_nolim\_ - discrepancies selected using \_total\_."  
  
$general.design.effect  
[1] 1.243531  
  
$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> 0.1369 NA NA NA NA NA NA  
<NA> 0.0489 NA NA NA NA NA NA  
<NA> 0.2593 NA NA NA NA NA NA  
<NA> 0.2473 NA NA NA NA NA NA  
<NA> 0.3076 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$weight\_2030\_dry\_hot <- unlist(outsave[1])  
  
# Calculate the sum of original weights  
original\_weight\_sum <- sum(rakedata$weight)  
  
# Target scaling for original weights  
  
original\_weight\_scaling\_factor <-  
 pop\_data$total\_population[pop\_data$year == 2030] /  
 pop\_data$total\_population[pop\_data$year == 2022]  
  
# 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$weight\_2030\_dry\_hot)  
  
# Scale the new weights to match the sum of the original weights  
scaling\_factor <- original\_weight\_sum / new\_weight\_sum  
rakedata$weight\_2030\_dry\_hot <- rakedata$weight\_2030\_dry\_hot \* scaling\_factor  
  
# Verify the adjustment  
head(rakedata[, c("weight", "weight\_2030\_dry\_hot")])

weight weight\_2030\_dry\_hot  
1 185.7685 174.1517  
2 185.7685 188.2564  
3 122.7176 101.8126  
4 185.7685 190.5865  
5 326.8796 263.8479  
6 326.8796 335.3575

summary(rakedata$weight\_2030\_dry\_hot)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 15.84 102.38 143.01 153.52 194.73 518.52

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 %>%   
 mutate(hh\_weight\_2030\_dry\_hot = weight\_2030\_dry\_hot / hhsize)  
  
pp\_microsim <- tibble(rakedata)  
rm(rakedata)

Let’s add the NZS scenario

# Since this uses base R, we need to turn the data frame into base R object  
rakedata <- as.data.frame(pp\_microsim)  
  
anesrake::anesrakefinder(targets\_nzs, rakedata, choosemethod = "total")

gender\_code cohort\_code lmarket   
 0.03626510 0.09677000 0.08132418

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

[1] "Raking...Iteration 1"  
[1] "Current iteration changed total weights by 2405.38337714935"  
[1] "Raking...Iteration 2"  
[1] "Current iteration changed total weights by 381.11433942213"  
[1] "Raking...Iteration 3"  
[1] "Current iteration changed total weights by 66.9715000528318"  
[1] "Raking...Iteration 4"  
[1] "Current iteration changed total weights by 20.6776651602141"  
[1] "Raking...Iteration 5"  
[1] "Current iteration changed total weights by 6.53725119305366"  
[1] "Raking...Iteration 6"  
[1] "Current iteration changed total weights by 2.0599945856778"  
[1] "Raking...Iteration 7"  
[1] "Current iteration changed total weights by 0.649111380457645"  
[1] "Raking...Iteration 8"  
[1] "Current iteration changed total weights by 0.204577406691525"  
[1] "Raking...Iteration 9"  
[1] "Current iteration changed total weights by 0.0644801211551565"  
[1] "Raking...Iteration 10"  
[1] "Current iteration changed total weights by 0.0203235937837268"  
[1] "Raking...Iteration 11"  
[1] "Current iteration changed total weights by 0.00640584165249312"  
[1] "Raking...Iteration 12"  
[1] "Current iteration changed total weights by 0.00201907300818058"  
[1] "Raking...Iteration 13"  
[1] "Current iteration changed total weights by 0.000636396628444766"  
[1] "Raking...Iteration 14"  
[1] "Current iteration changed total weights by 0.000200587436619007"  
[1] "Raking...Iteration 15"  
[1] "Current iteration changed total weights by 6.3223652922148e-05"  
[1] "Raking...Iteration 16"  
[1] "Current iteration changed total weights by 1.99276214380567e-05"  
[1] "Raking...Iteration 17"  
[1] "Current iteration changed total weights by 6.28103589818407e-06"  
[1] "Raking...Iteration 18"  
[1] "Current iteration changed total weights by 1.97973583832001e-06"  
[1] "Raking...Iteration 19"  
[1] "Current iteration changed total weights by 6.23997950852107e-07"  
[1] "Raking...Iteration 20"  
[1] "Current iteration changed total weights by 1.96680178823905e-07"  
[1] "Raking...Iteration 21"  
[1] "Current iteration changed total weights by 6.19911701138509e-08"  
[1] "Raking...Iteration 22"  
[1] "Current iteration changed total weights by 1.95397621555182e-08"  
[1] "Raking...Iteration 23"  
[1] "Current iteration changed total weights by 6.15833799233467e-09"  
[1] "Raking...Iteration 24"  
[1] "Current iteration changed total weights by 1.94083731031025e-09"  
[1] "Raking...Iteration 25"  
[1] "Current iteration changed total weights by 6.12484576945072e-10"  
[1] "Raking...Iteration 26"  
[1] "Current iteration changed total weights by 1.92554375222365e-10"  
[1] "Raking...Iteration 27"  
[1] "Current iteration changed total weights by 5.94729127056937e-11"  
[1] "Raking...Iteration 28"  
[1] "Current iteration changed total weights by 1.94124161190246e-11"  
[1] "Raking...Iteration 29"  
[1] "Current iteration changed total weights by 6.34514663033769e-12"  
[1] "Raking...Iteration 30"  
[1] "Current iteration changed total weights by 3.0190572264388e-12"  
[1] "Raking...Iteration 31"  
[1] "Current iteration changed total weights by 2.23793206188816e-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] "gender\_code" "cohort\_code" "lmarket"   
  
$weight.summary  
 Min. 1st Qu. Median Mean 3rd Qu. Max.   
 0.1064 0.6667 0.9342 1.0000 1.2586 3.4331   
  
$selection.method  
[1] "variable selection conducted using \_nolim\_ - discrepancies selected using \_total\_."  
  
$general.design.effect  
[1] 1.24885  
  
$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> 0.1251 NA NA NA NA NA NA  
<NA> 0.0516 NA NA NA NA NA NA  
<NA> 0.2623 NA NA NA NA NA NA  
<NA> 0.2516 NA NA NA NA NA NA  
<NA> 0.3094 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$weight\_2030\_nzs <- unlist(outsave[1])  
  
# Calculate the sum of original weights  
original\_weight\_sum <- sum(rakedata$weight)  
  
# Target scaling for original weights  
  
original\_weight\_scaling\_factor <-  
 pop\_data$total\_population[pop\_data$year == 2030] /  
 pop\_data$total\_population[pop\_data$year == 2022]  
  
# 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$weight\_2030\_nzs)  
  
# Scale the new weights to match the sum of the original weights  
scaling\_factor <- original\_weight\_sum / new\_weight\_sum  
rakedata$weight\_2030\_nzs <- rakedata$weight\_2030\_nzs \* scaling\_factor  
  
# Verify the adjustment  
head(rakedata[, c("weight", "weight\_2030\_nzs")])

weight weight\_2030\_nzs  
1 185.7685 175.7182  
2 185.7685 188.4089  
3 122.7176 103.4375  
4 185.7685 192.4717  
5 326.8796 272.2376  
6 326.8796 338.6745

summary(rakedata$weight\_2030\_nzs)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 16.33 102.36 143.42 153.52 193.22 527.05

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 %>%   
 mutate(hh\_weight\_2030\_nzs = weight\_2030\_nzs / hhsize)

Weights for the household database

# We calculate new weights for households in the hh database  
weights\_scenarios <- rakedata %>%   
 group\_by(hhid) %>%  
 summarize(  
 hh\_weight\_2030\_baseline =   
 sum(hh\_weight\_2030\_baseline, na.rm = TRUE),  
 hh\_weight\_2030\_dry\_hot =   
 sum(hh\_weight\_2030\_dry\_hot, na.rm = TRUE),  
 hh\_weight\_2030\_nzs =   
 sum(hh\_weight\_2030\_nzs, na.rm = TRUE)  
 )  
  
# We return rakedata to data frame pp\_microsim and get rid of rakedata  
pp\_microsim <- tibble(rakedata)  
rm(rakedata)

### 7.3 Rescaling labor income according to changes to the wage bill

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

# Wage rescale factor by sector from macro (Agriculture, Manufacturing, Services)  
wrf\_2030\_baseline <- c(1.250520168, 1.336828769, 1.378384149)  
wrf\_2030\_dry\_hot <- c(1.287103700, 1.297391076, 1.343485236)  
wrf\_2030\_nzs <- c(1.054278195, 1.317612666, 1.272275437)  
  
# We check the wage bill by sector  
wages\_by\_sector <- pp\_microsim %>%  
 filter(!is.na(sector\_w)) %>%   
 group\_by(sector\_w, .drop = TRUE) %>%   
 summarize(  
 wages\_2022 = sum(annual\_labor\_total \* weight, na.rm = TRUE),  
 wages\_2030\_baseline =   
 sum(annual\_labor\_total \* weight\_2030\_baseline, na.rm = TRUE),  
 wages2030\_dry\_hot =   
 sum(annual\_labor\_total \* weight\_2030\_dry\_hot, na.rm = TRUE),  
 wages2030\_nzs =   
 sum(annual\_labor\_total \* weight\_2030\_nzs, na.rm = TRUE)  
 )  
  
  
# 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\_baseline = case\_when(  
 sector\_w == 1 ~ wages\_2022 \* wrf\_2030\_baseline[1],  
 sector\_w == 2 ~ wages\_2022 \* wrf\_2030\_baseline[2],  
 sector\_w == 3 ~ wages\_2022 \* wrf\_2030\_baseline[3],  
 .default = NA  
 ),  
 wages\_target2030\_dry\_hot = case\_when(  
 sector\_w == 1 ~ wages\_2022 \* wrf\_2030\_dry\_hot[1],  
 sector\_w == 2 ~ wages\_2022 \* wrf\_2030\_dry\_hot[2],  
 sector\_w == 3 ~ wages\_2022 \* wrf\_2030\_dry\_hot[3],  
 .default = NA  
 ),  
 wages\_target2030\_nzs = case\_when(  
 sector\_w == 1 ~ wages\_2022 \* wrf\_2030\_nzs[1],  
 sector\_w == 2 ~ wages\_2022 \* wrf\_2030\_nzs[2],  
 sector\_w == 3 ~ wages\_2022 \* wrf\_2030\_nzs[3],  
 .default = NA  
 ),  
 wtc\_2030\_baseline = wages\_target\_2030\_baseline / wages\_2030\_baseline,  
 wtc\_2030\_dry\_hot = wages\_target2030\_dry\_hot / wages2030\_dry\_hot,  
 wtc\_2030\_nzs = wages\_target2030\_nzs / wages2030\_nzs  
 )   
  
# wages\_by\_sector %>%  
# gt()  
# write.table(wages\_by\_sector, "clipboard", sep="\t", row.names=FALSE)

We then add the coefficient to rescale each wage by sector

# Assign rescale the annual and monthly wage depending on the sector  
# Quick way, but needs to be put in a sapply statement or loop  
pp\_microsim <- pp\_microsim %>%   
 rename(monthly\_labor\_income\_2022 = monthly\_labor\_income,  
 annual\_labor\_total\_2022 = annual\_labor\_total) %>%   
 mutate(  
 monthly\_labor\_income\_2030\_baseline = case\_when(  
 sector\_w == 1 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_baseline[1],  
 sector\_w == 2 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_baseline[2],  
 sector\_w == 3 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_baseline[3],  
 TRUE ~ NA  
 ),  
 annual\_labor\_total\_2030\_baseline = case\_when(  
 sector\_w == 1 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_baseline[1],  
 sector\_w == 2 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_baseline[2],  
 sector\_w == 3 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_baseline[3],  
 TRUE ~ NA  
 ),  
 monthly\_labor\_income\_2030\_dry\_hot = case\_when(  
 sector\_w == 1 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_dry\_hot[1],  
 sector\_w == 2 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_dry\_hot[2],  
 sector\_w == 3 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_dry\_hot[3],  
 TRUE ~ NA  
 ),  
 annual\_labor\_total\_2030\_dry\_hot = case\_when(  
 sector\_w == 1 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_dry\_hot[1],  
 sector\_w == 2 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_dry\_hot[2],  
 sector\_w == 3 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_dry\_hot[3],  
 TRUE ~ NA  
 ),  
 monthly\_labor\_income\_2030\_nzs = case\_when(  
 sector\_w == 1 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_nzs[1],  
 sector\_w == 2 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_nzs[2],  
 sector\_w == 3 ~ monthly\_labor\_income\_2022 \* wages\_by\_sector$wtc\_2030\_nzs[3],  
 TRUE ~ NA  
 ),  
 annual\_labor\_total\_2030\_nzs = case\_when(  
 sector\_w == 1 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_nzs[1],  
 sector\_w == 2 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_nzs[2],  
 sector\_w == 3 ~ annual\_labor\_total\_2022 \* wages\_by\_sector$wtc\_2030\_nzs[3],  
 TRUE ~ NA  
 )  
 )  
  
# This takes care of different household members coming from different sectors  
hh\_li <- pp\_microsim %>%   
 group\_by(hhid) %>%   
 summarize(mli\_2022 = sum(monthly\_labor\_income\_2022, na.rm = TRUE),  
 mli\_2030\_baseline = sum(monthly\_labor\_income\_2030\_baseline, na.rm = TRUE),  
 mli\_2030\_dry\_hot = sum(monthly\_labor\_income\_2030\_dry\_hot, na.rm = TRUE),  
 mli\_2030\_nzs = sum(monthly\_labor\_income\_2030\_nzs, na.rm = TRUE),  
 mli\_coef\_2030\_baseline = if\_else(mli\_2022 == 0, 1, mli\_2030\_baseline / mli\_2022),  
 mli\_coef\_2030\_dry\_hot = if\_else(mli\_2022 == 0, 1, mli\_2030\_dry\_hot / mli\_2022),  
 mli\_coef\_2030\_nzs = if\_else(mli\_2022 == 0, 1, mli\_2030\_nzs / mli\_2022)  
 )%>%   
 select(hhid,  
 mli\_2022,  
 mli\_2030\_baseline,  
 mli\_2030\_dry\_hot,  
 mli\_2030\_nzs,  
 mli\_coef\_2030\_baseline,   
 mli\_coef\_2030\_dry\_hot,  
 mli\_coef\_2030\_nzs)  
   
ic\_microsim <- ic %>%   
 left\_join(hh\_li, join\_by(interview\_\_key == hhid)) %>%  
 left\_join(weights\_scenarios, join\_by(interview\_\_key == hhid)) %>%   
 rename(inc2\_2022 = inc2,  
 inc3\_2022 = inc3,  
 totalinc\_2022 = totalinc) %>%   
 mutate(  
 mli\_coef\_2030\_baseline =   
 if\_else(  
 is.na(mli\_coef\_2030\_baseline), 1,mli\_coef\_2030\_baseline),  
 mli\_coef\_2030\_dry\_hot =   
 if\_else(  
 is.na(mli\_coef\_2030\_dry\_hot), 1,mli\_coef\_2030\_dry\_hot),  
 mli\_coef\_2030\_nzs =   
 if\_else(  
 is.na(mli\_coef\_2030\_nzs), 1,mli\_coef\_2030\_nzs)  
 ) %>%   
 mutate(  
 inc2\_2030\_baseline = inc2\_2022 \* mli\_coef\_2030\_baseline,  
 inc3\_2030\_baseline = inc3\_2022 \* mli\_coef\_2030\_baseline,  
 inc2\_2030\_dry\_hot = inc2\_2022 \* mli\_coef\_2030\_dry\_hot,  
 inc3\_2030\_dry\_hot = inc3\_2022 \* mli\_coef\_2030\_dry\_hot,  
 inc2\_2030\_nzs = inc2\_2022 \* mli\_coef\_2030\_nzs,  
 inc3\_2030\_nzs = inc3\_2022 \* mli\_coef\_2030\_nzs  
 ) %>%   
 mutate(  
 totalinc\_2030\_baseline =   
 totalinc\_2022 - coalesce(inc2\_2022,0) - coalesce(inc3\_2022,0) +   
 coalesce(inc2\_2030\_baseline,0) + coalesce(inc3\_2030\_baseline,0),  
 totalinc\_2030\_dry\_hot =   
 totalinc\_2022 - coalesce(inc2\_2022,0) - coalesce(inc3\_2022,0) +   
 coalesce(inc2\_2030\_dry\_hot,0) + coalesce(inc3\_2030\_dry\_hot,0),  
 totalinc\_2030\_nzs\_noctr =   
 totalinc\_2022 - coalesce(inc2\_2022,0) - coalesce(inc3\_2022,0) +   
 coalesce(inc2\_2030\_nzs,0) + coalesce(inc3\_2030\_nzs,0)  
 )   
  
# Calculate quantiles and create 'breaks'  
breaks <- Hmisc::wtd.quantile(ic\_microsim$totalinc\_2030\_nzs\_noctr,   
 weights = ic\_microsim$hh\_weight\_2030\_nzs,   
 probs = seq(0.1, 0.9, 0.1))  
  
# Assign decile groups directly without creating an intermediate income\_decile column  
ic\_microsim <- ic\_microsim %>%  
 mutate(totalinc\_2030\_nzs\_noctr =  
 if\_else(is.na(totalinc\_2030\_nzs\_noctr), 0,totalinc\_2030\_nzs\_noctr)) %>%   
 mutate(income\_decile\_group = cut(totalinc\_2030\_nzs\_noctr,   
 breaks = c(-Inf, breaks, Inf),  
 labels = 1:10,   
 include.lowest = TRUE))  
  
income\_decile\_group <- ic\_microsim %>%   
 select(interview\_\_key, income\_decile\_group)

ic\_microsim <- ic\_microsim %>%   
 mutate(  
 totinc\_coef\_2030\_baseline =   
 if\_else(  
 totalinc\_2022 == 0,   
 1,   
 totalinc\_2030\_baseline / totalinc\_2022),  
 totinc\_coef\_2030\_dry\_hot =   
 if\_else(  
 totalinc\_2022 == 0,   
 1,   
 totalinc\_2030\_dry\_hot / totalinc\_2022),  
 totinc\_coef\_2030\_nzs\_noctr =   
 if\_else(  
 totalinc\_2022 == 0,   
 1,   
 totalinc\_2030\_nzs\_noctr / totalinc\_2022)  
 ) %>%   
 mutate(  
 totinc\_coef\_2030\_baseline =   
 if\_else(  
 is.na(totinc\_coef\_2030\_baseline),   
 1,   
 totinc\_coef\_2030\_baseline),  
 totinc\_coef\_2030\_dry\_hot =   
 if\_else(  
 is.na(totinc\_coef\_2030\_dry\_hot),   
 1,   
 totinc\_coef\_2030\_dry\_hot),  
 totinc\_coef\_2030\_nzs\_noctr =   
 if\_else(  
 is.na(totinc\_coef\_2030\_nzs\_noctr),   
 1,   
 totinc\_coef\_2030\_nzs\_noctr)  
 )  
  
  
ic\_coef\_scenarios <- ic\_microsim %>%  
 select(  
 interview\_\_key,   
 totinc\_coef\_2030\_baseline,   
 totinc\_coef\_2030\_dry\_hot,  
 totinc\_coef\_2030\_nzs\_noctr  
 )

We check that our reweighting was successful

# table <- pp\_microsim %>%  
# group\_by(cohort) %>%  
# # group\_by(lmarket) %>%  
# # group\_by(gender) %>%  
# summarize(no\_weight = sum(n(), na.rm = TRUE),  
# total\_pp = sum(weight, na.rm = TRUE)) %>%  
# ungroup()  
#   
# table %>%  
# gt() %>%  
# fmt\_number(columns = total\_pp, decimals = 0)  
#   
# write.table(  
# table,  
# "clipboard", sep="\t", row.names=FALSE  
# )

## 8 Microsimulation

We now implement different shocks according to various scenarios.

### 8.1 Macro scenarios without additional impacts

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\_scenarios, join\_by(hhid == hhid)) %>%   
 left\_join(ic\_coef\_scenarios, join\_by(hhid == interview\_\_key)) %>%   
 # We adjust total consumption by the income coefficient  
 rename(  
 totc\_2022 = totc,  
 poor\_Avpovln2022\_2022 = poor\_Avpovln2022  
 ) %>%   
 mutate(  
 totc\_2030\_baseline = totc\_2022 \* totinc\_coef\_2030\_baseline,  
 totc\_2030\_dry\_hot = totc\_2022 \* totinc\_coef\_2030\_dry\_hot,  
 totc\_2030\_nzs\_noctr = totc\_2022 \* totinc\_coef\_2030\_nzs\_noctr  
 )

Here we add back the tax revenue

Add back carbon tax revenue. We estimated weighted income deciles above and mapped number of households by decile. We divided the revenue by that number and we add that amount by the first four income deciles in this manner. Again, another one calculated in Excel because of time constraints. Needs fixing to make automatic.

ca\_microsim <- ca\_microsim %>%   
 left\_join(income\_decile\_group, join\_by(hhid == interview\_\_key)) %>%   
 mutate(  
 totc\_2030\_nzs =   
 # Urban 40% and Rural 60%  
 case\_when(  
 # Urban  
 income\_decile\_group == 1 & urb\_rur == 1 ~   
 totc\_2030\_nzs\_noctr + (1546.29 \* hhsize),  
 income\_decile\_group == 2 & urb\_rur == 1 ~   
 totc\_2030\_nzs\_noctr + (1610.65 \* hhsize),  
 income\_decile\_group == 3 & urb\_rur == 1 ~   
 totc\_2030\_nzs\_noctr + (1251.14 \* hhsize),  
 income\_decile\_group == 4 & urb\_rur == 1 ~   
 totc\_2030\_nzs\_noctr + (1134.21 \* hhsize),  
 # Rural  
 income\_decile\_group == 1 & urb\_rur == 2 ~   
 totc\_2030\_nzs\_noctr + (6323.09 \* hhsize),  
 income\_decile\_group == 2 & urb\_rur == 2 ~   
 totc\_2030\_nzs\_noctr + (6109.07 \* hhsize),  
 income\_decile\_group == 3 & urb\_rur == 2 ~   
 totc\_2030\_nzs\_noctr + (5599.16 \* hhsize),  
 income\_decile\_group == 4 & urb\_rur == 2 ~   
 totc\_2030\_nzs\_noctr + (4420.53 \* hhsize),  
 .default = totc\_2030\_nzs\_noctr  
 )  
 )

And recalculate poverty.

ca\_microsim <- ca\_microsim %>%   
 rename(  
 aec\_r\_2022 = aec\_r,  
 weight\_2022 = weight,  
 weight\_2030\_baseline = hh\_weight\_2030\_baseline,  
 weight\_2030\_dry\_hot = hh\_weight\_2030\_dry\_hot,  
 weight\_2030\_nzs = hh\_weight\_2030\_nzs  
 ) %>%   
 mutate(  
 aec\_r\_2030\_baseline =   
 totc\_2030\_baseline / ae\_r / PI,  
 aec\_r\_2030\_dry\_hot =   
 totc\_2030\_dry\_hot / ae\_r / PI,  
 aec\_r\_2030\_nzs =   
 totc\_2030\_nzs / ae\_r / PI  
 ) %>%   
 # Official poverty line  
 mutate(  
 poor\_Avpovln2022\_2030\_baseline =   
 if\_else(aec\_r\_2030\_baseline < 52883, 1, 0),  
 poor\_Avpovln2022\_2030\_dry\_hot =   
 if\_else(aec\_r\_2030\_dry\_hot < 52883, 1, 0),  
 poor\_Avpovln2022\_2030\_nzs =   
 if\_else(aec\_r\_2030\_nzs < 52883, 1, 0)  
 )

Test

test\_baseline <- ca\_microsim %>%  
 rename(  
 poor\_original = poor\_Avpovln2022\_2022,  
 poor\_2030\_baseline = poor\_Avpovln2022\_2030\_baseline,  
 poor\_2030\_dry\_hot = poor\_Avpovln2022\_2030\_dry\_hot,  
 poor\_2030\_nzs = poor\_Avpovln2022\_2030\_nzs  
 ) %>%  
 group\_by(poor\_original) %>%   
 summarize(  
 no\_hh\_2022 = sum(weight\_2022, na.rm = TRUE),  
 no\_pp\_2022 = sum(weight\_2022 \* hhsize, na.rm = TRUE),  
 # no\_hh\_baseline = sum(weight\_2030\_baseline, na.rm = TRUE),  
 # no\_pp\_baseline = sum(weight\_2030\_baseline \* hhsize, na.rm = TRUE),  
 # no\_hh\_dry\_hot = sum(weight\_2030\_dry\_hot, na.rm = TRUE),  
 # no\_pp\_dry\_hot = sum(weight\_2030\_dry\_hot \* hhsize, na.rm = TRUE),  
 # no\_hh\_nzs = sum(weight\_2030\_nzs, na.rm = TRUE),  
 # no\_pp\_nzs = sum(weight\_2030\_nzs \* hhsize, na.rm = TRUE)  
 ) %>%   
 ungroup()  
  
test\_baseline %>%   
 gt()

| Poor, Avpovln2022 | no\_hh\_2022 | no\_pp\_2022 |
| --- | --- | --- |
| 0 | 659638.8 | 2149568.6 |
| 1 | 143688.2 | 704957.6 |

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

Let’s plot the distributions.

# # Basic density plot comparing equivalized consumption per capita  
# ggplot(ca\_microsim,   
# aes(x = aec\_r\_2030\_nzs, fill = 'NZS')) +  
# geom\_density(alpha = 0.5) +  
# # geom\_density(  
# # data = ca\_microsim\_cc,  
# # aes(x = aec\_r\_2030\_dry\_hot, fill = 'Dry/Hot'),  
# # alpha = 0.5) +  
# # geom\_density(  
# # data = ca\_microsim\_cc,  
# # aes(x = aec\_r\_2030\_dry\_hot\_food2, fill = 'Dry/Hot + Food Price'),  
# # alpha = 0.5) +  
# geom\_density(  
# data = ca\_microsim,  
# aes(x = aec\_r\_2030\_dry\_hot, fill = 'Dry/Hot'),  
# alpha = 0.5) +  
# geom\_density(  
# data = ca\_microsim,  
# aes(x = aec\_r\_2030\_baseline, fill = 'Baseline'),  
# alpha = 0.5) +  
# labs(  
# fill = "Scenario Variant",   
# # title = "Comparison of Consumption Distributions",   
# x = "Equivalized consumption (Dram)",   
# y = "Probability") +  
# theme\_minimal()+  
# coord\_cartesian(xlim = c(0, 500000),  
# # ylim = c(0.000005,0.0000160)  
# ) + # Zoom in without removing data  
# scale\_x\_continuous(labels = scales::comma) +  
# scale\_y\_continuous(labels = scales::comma)+  
# geom\_vline(xintercept = 55883,   
# color = "red",   
# linetype = "dotted",   
# linewidth =0.8) +  
# annotate("text",   
# x = 55883,   
# y = 0.0000110,   
# #label = "Poverty line\nAMD 55,883",   
# label = "Poverty line",   
# color = "black",   
# hjust = -0.1,   
# # vjust = -3.5,  
# #angle = 90,   
# size = 3)

And we also plot the cumulative distributions.

# # Plot the cumulative distribution with left-facing arrows  
# ggplot(ca\_microsim,   
# aes(x = aec\_r\_2030\_baseline, color = 'Direct CC + Food Price')) +  
# stat\_ecdf(geom = "step") +  
# # stat\_ecdf(data = ca\_microsim\_cc,   
# # aes(x = aec\_r\_2030\_baseline\_lab\_avg, color = 'Baseline + Labor Productivity')) +  
# stat\_ecdf(data = ca\_microsim\_cc,   
# aes(x = aec\_r\_2030\_baseline\_lab\_cc\_avg, color = 'Direct CC')) +  
# stat\_ecdf(data = ca\_microsim\_cc,   
# aes(x = aec\_r\_2030\_baseline, color = 'Baseline')) +  
# labs(  
# color = "Scenario Variant",   
# # title = "Comparison of Cumulative Consumption Distributions",   
# x = "Equivalized consumption (Dram)",   
# y = "Cumulative Probability") +  
# theme\_minimal() +  
# coord\_cartesian(xlim = c(40000, 110000)) +   
# scale\_x\_continuous(labels = scales::comma) +  
# # geom\_vline(xintercept = 55883,   
# # color = "red",   
# # linetype = "dotted",   
# # linewidth = 0.8) +  
# # annotate("text",   
# # x = 55883,   
# # y = 0.5,   
# # label = "Poverty line",   
# # color = "black",   
# # hjust = -0.1,   
# # size = 3) +  
# annotate("segment", x = 70000, xend = 65000, y = 0.2, yend = 0.2,   
# arrow = arrow(length = unit(0.3, "cm")), color = "black") +  
# annotate("text", x = 72500, y = 0.2, label = "Shift due to shocks", hjust = 0)   
# # annotate("segment", x = 80000, xend = 75000, y = 0.4, yend = 0.4,   
# # arrow = arrow(length = unit(0.3, "cm")), color = "black") +  
# # annotate("text", x = 82500, y = 0.4, label = "Shift due to shocks", hjust = 0)

### 8.2 Climate change

In these sections we use Administrative Level 1 data on yield losses and labor productivity losses due to climate change that are provided in the study commissioned for Armenia’s CCDR *Estimating the Economic Impacts of Climate Change in Armenia* (Strzepek et al., 2024).

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  
labor\_productivity <- labor\_productivity %>%  
 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(labor\_productivity$clim\_scenario)  
  
# Moving average for year of interest  
lab\_loss\_avg <- labor\_productivity %>%  
 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 <- labor\_productivity %>%  
 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  
crops\_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 we do the same for livestock productivity.In this case, there is also disaggregation by Marz.

# First calculate moving window average  
livestock\_productivity <- livestock\_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(livestock\_productivity$climate.scenario)  
  
# Moving average for year of interest  
lvstk\_pdcvty\_loss <- livestock\_productivity %>%  
 filter(climate.scenario == cs[1],   
 year == yearsto[1]) %>%  
 select(-pct\_change\_prod,   
 -year,   
 -climate.scenario) %>%  
 rename(lvstk\_avg\_loss = moving\_avg,   
 lvstk\_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(crops\_pdcvty\_loss,   
 join\_by(NAM\_1 == NAM\_1)) %>%   
 left\_join(lvstk\_pdcvty\_loss,  
 join\_by(NAM\_1))  
  
##write.table(lab\_loss\_avg, "clipboard", sep="\t", row.names=FALSE)

And we first shock labor income.

# Labor income according to sector  
pp\_microsim\_cc <- pp\_microsim\_cc %>%  
 mutate(sector = as.numeric(sector)) %>%  
 mutate(  
 mli\_2030\_baseline\_lab\_avg =  
 case\_when(  
 sector == 1 ~  
 monthly\_labor\_income\_2030\_baseline \*   
 (1 + agr\_avg),  
 sector == 2 ~  
 monthly\_labor\_income\_2030\_baseline \*   
 (1 + man\_avg),  
 sector == 3 ~  
 monthly\_labor\_income\_2030\_baseline \*   
 (1 + ser\_avg),  
 TRUE ~ NA  
 )  
 ) %>%  
 mutate(  
 mli\_2030\_baseline\_lab\_max =  
 case\_when(  
 # \* 1000 because its thousands of Dram  
 sector == 1 ~  
 monthly\_labor\_income\_2030\_baseline \*   
 (1 + agr\_max),  
 sector == 2 ~  
 monthly\_labor\_income\_2030\_baseline \*   
 (1 + man\_max),  
 sector == 3 ~  
 monthly\_labor\_income\_2030\_baseline \*   
 (1 + ser\_max),  
 TRUE ~ NA  
 )  
 )

We aggregate at household level and take note of 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) %>%  
 summarize(  
 mli\_2030\_baseline\_lab\_avg =   
 sum(mli\_2030\_baseline\_lab\_avg, na.rm = TRUE),  
 mli\_2030\_baseline\_lab\_max =   
 sum(mli\_2030\_baseline\_lab\_max, na.rm = TRUE),  
 mli\_original =   
 sum(monthly\_labor\_income\_2030\_baseline, na.rm = TRUE)  
 ) %>%  
 mutate(  
 mli\_2030\_baseline\_lab\_avg\_coef =  
 if\_else(  
 mli\_original == 0 | is.na(mli\_original),  
 1,  
 mli\_2030\_baseline\_lab\_avg / mli\_original  
 ),  
 mli\_2030\_baseline\_lab\_max\_coef =  
 if\_else(  
 mli\_original == 0 | is.na(mli\_original),  
 1,  
 mli\_2030\_baseline\_lab\_max / mli\_original  
 )  
 ) %>%  
 ungroup()  
  
ic\_microsim\_cc <- ic\_microsim\_cc %>%  
 left\_join(ic\_new\_incomes,   
 join\_by(interview\_\_key == hhid)) %>%  
 mutate(  
 inc2\_2030\_baseline\_lab\_avg =   
 inc2\_2030\_baseline \* mli\_2030\_baseline\_lab\_avg\_coef,  
 inc2\_2030\_baseline\_lab\_max =   
 inc2\_2030\_baseline \* mli\_2030\_baseline\_lab\_max\_coef,  
 inc3\_2030\_baseline\_lab\_avg =   
 inc3\_2030\_baseline \* mli\_2030\_baseline\_lab\_avg\_coef,  
 inc3\_2030\_baseline\_lab\_max =   
 inc3\_2030\_baseline \* mli\_2030\_baseline\_lab\_max\_coef  
 )

And now we impact agricultural income cropinc and livestock incomelvstk.

ic\_microsim\_cc <- ic\_microsim\_cc %>%   
 mutate(  
 cropinc\_2030\_baseline\_cc\_avg =   
 cropinc \* (1 + crops\_avg\_loss),  
 cropinc\_2030\_baseline\_cc\_max =   
 cropinc \* (1 + crops\_max\_loss),  
 lvstk\_2030\_baseline\_cc\_avg =   
 lvstk \* (1 + lvstk\_avg\_loss),  
 lvstk\_2030\_baseline\_cc\_max =   
 lvstk \* (1 + lvstk\_max\_loss)  
 )

And recalculate total income.

ic\_microsim\_cc <- ic\_microsim\_cc %>%  
 mutate(  
 totalinc\_2030\_baseline\_lab\_avg =  
 totalinc\_2030\_baseline -  
 rowSums(select(., c(inc2\_2030\_baseline,   
 inc3\_2030\_baseline)), na.rm = TRUE) +  
 rowSums(select(  
 ., c(inc2\_2030\_baseline\_lab\_avg,   
 inc3\_2030\_baseline\_lab\_avg)), na.rm = TRUE),  
 totalinc\_2030\_baseline\_lab\_max =  
 totalinc\_2030\_baseline -  
 rowSums(select(., c(inc2\_2030\_baseline,   
 inc3\_2030\_baseline)), na.rm = TRUE) +  
 rowSums(select(  
 ., c(inc2\_2030\_baseline\_lab\_max,   
 inc3\_2030\_baseline\_lab\_max)), na.rm = TRUE)  
 ) %>%  
 mutate(  
 totalinc\_2030\_baseline\_lab\_avg\_coef =  
 if\_else(totalinc\_2030\_baseline == 0,   
 1, totalinc\_2030\_baseline\_lab\_avg /  
 totalinc\_2030\_baseline),  
 totalinc\_2030\_baseline\_lab\_max\_coef =  
 if\_else(totalinc\_2030\_baseline == 0,   
 1, totalinc\_2030\_baseline\_lab\_max /  
 totalinc\_2030\_baseline)  
 ) %>%  
 mutate(  
 totalinc\_2030\_baseline\_lab\_avg\_coef =  
 if\_else(is.na(totalinc\_2030\_baseline\_lab\_avg\_coef),   
 1, totalinc\_2030\_baseline\_lab\_avg\_coef),  
 totalinc\_2030\_baseline\_lab\_max\_coef =  
 if\_else(is.na(totalinc\_2030\_baseline\_lab\_max\_coef),   
 1, totalinc\_2030\_baseline\_lab\_max\_coef)  
 )

We do the same for agriculture and livestock income alone

ic\_microsim\_cc <- ic\_microsim\_cc %>%  
 mutate(  
 totalinc\_2030\_baseline\_cc\_avg =  
 totalinc\_2030\_baseline -  
 rowSums(select(., c(cropinc,  
 lvstk)), na.rm = TRUE) +  
 rowSums(select(  
 ., c(cropinc\_2030\_baseline\_cc\_avg,  
 lvstk\_2030\_baseline\_cc\_avg)), na.rm = TRUE),  
 totalinc\_2030\_baseline\_cc\_max =  
 totalinc\_2030\_baseline -  
 rowSums(select(., c(cropinc,  
 lvstk)), na.rm = TRUE) +  
 rowSums(select(  
 ., c(cropinc\_2030\_baseline\_cc\_max,  
 lvstk\_2030\_baseline\_cc\_max)), na.rm = TRUE)  
 ) %>%  
 mutate(  
 totalinc\_2030\_baseline\_cc\_avg\_coef =  
 if\_else(totalinc\_2030\_baseline == 0,   
 1, totalinc\_2030\_baseline\_cc\_avg   
 / totalinc\_2030\_baseline),  
 totalinc\_2030\_baseline\_cc\_max\_coef =  
 if\_else(totalinc\_2030\_baseline == 0,   
 1, totalinc\_2030\_baseline\_cc\_max   
 / totalinc\_2030\_baseline)  
 ) %>%  
 mutate(  
 totalinc\_2030\_baseline\_cc\_avg\_coef =  
 if\_else(is.na(totalinc\_2030\_baseline\_cc\_avg\_coef),   
 1, totalinc\_2030\_baseline\_cc\_avg\_coef),  
 totalinc\_2030\_baseline\_cc\_max\_coef =  
 if\_else(is.na(totalinc\_2030\_baseline\_cc\_max\_coef),   
 1, totalinc\_2030\_baseline\_cc\_max\_coef)  
 )

And yet again for the combined impacts

ic\_microsim\_cc <- ic\_microsim\_cc %>%  
 mutate(  
 totalinc\_2030\_baseline\_lab\_cc\_avg =  
 totalinc\_2030\_baseline -  
 rowSums(select(., c(inc2\_2030\_baseline,   
 inc3\_2030\_baseline,  
 cropinc,  
 lvstk)), na.rm = TRUE) +  
 rowSums(select(  
 ., c(inc2\_2030\_baseline\_lab\_avg,   
 inc3\_2030\_baseline\_lab\_avg,  
 cropinc\_2030\_baseline\_cc\_avg,  
 lvstk\_2030\_baseline\_cc\_avg)), na.rm = TRUE),  
 totalinc\_2030\_baseline\_lab\_cc\_max =  
 totalinc\_2030\_baseline -  
 rowSums(select(., c(inc2\_2030\_baseline,   
 inc3\_2030\_baseline,  
 cropinc,  
 lvstk)), na.rm = TRUE) +  
 rowSums(select(  
 ., c(inc2\_2030\_baseline\_lab\_max,   
 inc3\_2030\_baseline\_lab\_max,  
 cropinc\_2030\_baseline\_cc\_max,  
 lvstk\_2030\_baseline\_cc\_max)), na.rm = TRUE)  
 ) %>%  
 mutate(  
 totalinc\_2030\_baseline\_lab\_cc\_avg\_coef =  
 if\_else(totalinc\_2030\_baseline == 0,   
 1, totalinc\_2030\_baseline\_lab\_cc\_avg /  
 totalinc\_2030\_baseline),  
 totalinc\_2030\_baseline\_lab\_cc\_max\_coef =  
 if\_else(totalinc\_2030\_baseline == 0,   
 1, totalinc\_2030\_baseline\_lab\_cc\_max /  
 totalinc\_2030\_baseline)  
 ) %>%  
 mutate(  
 totalinc\_2030\_baseline\_lab\_cc\_avg\_coef =  
 if\_else(is.na(totalinc\_2030\_baseline\_lab\_cc\_avg\_coef),   
 1, totalinc\_2030\_baseline\_lab\_cc\_avg\_coef),  
 totalinc\_2030\_baseline\_lab\_cc\_max\_coef =  
 if\_else(is.na(totalinc\_2030\_baseline\_lab\_cc\_max\_coef),   
 1, totalinc\_2030\_baseline\_lab\_cc\_max\_coef)  
 )

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

income\_losses <- ic\_microsim\_cc %>%   
 select(interview\_\_key,  
 totalinc\_2030\_baseline\_lab\_avg\_coef,   
 totalinc\_2030\_baseline\_lab\_max\_coef,  
 totalinc\_2030\_baseline\_cc\_avg\_coef,  
 totalinc\_2030\_baseline\_cc\_max\_coef,  
 totalinc\_2030\_baseline\_lab\_cc\_avg\_coef,  
 totalinc\_2030\_baseline\_lab\_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\_2030\_baseline\_lab\_avg = totc\_2030\_baseline \*  
 totalinc\_2030\_baseline\_lab\_avg\_coef,  
 totc\_2030\_baseline\_lab\_max = totc\_2030\_baseline \*   
 totalinc\_2030\_baseline\_lab\_max\_coef,  
 totc\_2030\_baseline\_cc\_avg = totc\_2030\_baseline \*  
 totalinc\_2030\_baseline\_cc\_avg\_coef,  
 totc\_2030\_baseline\_cc\_max = totc\_2030\_baseline \*   
 totalinc\_2030\_baseline\_cc\_max\_coef,  
 totc\_2030\_baseline\_lab\_cc\_avg = totc\_2030\_baseline \*  
 totalinc\_2030\_baseline\_lab\_cc\_avg\_coef,  
 totc\_2030\_baseline\_lab\_cc\_max = totc\_2030\_baseline \*   
 totalinc\_2030\_baseline\_lab\_cc\_max\_coef  
 ) %>%   
 mutate(aec\_r\_2030\_baseline\_lab\_avg =   
 totc\_2030\_baseline\_lab\_avg / ae\_r / PI,  
 aec\_r\_2030\_baseline\_lab\_max =   
 totc\_2030\_baseline\_lab\_max / ae\_r / PI,  
 aec\_r\_2030\_baseline\_cc\_avg =   
 totc\_2030\_baseline\_cc\_avg / ae\_r / PI,  
 aec\_r\_2030\_baseline\_cc\_max =   
 totc\_2030\_baseline\_cc\_max / ae\_r / PI,  
 aec\_r\_2030\_baseline\_lab\_cc\_avg =   
 totc\_2030\_baseline\_lab\_cc\_avg / ae\_r / PI,  
 aec\_r\_2030\_baseline\_lab\_cc\_max =   
 totc\_2030\_baseline\_lab\_cc\_max / ae\_r / PI) %>%   
 mutate(poor\_2030\_baseline\_lab\_avg =   
 if\_else(aec\_r\_2030\_baseline\_lab\_avg < 52883, 1, 0),  
 poor\_2030\_baseline\_lab\_max =   
 if\_else(aec\_r\_2030\_baseline\_lab\_max < 52883, 1, 0),  
 poor\_2030\_baseline\_cc\_avg =   
 if\_else(aec\_r\_2030\_baseline\_cc\_avg < 52883, 1, 0),  
 poor\_2030\_baseline\_cc\_max =   
 if\_else(aec\_r\_2030\_baseline\_cc\_max < 52883, 1, 0),  
 poor\_2030\_baseline\_lab\_cc\_avg =   
 if\_else(aec\_r\_2030\_baseline\_lab\_cc\_avg < 52883, 1, 0),  
 poor\_2030\_baseline\_lab\_cc\_max =   
 if\_else(aec\_r\_2030\_baseline\_lab\_cc\_max < 52883, 1, 0)  
 )  
  
# We make a table to see who became poor.   
  
test <- ca\_microsim\_cc %>%  
 rename(poor\_original = poor\_Avpovln2022\_2030\_baseline,  
 poor\_cc = poor\_2030\_baseline\_lab\_avg) %>%  
 group\_by(income\_decile\_group, urb\_rur) %>%   
 summarize(no\_hh = round(sum(weight\_2030\_nzs, na.rm = TRUE)),  
 no\_pp = round(sum(weight\_2030\_nzs \* hhsize, na.rm = TRUE))) %>%   
 ungroup()

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

test %>%   
 gt()

| income\_decile\_group | urb\_rur | no\_hh | no\_pp |
| --- | --- | --- | --- |
| 1 | 1 | 55581 | 101934 |
| 1 | 2 | 27908 | 74783 |
| 2 | 1 | 49717 | 97861 |
| 2 | 2 | 28239 | 77403 |
| 3 | 1 | 51846 | 125981 |
| 3 | 2 | 26300 | 84452 |
| 4 | 1 | 50737 | 138969 |
| 4 | 2 | 27163 | 106969 |
| 5 | 1 | 52779 | 181812 |
| 5 | 2 | 24816 | 104470 |
| 6 | 1 | 48955 | 178108 |
| 6 | 2 | 29091 | 130978 |
| 7 | 1 | 52818 | 202914 |
| 7 | 2 | 25175 | 115496 |
| 8 | 1 | 49690 | 209561 |
| 8 | 2 | 28184 | 137425 |
| 9 | 1 | 49603 | 225308 |
| 9 | 2 | 28665 | 151247 |
| 10 | 1 | 52287 | 245685 |
| 10 | 2 | 25319 | 141672 |

##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.

# 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) %>%   
 summarize(index = mean(index, na.rm = TRUE)) %>%   
 gt()

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

| scenario | index |
| --- | --- |
| energy | |
| baseline | 1.1291820 |
| dry\_hot | 1.1248830 |
| nzs | 1.8014210 |
| food | |
| baseline | 1.0099860 |
| dry\_hot | 1.0756960 |
| nzs | 0.9670056 |

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( scenario,year,commodity\_group, 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)

So in this particular case, we don’t want to use the price index from the dry\_hot scenario, but we want to use the difference between the baseline and that scenario, so we are going to do those columns by hand, but we actually have to find a way to do it programmatically against the baseline.

price\_data\_yearsto <- price\_data\_yearsto %>%   
 mutate(food\_PI = dry\_hot\_2030\_food - baseline\_2030\_food +1,  
 energy\_PI = nzs\_2030\_energy - baseline\_2030\_energy + 1)

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) %>%  
 summarize(  
 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")

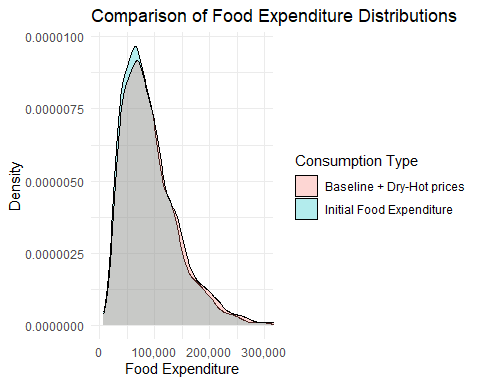
Let’s apply the elasticities to the new data.

# 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 = (baseline\_2030\_food - 1),  
 food\_2\_dprice = (dry\_hot\_2030\_food - 1),  
 food\_PI\_dprice = (food\_PI - 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),  
 food\_qPI\_sim = food\_quantity \*  
 (1 + food\_PI\_dprice \* price\_elasticity))  
  
# Calculate the new expenditure levels  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(food\_exp1\_sim = food\_q1\_sim \* baseline\_2030\_food,  
 food\_exp2\_sim = food\_q2\_sim \* dry\_hot\_2030\_food,  
 food\_exp3\_sim = food\_qPI\_sim \* food\_PI)  
  
# View the results  
print(ca\_microsim\_cc %>% select(decile, food1, baseline\_2030\_food, food\_1\_dprice, food\_q1\_sim, food\_exp1\_sim, food\_2\_dprice, food\_q2\_sim, food\_exp2\_sim,food\_exp2\_sim, food\_PI\_dprice))

# A tibble: 5,184 × 10  
 decile food1 baseline\_2030\_food food\_1\_dprice food\_q1\_sim food\_exp1\_sim  
 <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 9 92178. 1.01 0.00973 92073. 92969.  
 2 7 105290. 1.01 0.00963 105229. 106243.  
 3 2 22733. 1.01 0.0102 22650. 22881.  
 4 2 163379. 1.01 0.0102 162778. 164442.  
 5 4 102718. 1.01 0.0104 102400. 103461.  
 6 10 208995. 1.01 0.0104 208548. 210712.  
 7 8 116151. 1.01 0.00995 116140. 117296.  
 8 5 67911. 1.01 0.00965 67745. 68398.  
 9 6 152144. 1.01 0.00978 151971. 153457.  
10 8 80033. 1.01 0.00995 80025. 80822.  
# ℹ 5,174 more rows  
# ℹ 4 more variables: food\_2\_dprice <dbl>, food\_q2\_sim <dbl>,  
# food\_exp2\_sim <dbl>, food\_PI\_dprice <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 = 0.3) +   
 geom\_density(  
 data = ca\_microsim\_cc,   
 aes(x = food\_exp3\_sim, fill = 'Baseline + Dry-Hot prices'),   
 alpha = 0.3) +  
 labs(  
 fill = "Consumption Type",   
 title = "Comparison of Food Expenditure Distributions",   
 x = "Food Expenditure",   
 y = "Density") +  
 theme\_minimal() +  
 coord\_cartesian(xlim = c(0, 300000)) + # 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 Consumer Surplus loss for food1 and food2 scenarios  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(  
 across(  
 c(food1,  
 food\_exp1\_sim,  
 food\_exp2\_sim,  
 food\_exp3\_sim,  
 totc\_2030\_baseline\_lab\_cc\_avg), ~replace\_na(., 0))) %>%  
 mutate(food1\_CSloss = ((food\_quantity \* implicit\_price) /   
 totc\_2030\_baseline) \* food\_1\_dprice \* (  
 1 + (price\_elasticity / 2) \* food\_1\_dprice),  
 food2\_CSloss = ((food\_quantity \* implicit\_price) /  
 totc\_2030\_dry\_hot) \* food\_2\_dprice \* (  
 1 + (price\_elasticity / 2) \* food\_2\_dprice),  
 foodPI\_CSloss = ((food\_quantity \* implicit\_price) /  
 totc\_2030\_baseline\_lab\_cc\_avg) \* food\_PI\_dprice \* (  
 1 + (price\_elasticity / 2) \* food\_PI\_dprice),  
 ttl\_CSloss\_1 = food1\_CSloss,  
 ttl\_CSloss\_2 = food2\_CSloss,  
 ttl\_CSloss\_PI = foodPI\_CSloss)  
  
# Calculate Purchasing Power loss for food1 and food2 and PI scenarios  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(food1\_PPloss = (food1 /   
 totc\_2030\_baseline) \* food\_1\_dprice,  
 food2\_PPloss = (food1 /   
 totc\_2030\_dry\_hot) \* food\_2\_dprice,  
 foodPI\_PPloss = (food1 /   
 totc\_2030\_baseline\_lab\_cc\_avg) \* food\_PI\_dprice,  
 ttl\_PPloss\_1 = food1\_PPloss,  
 ttl\_PPloss\_2 = food2\_PPloss,  
 ttl\_PPloss\_PI = foodPI\_PPloss  
 )  
  
# Adjust total expenditure (totc) based on the purchasing power loss  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(totc\_2030\_baseline\_food1 = totc\_2030\_baseline \* (1 - ttl\_PPloss\_1),  
 totc\_2030\_dry\_hot\_food2 = totc\_2030\_dry\_hot \* (1 - ttl\_PPloss\_2),  
 totc\_2030\_baseline\_lab\_cc\_foodPI2 = totc\_2030\_baseline\_lab\_cc\_avg \* (1 - ttl\_PPloss\_PI),  
 totc\_2030\_baseline\_lab\_cc\_foodPI = totc\_2030\_baseline\_lab\_cc\_avg - (food\_exp3\_sim - food1)  
 )  
  
# View the results  
print(ca\_microsim\_cc %>% select(decile, totc\_2030\_baseline, totc\_2030\_baseline\_food1, totc\_2030\_dry\_hot\_food2, totc\_2030\_baseline\_lab\_cc\_foodPI))

# A tibble: 5,184 × 5  
 decile totc\_2030\_baseline totc\_2030\_baseline\_food1 totc\_2030\_dry\_hot\_food2  
 <dbl> <dbl> <dbl> <dbl>  
 1 9 298269. 297372. 291365.  
 2 7 345611. 344597. 331885.  
 3 2 130498. 130266. 126316.  
 4 2 365825. 364155. 348536.  
 5 4 285862. 284798. 272543.  
 6 10 525030. 522861. 499583.  
 7 8 232452. 231296. 223637.  
 8 5 326731. 326076. 315638.  
 9 6 479398. 477910. 458876.  
10 8 197776. 196980. 188004.  
# ℹ 5,174 more rows  
# ℹ 1 more variable: totc\_2030\_baseline\_lab\_cc\_foodPI <dbl>

Okay so now we estimate new welfare and poverty.

ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(aec\_r\_2030\_baseline\_food1 = totc\_2030\_baseline\_food1 / ae\_r / PI,  
 aec\_r\_2030\_dry\_hot\_food2 = totc\_2030\_dry\_hot\_food2 / ae\_r / PI,  
 aec\_r\_2030\_baseline\_foodPI = totc\_2030\_baseline\_lab\_cc\_foodPI / ae\_r / PI) %>%  
 mutate(poor\_2030\_baseline\_food1 =  
 if\_else(aec\_r\_2030\_baseline\_food1 < 52883, 1, 0),  
 poor\_2030\_dry\_hot\_food2 =  
 if\_else(aec\_r\_2030\_dry\_hot\_food2 < 52883, 1, 0),  
 poor\_2030\_baseline\_lab\_cc\_foodPI =  
 if\_else(aec\_r\_2030\_baseline\_foodPI < 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\_2030\_baseline,  
 poor\_cc = poor\_2030\_baseline\_lab\_cc\_avg,  
 poor\_food1 = poor\_2030\_baseline\_food1,  
 poor\_food2 = poor\_2030\_dry\_hot\_food2,  
 poor\_foodPI = poor\_2030\_baseline\_lab\_cc\_foodPI) %>%  
 group\_by(poor\_original) %>%   
 summarize(no\_hh = round(sum(weight\_2030\_baseline, na.rm = TRUE)),  
 no\_pp = round(sum(weight\_2030\_baseline\*hhsize, na.rm = TRUE)))  
  
test %>%   
 gt()

| poor\_original | no\_hh | no\_pp |
| --- | --- | --- |
| 0 | 697740 | 2412949 |
| 1 | 85849 | 420079 |

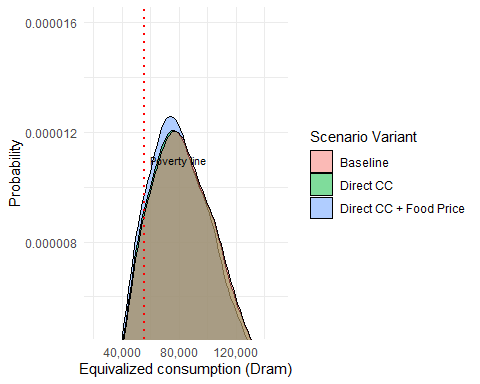
##write.table(test, "clipboard", sep="\t", row.names=FALSE)

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) %>%   
# summarize(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

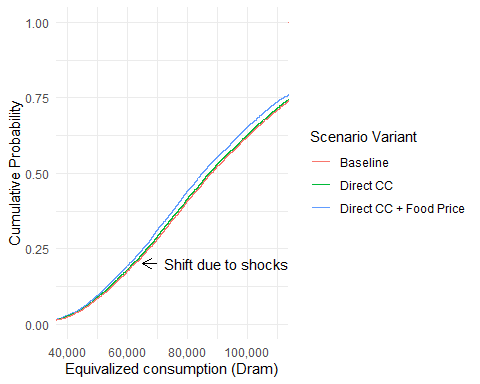
Let’s plot how the distribution moves with all these measures.

# Basic density plot comparing equivalized consumption per capita  
ggplot(ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_baseline\_foodPI, fill = 'Direct CC + Food Price')) +  
 geom\_density(alpha = 0.5) +  
 # geom\_density(  
 # data = ca\_microsim\_cc,  
 # aes(x = aec\_r\_2030\_dry\_hot, fill = 'Dry/Hot'),  
 # alpha = 0.5) +  
 # geom\_density(  
 # data = ca\_microsim\_cc,  
 # aes(x = aec\_r\_2030\_dry\_hot\_food2, fill = 'Dry/Hot + Food Price'),  
 # alpha = 0.5) +  
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = aec\_r\_2030\_baseline\_lab\_cc\_avg, fill = 'Direct CC'),  
 alpha = 0.5) +  
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = aec\_r\_2030\_baseline, fill = 'Baseline'),  
 alpha = 0.5) +  
 labs(  
 fill = "Scenario Variant",   
 # title = "Comparison of Consumption Distributions",   
 x = "Equivalized consumption (Dram)",   
 y = "Probability") +  
 theme\_minimal()+  
 coord\_cartesian(xlim = c(20000, 150000),  
 ylim = c(0.000005,0.0000160)) + # Zoom in without removing data  
 scale\_x\_continuous(labels = scales::comma) +  
 scale\_y\_continuous(labels = scales::comma)+  
 geom\_vline(xintercept = 55883,   
 color = "red",   
 linetype = "dotted",   
 linewidth =0.8) +  
 annotate("text",   
 x = 55883,   
 y = 0.0000110,   
 #label = "Poverty line\nAMD 55,883",   
 label = "Poverty line",   
 color = "black",   
 hjust = -0.1,   
 # vjust = -3.5,  
 #angle = 90,   
 size = 3)



And we also plot the cumulative distributions.

# Plot the cumulative distribution with left-facing arrows  
ggplot(ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_baseline\_foodPI, color = 'Direct CC + Food Price')) +  
 stat\_ecdf(geom = "step") +  
 # stat\_ecdf(data = ca\_microsim\_cc,   
 # aes(x = aec\_r\_2030\_baseline\_lab\_avg, color = 'Baseline + Labor Productivity')) +  
 stat\_ecdf(data = ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_baseline\_lab\_cc\_avg, color = 'Direct CC')) +  
 stat\_ecdf(data = ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_baseline, color = 'Baseline')) +  
 labs(  
 color = "Scenario Variant",   
 # title = "Comparison of Cumulative Consumption Distributions",   
 x = "Equivalized consumption (Dram)",   
 y = "Cumulative Probability") +  
 theme\_minimal() +  
 coord\_cartesian(xlim = c(40000, 110000)) +   
 scale\_x\_continuous(labels = scales::comma) +  
 # geom\_vline(xintercept = 55883,   
 # color = "red",   
 # linetype = "dotted",   
 # linewidth = 0.8) +  
 # annotate("text",   
 # x = 55883,   
 # y = 0.5,   
 # label = "Poverty line",   
 # color = "black",   
 # hjust = -0.1,   
 # size = 3) +  
 annotate("segment", x = 70000, xend = 65000, y = 0.2, yend = 0.2,   
 arrow = arrow(length = unit(0.3, "cm")), color = "black") +  
 annotate("text", x = 72500, y = 0.2, label = "Shift due to shocks", hjust = 0)



# annotate("segment", x = 80000, xend = 75000, y = 0.4, yend = 0.4,   
 # arrow = arrow(length = unit(0.3, "cm")), color = "black") +  
 # annotate("text", x = 82500, y = 0.4, label = "Shift due to shocks", hjust = 0)

### 8.4 Energy prices

We first establish energy elasticities. We only have quantities for liquefied gas hous\_29\_a and their purchase value hous\_29\_b with which we can compute price. Unfortunately there is no quantity for electricity, so we will use the same elasticity. We do not compute an elasticity by decile, because there are too few observations per decile, so we estimate an overall elasticity for all the distribution.

# We extract the liquefied gas (hous\_29), natural gas (hous\_38)   
# and electricity (hous\_23) information  
energy\_summary\_all <- hh %>%   
 mutate(l\_gas\_price =   
 if\_else(hous\_29\_a == 0, 0, hous\_29\_b/hous\_29\_a),  
 n\_gas\_price =   
 if\_else(hous\_36\_a == 0, 0, hous\_36\_b/hous\_36\_a)) %>%  
 select(interview\_\_key, weight, hous\_29\_a, hous\_29\_b,hous\_23,  
 hous\_36\_a, hous\_36\_b, l\_gas\_price, n\_gas\_price)  
  
# We estimate the weighted mean of liquefied gas prices  
avg\_l\_gas\_price <- weighted.mean(energy\_summary\_all$l\_gas\_price,  
 energy\_summary\_all$weight,  
 na.rm=TRUE)  
  
# And do the same for natural gas  
avg\_n\_gas\_price <- weighted.mean(energy\_summary\_all$n\_gas\_price,  
 energy\_summary\_all$weight,  
 na.rm=TRUE)  
  
# We replace missing 0 values with average gas price  
energy\_summary\_all <- energy\_summary\_all %>%   
 mutate(l\_gas\_price = if\_else(l\_gas\_price==0.0,  
 avg\_l\_gas\_price,  
 l\_gas\_price),  
 n\_gas\_price = if\_else(n\_gas\_price==0.0,  
 avg\_n\_gas\_price,  
 n\_gas\_price))  
  
# We subset to compute a single elasticity value for the entire distribution  
# Summarize the data at the household level  
l\_energy\_summary <- energy\_summary\_all %>%  
 filter(!is.na(l\_gas\_price))# %>%   
   
# Filter out rows with non-positive values in hous\_29\_a or l\_gas\_price  
l\_energy\_summary <- l\_energy\_summary[l\_energy\_summary$hous\_29\_a > 0 & l\_energy\_summary$l\_gas\_price > 0, ]  
  
# Compute the log of quantity and price  
l\_energy\_summary$log\_gas\_quantity <- log(l\_energy\_summary$hous\_29\_a)  
l\_energy\_summary$log\_l\_gas\_price <- log(l\_energy\_summary$l\_gas\_price)  
  
# Estimate a single price elasticity for the entire dataset  
model <- lm(log\_gas\_quantity ~ log\_l\_gas\_price, data = l\_energy\_summary)  
summary\_model <- summary(model)  
  
# Extract the price elasticity (coefficient of log\_l\_gas\_price)  
l\_gas\_price\_elasticity <- coef(summary\_model)["log\_l\_gas\_price", "Estimate"]  
  
# Print the results  
print(summary\_model)

Call:  
lm(formula = log\_gas\_quantity ~ log\_l\_gas\_price, data = l\_energy\_summary)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-1.79267 -0.42107 -0.01872 0.50991 1.87661   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 2.33837 0.63292 3.695 0.00025 \*\*\*  
log\_l\_gas\_price -0.08781 0.10115 -0.868 0.38587   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.5834 on 412 degrees of freedom  
Multiple R-squared: 0.001826, Adjusted R-squared: -0.0005972   
F-statistic: 0.7535 on 1 and 412 DF, p-value: 0.3859

print(paste("Estimated price elasticity of gas quantity demanded:", l\_gas\_price\_elasticity))

[1] "Estimated price elasticity of gas quantity demanded: -0.0878078581985716"

We see that this commodity is highly inelastic at -0.088781.The estimated price elasticity of -0.086 suggests that the demand for gas is inelastic. This means that a 1% increase in the price of gas would lead to only a 0.09% decrease in the quantity of gas demanded. The absolute value of the elasticity is much less than 1, indicating that consumers do not significantly reduce their gas consumption in response to price increases. This could be because gas is a necessity for many households, and they cannot easily reduce their usage or switch to alternative sources. We expect electricity, being so universal in the dataset to behave in the same manner. We wanted to use natural gas to compute a similar metric, but there is hardly any variation in prices. Everybody experiences the same price and so there is not enough variation to compute a valid model. We will use the elasticity from liquefied gas for our purposes.

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

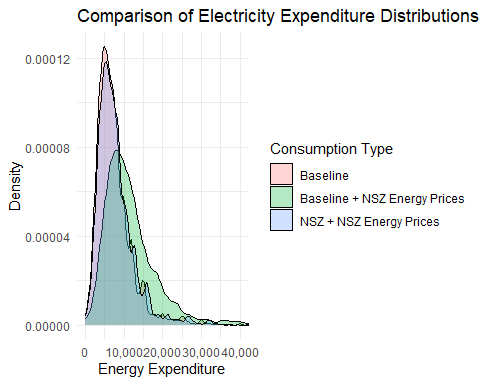
ca\_microsim\_cc$l\_gas\_price\_elasticity <- l\_gas\_price\_elasticity  
ca\_microsim\_cc <- ca\_microsim\_cc %>%   
 left\_join(energy\_summary\_all, join\_by(hhid==interview\_\_key))

Let’s apply the elasticities to the new data.

# Calculate the implicit price  
# Assuming implicit\_price can be calculated from the expenditure  
# If we assume baseline quantity consumed is proportional to expenditure/price  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(  
 # This is 1 as we don't have baseline price  
 electricity\_implicit\_price = if\_else(hous\_23 == 0,1,hous\_23 / hous\_23),   
 electricity\_quantity = hous\_23 / electricity\_implicit\_price,  
 energy\_price\_elasticity = l\_gas\_price\_elasticity)  
  
EM\_elec\_price <- 1.071353 # Price increase in Energy Model  
EM\_gas\_price <- 1.025514  
  
# Calculate the percentage change in prices by decile  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(energy\_baseline\_dprice = (energy\_PI - 1),  
 energy\_nzs\_dprice = (nzs\_2030\_energy - 1),  
 EM\_elec\_dprice = (EM\_elec\_price-1), # Interpolation from Energy Model  
 EM\_gas\_dprice = (EM\_gas\_price-1))   
  
# Estimate the new energy consumption levels  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(  
 electricity\_baseline\_q\_sim = electricity\_quantity \*   
 (1 + energy\_baseline\_dprice \* energy\_price\_elasticity),  
 electricity\_nzs\_q\_sim = electricity\_quantity \*  
 (1 + energy\_nzs\_dprice \* energy\_price\_elasticity),  
 electricity\_EM\_q\_sim = electricity\_quantity \*  
 (1 + EM\_elec\_dprice \* energy\_price\_elasticity),  
 l\_gas\_baseline\_q\_sim = hous\_29\_a \*   
 (1 + energy\_baseline\_dprice \* energy\_price\_elasticity),  
 l\_gas\_nzs\_q\_sim = hous\_29\_a \*  
 (1 + energy\_nzs\_dprice \* energy\_price\_elasticity),  
 l\_gas\_EM\_q\_sim = hous\_29\_a \*  
 (1 + EM\_gas\_dprice \* energy\_price\_elasticity),  
 n\_gas\_baseline\_q\_sim = hous\_36\_a \*   
 (1 + energy\_baseline\_dprice \* energy\_price\_elasticity),  
 n\_gas\_nzs\_q\_sim = hous\_36\_a \*  
 (1 + energy\_nzs\_dprice \* energy\_price\_elasticity),  
 n\_gas\_EM\_q\_sim = hous\_36\_a \*  
 (1 + EM\_gas\_dprice \* energy\_price\_elasticity)  
 )  
  
# Calculate the new expenditure levels  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(electricity\_baseline\_sim =   
 electricity\_baseline\_q\_sim \* energy\_PI, #because e-price = 1  
 l\_gas\_baseline\_sim =   
 l\_gas\_baseline\_q\_sim \* l\_gas\_price \* energy\_PI,  
 n\_gas\_baseline\_sim =   
 n\_gas\_baseline\_q\_sim \* n\_gas\_price \* energy\_PI,  
 electricity\_nzs\_sim =   
 electricity\_nzs\_q\_sim \* nzs\_2030\_energy,#because e-price = 1  
 l\_gas\_nzs\_sim =   
 l\_gas\_nzs\_q\_sim \* l\_gas\_price \* nzs\_2030\_energy,  
 n\_gas\_nzs\_sim =   
 n\_gas\_nzs\_q\_sim \* n\_gas\_price \* nzs\_2030\_energy,  
 electricity\_EM\_sim =   
 electricity\_EM\_q\_sim \* EM\_elec\_price,#because e-price = 1  
 l\_gas\_EM\_sim =   
 l\_gas\_EM\_q\_sim \* l\_gas\_price \* EM\_gas\_price,  
 n\_gas\_EM\_sim =   
 n\_gas\_EM\_q\_sim \* n\_gas\_price \* EM\_gas\_price)

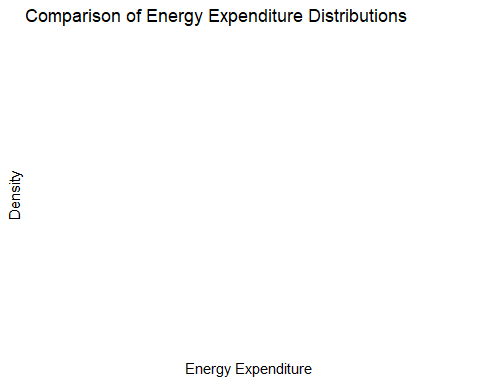
Let’s plot the distributions to see changes:

# Basic density plot comparing food1 and food\_exp\_sim  
ggplot(ca\_microsim\_cc, aes(x = hous\_23, fill = 'Baseline')) +   
 geom\_density(alpha = 0.3) +   
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = electricity\_baseline\_sim, fill = 'Baseline + NSZ Energy Prices'),  
 alpha = 0.3) +  
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = electricity\_EM\_sim, fill = 'NSZ + NSZ Energy Prices'),  
 alpha = 0.3) +  
 #facet\_wrap(~decile)+  
 labs(  
 fill = "Consumption Type",   
 title = "Comparison of Electricity Expenditure Distributions",   
 x = "Energy Expenditure",   
 y = "Density") +  
 theme\_minimal() +  
 coord\_cartesian(xlim = c(0, 40000)) + # Adjust the xlim for zoom  
 scale\_x\_continuous(labels = scales::comma) +  
 scale\_y\_continuous(labels = scales::comma)



# Basic density plot comparing food1 and food\_exp\_sim  
ggplot(ca\_microsim\_cc, aes(x = hous\_23+hous\_29\_b+hous\_36\_b, fill = 'Baseline')) +   
 geom\_density(alpha = 0.3) +   
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = electricity\_baseline\_sim+l\_gas\_baseline\_sim+n\_gas\_baseline\_sim, fill = 'Baseline + NSZ Energy Prices'),  
 alpha = 0.3) +  
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = electricity\_nzs\_sim+l\_gas\_nzs\_sim+n\_gas\_nzs\_sim, fill = 'NSZ + NSZ Energy Prices'),  
 alpha = 0.3) +  
 # facet\_wrap(~decile)+  
 labs(  
 fill = "Consumption Type",   
 title = "Comparison of Energy Expenditure Distributions",   
 x = "Energy Expenditure",   
 y = "Density") +  
 theme\_minimal() +  
 coord\_cartesian(xlim = c(0, 100000)) + # Adjust the xlim for zoom  
 scale\_x\_continuous(labels = scales::comma) +  
 scale\_y\_continuous(labels = scales::comma)

Warning: Removed 5184 rows containing non-finite outside the scale range  
(`stat\_density()`).  
Removed 5184 rows containing non-finite outside the scale range  
(`stat\_density()`).  
Removed 5184 rows containing non-finite outside the scale range  
(`stat\_density()`).



Calculate losses in consumer surplus and purchasing power loss.

# Calculate Purchasing Power loss for food1 and food2 and PI scenarios  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(  
 across(  
 c(hous\_23,  
 hous\_29\_b,  
 hous\_36\_b,   
 totc\_2030\_baseline,  
 totc\_2030\_nzs,  
 l\_gas\_baseline\_sim,  
 n\_gas\_baseline\_sim,  
 electricity\_baseline\_sim,  
 l\_gas\_nzs\_sim,  
 n\_gas\_nzs\_sim,  
 electricity\_nzs\_sim,  
 l\_gas\_EM\_sim,  
 n\_gas\_EM\_sim,  
 electricity\_EM\_sim), ~replace\_na(., 0))) %>%  
 mutate(energy\_baseline\_PPloss = ((hous\_23 + hous\_29\_b + hous\_36\_b) /   
 totc\_2030\_baseline) \* energy\_baseline\_dprice,  
 energy\_nzs\_PPloss = ((hous\_23 + hous\_29\_b + hous\_36\_b) /   
 totc\_2030\_nzs) \* energy\_nzs\_dprice,  
 ttl\_PPloss\_1 = energy\_baseline\_PPloss,  
 ttl\_PPloss\_2 = energy\_nzs\_PPloss  
 )  
  
# Adjust total expenditure (totc) based on the purchasing power loss  
ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 # mutate(nzs\_scale\_coef = if\_else(  
 # totc\_2030\_baseline==0, 1, totc\_2030\_nzs/totc\_2022)) %>%   
 mutate(#totc\_2030\_baseline\_energy2 = totc\_2030\_baseline \* (1 - ttl\_PPloss\_1),  
 totc\_2030\_baseline\_energy =   
 totc\_2030\_baseline -  
 ((electricity\_baseline\_sim +  
 l\_gas\_baseline\_sim +   
 n\_gas\_baseline\_sim) -   
 (hous\_23 +   
 hous\_36\_b +   
 hous\_29\_b)),  
 #totc\_2030\_nzs\_energy2 = totc\_2030\_nzs \* (1 - ttl\_PPloss\_2),  
 totc\_2030\_nzs\_energy =   
 totc\_2030\_nzs -  
 ((electricity\_nzs\_sim +  
 l\_gas\_nzs\_sim +   
 n\_gas\_nzs\_sim) -   
 (hous\_23 +   
 hous\_36\_b +   
 hous\_29\_b)),  
 totc\_2030\_EM\_baseline\_energy =   
 totc\_2030\_baseline -  
 ((electricity\_EM\_sim +  
 l\_gas\_EM\_sim +   
 n\_gas\_EM\_sim) -   
 (hous\_23 +   
 hous\_36\_b +   
 hous\_29\_b)),  
 totc\_2030\_EM\_nzs\_energy =   
 totc\_2030\_nzs -  
 ((electricity\_EM\_sim +  
 l\_gas\_EM\_sim +   
 n\_gas\_EM\_sim) -   
 (hous\_23 +   
 hous\_36\_b +   
 hous\_29\_b))  
 )  
  
# View the results  
print(ca\_microsim\_cc %>% select(decile, totc\_2030\_baseline, totc\_2030\_baseline\_energy, totc\_2030\_nzs\_energy, totc\_2030\_EM\_baseline\_energy,totc\_2030\_EM\_nzs\_energy))

# A tibble: 5,184 × 6  
 decile totc\_2030\_baseline totc\_2030\_baseline\_energy totc\_2030\_nzs\_energy  
 <dbl> <dbl> <dbl> <dbl>  
 1 9 298269. 250374. 242384.  
 2 7 345611. 326156. 297259.  
 3 2 130498. 128170. 120539.  
 4 2 365825. 332689. 305466.  
 5 4 285862. 261819. 234343.  
 6 10 525030. 484825. 438545.  
 7 8 232452. 203798. 203659.  
 8 5 326731. 294705. 262852.  
 9 6 479398. 439146. 395652.  
10 8 197776. 185229. 167076.  
# ℹ 5,174 more rows  
# ℹ 2 more variables: totc\_2030\_EM\_baseline\_energy <dbl>,  
# totc\_2030\_EM\_nzs\_energy <dbl>

Okay so now we estimate new welfare and poverty.

ca\_microsim\_cc <- ca\_microsim\_cc %>%  
 mutate(aec\_r\_2030\_baseline\_energy = totc\_2030\_baseline\_energy / ae\_r / PI,  
 aec\_r\_2030\_nzs\_energy = totc\_2030\_nzs\_energy / ae\_r / PI,  
 aec\_r\_2030\_EM\_baseline\_energy =  
 totc\_2030\_EM\_baseline\_energy / ae\_r / PI,  
 aec\_r\_2030\_EM\_nzs\_energy =  
 totc\_2030\_EM\_nzs\_energy / ae\_r / PI) %>%  
 mutate(poor\_2030\_baseline\_energy =  
 if\_else(aec\_r\_2030\_baseline\_energy < 52883, 1, 0),  
 poor\_2030\_nzs\_energy =  
 if\_else(aec\_r\_2030\_nzs\_energy < 52883, 1, 0),  
 poor\_2030\_EM\_baseline\_energy =   
 if\_else(aec\_r\_2030\_EM\_baseline\_energy < 52883, 1, 0),  
 poor\_2030\_EM\_nzs\_energy =   
 if\_else(aec\_r\_2030\_EM\_nzs\_energy < 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\_2030\_baseline,  
 poor\_baseline\_energy = poor\_2030\_baseline\_energy  
 ) %>%  
 group\_by(poor\_original) %>%   
 summarize(no\_hh = round(sum(weight\_2030\_baseline, na.rm = TRUE)),  
 no\_pp = round(sum(weight\_2030\_baseline\*hhsize, na.rm = TRUE)))  
  
test %>%   
 gt()

| poor\_original | no\_hh | no\_pp |
| --- | --- | --- |
| 0 | 697740 | 2412949 |
| 1 | 85849 | 420079 |

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

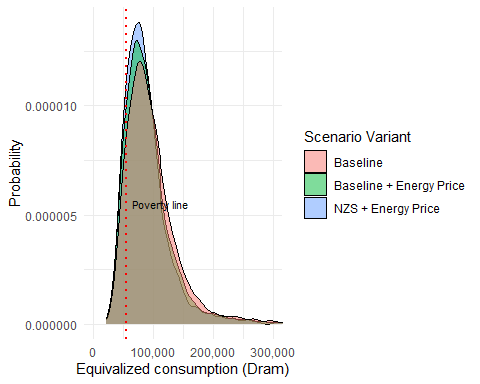
And we map these results.

# 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) %>%   
# summarize(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

Let’s plot how the distribution moves with all these measures.

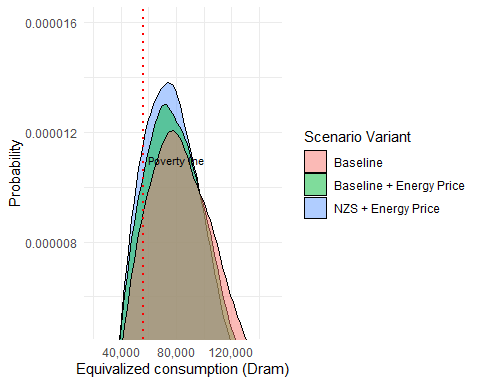
Food prices

# Basic density plot comparing equivalized consumption per capita  
ggplot(ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_nzs\_energy, fill = 'NZS + Energy Price')) +  
 geom\_density(alpha = 0.5) +  
 # geom\_density(  
 # data = ca\_microsim\_cc,  
 # aes(x = aec\_r\_2030\_dry\_hot, fill = 'Dry/Hot'),  
 # alpha = 0.5) +  
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = aec\_r\_2030\_baseline\_energy, fill = 'Baseline + Energy Price'),  
 alpha = 0.5) +  
 # geom\_density(  
 # data = ca\_microsim\_cc,  
 # aes(x = aec\_r\_2030\_baseline\_lab\_cc\_avg, fill = 'Direct CC'),  
 # alpha = 0.5) +  
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = aec\_r\_2030\_baseline, fill = 'Baseline'),  
 alpha = 0.5) +  
 labs(  
 fill = "Scenario Variant",   
 # title = "Comparison of Consumption Distributions",   
 x = "Equivalized consumption (Dram)",   
 y = "Probability") +  
 theme\_minimal()+  
 coord\_cartesian(xlim = c(00000, 300000))+  
 # ylim = c(0.000005,0.0000160)) + # Zoom in without removing data  
 scale\_x\_continuous(labels = scales::comma) +  
 scale\_y\_continuous(labels = scales::comma)+  
 geom\_vline(xintercept = 55883,   
 color = "red",   
 linetype = "dotted",   
 linewidth =0.8) +  
 annotate("text",   
 x = 55883,   
 y = 0.0000055,   
 #label = "Poverty line\nAMD 55,883",   
 label = "Poverty line",   
 color = "black",   
 hjust = -0.1,   
 # vjust = -3.5,  
 #angle = 90,   
 size = 3)



Energy

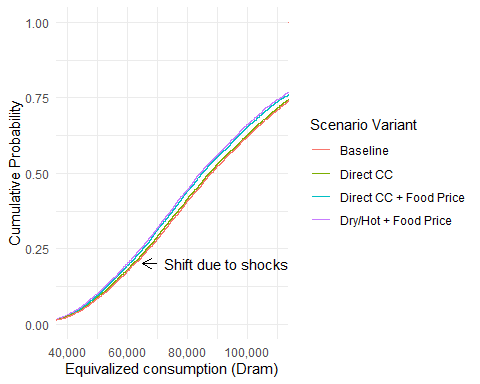
# Basic density plot comparing equivalized consumption per capita  
ggplot(ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_nzs\_energy, fill = 'NZS + Energy Price')) +  
 geom\_density(alpha = 0.5) +  
 # geom\_density(  
 # data = ca\_microsim\_cc,  
 # aes(x = aec\_r\_2030\_dry\_hot, fill = 'Dry/Hot'),  
 # alpha = 0.5) +  
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = aec\_r\_2030\_baseline\_energy, fill = 'Baseline + Energy Price'),  
 alpha = 0.5) +  
 # geom\_density(  
 # data = ca\_microsim\_cc,  
 # aes(x = aec\_r\_2030\_baseline\_lab\_cc\_avg, fill = 'Direct CC'),  
 # alpha = 0.5) +  
 geom\_density(  
 data = ca\_microsim\_cc,  
 aes(x = aec\_r\_2030\_baseline, fill = 'Baseline'),  
 alpha = 0.5) +  
 labs(  
 fill = "Scenario Variant",   
 # title = "Comparison of Consumption Distributions",   
 x = "Equivalized consumption (Dram)",   
 y = "Probability") +  
 theme\_minimal()+  
 coord\_cartesian(xlim = c(20000, 150000),  
 ylim = c(0.000005,0.0000160)) + # Zoom in without removing data  
 scale\_x\_continuous(labels = scales::comma) +  
 scale\_y\_continuous(labels = scales::comma)+  
 geom\_vline(xintercept = 55883,   
 color = "red",   
 linetype = "dotted",   
 linewidth =0.8) +  
 annotate("text",   
 x = 55883,   
 y = 0.0000110,   
 #label = "Poverty line\nAMD 55,883",   
 label = "Poverty line",   
 color = "black",   
 hjust = -0.1,   
 # vjust = -3.5,  
 #angle = 90,   
 size = 3)



And we also plot the cumulative distributions.

For food prices and cc

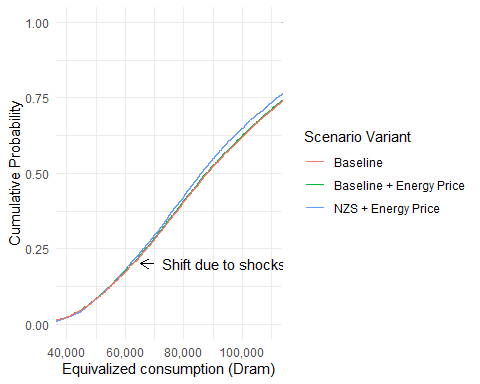
# Plot the cumulative distribution with left-facing arrows  
ggplot(ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_baseline\_foodPI, color = 'Direct CC + Food Price')) +  
 stat\_ecdf(geom = "step") +  
 # stat\_ecdf(data = ca\_microsim\_cc,   
 # aes(x = aec\_r\_2030\_baseline\_lab\_avg, color = 'Baseline + Labor Productivity')) +  
 stat\_ecdf(data = ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_dry\_hot\_food2, color = 'Dry/Hot + Food Price')) +  
 stat\_ecdf(data = ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_baseline\_lab\_cc\_avg, color = 'Direct CC')) +  
 stat\_ecdf(data = ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_baseline, color = 'Baseline')) +  
 labs(  
 color = "Scenario Variant",   
 # title = "Comparison of Cumulative Consumption Distributions",   
 x = "Equivalized consumption (Dram)",   
 y = "Cumulative Probability") +  
 theme\_minimal() +  
 coord\_cartesian(xlim = c(40000, 110000)) +   
 scale\_x\_continuous(labels = scales::comma) +  
 # geom\_vline(xintercept = 55883,   
 # color = "red",   
 # linetype = "dotted",   
 # linewidth = 0.8) +  
 # annotate("text",   
 # x = 55883,   
 # y = 0.5,   
 # label = "Poverty line",   
 # color = "black",   
 # hjust = -0.1,   
 # size = 3) +  
 annotate("segment", x = 70000, xend = 65000, y = 0.2, yend = 0.2,   
 arrow = arrow(length = unit(0.3, "cm")), color = "black") +  
 annotate("text", x = 72500, y = 0.2, label = "Shift due to shocks", hjust = 0)



# annotate("segment", x = 80000, xend = 75000, y = 0.4, yend = 0.4,   
 # arrow = arrow(length = unit(0.3, "cm")), color = "black") +  
 # annotate("text", x = 82500, y = 0.4, label = "Shift due to shocks", hjust = 0)

For energy prices

# Plot the cumulative distribution with left-facing arrows  
ggplot(ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_EM\_nzs\_energy, color = 'NZS + Energy Price')) +  
 stat\_ecdf(geom = "step") +  
 # stat\_ecdf(data = ca\_microsim\_cc,   
 # aes(x = aec\_r\_2030\_baseline\_lab\_avg, color = 'Baseline + Labor Productivity')) +  
 stat\_ecdf(data = ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_EM\_baseline\_energy, color = 'Baseline + Energy Price')) +  
 stat\_ecdf(data = ca\_microsim\_cc,   
 aes(x = aec\_r\_2030\_baseline, color = 'Baseline')) +  
 labs(  
 color = "Scenario Variant",   
 # title = "Comparison of Cumulative Consumption Distributions",   
 x = "Equivalized consumption (Dram)",   
 y = "Cumulative Probability") +  
 theme\_minimal() +  
 coord\_cartesian(xlim = c(40000, 110000)) +   
 scale\_x\_continuous(labels = scales::comma) +  
 # geom\_vline(xintercept = 55883,   
 # color = "red",   
 # linetype = "dotted",   
 # linewidth = 0.8) +  
 # annotate("text",   
 # x = 55883,   
 # y = 0.5,   
 # label = "Poverty line",   
 # color = "black",   
 # hjust = -0.1,   
 # size = 3) +  
 annotate("segment", x = 70000, xend = 65000, y = 0.2, yend = 0.2,   
 arrow = arrow(length = unit(0.3, "cm")), color = "black") +  
 annotate("text", x = 72500, y = 0.2, label = "Shift due to shocks", hjust = 0)



# annotate("segment", x = 80000, xend = 75000, y = 0.4, yend = 0.4,   
 # arrow = arrow(length = unit(0.3, "cm")), color = "black") +  
 # annotate("text", x = 82500, y = 0.4, label = "Shift due to shocks", hjust = 0)

### 8.5 Disaggregation of poverty measures

We bring back poverty to the people’s dataset.

# We extract household poverty designations from the data  
new\_poor <- ca\_microsim\_cc %>%   
 select(hhid,  
 weight\_2030\_baseline,  
 weight\_2030\_dry\_hot,  
 poor\_Avpovln2022\_2022,   
 poor\_Avpovln2022\_2030\_baseline,   
 poor\_Avpovln2022\_2030\_dry\_hot,  
 poor\_Avpovln2022\_2030\_nzs,  
 poor\_2030\_baseline\_lab\_avg,   
 poor\_2030\_baseline\_lab\_max,  
 poor\_2030\_baseline\_cc\_avg,  
 poor\_2030\_baseline\_cc\_max,   
 poor\_2030\_baseline\_lab\_cc\_avg,   
 poor\_2030\_baseline\_lab\_cc\_max,   
 poor\_2030\_baseline\_food1,  
 poor\_2030\_dry\_hot\_food2,  
 poor\_2030\_baseline\_lab\_cc\_foodPI,  
 poor\_2030\_baseline\_energy,  
 poor\_2030\_nzs\_energy,  
 poor\_2030\_EM\_baseline\_energy,  
 poor\_2030\_EM\_nzs\_energy  
 )  
  
# And merge them back into the people dataset  
pp\_microsim\_cc <- pp\_microsim\_cc %>%   
 select(-c(poor\_Avpovln2022,  
 weight\_2030\_baseline,  
 weight\_2030\_dry\_hot)) %>%   
 left\_join(new\_poor, join\_by(hhid)) %>%   
 mutate(female = if\_else(gender == 2, 1,0),  
 youth = if\_else(age < 15, 1, 0))

Let’s find homes where more than 50% of income comes from agriculture. We first find the fraction of household labor income that comes from agriculture.

ag\_labinc\_fraction <- pp\_microsim\_cc %>%   
 mutate(  
 ag\_lab\_income =  
 if\_else(  
 lmarket == 1,   
 monthly\_labor\_income\_2030\_baseline, NA)  
 ) %>%   
 group\_by(hhid) %>%  
 summarize(  
 ag\_labinc =  
 sum(ag\_lab\_income, na.rm = TRUE),  
 hh\_labinc =   
 sum(monthly\_labor\_income\_2030\_baseline, na.rm = TRUE)) %>%  
 mutate(ag\_lab\_fraction = if\_else(hh\_labinc == 0, 0, ag\_labinc/ hh\_labinc)) %>%   
 select(hhid, ag\_lab\_fraction)

And then we add ag income sources and evaluate if they are at least 50% of total income

ag\_income\_50 <- ic\_microsim\_cc %>%  
 left\_join(ag\_labinc\_fraction, join\_by(interview\_\_key==hhid)) %>%   
 rename(hhid = interview\_\_key) %>%   
 mutate(  
 across(  
 c(inc2\_2030\_baseline,  
 inc3\_2030\_baseline,  
 inc4,   
 totalinc\_2030\_baseline), ~replace\_na(., 0))) %>%  
 mutate(  
 ag\_income =  
 inc2\_2030\_baseline +  
 inc3\_2030\_baseline +  
 inc4 \* ag\_lab\_fraction,  
 ag\_fraction = if\_else(  
 totalinc\_2030\_baseline == 0, 0, ag\_income / totalinc\_2030\_baseline)  
 ) %>%  
 mutate(  
 is\_ag\_home = if\_else(ag\_fraction >= 0.5, "Ag. HH (>= 50%)", "Other HH")  
 ) %>%   
 select(hhid, is\_ag\_home)

We make a table to see who became poor.

### 8.6 The table

test <- pp\_microsim\_cc  
  
test <- test%>%  
 left\_join(ag\_income\_50, join\_by(hhid)) %>%   
# filter(rural ==1 & is\_ag\_home == "Ag. HH (>= 50%)") %>%   
 rename(poor\_original = poor\_Avpovln2022\_2030\_baseline,  
 poor\_dh = poor\_Avpovln2022\_2030\_dry\_hot,  
 poor\_nzs = poor\_Avpovln2022\_2030\_nzs,  
 poor\_lab = poor\_2030\_baseline\_lab\_avg,  
 poor\_cc = poor\_2030\_baseline\_cc\_avg,  
 poor\_lab\_cc = poor\_2030\_baseline\_lab\_cc\_avg,  
 poor\_foodPI = poor\_2030\_baseline\_lab\_cc\_foodPI,  
 poor\_dh\_food = poor\_2030\_dry\_hot\_food2,  
 poor\_b\_energy = poor\_2030\_baseline\_energy,  
 poor\_nzs\_energy = poor\_2030\_nzs\_energy,  
 poor\_EM\_b\_energy = poor\_2030\_EM\_baseline\_energy,  
 poor\_EM\_nzs\_energy = poor\_2030\_EM\_nzs\_energy  
 ) %>%  
 group\_by(poor\_nzs, poor\_EM\_nzs\_energy) %>%   
 summarize(no\_pp = sum(weight\_2030\_nzs, na.rm = TRUE),  
 female = sum(female\*weight\_2030\_nzs, na.rm = TRUE),  
 male = no\_pp - female,  
 youth = sum(youth\*weight\_2030\_nzs, na.rm = TRUE),  
 non\_youth = no\_pp - youth,  
 rural = sum(rural \*weight\_2030\_nzs, na.rm = TRUE),  
 urban = no\_pp - rural  
 )

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

test %>%   
 gt()

| poor\_EM\_nzs\_energy | no\_pp | female | male | youth | non\_youth | rural | urban |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | | | | | | | |
| 0 | 2376884.720 | 1304727.583 | 1072157.137 | 450102.388 | 1926782.332 | 906106.986 | 1470777.734 |
| 1 | 4549.283 | 2583.922 | 1965.362 | 1088.172 | 3461.112 | 3929.787 | 619.497 |
| 1 | | | | | | | |
| 1 | 451593.951 | 246849.815 | 204744.136 | 120979.071 | 330614.880 | 214857.559 | 236736.392 |

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

# foodpoor <- ca\_microsim\_cc %>%  
# mutate(new\_poor\_food\_base = if\_else(  
# poor\_2030\_baseline\_lab\_cc\_foodPI == 1 &  
# poor\_2030\_baseline\_lab\_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\_2022,   
# poor\_Avpovln2022\_2030\_baseline,   
# poor\_Avpovln2022\_2030\_dry\_hot,  
# poor\_2030\_baseline\_lab\_avg,   
# poor\_2030\_baseline\_lab\_max,  
# poor\_2030\_baseline\_cc\_avg,  
# poor\_2030\_baseline\_cc\_max,   
# poor\_2030\_baseline\_lab\_cc\_avg,   
# poor\_2030\_baseline\_lab\_cc\_max,   
# poor\_2030\_baseline\_food1,  
# poor\_2030\_dry\_hot\_food2,  
# poor\_2030\_baseline\_lab\_cc\_foodPI,  
# weight\_2030\_baseline,  
# weight\_2030\_dry\_hot,  
# hhsize)  
#   
# fp <-foodpoor %>%   
# group\_by(NAM\_1) %>%   
# summarize(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

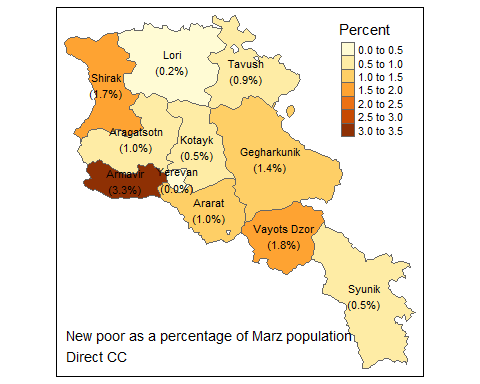
new\_poor\_scenarios <- pp\_microsim\_cc %>%  
 left\_join(ag\_income\_50, join\_by(hhid)) %>%   
 mutate(  
 poor\_baseline = poor\_Avpovln2022\_2030\_baseline,  
 poor\_dry\_hot = poor\_Avpovln2022\_2030\_dry\_hot,  
 poor\_nzs = poor\_Avpovln2022\_2030\_nzs,  
 new\_poor\_lab\_cc = if\_else(  
 poor\_Avpovln2022\_2030\_baseline == 0 &  
 poor\_2030\_baseline\_lab\_cc\_avg == 1,  
 1,  
 0  
 ),  
 new\_poor\_lab\_cc\_foodPI = if\_else(  
 poor\_Avpovln2022\_2030\_baseline == 0 &  
 poor\_2030\_baseline\_lab\_cc\_foodPI == 1,  
 1,  
 0  
 ),  
 new\_poor\_dry\_hot\_food2 = if\_else(  
 poor\_Avpovln2022\_2030\_dry\_hot == 0 &  
 poor\_2030\_dry\_hot\_food2 == 1,  
 1,  
 0  
 ),  
 new\_poor\_b\_energy = if\_else(  
 poor\_Avpovln2022\_2030\_baseline == 0 &  
 poor\_2030\_baseline\_energy == 1,  
 1,  
 0  
 ),  
 new\_poor\_nzs\_energy = if\_else(  
 poor\_Avpovln2022\_2030\_nzs == 0 &  
 poor\_2030\_nzs\_energy == 1,  
 1,  
 0  
 ),  
 new\_poor\_EM\_b\_energy = if\_else(  
 poor\_Avpovln2022\_2030\_baseline == 0 &  
 poor\_2030\_EM\_baseline\_energy == 1,  
 1,  
 0  
 ),  
 new\_poor\_EM\_nzs\_energy = if\_else(  
 poor\_Avpovln2022\_2030\_nzs == 0 &  
 poor\_2030\_EM\_nzs\_energy == 1,  
 1,  
 0  
 )  
 ) %>%  
 group\_by(NAM\_1) %>% # has to be NAM\_1 for the next chunk to work  
 summarize(  
 total\_population = sum(weight\_2030\_baseline, na.rm = TRUE),  
 poor\_baseline =  
 sum(poor\_baseline \* weight\_2030\_baseline, na.rm = TRUE),  
 poor\_dry\_hot =  
 sum(poor\_dry\_hot \* weight\_2030\_dry\_hot, na.rm = TRUE),  
 poor\_nzs =  
 sum(poor\_nzs \* weight\_2030\_nzs, na.rm = TRUE),  
 new\_p\_lab\_cc =  
 sum(new\_poor\_lab\_cc \* weight\_2030\_baseline, na.rm = TRUE),  
 new\_p\_lab\_cc\_foodPI =  
 sum(new\_poor\_lab\_cc\_foodPI \* weight\_2030\_baseline, na.rm = TRUE),  
 new\_p\_dry\_hot\_food2 =  
 sum(new\_poor\_dry\_hot\_food2 \* weight\_2030\_dry\_hot, na.rm = TRUE),  
 new\_p\_baseline\_energy =  
 sum(new\_poor\_b\_energy \* weight\_2030\_baseline, na.rm = TRUE),  
 new\_p\_nzs\_energy =   
 sum(new\_poor\_nzs\_energy \* weight\_2030\_nzs, na.rm = TRUE),  
 new\_p\_EM\_baseline\_energy =  
 sum(new\_poor\_EM\_b\_energy \* weight\_2030\_baseline, na.rm = TRUE),  
 new\_p\_EM\_nzs\_energy =   
 sum(new\_poor\_EM\_nzs\_energy \* weight\_2030\_nzs, na.rm = TRUE)  
 )  
  
write.table(new\_poor\_scenarios, "clipboard", sep="\t", row.names=FALSE)  
  
##write.table(test, "clipboard", sep="\t", row.names=FALSE)

And we create labels for our map.

new\_poor\_map <- adm1 %>% # previous chunk has to be grouped by NAM\_1  
 left\_join(new\_poor\_scenarios, join\_by(NAM\_1)) %>%  
 mutate(  
 new\_p\_lab\_cc\_pct = new\_p\_lab\_cc / total\_population \*  
 100,  
 new\_poor\_lab\_cc\_foodPI\_pct = new\_p\_lab\_cc\_foodPI /  
 total\_population \* 100,  
 new\_poor\_dry\_hot\_food2\_pct = new\_p\_dry\_hot\_food2 /  
 total\_population \* 100,  
 new\_poor\_baseline\_energy\_pct = new\_p\_baseline\_energy /  
 total\_population\*100,  
 new\_poor\_nzs\_energy\_pct = new\_p\_nzs\_energy / total\_population \* 100,  
 new\_poor\_EM\_baseline\_energy\_pct = new\_p\_EM\_baseline\_energy /  
 total\_population\*100,  
 new\_poor\_EM\_nzs\_energy\_pct = new\_p\_EM\_nzs\_energy / total\_population \* 100  
 ) %>%   
 mutate(  
 new\_p\_lab\_cc\_label = paste0(NAM\_1, "\n(", sprintf("%.1f%%", new\_p\_lab\_cc\_pct), ")"),  
 new\_p\_lab\_cc\_foodPI\_label = paste0(NAM\_1, "\n(", sprintf("%.1f%%", new\_poor\_lab\_cc\_foodPI\_pct), ")"),  
 new\_p\_dry\_hot\_food2\_label = paste0(NAM\_1, "\n(", sprintf("%.1f%%", new\_poor\_dry\_hot\_food2\_pct), ")"),  
 new\_p\_b\_energy\_label = paste0(NAM\_1, "\n(", sprintf("%.1f%%", new\_poor\_baseline\_energy\_pct), ")"),  
 new\_p\_nzs\_energy\_label = paste0(NAM\_1, "\n(", sprintf("%.1f%%", new\_poor\_nzs\_energy\_pct), ")"),  
 new\_p\_EM\_b\_energy\_label = paste0(NAM\_1, "\n(", sprintf("%.1f%%", new\_poor\_EM\_baseline\_energy\_pct), ")"),  
 new\_p\_EM\_nzs\_energy\_label = paste0(NAM\_1, "\n(", sprintf("%.1f%%", new\_poor\_EM\_nzs\_energy\_pct), ")")  
 )

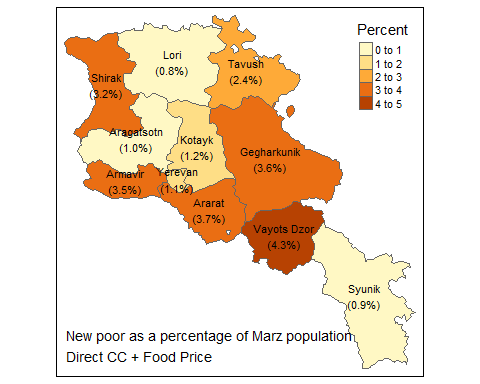
Let’s map different scenarios.

tm\_shape(new\_poor\_map)+  
 tm\_polygons("new\_p\_lab\_cc\_pct", title="Percent", legend.show = TRUE) +  
 tm\_text(c("new\_p\_lab\_cc\_label"), size = .7, col = "black")+  
 tm\_layout(legend.position = c("right", "top"),  
 #legend.outside = TRUE,  
 title= "New poor as a percentage of Marz population\nDirect CC",  
 title.position = c('left', 'bottom'),  
 title.size = 0.9)



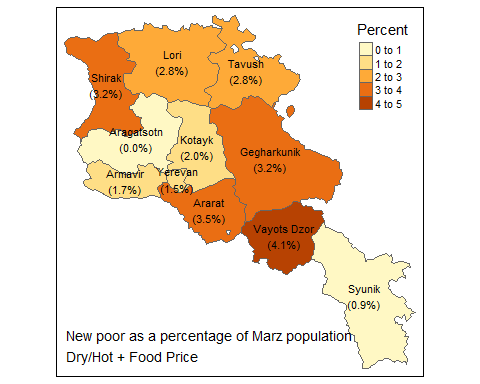
And the second variant

tm\_shape(new\_poor\_map)+  
 tm\_polygons("new\_poor\_lab\_cc\_foodPI\_pct",  
 title="Percent",   
 legend.show = TRUE) +  
 tm\_text(c("new\_p\_lab\_cc\_foodPI\_label"), size = .7, col = "black")+  
 tm\_layout(legend.position = c("right", "top"),   
 title= "New poor as a percentage of Marz population\nDirect CC + Food Price",  
# outer.margins=c(.10,.10, .10, .10),   
 title.position = c('left', 'bottom'),  
 title.size = 0.9)



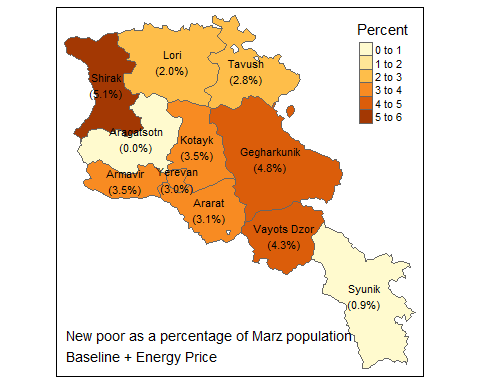
Second variant b

tm\_shape(new\_poor\_map)+  
 tm\_polygons("new\_poor\_dry\_hot\_food2\_pct",  
 title="Percent",   
 legend.show = TRUE) +  
 tm\_text(c("new\_p\_dry\_hot\_food2\_label"), size = .7, col = "black")+  
 tm\_layout(legend.position = c("right", "top"),   
 title= "New poor as a percentage of Marz population\nDry/Hot + Food Price",  
# outer.margins=c(.10,.10, .10, .10),   
 title.position = c('left', 'bottom'),  
 title.size = 0.9)



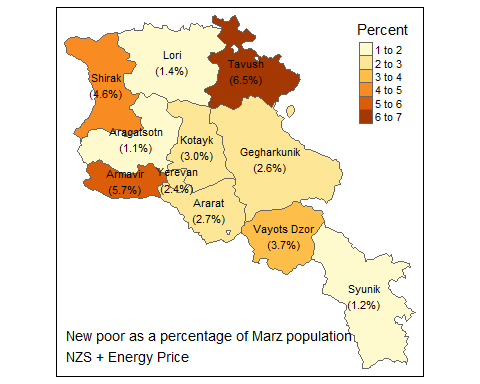
And the third variant

tm\_shape(new\_poor\_map)+  
 tm\_polygons("new\_poor\_baseline\_energy\_pct",  
 title="Percent",   
 legend.show = TRUE) +  
 tm\_text(c("new\_p\_b\_energy\_label"), size = .7, col = "black")+  
 tm\_layout(legend.position = c("right", "top"),   
 title= "New poor as a percentage of Marz population\nBaseline + Energy Price",  
# outer.margins=c(.10,.10, .10, .10),   
 title.position = c('left', 'bottom'),  
 title.size = 0.9)



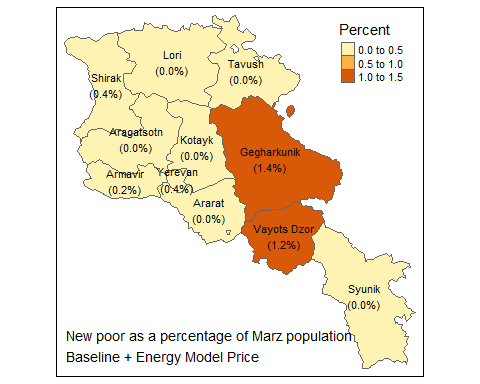
And the fourth variant

tm\_shape(new\_poor\_map)+  
 tm\_polygons("new\_poor\_nzs\_energy\_pct",  
 title="Percent",   
 legend.show = TRUE) +  
 tm\_text(c("new\_p\_nzs\_energy\_label"), size = .7, col = "black")+  
 tm\_layout(legend.position = c("right", "top"),   
 title= "New poor as a percentage of Marz population\nNZS + Energy Price",  
# outer.margins=c(.10,.10, .10, .10),   
 title.position = c('left', 'bottom'),  
 title.size = 0.9)



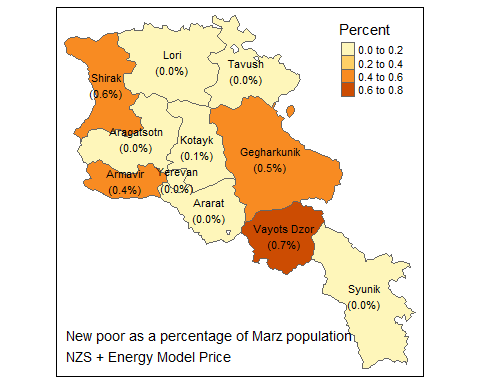
Fifth

tm\_shape(new\_poor\_map)+  
 tm\_polygons("new\_poor\_EM\_baseline\_energy\_pct",  
 title="Percent",   
 legend.show = TRUE) +  
 tm\_text(c("new\_p\_EM\_b\_energy\_label"), size = .7, col = "black")+  
 tm\_layout(legend.position = c("right", "top"),   
 title= "New poor as a percentage of Marz population\nBaseline + Energy Model Price",  
# outer.margins=c(.10,.10, .10, .10),   
 title.position = c('left', 'bottom'),  
 title.size = 0.9)



Sixth

tm\_shape(new\_poor\_map)+  
 tm\_polygons("new\_poor\_EM\_nzs\_energy\_pct",  
 title="Percent",   
 legend.show = TRUE) +  
 tm\_text(c("new\_p\_EM\_nzs\_energy\_label"), size = .7, col = "black")+  
 tm\_layout(legend.position = c("right", "top"),   
 title= "New poor as a percentage of Marz population\nNZS + Energy Model Price",  
# outer.margins=c(.10,.10, .10, .10),   
 title.position = c('left', 'bottom'),  
 title.size = 0.9)



Now let’s show average losses by decile as a percentage of total spending.

avg\_scenario\_losses <- ca\_microsim\_cc %>%   
 mutate(  
 totc\_loss\_lab\_cc = if\_else(  
 (totc\_2030\_baseline\_lab\_cc\_avg - totc\_2030\_baseline)< 0,  
 (totc\_2030\_baseline\_lab\_cc\_avg - totc\_2030\_baseline)/  
 totc\_2030\_baseline,NA),  
 totc\_loss\_lab\_cc\_foodPI = if\_else(  
 (totc\_2030\_baseline\_lab\_cc\_foodPI - totc\_2030\_baseline)< 0,  
 (totc\_2030\_baseline\_lab\_cc\_foodPI - totc\_2030\_baseline)/  
 totc\_2030\_baseline,NA)  
 ) %>%  
 group\_by(decile) %>%   
 summarize(no\_hh =   
 round(  
 sum(  
 weight\_2030\_baseline, na.rm = TRUE), digits = 0  
 ),  
 avg\_totc =   
 round(  
 weighted.mean(  
 totc\_2030\_baseline, weight\_2030\_baseline, na.rm = TRUE), digits = 2  
 ),  
 avg\_totc\_usd =   
 round(  
 weighted.mean(  
 totc\_2030\_baseline, weight\_2030\_baseline, na.rm = TRUE)\*er, digits = 1  
 ),  
 avg\_loss\_lab\_cc =   
 round(  
 weighted.mean(  
 totc\_loss\_lab\_cc,   
 weight\_2030\_baseline,   
 na.rm = TRUE),   
 digits = 4  
 ),  
 avg\_loss\_lab\_cc\_foodPI =   
 round(  
 weighted.mean(  
 totc\_loss\_lab\_cc\_foodPI,   
 weight\_2030\_baseline,   
 na.rm = TRUE), digits = 4),  
 # avg\_loss\_b\_energy =   
 # round(  
 # weighted.mean(  
 # totc\_loss\_lab\_cc\_foodPI,   
 # weight\_2030\_baseline,   
 # na.rm = TRUE), digits = 4),  
 # avg\_loss\_nzs\_energy =   
 # round(  
 # weighted.mean(  
 # totc\_loss\_lab\_cc\_foodPI,   
 # weight\_2030\_baseline,   
 # na.rm = TRUE), digits = 4),  
 # avg\_loss\_EM\_b\_energy =   
 # round(  
 # weighted.mean(  
 # totc\_loss\_lab\_cc\_foodPI,   
 # weight\_2030\_baseline,   
 # na.rm = TRUE), digits = 4),  
 # avg\_loss\_EM\_nzs\_energy =   
 # round(  
 # weighted.mean(  
 # totc\_loss\_lab\_cc\_foodPI,   
 # weight\_2030\_baseline,   
 # na.rm = TRUE), digits = 4)  
 )  
  
##write.table(avg\_scenario\_losses, "clipboard", sep="\t", row.names=FALSE)  
  
avg\_scenario\_losses %>%   
 gt()

| Decile of aec\_r, with pweight | no\_hh | avg\_totc | avg\_totc\_usd | avg\_loss\_lab\_cc | avg\_loss\_lab\_cc\_foodPI |
| --- | --- | --- | --- | --- | --- |
| 1 | 54366 | 150233.1 | 347.0 | -0.0088 | -0.0221 |
| 2 | 59969 | 173070.4 | 399.8 | -0.0109 | -0.0271 |
| 3 | 61790 | 197378.1 | 455.9 | -0.0100 | -0.0238 |
| 4 | 66999 | 208086.4 | 480.7 | -0.0091 | -0.0266 |
| 5 | 73341 | 210340.4 | 485.9 | -0.0110 | -0.0288 |
| 6 | 80796 | 213555.7 | 493.3 | -0.0113 | -0.0329 |
| 7 | 80304 | 236754.2 | 546.9 | -0.0097 | -0.0327 |
| 8 | 88195 | 244050.4 | 563.8 | -0.0119 | -0.0362 |
| 9 | 96945 | 254350.8 | 587.6 | -0.0090 | -0.0290 |
| 10 | 120886 | 310809.4 | 718.0 | -0.0105 | -0.0245 |

Energy price index interpolation

# Sample data  
data <- data.frame(  
 year = c(2020, 2030),  
 value = c(100.00000,103.18920)   
)  
  
# Define the years for interpolation  
years <- seq(2020, 2030, by = 1)  
  
# Perform linear interpolation  
interpolated\_values <- approx(data$year, data$value, xout = years)  
  
# Create a data frame with the interpolated results  
interpolated\_data <- data.frame(  
 year = interpolated\_values$x,  
 value = interpolated\_values$y  
)  
  
# Display the result  
print(interpolated\_data)

year value  
1 2020 100.0000  
2 2021 100.3189  
3 2022 100.6378  
4 2023 100.9568  
5 2024 101.2757  
6 2025 101.5946  
7 2026 101.9135  
8 2027 102.2324  
9 2028 102.5514  
10 2029 102.8703  
11 2030 103.1892

The inquiry is who is getting hit the most from changes to energy prices.

energy\_shares <- ca\_microsim\_cc %>%   
 left\_join(ag\_income\_50, join\_by(hhid)) %>%   
 mutate(  
 shr\_energy = if\_else(totc\_2022==0,NA, (hous\_23+hous\_36\_b+hous\_29\_b)/totc\_2022)  
 ) %>%   
 group\_by(decile, is\_ag\_home) %>%   
 summarize(no\_hh= sum(weight\_2022, na.rm = TRUE),  
 shr\_e = weighted.mean(shr\_energy, weight\_2022, na.rm=TRUE))

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

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

## 9 End

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

Strzepek, K., Boehlert, B., Castillo, D., & Smet, K. (2024). *Estimating the Economic Impacts of Climate Change in Armenia*. Industrial Economics, Incorporated / The World Bank.