Armenia CCDR Microsimulation

Renato Vargas Julie Rozenberg Colin Lenoble
Natsuko Kiso Nozaki Thomas Farole

Table of contents

0.1 Introduction

In this calculation file, we "age" the household survey according to demographic projections and different macroeconomic scenarios to explore the impact of climate-related risks and policy measures on the consumption expenditure distribution.

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

```r
# 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.

0.2 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.

```r
# Clean workspace
rm(list = ls())

# Armenia country ISO code
iso <- "ARM"

# Survey year
surveyyear <- 2022

# Exchange rate USD per dram
er <- 0.002310
```

We call the appropriate libraries.

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

```
# Load packages
library(tidyverse) # includes dplyr, ggplot2 and others
library(haven)     # to read SPSS and Stata datasets
library(readxl)    # to read from MS-Excel
library(openxlsx)  # to write to MS-Excel.
library(gt)        # pretty tables
library(car)       # Companion to applied regression
library(modelr)    # regression models
#library(ebal)      # Entropy reweighting
library(anesrake)  # Raking reweighting
#library(weights)   # Weigthed survey statistics
library(janitor)   # pretty subtotals
library(broom)     # More regressions
library(purrr)     # map vectors (aggregation)
library(zoo)       # Calculate moving window average and max value


# Geopackages
library(sf)        # to read and write shapefile maps
library(terra)     # to perform geocalculations
library(tmap)      # for static and interactive maps
```

Stata integration

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

0.3 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).

```r
# Original SPSS datasets
# Households (hh)
hh <- read_sav(
  "data/ARM-HH-survey/original-spss-files/ILCS-ARM-2022-Households.sav")
# Persons (pp)
pp <- read_sav(
  "data/ARM-HH-survey/original-spss-files/ILCS-ARM-2022-Persons.sav")

# Processed WB datasets
# Consumption aggregate at household level (ca)
ca  <- read_dta("data/ARM-HH-survey/CONSAGG2022.dta")
# Processed income at household level (ic)
ic  <- read_dta("data/ARM-HH-survey/totinc.dta")
```

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.

```r
# 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.

```r
# Population projections UN 2022
population_projections <- read_dta("data/UN2022_population.dta") %>%
  filter(country == iso)
```

Macro scenario dataset

```r
scenario_file <- "data/ARM-Microsimulation/ARM_MacroScenarioInformation.xlsx"
scenario_varlist <- read_xlsx(
  "data/ARM-Microsimulation/ARM_Macro_varlist.xlsx")
prices_2030 <-
  read.csv("data/ARM-Microsimulation/prices2030.csv")
```

Economic sectors.

```
sectors <- read_xlsx("data/ARM-HH-survey/economic_activity_codes.xlsx")
```

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

```
# Geodata
# Armenia marzes or administrative level 1 shapefile
adm1 <- read_sf("data/ARM-Geodata/ARM-ADM1.shp") %>%
  select(NAM_1, COD_HH_SVY, geometry) %>%
    # Make sure that names match the rest of datasets
  mutate(NAM_1 = if_else(NAM_1 == "Gergharkunik", "Gegharkunik", NAM_1))
names(adm1)[2] <- "hh_02"

tm_shape(adm1)+
  tm_polygons("NAM_1", legend.show = FALSE) +
  tm_text("NAM_1", size = 3/4)
```

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

```
hh <- hh %>%
  left_join(adm1, join_by(hh_02 == hh_02)) %>%
  select(-geometry)

ic <- ic %>%
  left_join(adm1, join_by(hh_02 == hh_02)) %>%
  select(-geometry)
```

Labor productivity

```
file <- r"(data/ARM-Microsimulation/LaborProductivityChanges.xlsx)"
sheets <- excel_sheets(file)

# Use lapply to read and process each sheet
heat_l_pdcty <- lapply(sheets, function(sheet) {
  info <- read_excel(
    file,
    sheet = sheet,
    col_names = TRUE,
    col_types = c("text", "text", "numeric", "text", "numeric")
  )
  info$sector <- sheet
  return(info)
})

# Bind all data frames in the list into a single data frame
heat_l_pdcty <- bind_rows(heat_l_pdcty)
```

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

```
buildings_aal <-
  read_xlsx("data/ARM-Vulnerability-Analysis/Data_AAL_AAE.xlsx",
            sheet = "Building_AAL") %>%
    # Make sure that names match the rest of datasets
  mutate(NAM_1 = if_else(NAM_1 == "Gergharkunik", "Gegharkunik", NAM_1))
buildings_1in100 <-
  read_xlsx("data/ARM-Vulnerability-Analysis/Data_AAL_AAE.xlsx",
            sheet = "Building_1in100")
crops_productivity <-
  read.csv("data/ARM-Vulnerability-Analysis/ARM_crops_combined_REF_shock_admin1.csv") %>%
  rename(NAM_1 = Province)
```

```
crops_aal <-
  read_xlsx("data/ARM-Vulnerability-Analysis/Data_AAL_AAE.xlsx",
            sheet = "Agriculture_AAL")
crops_1in100 <-
  read_xlsx("data/ARM-Vulnerability-Analysis/Data_AAL_AAE.xlsx",
            sheet = "Agriculture_1in100")
```

0.4 Data preparation income outliers and missings

0.4.1 Household consumption aggregates and characteristics

Initial necessary variables.

```
consumption_aggregates <- ca %>%
  mutate(rural = ifelse(urb_rur == 2, 1, 0),  # Create rural indicator
         yhh = totc,              # Total household expenditure
         wgt_adj = pweight) %>%        # Make a copy of the weight variable
  select(hhid, rural, hhsize,hhsize_R, marz, aepc, yhh, wgt_adj, weight,
         Foodpovln2022, Lpovln2022, Upovln2022, Avpovln2022,
         poor_Foodpovln2022, poor_Lpovln2022, poor_Upovln2022,
         poor_Avpovln2022, decile )  # Keep only necessary columns
```

0.4.2 Demographic characteristics, education, Labor Force

Here the original code calls for Zone data, which is not present in our dataset, due to the different administrative structure of Armenia. However, we use `hh_01_code` (settlement) for this purpose.

```
# Zone data
zone_data <- hh %>%
  select(interview__key, hh_01_code, hh_02, hh_03, NAM_1) %>%
  mutate(
    hhid = interview__key, # Household id
    zone = hh_01_code,     # Settlement
    marz = hh_02,          # Marz
    NAM_1 = NAM_1,          # Marz name
    urb_rur = hh_03        # Urban / rural
  )
```

Demographic data, merge with zone data Note that ed_03 (educy) below is not years of education, but education level (primary, general, secondary, etc.) However, it is ordered in a way that higher levels imply more years of education. We perform several steps within the first pipe call.

```
pp_microsim <- pp %>%
  rename(hhid = interview__key) %>%
  left_join(zone_data, join_by( hhid == hhid))  %>%
  mutate(# Demographic characteristics
         pid = paste0(interview__key, "-",
                      str_pad(mem_001__id, 2, pad = "0")), # Unique person id
         gender = mem_02,
         age = mem_05,
         head = ifelse(mem_03 == 1, 1, 0),
         # Education level
         educy = ifelse(is.na(ed_03) | ed_03 == 8, 0, ed_03),
         # Labor Force Status
         lstatus = case_when(
           # 1. Employed
           est_03 == 1 | est_04 == 1 | est_05 == 1 | est_06 == 1 | est_08 == 1 ~ 1L,
           # 2. Unemployed (available, and searching)
           est_10 == 1 ~ 2L,
           # 3. Inactive (available, not searching)
           est_10 == 2 ~ 3L,
           # Out of the labor force
           .default = 4L # Default to OLF
         ),
         employed = (lstatus == 1),
         # Salaried status (1. paid employee; 2 self-employed)
         salaried = ifelse(!is.na(emp_11a), 1L,
                           ifelse(is.na(emp_11a) & employed == TRUE, 0L, NA_integer_))
         ) %>%
  rename(rel = mem_03) # %>%
  # select(hhid, pid, gender, age, head, rel, zone, marz, urb_rur, educy,
  #        lstatus, employed, salaried, )
```

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

```
pp_microsim <- pp_microsim %>%
  mutate(calif = case_when(
    educy >= 0 & educy <= 2 ~ "None - General",
```

```r
    educy > 3 & educy <= 7 ~ "Secondary - Vocational",
    educy > 7 & educy <= 11 ~ "Higher +",
    TRUE ~ NA_character_  # This handles any values outside the specified ranges
  ))

# View the first few rows to confirm the recoding
head(pp_microsim[,c("calif")])
```

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

Count the number of employed persons by household.

```r
pp_microsim <- pp_microsim %>%
  mutate(employed = (lstatus == 1)) %>%
  group_by(hhid) %>%
  mutate(employed_hh = sum(employed, na.rm = TRUE)) %>%  # Count within each household
  ungroup()
```

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

Primary and Secondary Job income

- **emp_11** 11.How much was %rostertitle%'s payment for wages/salary/income for last month?
- **emp_12** 12.What period of time was the wage/income for?
- **emp_25** 25.How much was %rostertitle%'s payment for wages/salary/income for last month?
- **emp_26** 26.What period of time was the wage/income for?

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

```r
pp_microsim <- pp_microsim %>%
  # Labor income primary job
  mutate(annual_labor_income_primary = case_when(
    emp_12 == 1 ~ emp_11 * 365,
    emp_12 == 2 ~ (emp_11/7) * 365,  # Assuming weekly rate
    emp_12 == 3 ~ (emp_11/14) * 365,
    emp_12 == 4 ~ emp_11 * 12,
    emp_12 == 5 ~ emp_11 * 2,
    emp_12 == 6 ~ emp_11,
    emp_12 == 7 ~ NA
  ))   %>%
  # Labor income secondary job
  mutate(annual_labor_income_secondary = case_when(
    emp_26 == 1 ~ emp_25 * 365,
    emp_26 == 2 ~ (emp_25/7) * 365,  # Assuming weekly rate
    emp_26 == 3 ~ (emp_25/14) * 365,
    emp_26 == 4 ~ emp_25 * 12,
    emp_26 == 5 ~ emp_25 * 2,
    emp_26 == 6 ~ emp_25,
    emp_26 == 7 ~ NA
  )) %>%
  # Annual labor total in thousands of dram
  mutate(annual_labor_total = (coalesce(annual_labor_income_primary, 0) +
         coalesce(annual_labor_income_secondary, 0))/1000)

# Restore annual_labor_total to NA if both NA
pp_microsim <- pp_microsim %>%
  mutate(annual_labor_total =
           if_else(
             is.na(annual_labor_income_primary)
             & is.na(annual_labor_income_secondary),
         NA,
         annual_labor_total))
```

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

```r
total_employed_no_income <- pp_microsim %>%
  filter(employed == TRUE & is.na(annual_labor_total)) %>%
  nrow()

total_employed <- pp_microsim %>%
  filter(employed == TRUE) %>%
```

```
  nrow()

percent_employed_no_income <- (total_employed_no_income / total_employed) * 100

print(percent_employed_no_income)
```

```
[1] 28.57496
```

Let's flag outliers now

```
pp_microsim <- pp_microsim  %>%
  # Filter for employed and positive income
  #filter(employed == TRUE & annual_labor_total > 0) %>%
  mutate(
    sd = sd(annual_labor_total, na.rm = TRUE),   # Calculate standard deviation
    d = annual_labor_total / sd,
    # Combined outlier condition
    outlier = (d > 5) | (employed == TRUE & annual_labor_total == 0),
    # Mark potential missings
    missings = if_else(employed == TRUE, is.na(annual_labor_total), NA)
  )
```

Economic sector

```
pp_microsim <- pp_microsim %>%
  mutate(emp_04 = as.integer(emp_04)) %>%
  left_join(sectors, join_by("emp_04" == "economic_activity_code") ) %>%
  rename(sector = ea_shortcode)
```

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

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

```
pp_microsim <- pp_microsim %>%
  group_by(hhid) %>%
  mutate(
    # Create a temporary variable 'other_sector' which captures the sector of any employed i
    other_sector = if_else(employed == TRUE & !is.na(sector), sector, NA_real_)
  ) %>%
  # Use 'fill' to propagate 'other_sector' values within the household
  fill(other_sector, .direction = "downup") %>%
```

```
  mutate(
    # Impute missing 'sector' values based on the 'other_sector'
    sector = if_else(is.na(sector) & employed == TRUE, other_sector, sector)
  ) %>%
  # Drop the temporary 'other_sector' variable
  select(-other_sector) %>%
  ungroup()
```

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

```
pp_microsim <- pp_microsim %>%
  mutate(sector = if_else(is.na(sector) & employed == TRUE, 3, sector))
```

Step 4: Label the sector variable.

```
pp_microsim <- pp_microsim %>%
  mutate(sector_name = factor(sector, levels = c(1, 2, 3),
                              labels = c("Agriculture",
                                         "Manufacturing", "Services"))
         )
```

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

```
pp_microsim <- pp_microsim %>%
  mutate(lstatus = as.numeric(lstatus),
         sector = if_else(lstatus == 4, as.character(NA), as.character(sector)),
         industry = as.factor(sector)) %>%
   mutate(sector_w = sector) # We need this for reweighting and not messing up
                             # and not mess up the regression below.
```

0.4.3 The regression

Prepare the data.

```
pp_microsim <- pp_microsim %>%
  mutate(
    educy2 = educy^2,
    age2 = age^2,
```

```
    male = case_when(
      gender == 1 ~ 1,
      gender == 2 ~ 0
    ),
    lnlab = log(annual_labor_total),
    simuli = NA_real_ # Initialize simuli
  )
```

Filter the data for regression conditions.

```
regression_data <- pp_microsim %>%
  filter(employed == TRUE & outlier == FALSE & missings == FALSE)
```

Regression model.

```
model <- lm(lnlab ~ age + gender + educy + age2 + marz + sector,
            data = regression_data)
```

Predict for specific conditions

```
pp_microsim <- pp_microsim %>%
  mutate(
    condition = (lstatus == 1 & (outlier == TRUE | missings == TRUE))
  )
```

Applying predictions.

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

temp2 equivalent - Note: 'type = "response"' might be needed depending on model type.

```
pp_microsim$simuli[pp_microsim$condition==TRUE] <- exp(
  predict(model, pp_microsim[pp_microsim$condition==TRUE, ], type = "response"))
```

Handling negative values in 'simuli'.

```
pp_microsim <- pp_microsim %>%
  mutate(
    simuli = if_else(simuli < 0, 0, simuli)
  )
```

There were 8 observations that met the criteria:

We will replace `annual_labor_total` with this value for those observations.

```
pp_microsim <- pp_microsim %>%
  mutate(annual_labor_total = if_else(
    employed == TRUE & (outlier == TRUE | missings == TRUE),
    simuli, annual_labor_total))

# And get monthly incomes for everyone
pp_microsim <- pp_microsim %>%
  mutate(monthly_labor_income = annual_labor_total / 12)
```

Merging datasets.

```
pp_microsim <- pp_microsim %>%
  left_join(consumption_aggregates, by = "hhid")
```

0.4.4 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 `.`.

```r
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:

```r
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.

```r
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.

```r
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)
```

0.5 UN Population Projections

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

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

```
population_projections <- population_projections  %>%
  filter(country == iso)
```

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

```
population_projections <- population_projections %>%
  group_by(Variant, country, cohort) %>%
  summarise(across(starts_with(c("yf", "ym")), sum)) %>%
  ungroup()
```

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

```
population_projections <- pivot_longer(population_projections,
                              cols = starts_with(c("yf", "ym")),
                              names_to = c(".value", "year"),
                              names_pattern = "(yf|ym)(.*)")
```

Creating new variable `total_population` as the sum of `yf` and `ym`. Dropping `country` variables.

```
population_projections <- population_projections %>%
  mutate(total_population = yf + ym) %>%
  select( -country) %>%
  mutate(year = as.numeric(year))
```

Summarizing the year to find the range.

```
minyear <- surveyyear # Make sure `surveyyear` is correctly defined
maxyear <- max(as.numeric(population_projections$year))

# Print the year range as a check
print(paste("Min Year:", minyear, "- Max Year:", maxyear))
```

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

```
# With minyear and maxyear defined above
# Initialize a list to store growth data
pop_growth <- list()

# Loop over variants
variants <- unique(population_projections$Variant)
for (variant in variants) {
  for (t in minyear:maxyear) {

    # Calculate population for year t
    pop_t <- population_projections %>%
      filter(year == t, Variant == variant) %>%
      summarise(sum_pop = sum(total_population)) %>%
      pull(sum_pop)

    # Calculate population for base year
    pop_base <- population_projections %>%
      filter(year == minyear, Variant == variant) %>%
      summarise(sum_pop = sum(total_population)) %>%
      pull(sum_pop)

    # Calculate growth rate and store in list with dynamic naming
    growth_rate <- pop_t / pop_base
    pop_growth[[paste0(t, "_", variant)]] <- list(
      growth_rate = growth_rate, pop_t = pop_t
      )
```

```
  }
}

# Convert list to dataframe
pop_growth <- do.call(rbind, lapply(names(pop_growth), function(x) {
  # Extract year and variant from the name
  parts <- unlist(strsplit(x, "_"))
  year <- as.integer(parts[1])
  variant <- parts[2]

  # Create a tibble for each entry
  tibble(year = year,
         variant = variant,
         total_population = pop_growth[[x]]$pop_t,
         pop_growth_rate = pop_growth[[x]]$growth_rate)
}))

# Arrange the dataframe for better readability
pop_growth <- arrange(pop_growth, variant, year)

# Display the first few rows of the dataframe
pop_growth[c(1:09),]
```

```
# A tibble: 9 x 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)
```

0.6 Macro Scenarios

The following code accomplishes the following:

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

```
# Macro Scenario File imported in "Datasets" section (scenario_file)
sheets <- excel_sheets(scenario_file)

# Define the names of the scenarios and the variants
scenarios <- sheets[c(1,2)] # modify list with the tab numbers or names with scenarios

# Create an empty list to store data frames for each scenario
scen_data_list <- list()

# Import data for each scenario and store it in the list
for (i in seq_along(scenarios)) {
  sheet_data <- read_excel(scenario_file,
                           sheet = scenarios[i],
                           range = "B3:AT31",
                           col_names = FALSE)
  sheet_data$scenario_id <- scenarios[i]
  colnames(sheet_data) <- scenario_varlist$var_short_name
  scen_data_list[[i]] <- sheet_data
}
```

```
New names:
New names:
* `` -> `...1`
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
* `` -> `...6`
* `` -> `...7`
* `` -> `...8`
* `` -> `...9`
* `` -> `...10`
```

```
* `` -> `...11`
* `` -> `...12`
* `` -> `...13`
* `` -> `...14`
* `` -> `...15`
* `` -> `...16`
* `` -> `...17`
* `` -> `...18`
* `` -> `...19`
* `` -> `...20`
* `` -> `...21`
* `` -> `...22`
* `` -> `...23`
* `` -> `...24`
* `` -> `...25`
* `` -> `...26`
* `` -> `...27`
* `` -> `...28`
* `` -> `...29`
* `` -> `...30`
* `` -> `...31`
* `` -> `...32`
* `` -> `...33`
* `` -> `...34`
* `` -> `...35`
* `` -> `...36`
* `` -> `...37`
* `` -> `...38`
* `` -> `...39`
* `` -> `...40`
* `` -> `...41`
* `` -> `...42`
* `` -> `...43`
* `` -> `...44`
* `` -> `...45`
```

```
# Combine all data frames into one
combined_data <- bind_rows(scen_data_list)

# Rename population_m from the data set because we will use
# UN pop projections from the other data set.
combined_data <- combined_data %>%
```

```
  rename(population_m_macrodata = population_m)

# Calculate real wages
combined_data <- combined_data %>%
  mutate(rwage_agr_m_amd = wage_agr_m_amd / cpi,
         rwage_man_m_amd = wage_man_m_amd / cpi,
         rwage_ser_m_amd = wage_ser_m_amd / cpi)

pop_data <- population_projections %>%
  group_by(Variant, year) %>%
  summarise(female = sum(yf),
            male = sum(ym),
            total_population = sum(total_population) ) %>%
  ungroup()
```

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

```
# Filter population data to macro model years
pop_data <- pop_data %>%
  filter(year <= max(combined_data$year),
         Variant == variants[7])
# Merge the combined data with population projections
macro_data <- combined_data %>%
  left_join(pop_data, by = c("year"))

# Calculate consumption per capita and other totals
macro_data <- macro_data %>%
  mutate(
    consumption_pc = consumption_b_amd / (total_population),
    total_employment = lab_agr_1000p + lab_man_1000p + lab_ser_1000p,
    employment_rate = working_age_pop_m / total_population
    )

# Function to add growth rate columns directly in the dataframe
calculate_growth <- function(data, value_column) {
  growth_col_name <- paste0(value_column, "_growth") # dynamic name for growth column
  data %>%
    arrange(year) %>%
    group_by(Variant, scenario_id) %>%
    mutate(
```

```
      base_value = first(!!sym(value_column)),
      !!sym(growth_col_name) := !!sym(value_column) / base_value
    ) %>%
    select(-base_value) %>% # optionally remove base_value column if not needed
    ungroup()
}

# Columns to calculate growth for
value_columns <- c(
  "gdp_b_amd",           # GDP
  "consumption_b_amd",   # Consumption
  "consumption_pc",      # Consumption PC
  "remittances_b_amd",   # Remittances
  "total_employment",    # Employment
  "employment_rate",     # Employment rate
  "working_age_pop_m",   # Working age population
  "va_agr_b_amd",        # Value added agriculture
  "va_man_b_amd",        # Value added manufacturing
  "va_ser_b_amd",        # Value added services
  "wage_agr_m_amd",      # Nominal wage agriculture
  "wage_man_m_amd",      # Nominal wage manufacturing
  "wage_ser_m_amd",      # Nominal wage services
  "rwage_agr_m_amd",     # Real wage agriculture
  "rwage_man_m_amd",     # Real wage manufacturing
  "rwage_ser_m_amd"      # Real wage services
  )

# Applying the growth calculation to the macro_data for each column
for (col in value_columns) {
  macro_data <- calculate_growth(macro_data, col)
}

# Now `macro_data` will have growth rate columns for each of the variables listed
# We rearrange the dataset for clarity
macro_data <- macro_data %>%
  relocate(scenario_id, Variant, .before = year) %>%
  arrange(scenario_id, Variant, year)
```

0.7 Reweighting of the dataset

0.7.1 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.

```r
population_projections <- population_projections %>%
  # filter(Variant == "Medium") %>%
  # Recoding cohorts into ordered factors
    mutate(cohort_short = factor(case_when(
    cohort %in% c("P0004", "P0509","P1014",
                  "P1519","P2024", "P2529") ~ "P0029",
    cohort %in% c("P3034", "P3539") ~ "P3039",
    cohort %in% c("P4044", "P4549") ~ "P4049",
    cohort %in% c("P5054", "P5559") ~ "P5059",
    cohort %in% c("P6064", "P6569","P7074", "P7579",
                  "P8084", "P8589", "P9094", "P9599",
                  "P100up") ~ "P60up"
  ), levels = c("P0029", "P3039",
                "P4049", "P5059", "P60up"))) %>%
  # mutate(cohort = factor(case_when(
    # cohort %in% c("P0004", "P0509") ~ "P0009",
    # cohort %in% c("P1014", "P1519") ~ "P1019",
    # cohort %in% c("P2024", "P2529") ~ "P2029",
  #   cohort %in% c("P3034", "P3539") ~ "P3039",
  #   cohort %in% c("P4044", "P4549") ~ "P4049",
  #   cohort %in% c("P5054", "P5559") ~ "P5059",
  #   cohort %in% c("P6064", "P6569") ~ "P6069",
  #   cohort %in% c("P7074", "P7579", "P8084", "P8589", "P9094", "P9599", "P100up") ~ "P70up"
  # ), levels = c("P0009", "P1019", "P2029", "P3039", "P4049", "P5059", "P6069", "P70up"))) %
  # Convert factor 'cohort' to numeric codes
  mutate(cohort_code = as.integer(cohort_short))

# Checking the resulting dataset
print(pop_data)
```

```
# A tibble: 60 x 5
   Variant  year female  male total_population
   <chr>   <dbl>  <dbl> <dbl>            <dbl>
```

```
 1 Medium   1991  1867. 1750.            3618.
 2 Medium   1992  1850. 1724.            3575.
 3 Medium   1993  1799. 1658.            3457.
 4 Medium   1994  1763. 1610.            3374.
 5 Medium   1995  1741. 1581.            3323.
 6 Medium   1996  1731. 1568.            3299.
 7 Medium   1997  1719. 1552.            3271.
 8 Medium   1998  1705. 1535.            3241.
 9 Medium   1999  1689. 1517.            3206.
10 Medium   2000  1672. 1496.            3169.
# i 50 more rows
```

Let's now create cohorts in our `pp_microsim` data to match our population projection data.

```
# Convert 'age' into 'cohort' factor with levels ordered as specified
pp_microsim <- pp_microsim %>%
    mutate(cohort = factor(case_when(
    age >= 0  & age <= 29 ~ "P0029",
    age >= 30 & age <= 39 ~ "P3039",
    age >= 40 & age <= 49 ~ "P4049",
    age >= 50 & age <= 59 ~ "P5059",
    age >= 60  ~ "P60up"
  ), levels = c("P0029", "P3039", "P4049", "P5059", "P60up")))
  # mutate(cohort = factor(case_when(
  #   age >= 0 & age <= 9 ~ "P0009",
  #   age >= 10 & age <= 19 ~ "P1019",
  #   age >= 20 & age <= 29 ~ "P2029",
  #   age >= 30 & age <= 39 ~ "P3039",
  #   age >= 40 & age <= 49 ~ "P4049",
  #   age >= 50 & age <= 59 ~ "P5059",
  #   age >= 60 & age <= 69 ~ "P6069",
  #   age >= 70 ~ "P70up"
  # ), levels = c("P0009", "P1019", "P2029", "P3039", "P4049", "P5059", "P6069", "P70up")))

# Convert the 'cohort' and 'gender' factor to numeric codes
pp_microsim <- pp_microsim %>%
  mutate(cohort_code = as.integer(cohort)) %>%
  mutate(gender_code = as.integer(gender))
```

We also need demographic targets for 2030

```r
# Ensure pop_targets_2030 is correctly prepared
pop_targets_2030 <- population_projections  %>%
  filter(year == 2030, Variant == variants[7])  %>%
  group_by(cohort_code, cohort_short) %>%
    summarise(female = sum(yf),
              male   = sum(ym),
              total = sum(total_population),
              ) %>%
  ungroup()
```

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

```r
pop_total <- sum(pop_targets_2030$total)

pop_targets_2030 <- pop_targets_2030 %>%
  mutate(pct_total = total / pop_total)
```

And economic targets from our macroeconomic scenario data.

```r
economic_targets_2030 <- macro_data %>%
  filter(year == 2030, Variant == "Medium", scenario_id == "baseline") %>%
  summarise(
    target_lab_agr = sum(lab_agr_1000p * 1000),
    target_lab_man = sum(lab_man_1000p * 1000),
    target_lab_ser = sum(lab_ser_1000p * 1000)
  )
```

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

```r
pp_microsim <- pp_microsim %>%
  mutate(lmarket = case_when(
    lstatus == 1 & sector_w   == 1   ~ 1,    # Agriculture
    lstatus == 1 & sector_w   == 2   ~ 2,    # Manufactures
    lstatus == 1 & sector_w   == 3   ~ 3,    # Services
    lstatus == 2 & is.na(sector_w) ~ 4,    # Unemployed
    lstatus == 3 & is.na(sector_w) ~ 4,    # Unemployed
```

```
    lstatus == 4 & is.na(sector_w) ~ 5,    # OLF

  ))
```

And we check the values.

```
write.table(
  pp_microsim %>%
  group_by(lmarket) %>%
  summarise(total_pp = sum(weight, na.rm = TRUE)) %>%
    ungroup(),
  "clipboard", sep="\t", row.names=FALSE
  )
```

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

0.7.2 Reweigting

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

```
# Target for each variable

gender_code <- c(
  sum(pop_targets_2030$male)   /
    (sum(pop_targets_2030$male)+ sum(pop_targets_2030$female)),
  sum(pop_targets_2030$female) /
    (sum(pop_targets_2030$male)+ sum(pop_targets_2030$female)))

cohort_code <- pop_targets_2030$pct_total

# Four digits are better than two in this case, raking is quite accurate.
lmarket <- c(0.1341, 0.0494, 0.2611, 0.2472, 0.3082)

# Target list

targets <- list(gender_code
                , cohort_code
                , lmarket
```

```r
                 )

names(targets) <- c("gender_code",
                     "cohort_code",
                     "lmarket"
                     )

# Since this uses base R, we need to turn the data frame into base R object
rakedata <- as.data.frame(pp_microsim)
```

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

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

```
gender_code cohort_code     lmarket 
 0.03626510  0.09677000  0.07212418 
```

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

```
[1] "Raking...Iteration 1"
[1] "Current iteration changed total weights by 2336.85909302283"
[1] "Raking...Iteration 2"
[1] "Current iteration changed total weights by 196.013077923158"
[1] "Raking...Iteration 3"
[1] "Current iteration changed total weights by 48.9461383276265"
[1] "Raking...Iteration 4"
[1] "Current iteration changed total weights by 15.2825935005305"
[1] "Raking...Iteration 5"
```

```
[1] "Current iteration changed total weights by 4.79740637305395"
[1] "Raking...Iteration 6"
[1] "Current iteration changed total weights by 1.50755591186616"
[1] "Raking...Iteration 7"
[1] "Current iteration changed total weights by 0.473847760331075"
[1] "Raking...Iteration 8"
[1] "Current iteration changed total weights by 0.148945583420334"
[1] "Raking...Iteration 9"
[1] "Current iteration changed total weights by 0.0468190426274527"
[1] "Raking...Iteration 10"
[1] "Current iteration changed total weights by 0.0147169948904372"
[1] "Raking...Iteration 11"
[1] "Current iteration changed total weights by 0.00462611317398909"
[1] "Raking...Iteration 12"
[1] "Current iteration changed total weights by 0.00145416443982054"
[1] "Raking...Iteration 13"
[1] "Current iteration changed total weights by 0.00045709959257946"
[1] "Raking...Iteration 14"
[1] "Current iteration changed total weights by 0.000143683917964627"
[1] "Raking...Iteration 15"
[1] "Current iteration changed total weights by 4.51653621963283e-05"
[1] "Raking...Iteration 16"
[1] "Current iteration changed total weights by 1.41972053157718e-05"
[1] "Raking...Iteration 17"
[1] "Current iteration changed total weights by 4.46272523452196e-06"
[1] "Raking...Iteration 18"
[1] "Current iteration changed total weights by 1.40280532930082e-06"
[1] "Raking...Iteration 19"
[1] "Current iteration changed total weights by 4.40955060437842e-07"
[1] "Raking...Iteration 20"
[1] "Current iteration changed total weights by 1.38610069977396e-07"
[1] "Raking...Iteration 21"
[1] "Current iteration changed total weights by 4.35708336493645e-08"
[1] "Raking...Iteration 22"
[1] "Current iteration changed total weights by 1.36949823009713e-08"
[1] "Raking...Iteration 23"
[1] "Current iteration changed total weights by 4.30483908242518e-09"
[1] "Raking...Iteration 24"
[1] "Current iteration changed total weights by 1.35239161536127e-09"
[1] "Raking...Iteration 25"
[1] "Current iteration changed total weights by 4.25884119636954e-10"
[1] "Raking...Iteration 26"
[1] "Current iteration changed total weights by 1.34918465288791e-10"
```

```
[1] "Raking...Iteration 27"
[1] "Current iteration changed total weights by 4.08980349586585e-11"
[1] "Raking...Iteration 28"
[1] "Current iteration changed total weights by 1.39531441956109e-11"
[1] "Raking...Iteration 29"
[1] "Current iteration changed total weights by 4.07238132105192e-12"
[1] "Raking...Iteration 30"
[1] "Current iteration changed total weights by 2.60423627107542e-12"
[1] "Raking...Iteration 31"
[1] "Current iteration changed total weights by 2.40631126136037e-12"
[1] "Raking...Iteration 32"
[1] "Current iteration changed total weights by 0"
[1] "Raking...Iteration 33"
[1] "Current iteration changed total weights by 0"
[1] "Raking converged in 33 iterations"
```

```
summary(outsave)
```

```
$convergence
[1] "Complete convergence was achieved after 33 iterations"

$base.weights
[1] "Using Base Weights Provided"

$raking.variables
[1] "cohort_code" "lmarket"

$weight.summary
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1103  0.6867  0.9272  1.0000  1.2758  3.2390

$selection.method
[1] "variable selection conducted using _pctlim_ - discrepancies selected using _total_."

$general.design.effect
[1] 1.241567

$gender_code
         Target Old Weights N Old Weights % Wtd N Wtd % Change in %
<NA>  0.4514133            NA            NA    NA    NA           NA
<NA>  0.5485867            NA            NA    NA    NA           NA
Total 1.0000000             0             0     0     0            0
```

```
      Resid. Disc. Orig. Disc.
<NA>            NA          NA
<NA>            NA          NA
Total            0           0

$cohort_code
         Target Old Weights N Old Weights % Wtd N Wtd % Change in %
<NA>  0.3638899            NA            NA    NA    NA          NA
<NA>  0.1420277            NA            NA    NA    NA          NA
<NA>  0.1630905            NA            NA    NA    NA          NA
<NA>  0.1042513            NA            NA    NA    NA          NA
<NA>  0.2267405            NA            NA    NA    NA          NA
Total 1.0000000             0             0     0     0           0
      Resid. Disc. Orig. Disc.
<NA>            NA          NA
<NA>            NA          NA
<NA>            NA          NA
<NA>            NA          NA
<NA>            NA          NA
Total            0           0

$lmarket
      Target Old Weights N Old Weights % Wtd N Wtd % Change in % Resid. Disc.
<NA>  0.1341            NA            NA    NA    NA          NA           NA
<NA>  0.0494            NA            NA    NA    NA          NA           NA
<NA>  0.2611            NA            NA    NA    NA          NA           NA
<NA>  0.2472            NA            NA    NA    NA          NA           NA
<NA>  0.3082            NA            NA    NA    NA          NA           NA
Total 1.0000             0             0     0     0           0            0
      Orig. Disc.
<NA>           NA
<NA>           NA
<NA>           NA
<NA>           NA
<NA>           NA
Total           0
```

```
# add weights to the dataset

rakedata$weightvec  <- unlist(outsave[1])
n  <- length(rakedata$sector)
```

```r
# Calculate the sum of original weights
original_weight_sum <- sum(rakedata$weight)

# # Target scaling for original weights
# In a first version, we used the population scaling from UN,
# But to match with the labor market given by the scenario, we take
# the total population from the macro scenario information.
# original_weight_scaling_factor <- pop_data$total_population[pop_data$year == 2030] /
#   pop_data$total_population[pop_data$year == 2022]

# For year 2030 (Calculated in Excel, but need a more elegant solution from dataset)
original_weight_scaling_factor <- 0.992805755

# Scaled original weights
original_weight_sum <- (original_weight_sum
                        * original_weight_scaling_factor)

# Calculate the sum of the new weights
new_weight_sum <- sum(rakedata$weightvec)

# Scale the new weights to match the sum of the original weights
scaling_factor <- original_weight_sum / new_weight_sum
rakedata$weightvec <- rakedata$weightvec * scaling_factor

# Verify the adjustment
head(rakedata[, c("weight", "weightvec")])
```

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

```r
summary(rakedata$weightvec)
```

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

```r
summary(rakedata$weight)
```

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

```r
rakedata <- rakedata %>%
  relocate(weightvec, .after = weight) %>%
  mutate(hh_weight = weightvec / hhsize)

# We calculate new weights for households in the hh database
weights_2030 <- rakedata %>%
  group_by(hhid) %>%
  summarise(hh_weight_2030 = sum(hh_weight, na.rm = TRUE))

pp_microsim <- tibble(rakedata)
rm(rakedata)
```

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

```r
# Wage rescale factor by sector from macro (Agriculture, Manufacturing, Services)
wrf <- c(1.269821454, 1.284834838, 1.328623737)

# We check the wage bill by sector
wages_by_sector <- pp_microsim %>%
  group_by(sector_w, .drop = TRUE) %>%
  summarise(
    wages_2022 = sum(annual_labor_total * weight, na.rm = TRUE),
    wages_2030 = sum(annual_labor_total * weightvec, na.rm = TRUE)
  )

wages_by_sector <- wages_by_sector[c(-4),]

# Compare how much it changed with reweighting with how it should have changed
# Derive coefficients (wtc_2030) from that
wages_by_sector <- wages_by_sector %>%
  mutate(wages_target_2030 = case_when(
      sector_w == 1 ~ wages_2022 * wrf[1],
      sector_w == 2 ~ wages_2022 * wrf[2],
      sector_w == 3 ~ wages_2022 * wrf[3],
      .default = NA
```

```r
    ),
    wtc_2030 = wages_target_2030 / wages_2030)

wages_by_sector %>%
  gt()
```

sector_w	wages_2022	wages_2030	wages_target_2030	wtc_2030
1	329980600	332988623	419016446	1.258351
2	250074262	262381786	321304124	1.224567
3	1255196730	1317702602	1667684171	1.265600

We then add the coefficient to rescale each weight by sector

```r
# Assign rescale the annual and monthly wage depending on the sector
pp_microsim <- pp_microsim %>%
  rename(monthly_labor_income_old = monthly_labor_income,
         annual_labor_total_old = annual_labor_total) %>%
  mutate(
    monthly_labor_income = case_when(
      sector_w == 1 ~ monthly_labor_income_old * wages_by_sector$wtc_2030[1],
      sector_w == 2 ~ monthly_labor_income_old * wages_by_sector$wtc_2030[2],
      sector_w == 3 ~ monthly_labor_income_old * wages_by_sector$wtc_2030[3],
      TRUE ~ NA
    ),
    annual_labor_total = case_when(
      sector_w == 1 ~ annual_labor_total_old * wages_by_sector$wtc_2030[1],
      sector_w == 2 ~ annual_labor_total_old * wages_by_sector$wtc_2030[2],
      sector_w == 3 ~ annual_labor_total_old * wages_by_sector$wtc_2030[3],
      TRUE ~ NA
    )
    )

# This takes care of different household members coming from different sectors
hh_li_baseline <- pp_microsim %>%
  group_by(hhid) %>%
  summarise(mli_2022 = sum(monthly_labor_income_old),
            mli_2030 = sum(monthly_labor_income),
            mli_coef_2030 = mli_2030 / mli_2022) %>%
  select(hhid, mli_coef_2030)
```

```r
ic_microsim <- ic %>%
  left_join(hh_li_baseline, join_by(interview__key == hhid)) %>%
  rename(inc2_old = inc2,
         inc3_old = inc3,
         totalinc_old = totalinc) %>%
  mutate(mli_coef_2030 = if_else(is.na(mli_coef_2030), 1,mli_coef_2030)) %>%
  mutate(inc2 = inc2_old * mli_coef_2030,
         inc3 = inc3_old * mli_coef_2030) %>%
  mutate(totalinc = totalinc_old - coalesce(inc2_old,0) - coalesce(inc3_old,0)
                         + coalesce(inc2,0) + coalesce(inc3,0)) %>%
  mutate(totinc_coef_2030 = if_else(totalinc_old == 0, 1, totalinc / totalinc_old)) %>%
  mutate(totinc_coef_2030 = if_else(is.na(totinc_coef_2030), 1, totinc_coef_2030))

ic_coef_2030 <- ic_microsim %>%
  select(interview__key, totinc_coef_2030)
```

0.8 Microsimulation

We now implement different shocks according to various scenarios.

0.8.1 Baseline

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

```r
ca_microsim <- ca %>%
  left_join(weights_2030, join_by(hhid == hhid)) %>%
  left_join(ic_coef_2030, join_by(hhid == interview__key)) %>%
  # We adjust total consumption by the income coefficient
  # Make sure to remember that totc is baseline 2030
  rename(totc_old = totc,
         # Make sure to remember that pov2022 is baseline 2030
         poor_Avpovln2022_old = poor_Avpovln2022) %>%
  mutate(totc = totc_old * totinc_coef_2030 )
```

And recalculate poverty.

```r
ca_microsim <- ca_microsim %>%
  rename(aec_r_old = aec_r,
         weight_old = weight,
```

```
         weight = hh_weight_2030) %>%
  mutate(aec_r = totc / ae_r / PI) %>%
  mutate(poor_Avpovln2022 =
           if_else(aec_r < 52883, 1, 0)
         )
```

Test

```
test <- ca_microsim %>%
  rename(poor_original = poor_Avpovln2022_old,
         poor_2030 = poor_Avpovln2022) %>%
  group_by(poor_2030) %>%
  summarise(no_hh = round(sum(weight, na.rm = TRUE)),
            no_pp = round(sum(weight * hhsize, na.rm = TRUE))) %>%
  ungroup()

test <- ca_microsim %>%
  rename(poor_original = poor_Avpovln2022_old,
         poor_2030 = poor_Avpovln2022) %>%
  group_by(poor_original) %>%
  summarise(no_hh = round(sum(weight, na.rm = TRUE)),
            no_pp = round(sum(weight * hhsize, na.rm = TRUE))) %>%
  ungroup()

test %>%
  gt()
```

Poor, Avpovln2022	no_hh	no_pp
0	638528	2126723
1	143033	707267

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

0.8.2 Climate change

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

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

```r
# First calculate moving window average
heat_l_pdcty <- heat_l_pdcty %>%
  group_by(ADM1_EN, clim_scenario) %>%
  arrange(year) %>%
  # Moving window average 5 years before, 5 after
  mutate(
    moving_avg = rollapply(
      pct_change_productivity,
      width = 11,
      FUN = mean,
      partial = TRUE,
      align = "center",
      fill = NA,
      na.rm = TRUE),
    # Moving window max value 5 years before, 5 after
    # Since it's expressed in negative values (min) is the maximum
    moving_max = rollapply(
      pct_change_productivity,
      width = 11,
      FUN = min,
      partial = TRUE,
      align = "center",
      fill = NA,
      na.rm = TRUE)) %>%
  ungroup()

# Clim scenarios to select
cs <- unique(heat_l_pdcty$clim_scenario)

# Moving average for year of interest
lab_loss_avg <- heat_l_pdcty %>%
  filter(clim_scenario == cs[1], year == yearsto[1] ) %>%
  select(-pct_change_productivity,-ADM1_PCODE,
         -year,-clim_scenario, -moving_max) %>%
  pivot_wider(names_from = sector,
              values_from = moving_avg) %>%
  rename(agr_avg = Agriculture,
         man_avg = Manufacturing,
         ser_avg = Services)

# Max value for year of interest
lab_loss_max <- heat_l_pdcty %>%
```

```
  filter(clim_scenario == cs[1], year == yearsto[1] ) %>%
  select(-pct_change_productivity, -ADM1_PCODE,
         -year,-clim_scenario,-moving_avg) %>%
  pivot_wider(names_from = sector,
              values_from = moving_max) %>%
  rename(agr_max = Agriculture,
         man_max = Manufacturing,
         ser_max = Services)
```

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

```
# First calculate moving window average
crops_productivity <- crops_productivity %>%
  group_by(NAM_1, climate_scenario) %>%
  arrange(year) %>%
  # Moving window average
  mutate(
    moving_avg = rollapply(
      pct_change_prod,
      width = 11,    # 5 years before, 5 after + reference year = 11
      FUN = mean,
      partial = TRUE,
      align = "center",
      fill = NA,
      na.rm = TRUE),
    # Moving window max value 5 years before, 5 after
    # Since it's expressed in negative values (min) is the maximum
    moving_max = rollapply(
      pct_change_prod,
      width = 11,
      FUN = min,
      partial = TRUE,
      align = "center",
      fill = NA,
      na.rm = TRUE)) %>%
  ungroup()

# Clim scenarios to select
cs <- unique(crops_productivity$climate_scenario)

# Moving average for year of interest
```

```
ag_pdcvty_loss <- crops_productivity %>%
  filter(climate_scenario == cs[1], year == yearsto[1] ) %>%
  select(-pct_change_prod,-GID_1, -year,-climate_scenario) %>%
  rename(crops_avg_loss = moving_avg,
         crops_max_loss = moving_max)
```

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

```
# Persons processed dataset
pp_microsim_cc <- pp_microsim %>%
  left_join(lab_loss_avg, join_by(NAM_1==ADM1_EN)) %>%
  left_join(lab_loss_max, join_by(NAM_1==ADM1_EN))

# Household income processed dataset
ic_microsim_cc <- ic_microsim %>%
  left_join(ag_pdcvty_loss, join_by(NAM_1==NAM_1))
```

And we first shock labor income.

```
# Labor income according to sector
pp_microsim_cc <- pp_microsim_cc %>%
  mutate(sector = as.numeric(sector)) %>%
  mutate(mli_cc_avg = case_when(
    # * 1000 because its thousands of Dram
    sector == 1 ~ monthly_labor_income * (1 + agr_avg)* 1000,
    sector == 2 ~ monthly_labor_income * (1 + man_avg)* 1000,
    sector == 3 ~ monthly_labor_income * (1 + ser_avg)* 1000,
    TRUE ~ NA
  )) %>%
  mutate(mli_cc_max = case_when(
    # * 1000 because its thousands of Dram
    sector == 1 ~ monthly_labor_income * (1 + agr_max)* 1000,
    sector == 2 ~ monthly_labor_income * (1 + man_max)* 1000,
    sector == 3 ~ monthly_labor_income * (1 + ser_max)* 1000,
    TRUE ~ NA
  ))
```

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

```
ic_new_incomes <- pp_microsim_cc %>%
  group_by(hhid) %>%
  summarise(
    mli_cc_avg = sum(mli_cc_avg, na.rm = TRUE),
    mli_cc_max = sum(mli_cc_max, na.rm = TRUE),
    mli_original = sum(monthly_labor_income*1000, na.rm = TRUE)
  ) %>%
  mutate(mli_avg_coef =
           if_else(mli_original == 0 | is.na(mli_original), 1,
                   mli_cc_avg / mli_original),
         mli_max_coef =
           if_else(mli_original == 0 | is.na(mli_original), 1,
                   mli_cc_max / mli_original)
         ) %>%
  ungroup()

ic_microsim_cc <- ic_microsim_cc %>%
  left_join(ic_new_incomes, join_by(interview__key == hhid)) %>%
  mutate(inc2_cc_avg = inc2 * mli_avg_coef,
         inc2_cc_max = inc2 * mli_max_coef,
         inc3_cc_avg = inc3 * mli_avg_coef,
         inc3_cc_max = inc3 * mli_max_coef)
```

And now we impact agricultural income `inc4`.

```
ic_microsim_cc <- ic_microsim_cc %>%
  mutate(inc4_cc_avg = inc4 * (1 + crops_avg_loss),
         inc4_cc_max = inc4 * (1 + crops_max_loss))
```

And recalculate total income.

```
ic_microsim_cc <- ic_microsim_cc %>%
  mutate(totalinc_cc_avg = totalinc - rowSums(select(., c(inc2, inc3, inc4)),
                                              na.rm = TRUE) +
                           rowSums(select(., c(inc2_cc_avg, inc3_cc_avg, inc4_cc_avg)),
                                   na.rm = TRUE),
         totalinc_cc_max = totalinc - rowSums(select(., c(inc2, inc3, inc4)),
                                              na.rm = TRUE) +
                           rowSums(select(., c(inc2_cc_max, inc3_cc_max, inc4_cc_max)),
                                   na.rm = TRUE)) %>%
  mutate(totalinc_cc_avg_coef =
```

```
            if_else(totalinc == 0, 1,
                    totalinc_cc_avg / totalinc),
          totalinc_cc_max_coef =
            if_else(totalinc == 0, 1,
                    totalinc_cc_max / totalinc)
          ) %>%
  mutate(totalinc_cc_avg_coef = if_else(is.na(totalinc_cc_avg_coef), 1, totalinc_cc_avg_coef
         totalinc_cc_max_coef = if_else(is.na(totalinc_cc_max_coef), 1, totalinc_cc_max_coef
```

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

```
income_losses <- ic_microsim_cc %>%
  select(interview__key,totalinc_cc_avg_coef, totalinc_cc_max_coef)


ca_microsim_cc <- ca_microsim %>%
  left_join(income_losses, join_by(hhid == interview__key))

# And now reduce total consumption

ca_microsim_cc <- ca_microsim_cc %>%
  mutate(totc_cc_avg = totc * totalinc_cc_avg_coef,
         totc_cc_max = totc * totalinc_cc_max_coef) %>%
  mutate(aec_r_cc_avg = totc_cc_avg / ae_r / PI,
         aec_r_cc_max = totc_cc_max / ae_r / PI) %>%
  mutate(poor_cc_avg =
           if_else(aec_r_cc_avg < 52883, 1, 0),
         poor_cc_max =
           if_else(aec_r_cc_max < 52883, 1, 0))

# We make a table to see who became poor.

test <- ca_microsim_cc %>%
  rename(poor_original = poor_Avpovln2022,
         poor_cc = poor_cc_avg) %>%
  group_by(poor_original, poor_cc) %>%
  summarise(no_hh = round(sum(weight, na.rm = TRUE)),
            no_pp = round(sum(weight * hhsize, na.rm = TRUE))) %>%
  ungroup()
```

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

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

0.8.3 Food prices

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

```
scenarios <- unique(macro_data$scenario_id)

# We extract and reformat the price data
price_data <- macro_data %>%
  select(year, scenario_id, starts_with( c("fpi" , "epi") )) %>%
  rename(scenario = scenario_id) %>%
  pivot_longer(starts_with( c("fpi" , "epi") ),
               names_to = "type_decile",
               values_to = "index") %>%
  mutate(decile = parse_number(type_decile)) %>%
  mutate(commodity_group =
           case_when(
             str_starts(type_decile, "fpi") ~ "food",
             str_starts(type_decile, "epi") ~ "energy",
             TRUE ~ NA_character_
           )) %>%
  select(-type_decile) %>%
  relocate(index, .after = commodity_group)

# We take a look at price information in 2030
price_data %>%
  filter(year == 2030) %>%
  group_by(commodity_group, scenario) %>%
  summarise(index = mean(index, na.rm = TRUE)) %>%
  gt()
```

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

scenario	index
energy	
Dry-hot scenario	1.124883

baseline	1.129182
food	
Dry-hot scenario	1.075696
baseline	1.009986

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

```
# Filter `price_data` for the years of interest
price_data_yearsto <- price_data %>%
  filter(year %in% yearsto)

# Create a named vector for scenario indices
scenario_indices <- setNames(seq_along(scenarios), scenarios)

# Create the composite string column
price_data_yearsto <- price_data_yearsto %>%
  mutate(
    scenario_index = scenario_indices[scenario],
    composite_column = paste(commodity_group, scenario_index, year, sep = "_")
  ) %>%
  select(decile,index,composite_column)

composite_column_names <- unique(price_data_yearsto$composite_column)

price_data_yearsto <- price_data_yearsto %>%
  pivot_wider(names_from="composite_column", values_from = index)
```

And we join with our household's dataset.

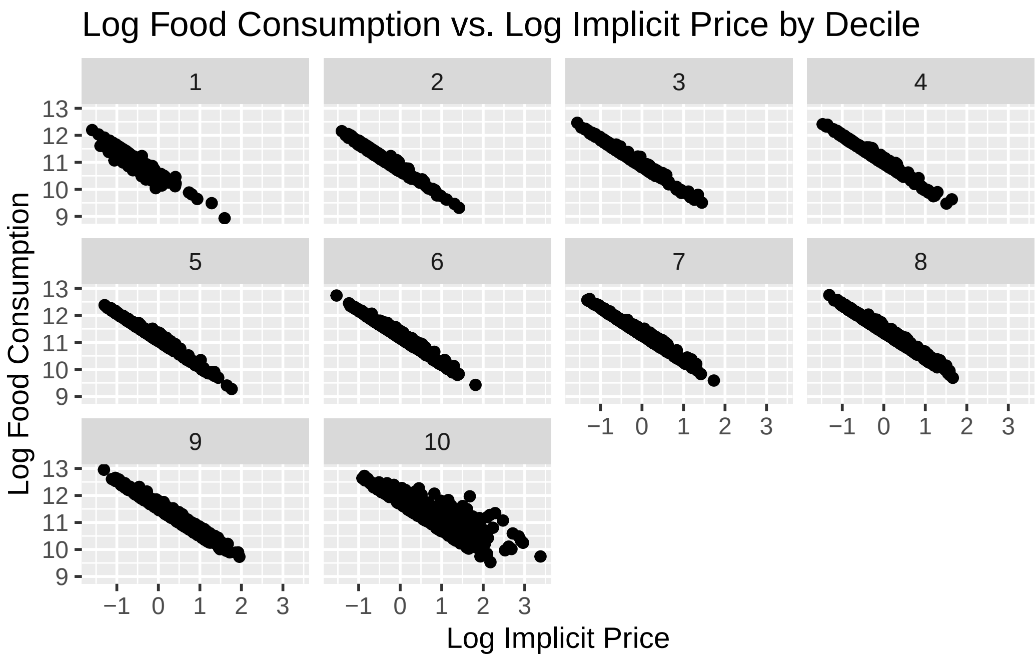
```
# PP microsim already has decile information from previous join
ca_microsim_cc <- ca_microsim_cc %>%
  left_join(price_data_yearsto, join_by(decile==decile))
```

Since we don't have quantities for the aggregate food expenditure category or for the aggregate energy bundle, we assume a price of 1 in the survey year. We estimate elasticitiesand assign them to homes by decile.

```
# Calculate implicit price as total expenditure on food divided by food consumption
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(implicit_price = aec_r / food1)

# Log-transform the relevant variables
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(log_food1 = log(food1),
         log_implicit_price = log(implicit_price),
         log_aec_r = log(aec_r))

# Check for missing values and data distribution
#summary(ca_microsim_cc)

# summarise the data for variability within each decile
ca_summary <- ca_microsim_cc %>%
  group_by(decile) %>%
  summarise(
    log_food1_mean = mean(log_food1, na.rm = TRUE),
    log_food1_sd = sd(log_food1, na.rm = TRUE),
    log_implicit_price_mean = mean(log_implicit_price, na.rm = TRUE),
    log_implicit_price_sd = sd(log_implicit_price, na.rm = TRUE)
  )

print(ca_summary)
```

```
# A tibble: 10 x 5
   decile log_food1_mean log_food1_sd log_implicit_price_mean
    <dbl>          <dbl>        <dbl>                   <dbl>
 1      1           11.1        0.415                  -0.562
 2      2           11.1        0.459                  -0.397
 3      3           11.2        0.510                  -0.329
 4      4           11.3        0.528                  -0.250
 5      5           11.3        0.541                  -0.153
 6      6           11.3        0.577                  -0.0370
 7      7           11.3        0.580                  -0.0258
 8      8           11.3        0.592                   0.0875
 9      9           11.3        0.620                   0.283
10     10           11.2        0.582                   0.761
# i 1 more variable: log_implicit_price_sd <dbl>
```

```r
# Visual inspection
ggplot(ca_microsim_cc, aes(x = log_implicit_price, y = log_food1)) +
  geom_point() +
  facet_wrap(~ decile) +
  labs(title = "Log Food Consumption vs. Log Implicit Price by Decile",
       x = "Log Implicit Price", y = "Log Food Consumption")
```




```r
# Create a function to fit the simplified model and extract the price elasticity
fit_model <- function(data) {
  model <- lm(log_food1 ~ log_implicit_price, data = data)
  tidy(model)
}

# Apply the simplified model fitting function by decile
decile_models <- ca_microsim_cc %>%
  group_by(decile) %>%
  nest() %>%
  mutate(model = map(data, fit_model)) %>%
  unnest(model) %>%
  filter(term == "log_implicit_price") %>%
  select(decile, price_elasticity = estimate)
```

```r
# Extracting the price elasticity for each decile
food_decile_elasticities <- decile_models %>%
  select(decile, price_elasticity)

# Print the results
print(food_decile_elasticities)
```

```
# A tibble: 10 x 2
# Groups:   decile [10]
   decile price_elasticity
    <dbl>            <dbl>
 1      9           -0.964
 2      7           -0.971
 3      2           -0.980
 4      4           -0.977
 5     10           -0.759
 6      8           -0.968
 7      5           -0.984
 8      6           -0.977
 9      3           -0.972
10      1           -0.929
```

Let's apply the elasticities to the new data.

```r
# Merge elasticities
ca_microsim_cc <- ca_microsim_cc %>%
  left_join(food_decile_elasticities, by = "decile")

# Calculate the implicit price
# Assuming implicit_price can be calculated from the expenditure (food1)
# If we assume baseline quantity consumed is proportional to expenditure/price
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(implicit_price = food1 / food1,  # This is 1 as we don't have baseline price
         food_quantity = food1 / implicit_price)

# Calculate the percentage change in prices for each decile
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(food_1_dprice = (food_1_2030 - 1),
         food_2_dprice = (food_2_2030 - 1))

# Estimate the new food consumption levels
```

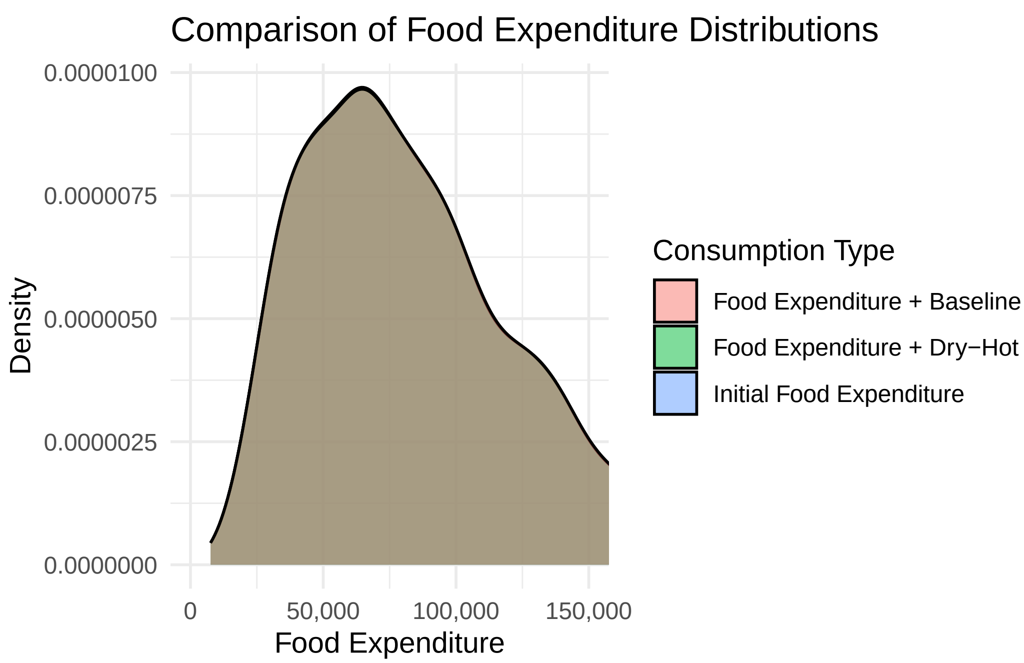
```r
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(food_q1_sim = food_quantity * (1 + food_1_dprice * price_elasticity),
         food_q2_sim = food_quantity * (1 + food_2_dprice * price_elasticity))

# Calculate the new expenditure levels
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(food_exp1_sim = food_q1_sim * food_1_2030,
         food_exp2_sim = food_q2_sim * food_2_2030)

# View the results
print(ca_microsim_cc %>% select(decile, food1, food_1_2030, food_1_dprice, food_q1_sim, food_
```

```
# A tibble: 5,184 x 9
   decile   food1 food_1_2030 food_1_dprice food_q1_sim food_exp1_sim
    <dbl>   <dbl>       <dbl>         <dbl>       <dbl>         <dbl>
 1      9  92178.        1.07        0.0749      85520.        91925.
 2      7 105290.        1.07        0.0743      97691.       104953.
 3      2  22733.        1.08        0.0761      21039.        22640.
 4      2 163379.        1.08        0.0761     151199.       162705.
 5      4 102718.        1.08        0.0774      94956.       102303.
 6     10 208995.        1.08        0.0777     196678.       211958.
 7      8 116151.        1.08        0.0759     107620.       115788.
 8      5  67911.        1.07        0.0742      62954.        67626.
 9      6 152144.        1.07        0.0749     141003.       151568.
10      8  80033.        1.08        0.0759      74155.        79783.
# i 5,174 more rows
# i 3 more variables: food_2_dprice <dbl>, food_q2_sim <dbl>,
#   food_exp2_sim <dbl>
```

Let's plot the distributions to see changes:

```r
# Basic density plot comparing food1 and food_exp_sim
ggplot(ca_microsim_cc, aes(x = food1, fill = 'Initial Food Expenditure')) +
  geom_density(alpha = 0.5) +
  geom_density(
    data = ca_microsim_cc,
    aes(x = food_exp1_sim, fill = 'Food Expenditure + Dry-Hot'),
    alpha = 0.5) +geom_density(
    data = ca_microsim_cc,
    aes(x = food_exp2_sim, fill = 'Food Expenditure + Baseline'),
    alpha = 0.5) +
```

```
  labs(
    fill = "Consumption Type",
    title = "Comparison of Food Expenditure Distributions",
    x = "Food Expenditure",
    y = "Density") +
  theme_minimal() +
  coord_cartesian(xlim = c(0, 150000)) + # Adjust the xlim for zoom
  scale_x_continuous(labels = scales::comma) +
  scale_y_continuous(labels = scales::comma)
```



Calculate losses in consumer surplus and purchasing power loss.

```
# Calculate the implicit price (assuming baseline implicit price is 1 for simplicity)
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(implicit_price = 1,  # Simplified assumption
         food_quantity = food1 / implicit_price)

# Calculate the percentage change in prices for each decile
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(food_1_dprice = (food_1_2030 - 1),
         food_2_dprice = (food_2_2030 - 1))
```

```r
# Use the existing new food consumption levels (already calculated)
# ca_microsim_cc <- ca_microsim_cc %>%
#   mutate(food_q1_sim = food_quantity * (1 + food_1_dprice * price_elasticity),
#          food_q2_sim = food_quantity * (1 + food_2_dprice * price_elasticity))

# Calculate Consumer Surplus loss for food1 and food2 scenarios
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(food1_CSloss = ((food_quantity * implicit_price) / totc) * food_1_dprice * (1 + (pr
         food2_CSloss = ((food_quantity * implicit_price) / totc) * food_2_dprice * (1 + (pr
         ttl_CSloss_1 = food1_CSloss,
         ttl_CSloss_2 = food2_CSloss)

# Calculate Purchasing Power loss for food1 and food2 scenarios
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(food1_PPloss = (food1 / totc) * food_1_dprice,
         food2_PPloss = (food1 / totc) * food_2_dprice,
         ttl_PPloss_1 = food1_PPloss,
         ttl_PPloss_2 = food2_PPloss)

# Adjust total expenditure (totc) based on the purchasing power loss
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(totc_cc_avg_food1 = totc_cc_avg * (1 - ttl_PPloss_1),
         totc_cc_avg_food2 = totc_cc_avg * (1 - ttl_PPloss_2))

# View the results
print(ca_microsim_cc %>% select(decile, food1, food_1_2030, food_1_dprice, food_q1_sim, food_
```

```
# A tibble: 5,184 x 19
   decile   food1 food_1_2030 food_1_dprice food_q1_sim food_exp1_sim
    <dbl>   <dbl>       <dbl>         <dbl>       <dbl>         <dbl>
 1      9  92178.        1.07        0.0749      85520.        91925.
 2      7 105290.        1.07        0.0743      97691.       104953.
 3      2  22733.        1.08        0.0761      21039.        22640.
 4      2 163379.        1.08        0.0761     151199.       162705.
 5      4 102718.        1.08        0.0774      94956.       102303.
 6     10 208995.        1.08        0.0777     196678.       211958.
 7      8 116151.        1.08        0.0759     107620.       115788.
 8      5  67911.        1.07        0.0742      62954.        67626.
 9      6 152144.        1.07        0.0749     141003.       151568.
10      8  80033.        1.08        0.0759      74155.        79783.
# i 5,174 more rows
# i 13 more variables: food_2_dprice <dbl>, food_q2_sim <dbl>,
```

```
#   food_exp2_sim <dbl>, food1_CSloss <dbl>, food2_CSloss <dbl>,
#   ttl_CSloss_1 <dbl>, ttl_CSloss_2 <dbl>, food1_PPloss <dbl>,
#   food2_PPloss <dbl>, ttl_PPloss_1 <dbl>, ttl_PPloss_2 <dbl>,
#   totc_cc_avg_food1 <dbl>, totc_cc_avg_food2 <dbl>
```

Okay so now we estimate new welfare and poverty.

```r
ca_microsim_cc <- ca_microsim_cc %>%
  mutate(aec_r_cc_avg_food1 = totc_cc_avg_food1 / ae_r / PI,
         aec_r_cc_avg_food2 = totc_cc_avg_food2 / ae_r / PI) %>%
  mutate(poor_cc_avg_food1 =
           if_else(aec_r_cc_avg_food1 < 52883, 1, 0),
         poor_cc_avg_food2 =
           if_else(aec_r_cc_avg_food2 < 52883, 1, 0))
```

And now we see who became poor

```r
# We make a table to see who became poor.
test <- ca_microsim_cc

test$ones <- 1

test <- test%>%
  rename(poor_original = poor_Avpovln2022,
         poor_cc = poor_cc_avg,
         poor_cc_food1 = poor_cc_avg_food1,
         poor_cc_food2 = poor_cc_avg_food2) %>%
  group_by(poor_original, poor_cc) %>%
  summarise(no_hh = round(sum(weight, na.rm = TRUE)),
            no_pp = round(sum(weight*hhsize, na.rm = TRUE)))
```

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

```r
test %>%
  gt()
```

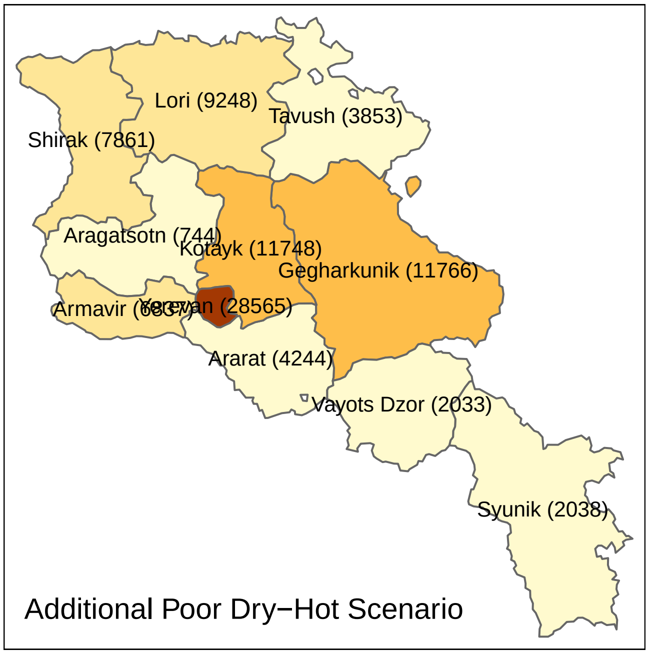
poor_cc	no_hh	no_pp
0		

	0	635512	2102222
	1	6156	33962
1			
	1	139892	697806

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

And we map these results.

```
foodpoor <- ca_microsim_cc %>%
  mutate(new_poor_food_base = if_else(
    poor_cc_avg_food2 == 1 & poor_cc_avg == 0, 1, 0),
         new_poor_food_dryhot = if_else(
           poor_cc_avg_food1 == 1 & poor_cc_avg == 0, 1, 0),
    NAM_1 = as_factor(marz)) %>%
  mutate(NAM_1 = if_else(NAM_1 == "VayotsDzor", "Vayots Dzor", NAM_1)) %>%
  mutate(NAM_1 = if_else(NAM_1 == "Sjunik", "Syunik", NAM_1)) %>%
  select(NAM_1, poor_Avpovln2022, poor_cc_avg, poor_cc_max,
         poor_cc_avg_food1, poor_cc_avg_food2, new_poor_food_base,
         new_poor_food_dryhot, weight, hhsize)

fp <-foodpoor %>%
  group_by(NAM_1) %>%
  summarise(new_poor = round(sum(new_poor_food_dryhot * weight*hhsize, na.rm = TRUE))) %>%
  mutate(label = paste0(NAM_1," (", new_poor, ")"))


#write.table(fp, "clipboard", sep="\t", row.names=FALSE)
fp_map <- adm1 |>
  left_join(fp, join_by(NAM_1 == NAM_1))

fp_map <-tm_shape(fp_map)+
  tm_polygons("new_poor", legend.show = FALSE) +
  tm_text("label", size = .7, col = "black")+
  tm_layout(legend.position = c("right", "top"),
            title= "Additional Poor Dry-Hot Scenario",
            title.position = c('left', 'bottom'),
            title.size = 0.9)
fp_map
```




```r
# We check that our reweighting was successful
table <- pp_microsim_cc %>%
  group_by(lmarket,sector_w) %>%
  summarise(total_pp = sum(weightvec, na.rm = TRUE)) %>%
  ungroup()
```

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

```r
table %>%
  gt() %>%
  fmt_number(columns = total_pp, decimals = 0)
```

lmarket	sector_w	total_pp
1	1	380,038
2	2	139,999
3	3	739,955
4	NA	700,562
5	NA	873,436

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

0.9 End

Tasks for today:

Reweight the dataset

- Determine the numbers for the labor market reweighting for 2030. [v]
- Reweight and rescale the weights. [v]
- Determine by how much to rescale the wage bill. [v]
- Recalculate labor income. [v]
- Recalculate total income. [v]
- Determine delta total income and change total expenditure. [v]
- Poverty [v]

Review price elasticities

- Look for average prices in underlying data.
- Reestimate based on adjusted food expenditure by family members.
- Rerun elasticities.
- Reestimate poverty.

Bring back poverty to people's dataset.

- Evaluate poverty among women.
- Poverty among urban rural.
- Poverty by marz.
- Youth?

Write draft.

- Make tables of everything.
- Make maps of everything.

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