Vulnerability Analysis Calculations

Renato Vargas

Table of contents

0.1 Introduction

In the context of the development of the Country Climate and Development Report (CCDR) for Armenia, the poverty team is contributing with inputs for vulnerability analysis at the household level. The methods for these inputs are in active development and benefit greatly from the practical applications and interdisciplinary discussions that take place during the creation of these CCDRs. This guide aims to document the steps carried out to link vulnerability impacts and household survey data.

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:

1

```
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 oportunity 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"

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

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(janitor)   # pretty subtotals
library(purrr)

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

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

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

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

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)
```

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)
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)
```

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

```
buildings_aal <- read_xlsx("data/ARM-Vulnerability-Analysis/Data_AAL_Buildings.xlsx", sheet =
buildings_1in100 <- read_xlsx("data/ARM-Vulnerability-Analysis/Data_AAL_Buildings.xlsx", shee
crops_aal <- read.csv("data/ARM-Vulnerability-Analysis/ARM_crops_combined_REF_shock_admin1.cs
  rename(NAM_1 = Province) |>
  # Make sure that names match the rest of datasets
  mutate(NAM_1 = if_else(NAM_1 == "Gegharkunik", "Gergharkunik", NAM_1))
crops_1in100 <- read_xlsx("data/ARM-Vulnerability-Analysis/damages_1in100_agri_ADM2.xlsx", sh
```

0.3 Asset value of income flows

0.3.1 Imputed rent

"Housing, measured as the welfare value of the flow of services households derive from their dwelling, is one of the most relevant components of households' welfare aggregate, which is used as a basis for distributional analysis" (Deaton & Zaidi, 2002; cited by Ceriani, Olivieri, & Ranzani, 2019). In Armenia, most households own their home, so the emergent rental market information is used to impute rent to non-renters using a log linear modeling approach described by (Ceriani et al., 2019), in which imputed rent is predicted using a combination of household characteristics (urban/rural, Marz, number of rooms, presence of an indoor toilet, number of household, square meters, type of dwelling, household members) and head of household characteristics (i.e. sex, highest completed schooling level, age group). The first step is to identify these characteristics for the regression.

We first extract relevant characteristics of the heads of household and create a heads subset of our person's database, which we call `heads`. It has our household id (`interview__key`), sex (`mem_02`), age (`mem_05`), and education level.

```
heads <- pp |>
  filter(mem_03 == 1) |>
  select( interview__key ,mem_02, mem_05,ed_03)
```

Since we only have one head of household per household, we can join this data with our household information. We now create a subset of our household data, which we call `imputed_rent` with the relevant dwelling and head of household variables according to the model suggested by Ceriani et al. (2019).

```
imputed_rent <- hh |>
  left_join( heads , join_by(interview__key == interview__key)) |>
  select( interview__key, hh_02, hh_03, hous_02, hous_10, hous_04,mem_02,
          mem_05, ed_03, mem_num, hous_41, hous_19,hous_09, weight)
```

To save on the creation of unnecessary dummy variables for our regression, we take advantage of the factors present in the original SPSS files, which carry over when importing into R and are used by it to create them automatically at prediction time. Pay attention to the creation of age groups using `cut()` .

```
# Convert categorical variables to factors and create dummy variables
imputed_rent <- imputed_rent  |>
  mutate(hh_02 = as.factor(hh_02),           # Marz
         hh_03 = as.factor(hh_03),           # Urban / Rural
         hous_02 = as.factor(hous_02),       # Ownership or rental
         mem_02 = as.factor(mem_02),         # Sex
         ed_03 = as.factor(ed_03),           # Education level
         hous_41 = as.factor(hous_41),       # Type of toilet
         hous_19 = as.factor(hous_19))  |>  # Source of electricity
  mutate(age_group = cut(mem_05, breaks = c(0, 24, 34, 44,
                                            54, 64, Inf),
                         labels = c("15-24", "25-34", "35-44",
                                    "45-54", "55-64", "65+"),
                         right = TRUE)) |>
  mutate(age_group = as.factor(age_group)) |>
  mutate(bathroom_dummy = ifelse(hous_41 == 1, 1, 0)) |>
  mutate(bathroom_dummy = as.factor(bathroom_dummy)) |>
  select(-mem_05, -hous_41)  # Remove the original age variable
```

For our model, we need to concentrate on tenants who pay rent. So we subset further creating a data set called `renters_df`. Variable `hous_02` asks whether the household owns this dwelling or it is rented (with possible values 1. own, 2. rent, 3. other). And for renters, we want those whose value is larger than zero.

```
renters_df <- imputed_rent |>
  filter(hous_02 == 2) |>
  filter(!is.na(hous_04)) |>
  filter(hous_04 >0)
```

We are now ready to build our model:

```
log_linear_model <- lm(log(hous_04) ~        # Rent, which depends on:
                         hh_02 +             # Marz
                         hh_03 +             # Urban / Rural
                         hous_10 +           # Number of rooms
                         mem_02 +            # Sex of head of HH
                         ed_03 +             # Education level
```

6

```
                        mem_num +           # Number of HH members
                        bathroom_dummy +    # Flushing toilet dummy
                        hous_09 +           # Total square meters
                        age_group,          # Age brackets
                    data = renters_df)
```

For space considerations, we omit the output of the model, but you can inspect the results of the model with `summary(log_linear_model)`. This particular application for Armenia results in small positive significance for total square meters and having a flushing toilet, small negative significance for being female and high negative significance for the Marzes in relation to Yerevan, as well as high negative significance for rural areas (Multiple R-squared: 0.4883). In other words, rent for Armenians will be higher if they live in urban areas, have a working toilet, have a larger imputed_rent and the head of household is male. With our coefficients we can now impute rent for our non renters.

Before we move on, we need to de-factor some variables and re-code them so that our predictions run smoothly. We did not get predictions for education level 0 in our renters database, but there are some in our `non_renters_df` data set. Since they are factors (ie. categorical values) and not years of education, when R is running the regression, it creates dummy variables in the background for each level that it encounters in the data. Since that level was missing in the renters data, the prediction does not include it. So when it encounters that value in the non-renters data frame, R does not know how to handle it. This might not happen in your data set, but beware that if it does, this is the reason why your model won't predict. The error that gave this away read.

```
Error in model.frame.default(Terms, newdata,
na.action = na.action, xlev = object$xlevels):
factor ed_03 has new levels 0
```

So let's take care of non-trained values by making the decision to change the 12 cases that responded "none" to "primary". Another option would be to change it to "other", but since the prediction there was made with 1 observation we felt it was less of a disturbance this way. We find the values to change by indexing in square brackets; a powerful way of base R to slice data sets in multiple ways.

```
# Take care of non training values in  the original data set
# Convert to numeric to perform the operation
imputed_rent$ed_03 <- as.numeric(as.character(imputed_rent$ed_03))
imputed_rent$ed_03[imputed_rent$ed_03 == 0] <- 1 # Re-code 0 to 1
imputed_rent$ed_03 <- as.factor(imputed_rent$ed_03) # Convert back to factor
```

With everything in place, we can now predict the imputed rent. Actually, the **log** of predicted rent, so we transform the log value to value in the next pipe. We can do two things. One is to create a non-renters data set, predict rent there and then join with the renters data frame. Another is just to apply the prediction to the entire imputed_rent data set and then just replace the result with missing values for the renters. We will do the latter.

```r
imputed_rent <- imputed_rent |>
  add_predictions(log_linear_model,
                  var = "log_rent_predicted") |>
  mutate(imputed_rent = exp(log_rent_predicted)) |>
  # Replace renters imputed value with "missing"
  mutate(imputed_rent = if_else(hous_02 %in% c("2", "3"),
                                NA, imputed_rent)) |>
  # We just keep the household id and the imputed value going forward
  select( interview__key, imputed_rent)

# Remove intermediate products
rm(heads, log_linear_model, renters_df)
```

At this point we can save the prediction if we wish to do so to disk, but it is not necessary for our purposes here as we can continue using the created object `imputed_rent` in our calculations going forward. For example to output to Excel, Stata, SPSS, and CSV we would write (make sure your outputs directory exists):

```r
# Stata
write_dta(imputed_rent, "outputs/imputed_rent.dta", version = 10)
# Excel
write.xlsx(imputed_rent,"ouptuts/imputed_rent.xlsx",
           sheetName = "imputed_rent",
           rowNames = FALSE,
           colnames = FALSE,
           overwrite = TRUE,
           asTable = FALSE
)
# SPSS
write_sav(data, "outputs/imputed_rent.sav")
# Comma Separated Values
write.csv(imputed_rent,"outputs/imputed_rent.csv" sep = ",")
```

Let's explore the results, by first summarizing the data.

```
# Average imputed rent by marz
imputed_rent_marz <- hh |>
  left_join( deciles, join_by( interview__key == hhid)) |>
  left_join( imputed_rent, join_by( interview__key == interview__key)) |>
  select(decile, hh_02, hh_03, hous_10, imputed_rent,weight, NAM_1) |>
  group_by(NAM_1) |>
  summarize(avg_dwelling_m2 =
              weighted.mean(hous_10, as.integer(weight), na.rm = TRUE),
            avg_imputed_rent =
              weighted.mean(imputed_rent, as.integer(weight), na.rm = TRUE)*er)
```

And then making a table.

```
gt_table <- imputed_rent_marz |>
  gt() |>
   tab_header(
    title = "Imputed rent in Armenia",
    subtitle = "Average dwelling area and imputed rent (Year 2022)"
  ) |>
  grand_summary_rows(
    columns = c(avg_dwelling_m2,avg_imputed_rent),
    fns= list(
      Average = ~mean(., na.rm = TRUE)
      ),
    fmt = list(~ fmt_number(., decimals = 1))
  ) |>
  fmt_number(
    columns = c(avg_dwelling_m2, avg_imputed_rent),
    decimals = 1
  ) |>
  cols_label(
    NAM_1 = "Marz",
    avg_dwelling_m2 = "Average dwelling area ({{m^2}})",
    avg_imputed_rent = "Average imputed monthly rent (USD)"
  ) |>
    tab_source_note(
    source_note = md("Own elaboration based on Armenia Integrated Living Conditions Survey (A
  )
# gt_table |>
#   as_raw_html()
gt_table
```

Imputed rent in Armenia
Average dwelling area and imputed rent (Year 2022)

	Marz		Average imputed monthly rent (USD)
	Aragatsotn	103.8	87.2
	Ararat	102.6	93.3
	Armavir	109.3	92.9
	Gergharkunik	115.4	68.1
	Kotayk	96.2	100.4
	Lori	87.9	51.8
	Shirak	82.7	65.9
	Syunik	95.1	89.2
	Tavush	97.5	53.4
	Vayots Dzor	104.7	66.6
	Yerevan	70.9	216.2
Average	—	96.9	89.5

Own elaboration based on Armenia Integrated Living Conditions Survey (ARMSTAT, 2023).

0.3.2 Net present value of imputed rent

The previous steps help us determine the imputed monthly rent for home owners. We can treat this income as an asset by considering the net present value of future rents. We use the traditional formula:

$$\text{NPV} = \sum_{t=0}^{N} \frac{C_t}{(1+r)^t} \tag{1}$$

Where:

- NPV = Net Present Value
- C_t = Net cash inflow during the period t
- r = Discount rate
- t = Time period
- N = Total number of periods

We will use an annual discount rate of 5%, which is customary for homes, an inflation of 5% for 27 years, since the survey was conducted in the last few months of 2022 and we are making the calculation from January 01 2023 to December 31, 2050.

```
# Parameters
discount_rate <- 0.05 # Annual discount rate, for example, 5%
inflation_rate <- 0.05 # Annual inflation rate, for example, 5%
years <- 28 # Number of years to discount

# Adjust rates for monthly compounding, to avoid overestimation
monthly_discount_rate <- (1 + discount_rate)^(1/12) - 1
monthly_inflation_rate <- (1 + inflation_rate)^(1/12) - 1

# Annual imputed rent
imputed_rent$annual_imputed_rent <- imputed_rent$imputed_rent * 12
```

We can do two things, either annualize the monthly income or divide our rates by 12 and have the periods in the formula be months. It depends on the kind of shocks that we want to do. For example, if we know that a 1 in a 100 year event will have an impact that will last, let's say one and a half year, then having months is useful as we can introduce the shock as a tax that has an effect on 18 months worth of net present value. However, if we know that our shocks will have annual consequences, then doing our calculations year by year is enough.

To calculate each months worth of discounted present value, we use `sapply()` to perform the calculation over $years * 12$ months. This is similar to using for loops in other languages, but it is much more efficient, because it works hard to summarize results as vectors, and avoids iterations. In this case, since we are operating the formula over the entire vector of imputed rents, month, the result `return(present_value)` is not a vector but a matrix called `present_value` that gets attached to our data set at once (not column by column) where each column represents a month's worth of discounted present value for each household in the rows. Notice that we are using our modified monthly rates, which are adjusted (not just the annual divided by 12) to more accurately reflect the compounding value of money.

```
# Monthly periods
imputed_rent$present_value_rent <- sapply(1:(years * 12), function(n) {
  future_rent <- imputed_rent$imputed_rent * (1 + monthly_inflation_rate)^n
  present_value <- future_rent / ((1 + monthly_discount_rate)^n)
  return(present_value)
})
```

That results in the creation of a matrix containing the monthly discounted values by month. We can see a snippet of this attached matrix, filtering for the observations that are not missing values and showing only the first four valid households and months using:

```
imputed_rent$present_value_rent[
  !is.na(imputed_rent$present_value_rent[, 1]), ][1:4, 1:4]
```

```
       [,1]      [,2]      [,3]      [,4]
1 20134.17 20134.17 20134.17 20134.17
2 20635.94 20635.94 20635.94 20635.94
3 20491.28 20491.28 20491.28 20491.28
4 25328.52 25328.52 25328.52 25328.52
```

After that we sum over the columns corresponding to our monthly discounted values:

```
# Sum up the present values for the total present value over the period
imputed_rent$net_present_value_rent <- rowSums(imputed_rent$present_value_rent)
```

For comparison, we can do it annually as well. Here we use our annual rates (discount, and inflation) to generate a similar matrix, where all columns refer to years.

```
# Annual periods
imputed_rent$present_value_rent2 <- sapply(1:(years), function(n) {
  future_rent2 <- imputed_rent$annual_imputed_rent * (1 + inflation_rate)^(n)
  present_value2 <- future_rent2 / ((1 + discount_rate)^n)
  return(present_value2)
})
```

```
# Sum up the present values for the total present value over the period
imputed_rent$net_present_value_rent2 <- rowSums(imputed_rent$present_value_rent2)
```

```
# Delete partial calculations
imputed_rent <- imputed_rent |>
  select(interview__key, imputed_rent, net_present_value_rent, net_present_value_rent2)
```

For a comparison between the two calculations of net present value, we can compare their means and their difference should be zero:

```
hh |>
  left_join( deciles, join_by( interview__key == hhid)) |>
  left_join( imputed_rent, join_by( interview__key == interview__key)) |>
  select(decile, hh_02, hh_03, net_present_value_rent, net_present_value_rent2, NAM_1, weight
  group_by(NAM_1) |>
  summarize(avg_net_present_value1 = weighted.mean(net_present_value_rent, as.integer(weight)
            avg_net_present_value2 = weighted.mean(net_present_value_rent2, as.integer(weight
            difference =
                round(avg_net_present_value1 - avg_net_present_value2)) |>
  gt() |>
```

```
  fmt_number(
    columns = everything(),
    decimals = 1
  ) |>
  cols_label(
    NAM_1 = "Marz",
    avg_net_present_value1 = "Average net present value in USD (calculated monthly)",
    avg_net_present_value2 = "Average net present value in USD (calculated annual)",
    difference = "Difference"
  )
```

Marz	Average net present value in USD (calculated monthly)	Average net present value in USD (
Aragatsotn	29, 315.1	
Ararat	31, 349.4	
Armavir	31, 230.6	
Gergharkunik	22, 881.6	
Kotayk	33, 741.8	
Lori	17, 397.0	
Shirak	22, 134.2	
Syunik	29, 964.8	
Tavush	17, 951.5	
Vayots Dzor	22, 362.3	
Yerevan	72, 641.5	

Using our Shapefile, we can explore how the average net present value of imputed rent distributes geographically (labels in million dram).

```
npv <- hh |>
  left_join( imputed_rent, join_by( interview__key == interview__key)) |>
  select(hh_02, hh_03, net_present_value_rent,
         net_present_value_rent2, NAM_1, weight) |>
  group_by(NAM_1) |>
  summarize(
    avg_npv = weighted.mean(
      net_present_value_rent, as.integer(weight), na.rm = TRUE
      ),
    avg_npv_labels =
      formatC(
        weighted.mean(net_present_value_rent,
                      as.integer(weight), na.rm = TRUE)*er,
        big.mark = ",", format = "f", digits = 1)
```

```
      )

npv_map <- adm1 |>
  left_join(npv, join_by(NAM_1 == NAM_1))

npv_map <-tm_shape(npv_map)+
  tm_polygons("avg_npv", legend.show = FALSE) +
  tm_text("avg_npv_labels", size = .7, col = "black")+
  tm_layout(legend.position = c("right", "top"),
            title= "Average Imputed Rent NPV (USD)",
            title.position = c('left', 'bottom'),
            title.size = 0.9)
npv_map
```
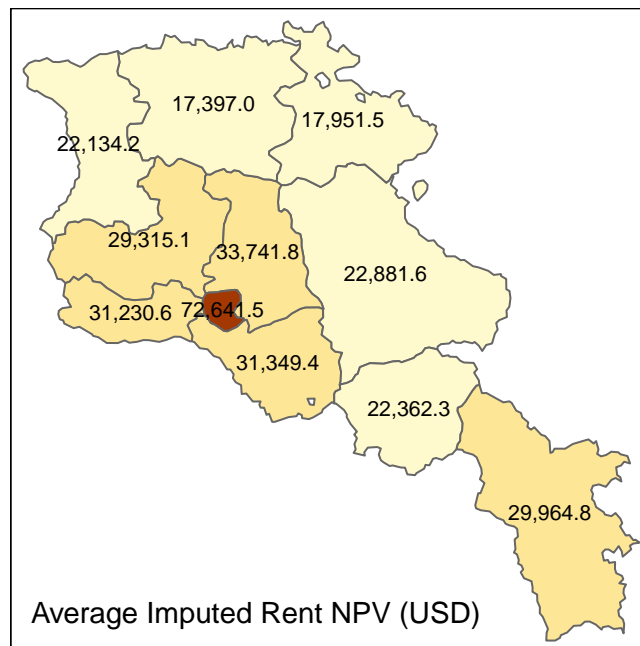

Figure 1: Average Net Present Value of Imputed Rent by Marz (Year 2022, USD)

In the next section we will perform the same calculations to treat agricultural income as a future discounted asset so that we can implement shocks, according to vulnerability data.

14

0.3.3 Net Present Value of Agricultural Income

For the Armenian case, we have access to an already processed data set with income aggregations from hired employment, self-employment, income on property, public pensions, transfers, other income, and, especially important for this section, agricultural income. Let's calculate the net present value for that income in the same way we did our imputed rent. We will focus on the annual version of our calculations. There is one caveat, which is related to the fact that agricultural income shows seasonal variations. So multiplying the monthly income by 12 is likely to overestimate the net present value so we use a scaling factor which should reflect that seasonality. We will use the same rates as before so we will not create new ones.

```
ag_scaling <- 0.65
ag_income <- ic |>
  mutate(lvstk = if_else(lvstk<0, 0, lvstk)) |>
  mutate(annual_ag_income = ((inc4 - lvstk) * 12 * ag_scaling))
```

And we perform the same Net Present Value calculations as before (see Equation 1).

```
# Annual periods
ag_income$present_value_ag_income <- sapply(1:(years), function(n) {
  future_ag_income <- ag_income$annual_ag_income * (1 + inflation_rate)^(n)
  present_ag_value <- future_ag_income / ((1 + discount_rate)^n)
  return(present_ag_value)
})
```

```
# Sum up the present values for the total present value over the period
ag_income$net_present_ag_value <- rowSums(ag_income$present_value_ag_income)
```

```
# Delete partial calculations
ag_income <-ag_income |>
  select(interview__key, present_value_ag_income, annual_ag_income, net_present_ag_value)
```

We will create a map object to show a side-by-side comparison of both NPVs.

```
npv_ag <- hh |>
  left_join( ag_income, join_by( interview__key == interview__key)) |>
  select(hh_02, hh_03, annual_ag_income, net_present_ag_value, NAM_1, weight) |>
  group_by(NAM_1) |>
  summarize(
    avg_ag_npv = weighted.mean(
      net_present_ag_value, as.integer(weight), na.rm = TRUE ),
```

```
    avg_ag_npv_labels =
        formatC(
          weighted.mean(
            net_present_ag_value *er,
            as.integer(weight), na.rm = TRUE),
          big.mark = ",",
          format = "f", digits = 1) )

npv_ag_map <- adm1 |>
  left_join(npv_ag, join_by(NAM_1 == NAM_1))

npv_ag_map <- tm_shape(npv_ag_map)+
  tm_polygons("avg_ag_npv", legend.show = FALSE) +
  tm_text("avg_ag_npv_labels", size = .7, col = "black")+
  tm_layout(legend.position = c("right", "top"),
            title= "Average Ag. Income NPV (USD)",
            title.position = c('left', 'bottom'),
            title.size = 0.9)
```

And now we can compare the spatial distributions of both Net Present Values from the map objects `npv_map` and `npv_ag_map` that we created before.

```
tmap_arrange(npv_map, npv_ag_map)
```
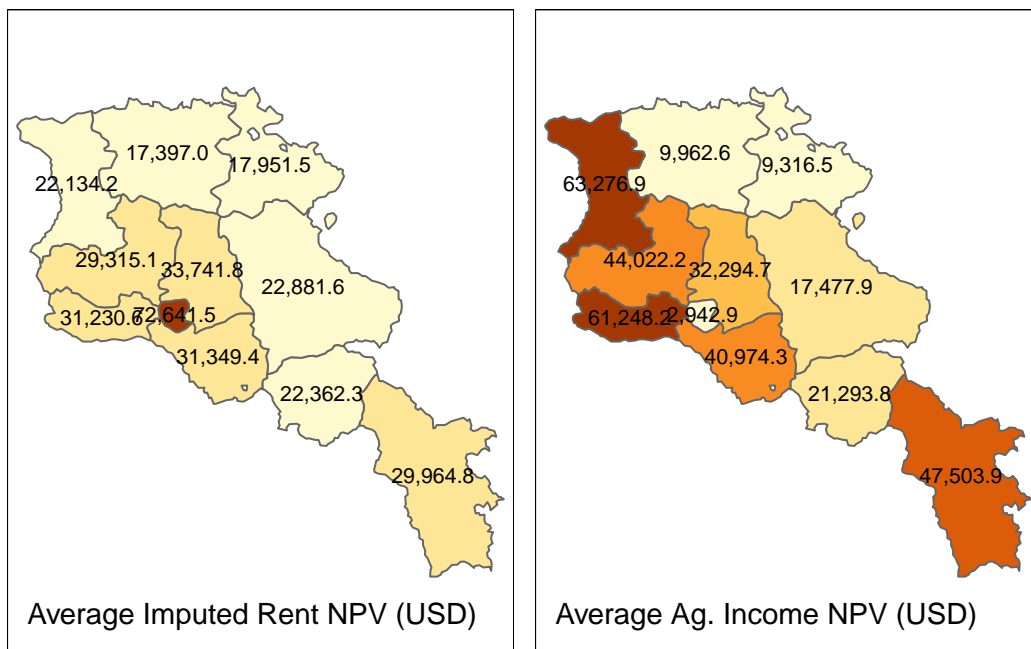
Figure 2: Average Net Present Value of Imputed Rent and Agricultural Income (Year 2022, million AMD)

With both Net Present Values calculated, in the following section we will apply our vulnerability shocks to a selection of Armenian households.

0.4 Vulnerability shocks

0.4.1 Buildings

We previously estimated the imputed rent values for households that own their homes, assuming that they derive welfare from owning those assets. We then treated that discounted future income flow as an asset value. Our data suggests that some of those buildings are damaged due to increased rain and flood events. In each administrative region a percentage of buildings receive these shocks, effectively taxing their monthly imputed value by a percentage. Let's find a way to randomly select from our data set a number of weighted households that matches the shocks. Let's move step by step.

We first merge the household data set with the imputed_rent data set to have the descriptive variables per household.

```
# Create a placeholder for our chosen HH's
rent_dataset <- hh |>
  mutate(selected_for_tax = FALSE, # initialize with FALSE
         is_dilapidated = if_else(hous_45__7 == 1, TRUE, FALSE)
         ) |>
  mutate(is_dilapidated = if_else(is.na(hous_45__7), FALSE, is_dilapidated)) |>
  left_join(imputed_rent, join_by(interview__key == interview__key )) |>
  rename(household_id = interview__key)

exposure_dataset <- buildings_aal
```

We merge with the exposure data set and prepare the necessary columns.

```
rent_dataset <- rent_dataset |>
  left_join(exposure_dataset, by = "NAM_1") |>
  mutate(is_dilapidated = if_else(is.na(hous_45__7) | hous_45__7 == 0, FALSE, TRUE))
```

The next step involves calculating the target weight for each marz (`NAM_1`) and randomly
selecting households based on their weight until the cumulative sum matches the target.

```
set.seed(123)  # Ensure reproducibility

rent_dataset <- rent_dataset |>
  group_by(NAM_1) |>
  mutate(total_weight = sum(weight, na.rm = TRUE),
         target_weight = total_weight * pct_AA_exposed_buildings / 100) |>
  ungroup() |>
  arrange(NAM_1, runif(n())) |>
  group_by(NAM_1) |>
  mutate(cum_weight = cumsum(weight),
         selected_for_tax = cum_weight <= target_weight) |>
  ungroup()
```

We can explore how many observations were targeted and their weighted values.

```
rent_dataset |>
  filter(selected_for_tax == TRUE) |>
  group_by(hh_02,NAM_1)  |>
  summarise(Selected_Cases = sum(selected_for_tax, na.rm = TRUE),
            Weighted_no_HHs = as.integer(sum(weight, na.rm = TRUE))) |>
  arrange(hh_02) |>
```

```
  adorn_totals("row") |>
  gt() |>
  fmt_number(
    columns = everything(),
    decimals = 0
  ) |>
  cols_label(
    hh_02 = "",
    NAM_1 = "Marz",
    Selected_Cases = "No. of targeted observations",
    Weighted_no_HHs = "No. of targeted weighted households"
  )
```

	Marz	No. of targeted observations	No. of targeted weighted households
1	Yerevan	31	5,973
2	Aragatsotn	13	1,503
3	Ararat	19	3,434
4	Armavir	12	1,647
5	Gergharkunik	10	1,675
6	Lori	15	2,286
7	Kotayk	10	2,355
8	Shirak	13	2,060
9	Syunik	15	1,761
10	Vayots Dzor	14	623
11	Tavush	6	752
Total	-	158	24,069

Now, we apply the "vulnerability tax" to the imputed_rent according to the building's state of dilapidation and the specific tax rates for normal and dilapidated conditions.

```
rent_dataset <- rent_dataset |>
  mutate(adjusted_rent = case_when(
    selected_for_tax & is_dilapidated ~ imputed_rent * (1 - perc_AAL_dilapidated / 100),
    selected_for_tax & !is_dilapidated ~ imputed_rent * (1 - perc_AAL_normal / 100),
    TRUE ~ imputed_rent
  ))
```

We then compare our values in a table. In this case, we see that, although the percentages of impacted households are small per marz, the mean imputed rent decrease, which is a value that is expected to compound over time.

And we can now estimate a new net present value with the adjusted values in the same manner as before.

```
# Annual adjusted imputed rent
rent_dataset$annual_adjusted_rent <- rent_dataset$adjusted_rent * 12


rent_dataset$adjusted_present_value_rent <- sapply(1:(years), function(n) {
  future_adjusted_rent <- rent_dataset$annual_adjusted_rent * (1 + inflation_rate)^(n)
  present_adjusted_value <- future_adjusted_rent / ((1 + discount_rate)^n)
  return(present_adjusted_value)
})
```

```
# Sum up the present values for the total present value over the period
rent_dataset$adjusted_net_present_value_rent <-
  rowSums(rent_dataset$adjusted_present_value_rent)
```

And we can view the compounded differences in NPVs for those affected by Marz and Poverty condition. Note that the weighted sum of households (24,069-24,075) in the tables differs slightly, because of rounding during the sliced calculations to avoid showing fractions of individual homes.

```
building_losses <- rent_dataset |>
  left_join(deciles, join_by(household_id == hhid)) |>
  rename(poor = poor_Avpovln2022) |>
  filter(selected_for_tax == TRUE) |>
  group_by(hh_02,NAM_1) |>
  summarize(
    Average_NPV = weighted.mean(net_present_value_rent, weight, na.rm = TRUE)*er,
    Average_Adjusted_NPV = weighted.mean(
      adjusted_net_present_value_rent, weight, na.rm = TRUE) * er,
    Difference =  Average_NPV - Average_Adjusted_NPV,
    No_HH = round(sum(weight, na.rm = TRUE))
  ) |>
   ungroup()

building_losses |>
    gt() |>
  fmt_number(
    columns = c(Average_NPV,Average_Adjusted_NPV, Difference) ,
    decimals = 1
  ) |>
```

```
cols_label(
  hh_02 = "",
  NAM_1 = "Marz",
  Average_NPV = "Average NPV of imputed rent (USD)",
  Average_Adjusted_NPV = "Adjusted average NPV of imputed rent (USD)",
  Difference = "Loss (USD)",
  No_HH = "No. HH"
)
```

	Marz	Average NPV of imputed rent (USD)	Adjusted average NPV of imputed rent (USD)
1	Yerevan	79,039.9	73,027.3
2	Aragatsotn	27,581.9	25,179.9
3	Ararat	24,145.6	22,245.4
4	Armavir	37,176.7	34,256.1
5	Gergharkunik	24,358.7	22,134.0
6	Lori	17,253.5	13,626.3
7	Kotayk	27,414.9	24,925.5
8	Shirak	25,245.4	22,944.3
9	Syunik	30,587.9	25,203.3
10	Vayots Dzor	23,868.1	20,056.4
11	Tavush	24,120.8	19,180.6

```
rent_dataset |>
  left_join(deciles, join_by(household_id == hhid)) |>
  #rename(poor = poor_Avpovln2022) |>
  filter(selected_for_tax == TRUE) |>
  mutate(poor = if_else(poor_Avpovln2022==1,
                        "Below poverty line",
                        "Above poverty line")) |>
  group_by( poor) |>
  summarize(
    Average_NPV = weighted.mean(net_present_value_rent, weight, na.rm = TRUE) * er,
    Average_Adjusted_NPV = weighted.mean(
      adjusted_net_present_value_rent, weight, na.rm = TRUE) * er,
    Difference =  Average_NPV - Average_Adjusted_NPV,
    No_HH = round(sum(weight, na.rm = TRUE))
  ) |>
  ungroup() |>
  gt() |>
  fmt_number(
    columns = c(Average_NPV,Average_Adjusted_NPV, Difference) ,
```

```
    decimals = 1
  ) |>
  cols_label(
    poor = "Poverty",
    Average_NPV = "Average NPV of imputed rent (USD)",
    Average_Adjusted_NPV = "Adjusted average NPV of imputed rent (USD)",
    Difference = "Loss (USD)",
    No_HH = "No. HH"
  )
```

Poverty	Average NPV of imputed rent (USD)	Adjusted average NPV of imputed rent (USD)
Above poverty line	$35,305.8$	$31,792.8$
Below poverty line	$46,736.4$	$42,633.1$

0.4.2 Agriculture

In the case of agriculture, we have percentage losses in agricultural productivity per year and per climate scenario from our `crops_aal` data set.

```
crops_aal |>
    select(-GID_1) |>
  group_by(climate_scenario) |>
  summarize(`Mean percent change all years` = mean(pct_change_prod)) |>
  gt()
```

climate_scenario	Mean percent change all years
Dry/Hot mean	-0.19475663
SSP1-1.9 mean	-0.04033261
SSP2-4.5 CAMS-CSM1-0	-0.06785020
SSP2-4.5 CMCC-CM2-SR5	-0.03269785
SSP2-4.5 CNRM-CM6-1	-0.08440299
SSP3-7.0 BCC-CSM2-MR	-0.15777422
SSP3-7.0 KACE-1-0-g	-0.20606020
SSP3-7.0 UKESM1-0-LL	-0.21341230
SSP3-7.0 mean	-0.08120867
Wet/Warm mean	-0.06320857

So we will apply this to our agricultural net present value calculations per year. Remember that when we calculated the present values of each year we were left with a matrix identifying

each year's value per household id. So we are going to pivot the data set into long format, so that we can match the appropriate loss in productivity according to year and marz. To test out our methodology we will keep one scenario only in our `ag_exposure` data set. Also, we will change the percent change to decimal so that we are not over-estimating productivity losses. We also leave out 2021-2022 because our NPV calculations started from 2023 to 2050.

```
ag_exposure <- crops_aal |>
  filter(climate_scenario=="Dry/Hot mean" & year > 2022 ) |>
    select(-GID_1, -climate_scenario) #|>
  #mutate(pct_change_prod = pct_change_prod / 100)
```

Our `ag_income` data set from before has the present values of each individual household in columns that represent each year. We can rename those columns with our year variables so that we can match them with our exposure data set.

```
# Define the years range
years <- as.character(2023:2050)

# Rename the columns of the matrix
colnames(ag_income$present_value_ag_income) <- years
```

We extract our present value calculations from the matrix column and convert it into a data set on its own, which we can manipulate into long format to merge with our exposure data set.

```
# Convert the matrix to a data.frame.
ag_income_long <- as.data.frame(ag_income$present_value_ag_income)  |>
  mutate(household_id = ag_income$interview__key) |>
  pivot_longer(cols = -household_id, names_to = "year",
               values_to = "present_value_ag_income") |>
  left_join(hh, join_by(household_id == interview__key)) |>
  select(household_id, present_value_ag_income, year, hh_02, NAM_1)

ag_income_long$year <- as.integer(ag_income_long$year)

# inspect the final result
head(ag_income_long[!is.na(
  ag_income_long$present_value_ag_income),])
```

```
# A tibble: 6 x 5
  household_id present_value_ag_income  year hh_02           NAM_1
```

```
  <chr>                        <dbl> <int> <dbl+lbl>          <chr>
1 00-03-45-50                  58500  2023 10 [VAYOTS DZOR] Vayots Dzor
2 00-03-45-50                  58500  2024 10 [VAYOTS DZOR] Vayots Dzor
3 00-03-45-50                  58500  2025 10 [VAYOTS DZOR] Vayots Dzor
4 00-03-45-50                  58500  2026 10 [VAYOTS DZOR] Vayots Dzor
5 00-03-45-50                  58500  2027 10 [VAYOTS DZOR] Vayots Dzor
6 00-03-45-50                  58500  2028 10 [VAYOTS DZOR] Vayots Dzor
```

So now we can match our exposure values according to year and marz and modify our annual
present values.

```
ag_income_long <- ag_income_long |>
  left_join(ag_exposure, join_by(NAM_1, year)) |>
  mutate(adjusted_present_value_ag = present_value_ag_income * (1 + pct_change_prod))
```

And we can collapse our dataset again to sum over the years for each household and join it
back with our `ag_income` dataset.

```
ag_income_long <- ag_income_long |>
  select(household_id, adjusted_present_value_ag) |>
  group_by(household_id) |>
  summarize( net_present_ag_value_adjusted =
               sum(adjusted_present_value_ag, na.rm = TRUE))

ag_income <- ag_income |>
  left_join(ag_income_long, join_by(interview__key == household_id)) |>
  select(interview__key, net_present_ag_value, net_present_ag_value_adjusted)
```

Let's compare the two mean values.

```
ag_losses <- hh |>
  left_join(deciles, join_by(interview__key == hhid)) |>
  left_join(ag_income, join_by(interview__key==interview__key)) |>
  filter(!is.na(net_present_ag_value)) |>
  rename(poor = poor_Avpovln2022) |>
  group_by(hh_02,NAM_1) |>
  summarize(
    Average_NPV = weighted.mean(net_present_ag_value, weight, na.rm = TRUE)*er,
    Average_Adjusted_NPV = weighted.mean(
      net_present_ag_value_adjusted, weight, na.rm = TRUE) * er,
    Difference =  Average_NPV - Average_Adjusted_NPV,
```

```
    No_HH = round(sum(weight, na.rm = TRUE))
  ) |>
  ungroup()

ag_losses |>
    gt() |>
  fmt_number(
    columns = c(Average_NPV,Average_Adjusted_NPV, Difference) ,
    decimals = 1
  ) |>
  cols_label(
    hh_02 = "",
    NAM_1 = "Marz",
    Average_NPV = "Avg. NPV of ag income (USD)",
    Average_Adjusted_NPV = "Adjusted avg. NPV of ag income (USD)",
    Difference = "Loss (USD)",
    No_HH = "No. HH"
  )
```

	Marz	Avg. NPV of ag income (USD)	Adjusted avg. NPV of ag income (USD)	Loss (USD)
1	Yerevan	$2,942.9$	$2,376.1$	566.9
2	Aragatsotn	$44,013.8$	$35,240.6$	$8,773.3$
3	Ararat	$40,974.3$	$31,920.8$	$9,053.4$
4	Armavir	$61,242.7$	$48,313.5$	$12,929.2$
5	Gergharkunik	$17,478.0$	$14,176.5$	$3,301.4$
6	Lori	$9,963.7$	$8,148.5$	$1,815.3$
7	Kotayk	$32,294.9$	$26,326.4$	$5,968.5$
8	Shirak	$63,270.8$	$51,073.9$	$12,196.9$
9	Syunik	$47,500.3$	$40,066.0$	$7,434.3$
10	Vayots Dzor	$21,296.9$	$15,854.6$	$5,442.3$
11	Tavush	$9,322.4$	$7,590.9$	$1,731.5$

And like in our previous example, we can see how the change affects the poor. It appears that
in this case those above the poverty line have the greatest average impact.

```
hh |>
  left_join(deciles, join_by(interview__key == hhid)) |>
  left_join(ag_income, join_by(interview__key==interview__key)) |>
  filter(!is.na(net_present_ag_value)) |>
  mutate(poor = if_else(poor_Avpovln2022==1,
                    "Below poverty line",
```

```
                        "Above poverty line")) |>
  group_by(poor) |>
  summarize(
    Average_NPV = weighted.mean(net_present_ag_value, weight, na.rm = TRUE)*er,
    Average_Adjusted_NPV = weighted.mean(
      net_present_ag_value_adjusted, weight, na.rm = TRUE) * er,
    Difference =  Average_NPV - Average_Adjusted_NPV,
    No_HH = round(sum(weight, na.rm = TRUE))
  ) |>
    gt() |>
  fmt_number(
    columns = c(Average_NPV,Average_Adjusted_NPV, Difference) ,
    decimals = 1
  ) |>
  cols_label(
    poor = "Poverty",
    Average_NPV = "Average NPV of ag income (USD)",
    Average_Adjusted_NPV = "Adjusted average NPV of ag income (USD)",
    Difference = "Loss (USD)",
    No_HH = "No. HH"
  )
```

Poverty	Average NPV of ag income (USD)	Adjusted average NPV of ag income (USD)	Loss
Above poverty line	35, 286.4	28, 423.8	6
Below poverty line	20, 490.2	16, 423.4	4

We can also explore how both losses are geographically distributed.

```
# For Buildings
building_losses <- adm1 |>
  left_join(building_losses, join_by(NAM_1 == NAM_1)) |>
  mutate(map_labels = formatC(building_losses$Difference,
                                        big.mark = ",",
                                        format = "f",
                                        digits = 1))


building_losses_map <- tm_shape(building_losses)+
  tm_polygons(col = "Difference", legend.show = FALSE,
              palette = c("YlGn"))+
  tm_text("map_labels", size = .7, col = "black")+
```

```
  tm_layout(legend.position = c("right", "top"),
           title= "Average Imputed Rent NPV Losses (USD)",
           title.position = c('left', 'bottom'),
           title.size = 0.9)
```

```
ag_losses <- adm1 |>
  left_join(ag_losses, join_by(NAM_1 == NAM_1)) |>
  mutate(map_labels = formatC(ag_losses$Difference,
                                          big.mark = ",",
                                          format = "f",
                                          digits = 1))


ag_losses_map <- tm_shape(ag_losses)+
  tm_polygons(col = "Difference", legend.show = FALSE,
              palette = c("YlGn"))+
  tm_text("map_labels", size = .7, col = "black")+
  tm_layout(legend.position = c("right", "top"),
           title= "Average Ag. Income NPV Losses (USD)",
           title.position = c('left', 'bottom'),
           title.size = 0.9)
```

And now we can compare the spatial distributions of both Net Present Values from the map objects `npv_map` and `npv_ag_map` that we created before.
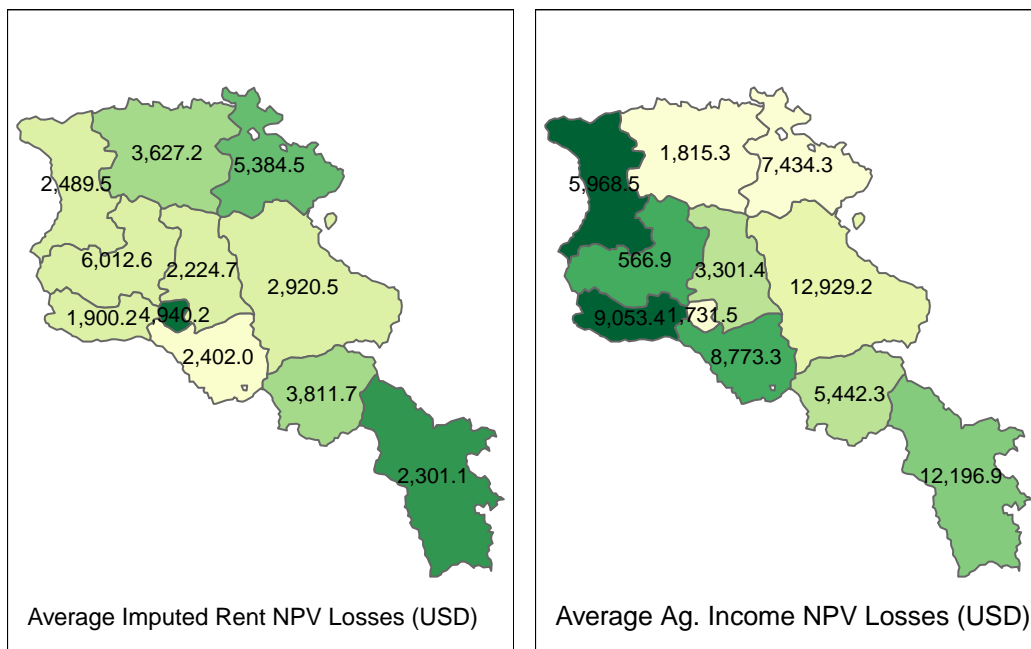
```
tmap_arrange(building_losses_map, ag_losses_map)
```

Figure 3: Average Losses of NPV of Imputed Rent and Agricultural Income (Year 2022, USD)

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

Ceriani, L., Olivieri, S., & Ranzani, M. (2019). Housing, imputed rent, and households' welfare. *Poverty & Equity Global Practice Working Papers*, (213). Retrieved from https://documents1.worldbank.org/curated/pt/336451565194643402/pdf/Housing-Imputed-Rent-and-Households-Welfare.pdf

Deaton, A., & Zaidi, S. (2002). *Guidelines for Constructing Consumption Aggregates for Welfare Analysis* (No. 135).