Vulnerability Analysis Calculations

Renato Vargas

Natsuko Kiso Nozaki

Julie Rozenberg

Colin Lenoble

Table of contents

## 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:

root/  
├── scripts  
│ └── my\_script.R  
├── data/  
| ├── my\_data.sav  
| ├── my\_data.dta  
| └── my\_data.csv  
└── output  
 ├── my\_output1.csv  
 └── my\_output2.xlsx

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

## 2 Preamble

We start with a clean environment, making sure that any objects from a previous session are not present. We take this 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"

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).

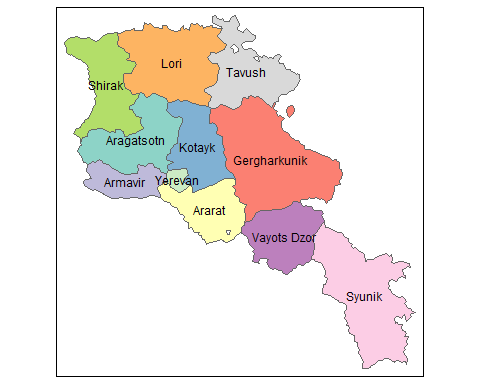
# Original SPSS datasets  
# Households (hh)  
hh <- read\_sav(  
 "data/ARM-HH-survey/original-spss-files/ILCS-ARM-2022-Households.sav")  
# Persons (pp)  
pp <- read\_sav(  
 "data/ARM-HH-survey/original-spss-files/ILCS-ARM-2022-Persons.sav")  
  
# Processed WB datasets  
# Consumption aggregate at 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 = "ADM1\_building\_AAL")  
buildings\_1in100 <- read\_xlsx("data/ARM-Vulnerability-Analysis/Data\_AAL\_Buildings.xlsx", sheet = "ADM1\_building\_1in100")  
crops\_aal <- read.csv("data/ARM-Vulnerability-Analysis/ARM\_crops\_combined\_REF\_shock\_admin1.csv")  
crops\_1in100 <- read\_xlsx("data/ARM-Vulnerability-Analysis/damages\_1in100\_agri\_ADM2.xlsx", sheet = "crops\_1in100")

## 3 Asset value of income flows

### 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  
 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.

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):

# 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))

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 (AMD)"  
 ) |>   
 tab\_source\_note(  
 source\_note = md("Own elaboration based on Armenia Integrated Living Conditions Survey (ARMSTAT, 2023).")  
 )  
# gt\_table |>   
# as\_raw\_html()  
gt\_table

Table 1: Imputed rent in Armenia

Average dwelling area and imputed rent (Year 2022)

|  | Marz | Average dwelling area (m<w:r><w:rPr><w:vertAlign w:val="superscript"></w:vertAlign></w:rPr><w:t xml:space="default">2</w:t></w:r>) | Average imputed monthly rent (AMD) |
| --- | --- | --- | --- |
|  | Aragatsotn | 103.8 | 37,769.4 |
|  | Ararat | 102.6 | 40,390.4 |
|  | Armavir | 109.3 | 40,237.3 |
|  | Gergharkunik | 115.4 | 29,480.5 |
|  | Kotayk | 96.2 | 43,472.8 |
|  | Lori | 87.9 | 22,414.2 |
|  | Shirak | 82.7 | 28,517.6 |
|  | Syunik | 95.1 | 38,606.5 |
|  | Tavush | 97.5 | 23,128.6 |
|  | Vayots Dzor | 104.7 | 28,811.5 |
|  | Yerevan | 70.9 | 93,590.9 |
| Average | — | 96.9 | 38,765.4 |
| Own elaboration based on Armenia Integrated Living Conditions Survey (ARMSTAT, 2023). | | |

### 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:

Where:

* = Net Present Value
* = Net cash inflow during the period
* = Discount rate
* = Time period
* = 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 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), na.rm = TRUE),  
 avg\_net\_present\_value2 = weighted.mean(net\_present\_value\_rent2, as.integer(weight), na.rm = TRUE),  
 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 (monthly)",  
 avg\_net\_present\_value2 = "Average net present value (annual)",  
 difference = "Difference"  
 )

| Marz | Average net present value (monthly) | Average net present value (annual) | Difference |
| --- | --- | --- | --- |
| Aragatsotn | 12,690,534.7 | 12,690,534.7 | 0.0 |
| Ararat | 13,571,189.9 | 13,571,189.9 | 0.0 |
| Armavir | 13,519,735.8 | 13,519,735.8 | 0.0 |
| Gergharkunik | 9,905,454.9 | 9,905,454.9 | 0.0 |
| Kotayk | 14,606,844.0 | 14,606,844.0 | 0.0 |
| Lori | 7,531,178.0 | 7,531,178.0 | 0.0 |
| Shirak | 9,581,921.6 | 9,581,921.6 | 0.0 |
| Syunik | 12,971,789.6 | 12,971,789.6 | 0.0 |
| Tavush | 7,771,222.4 | 7,771,222.4 | 0.0 |
| Vayots Dzor | 9,680,665.4 | 9,680,665.4 | 0.0 |
| Yerevan | 31,446,547.1 | 31,446,547.1 | 0.0 |

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 = paste(as.character(  
 formatC(  
 weighted.mean(net\_present\_value\_rent/1000000,   
 as.integer(weight), na.rm = TRUE),   
 big.mark = ",", format = "f", digits = 1)  
 ), " million", sep = ""  
 ))  
  
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",   
 title.position = c('left', 'bottom'),  
 title.size = 0.9)  
npv\_map

|  |
| --- |
| Figure 1: Average Net Present Value of Imputed Rent by Marz (Year 2022, million AMD) |

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.

### 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](#eq-npv)).

# 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, 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 = paste(  
 as.character(   
 formatC(   
 weighted.mean(  
 net\_present\_ag\_value/1000000,   
 as.integer(weight), na.rm = TRUE),   
 big.mark = ",",   
 format = "f", digits = 1) ), " million", sep = "" )  
 )  
  
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",   
 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.

## 4 Vulnerability shocks

### 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 = "Code",  
 NAM\_1 = "Marz",  
 Selected\_Cases = "No. of targeted observations",  
 Weighted\_no\_HHs = "No. of targeted weighted households"  
 )

| Code | 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.

rent\_dataset |>   
 left\_join(deciles, join\_by(household\_id == hhid)) |>   
 rename(poor = poor\_Avpovln2022) |>   
 filter(selected\_for\_tax == TRUE) |>   
 group\_by(NAM\_1) |>   
 summarize(  
 Average\_NPV = weighted.mean(net\_present\_value\_rent, weight, na.rm = TRUE),  
 Average\_Adjusted\_NPV = weighted.mean(  
 adjusted\_net\_present\_value\_rent, weight, na.rm = TRUE),  
 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(  
 NAM\_1 = "Marz",  
 Average\_NPV = "Average NPV of imputed rent",  
 Average\_Adjusted\_NPV = "Adjusted average NPV of imputed rent",  
 Difference = "Loss",  
 No\_HH = "No. HH"  
 )

| Marz | Average NPV of imputed rent | Adjusted average NPV of imputed rent | Loss | No. HH |
| --- | --- | --- | --- | --- |
| Aragatsotn | 11,940,217.6 | 10,900,402.4 | 1,039,815.2 | 1504 |
| Ararat | 10,452,620.9 | 9,630,026.7 | 822,594.2 | 3434 |
| Armavir | 16,093,795.0 | 14,829,495.7 | 1,264,299.3 | 1647 |
| Gergharkunik | 10,544,905.6 | 9,581,830.2 | 963,075.3 | 1676 |
| Kotayk | 11,867,930.2 | 10,790,245.2 | 1,077,685.0 | 2355 |
| Lori | 7,469,055.9 | 5,898,846.4 | 1,570,209.5 | 2287 |
| Shirak | 10,928,728.5 | 9,932,589.5 | 996,139.0 | 2060 |
| Syunik | 13,241,495.6 | 10,910,535.5 | 2,330,960.1 | 1761 |
| Tavush | 10,441,922.1 | 8,303,288.1 | 2,138,634.0 | 752 |
| Vayots Dzor | 10,332,516.1 | 8,682,429.2 | 1,650,086.9 | 624 |
| Yerevan | 34,216,391.3 | 31,613,534.9 | 2,602,856.4 | 5974 |

rent\_dataset |>   
 left\_join(deciles, join\_by(household\_id == hhid)) |>   
 rename(poor = poor\_Avpovln2022) |>   
 filter(selected\_for\_tax == TRUE) |>   
 group\_by( poor) |>   
 summarize(  
 Average\_NPV = weighted.mean(net\_present\_value\_rent, weight, na.rm = TRUE),  
 Average\_Adjusted\_NPV = weighted.mean(  
 adjusted\_net\_present\_value\_rent, weight, na.rm = TRUE),  
 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 = "Poor = 1",  
 Average\_NPV = "Average NPV of imputed rent",  
 Average\_Adjusted\_NPV = "Adjusted average NPV of imputed rent",  
 Difference = "Loss",  
 No\_HH = "No. HH"  
 )

| Poor = 1 | Average NPV of imputed rent | Adjusted average NPV of imputed rent | Loss | No. HH |
| --- | --- | --- | --- | --- |
| 0 | 15,283,899.5 | 13,763,117.6 | 1,520,781.9 | 19246 |
| 1 | 20,232,215.1 | 18,455,866.5 | 1,776,348.6 | 4829 |

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).