Village 2022 Income and Wealth: Code and Summaries

April 19, 2023

I did some minor changes in September 2023

1. use price of fertilizer given by Augus

This document presents the code and outputs from the file *income_wealth_22.py*. This file

- cleans the raw data
- check and correct values
- present summary statistics
- outputs clean data
  1. *income_wealth_22_rainseas.csv*
  2. *income_wealth_22_year.csv*

on agriculture, income, wealth and other variables from the raw dataset of phase-3 in the village for the year 2022. All codes, and datasets are in the dropbox folder.

Contents

1 Importing Data and Initial Checks **2**

2 Land **6**

3 Farming Capital and Livestock **11**

4 Agricultural Production **13**

5 Agricultural inputs and Production (at Plot Level) **28**

6 Cash-Transfer Subsidy **37**

7 Coupons and Fertilizer **38**

8 Agricultural Inputs: Labor, Fertilizer, and other Intermediates **42**

9 Shocks **47**

10 Formal Labor and Ganyu **49**

11 Business Income **53**

12 Transfers: Government, NGO, and Remittances **54**

13 Aggregate Income 55

14 Wealth 57

1 Importing Data and Initial Checks

[1]:
```
# ==============================================================================
# Village Income and Wealth July 2022
# ==============================================================================
# INCLUDES ============
# agriculture: output (in kg and monetary value) and inputs (including land,␣
↪labor, fertilizers, etc.).
# non-agric income: labor, ganyu, business, other.
# Wealth: farming capital, hh assets, etc.
# labor supply.
# shocks
# Checks and correction of the data:
    #missing hhs, duplicates hhs.
    # outliers
    # Apply Augustine corrections from the feedback in November.

# OUTPUT ==============
    # income_wealth_22_rainseas.csv
    # income_wealth_22_rainseas.csv
    # income_wealth_22_year.csv

## MISSING ===========
# clean land quality variables.
# clean variables on expectations in finding workers/jobs (Ying questions)
# More questions/variables.

# I think if we have a more narrow question then I can clean some of the␣
↪variables missing or have
# a better idea of what final dataset we are looking for.

root_path = 'C:/Users/rodri/Dropbox/Malawi/SIEG2021 (1)/Data Collection July␣
↪2022'
path_19 = 'C:/Users/rodri/Dropbox/Malawi/Chied_Field_June_19/Data/'
save=True

folder_fig = root_path+'Figures'
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
import statsmodels.api as sm
os.chdir(root_path+'/Code/Phase 3/Auxiliary files/')
from data_functions_albert import remove_outliers, gini

# Set the working directory
os.chdir(root_path+'/Data/Phase 3/Income')

## Display  set-up
pd.options.display.float_format = '{:,.2f}'.format
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

os.environ['PYTHONWARNINGS']='ignore::FutureWarning'
import warnings
warnings.filterwarnings("ignore")

#July 14th 2022 MWK vs US dollar
dollar_MWK = 1030.36


# Import village 19 data
data19 = pd.read_csv(path_19+'/Finished dataframes/data19_w18.csv')


# ==============================================================================
# Import data
# ==============================================================================

data = pd.read_stata(root_path+"/Data/Raw/2022-SIEG-Phase 3-Final Data.dta",␣
 ↪convert_categoricals=False)
data.rename(columns={'householdid':'hhid'}, inplace=True)

# roster
roster = pd.read_csv(root_path+"/Data/Phase 1 - Roster/roster_22.csv")
roster = roster[['hhid','oldhhid','inter1_fullnam']]

# Check households in the data but not in the roster: None
merge_rost = data.merge(roster,on='hhid', how='inner')
missing_roster = data[~data.hhid.isin(merge_rost.hhid)]

# Households in the roster but not in the data:
missing_data =  roster[~roster.hhid.isin(merge_rost.hhid)]

print('Households in the roster but not on phase 3. The issue we discussed about␣
 ↪households that left to the mines and so on.')
print('The final dataset should not include these households.')
print(missing_data[['hhid','inter1_fullnam']].to_string(index=False))
```

```
## consumption prices in the village 2019
p_22 = pd.read_csv(root_path+'/Data/Phase 3/Income/prices/crop_prices_22.csv')

# Isa-lsms prices old survey (for missing prices)
p_isalsms = pd.read_stata(root_path+'/Data/Phase 3/Income/prices/
 ↪price_prioduction_kg.dta')
p_isalsms = p_isalsms.groupby(by=['crop_code']).median()


## Look at duplicates:
duplicates = pd.value_counts(data['hhid'])
print('===============================================================')
print('These households are duplicate')
print('===============================================================')
print(duplicates[duplicates>1])

### Check that no duplicates:
duplicates = pd.value_counts(data['hhid'])
data.reset_index(drop=True, inplace=True)


# Check households in the data but not in the roster
merge_rost = data.merge(roster,on='hhid', how='inner')
missing = data[~data.hhid.isin(merge_rost.hhid)]

# merge to get old hhids and be able to see panel
data = data.merge(roster[['hhid','oldhhid']], on='hhid', how='left')

percentiles = [ 0.1, .25, .5, .75, 0.9, 0.99]
list_crops = ['maize', 'groundnut', 'groundbean', 'sweetpotatoe',␣
 ↪'fingermillet', 'sorghum', 'pearlmillet', 'soyabean', 'pigeonpeas', 'cotton',␣
 ↪'nkhwani', 'cassava',  'sugarcane',  'tomatoes',  'therereokra', 'tanaposi' ]

# Rename some variables
data.rename(columns={'unitssoldpearlmillet2':'unitssoldpearlmilletout2'},␣
 ↪inplace=True)
data.rename(columns={'unitssoldsoyabean2':'unitssoldsoyabeanout2'}, inplace=True)
data.rename(columns={'soldquantitygroundbeanin':'soldquantitygroundbeanin'},␣
 ↪inplace=True)


## Remove 9999 observations=====================================
data.replace([9999, 9999.00], np.nan, inplace=True)
```

Households in the roster but not on phase 3. The issue we discussed about households that left to the mines and so on.

```
The final dataset should not include these households.
 hhid   inter1_fullnam
 1301      Marium Adam
 1102     Zione Matius
 1106       Rose World
 1004    Elube Kachere
 1407   Patuma Billiat
 1410       Ema Makiyi
 1415     Niah Shabani
 1422  Pilirani Ngunga
 1121    Wilson Khonje
 1222  Lydia Kacholora
 1228     Yusuf Thomas
 1426      Akibu Kaisi
 1134     Annie Jamali
 1034    Esnart Jamari
 1327      Mary Wisiki
 1328  William Dickson
 1142       Zione Luka
 1540   Modester James
 1237     Zainab James
 1146     Joice Mphepo
 1147    Esther Kalipo
 1450  Pemphero Wadeka
 1351     Afiki Labana
===============================================================
These households are duplicate
===============================================================
Series([], Name: hhid, dtype: int64)
```

2 Land

[2]:
```
# ==============================================================================
# Check number, size, and value plots
# ==============================================================================
data['total_plots'] = data['manyplot'].fillna(0) + data['rentinmany'].fillna(0)
sumplots = data[['manyplot','rentinmany','total_plots']].
 ↪describe(percentiles=percentiles)
N_plots = int(data[['manyplot']].max())

print('================================================================')
print('Summary number of plots')
print('================================================================')
print(sumplots)

## Check plot size and value =========================================
#units area plots
units_plot = pd.value_counts(data['unitsareaplot_1'])
# small futbol fields are around 1 acre
# square meters to acres: 0.000247105

for i in range(1,N_plots+1):
    data['area_plot_acr_'+str(i)] = data['areaplot_'+str(i)]
    data.loc[data['unitsareaplot_'+str(i)]==2.0, 'area_plot_acr_'+str(i)] = data.
 ↪loc[data['unitsareaplot_'+str(i)]==2.0, 'areaplot_'+str(i)]*2.47105
    data.loc[data['unitsareaplot_'+str(i)]==4.0, 'area_plot_acr_'+str(i)] = data.
 ↪loc[data['unitsareaplot_'+str(i)]==4.0, 'areaplot_'+str(i)]*0.000247105


for i in range(1,N_plots+1):
    data['ratio_value_rent_'+str(i)] = np.nan
    data['p_acre_plot_'+str(i)]= np.nan

#Check ratio value vs rentout:
for i in range(1,N_plots+1):
    data['ratio_value_rent_'+str(i)]  =  data['valueplot_'+str(i)] /␣
 ↪data['rentoutplot_'+str(i)]

## Check price per acre:
for i in range(1,N_plots+1):
    data['p_acre_plot_'+str(i)]  =  data['valueplot_'+str(i)] /␣
 ↪data['area_plot_acr_'+str(i)]


# working on land quality (categorical variable). TO BE DONE
'''
for i in range(1,N_plots+1):
```

```
    data[['soilplot_'+str(i)]].replace([1,2,6],['Red Soil', 'Red Sandy Soil',␣
 ↪'Other'],inplace=True)

          ,'topoplot_','tavelplot_1']]

1= Red Soil
2= Red Sandy Soil
6= Other (Specify)

1 = Hilly
2 = Flat
3 = Gentle slope
4 = Steep slope
5 = Valley
6 = Other (specify)

1= Less than 15mn
2= 15mn - 30mn
3= 30mn - 60mn
4=1hour - 2hours
6=Over 2 hours
'''


# hh aggregate variables
data['hh_area_plots'] = 0
data['hh_rentout_plots'] = 0
data['hh_value_plots'] = 0


### Add at household level:
for i in range(1,N_plots+1):
    data['hh_area_plots'] += data['area_plot_acr_'+str(i)].fillna(0)
    data['hh_rentout_plots'] += data['rentoutplot_'+str(i)].fillna(0)
    data['hh_value_plots'] += data['valueplot_'+str(i)].fillna(0)


# Implement Augustine corrections on land size and value ---------------------
'''
6.        Check land size for the following households:  Ask enumerators if from␣
 ↪notebook the number is correct and if there is a particular reason for such␣
 ↪big numbers. If from the notebooks numbers are correct and there is no reason␣
 ↪for the numbers, reask the households.
-        Hhid=1211 reported 21 acres (the max in the data). In 2019 it reported␣
 ↪7.7.
-        Hhid=1317 reported 19 acres (2nd largest). In 2019 it reported 8.5.
```

```
-       Hhid= 1002 reported 9.5 acres (quite big) and a total value of land␣
↪equal to 12,430,000.00 MWK. Can you check acres is correct? (it might be) Can␣
↪you check total value? (12 million kwachas definitely seems a wrong number.␣
↪Perhaps enumerator added an extra 0. Check).
'''
# - 1211: Augus confirmed has 21 acres.
# - 1317: total land size of 3.5 acres.
# - 1002: confirmed 9.5 acres. Value of land is 4,000,000.00 MWK.

data.loc[data['hhid']==1317, 'hh_area_plots'] = 1.5
data.loc[data['hhid']==1002, 'hh_value_plots'] = 4000000


data['hh_p_acre_plots']  =  data['hh_value_plots'] / data['hh_area_plots'].
↪replace([0, 0.0],np.nan)
data['hh_rent_per_acre']  =  data['hh_rentout_plots'] / data['hh_area_plots'].
↪replace([0, 0.0],np.nan)
data['hh_ratio_value_rent'] = data['hh_value_plots'] / data['hh_rentout_plots'].
↪replace([0, 0.0],np.nan)


print('=================================================================')
print('Check:  Size and value First Plot')
print('=================================================================')
sum_1plot2 = data[['area_plot_acr_1','rentoutplot_1','valueplot_1',␣
↪'ratio_value_rent_1', 'p_acre_plot_1']].describe(percentiles=percentiles)
print(sum_1plot2)
### STOP RUN


#print('=================================================================')
#print('Check: Distribution Second Reported Plot')  # I skip it
#print('=================================================================')
sum_2plot = data[['area_plot_acr_2','rentoutplot_2','valueplot_2',␣
↪'ratio_value_rent_2','p_acre_plot_2']].describe(percentiles=percentiles)
#print(sum_2plot)


# ============================================================================
# Check: land area, rentout value, and land value at household level
# ============================================================================
sum_hhplots =␣
↪data[['total_plots','hh_area_plots','hh_rentout_plots','hh_value_plots',␣
↪'hh_ratio_value_rent', 'hh_p_acre_plots', 'area_plot_acr_1',␣
↪'area_plot_acr_2']].describe()
print('')
```

```python
print('================================================================')
print('Check: Distribution land at household level')
print('================================================================')
print(sum_hhplots)

# ==============================================================================
#  Summarize land rights
# ==============================================================================

data[['rightsellland', 'rightbequeathplot', 'chiefpreventsell',␣
↪'chiefpreventbequeat', 'landdispute' ]] = data[['rightsellland',␣
↪'rightbequeathplot', 'chiefpreventsell', 'chiefpreventbequeat', 'landdispute'␣
↪]].replace([2.00, 2], 0.00)
sum_landrights = (data[['rightsellland', 'rightbequeathplot',␣
↪'chiefpreventsell', 'chiefpreventbequeat', 'landdispute' ]].describe()).iloc[0:
↪3,:]
print(sum_landrights)
```

```
================================================================
Summary number of plots
================================================================
       manyplot  rentinmany  total_plots
count    273.00       31.00       273.00
mean       1.59        1.19         1.72
std        0.72        0.48         0.79
min        0.00        1.00         0.00
10%        1.00        1.00         1.00
25%        1.00        1.00         1.00
50%        1.00        1.00         2.00
75%        2.00        1.00         2.00
90%        2.00        2.00         3.00
99%        3.28        2.70         4.00
max        4.00        3.00         5.00
================================================================
Check:  Size and value First Plot
================================================================
       area_plot_acr_1  rentoutplot_1  valueplot_1  ratio_value_rent_1  \
count           268.00         268.00       268.00              268.00
mean              1.61      24,166.04   336,201.49               14.05
std               1.42      16,011.63   427,310.89               13.32
min               0.25       5,000.00    40,000.00                1.00
10%               0.50      10,000.00    90,000.00                5.00
25%               1.00      15,000.00   120,000.00                6.38
50%               1.50      20,000.00   200,000.00               10.00
75%               2.00      30,000.00   400,000.00               16.67
90%               2.50      40,000.00   700,000.00               26.85
99%               7.33      93,300.00 2,665,000.00               69.42
```

```
max              15.00     130,000.00 3,000,000.00              100.00

       p_acre_plot_1
count         268.00
mean      214,262.97
std       170,895.55
min        15,000.00
10%        83,333.33
25%       100,000.00
50%       150,000.00
75%       250,000.00
90%       455,000.00
99%       800,000.00
max     1,000,000.00

==============================================================
Check: Distribution land at household level
==============================================================
       total_plots  hh_area_plots  hh_rentout_plots  hh_value_plots  \
count       273.00         273.00            273.00          273.00
mean          1.72           2.22         33,509.16      459,329.67
std           0.79           2.15         25,800.30      640,554.52
min           0.00           0.00              0.00            0.00
25%           1.00           1.00         15,000.00      150,000.00
50%           2.00           1.50         25,000.00      300,000.00
75%           2.00           3.00         43,000.00      500,000.00
max           5.00          21.00        190,000.00    5,900,000.00

       hh_ratio_value_rent  hh_p_acre_plots  area_plot_acr_1  area_plot_acr_2
count               268.00           268.00           268.00           135.00
mean                 13.84       213,635.75             1.61             1.09
std                  12.62       161,895.90             1.42             0.86
min                   1.00        15,000.00             0.25             0.00
25%                   6.66       100,000.00             1.00             0.50
50%                  10.00       156,904.76             1.50             1.00
75%                  15.79       250,000.00             2.00             1.50
max                 100.00       885,714.29            15.00             6.00
       rightsellland  rightbequeathplot  chiefpreventsell  \
count         266.00             266.00            266.00
mean            0.42               0.47              0.06
std             0.49               0.50              0.23

       chiefpreventbequeat  landdispute
count               266.00       266.00
mean                  0.03         0.13
std                   0.18         0.33
```

3 Farming Capital and Livestock

[3]:
```
#%% ================================================================================
# INPUTS: Capital and livestock
# ================================================================================

#livestock
data['hhlivestock'] =0
for i in range(1,16):
    if (i==2) or (i==5):
        continue
    data['hhlivestock'] += (data['selllivstck_'+str(i)].replace(9999,np.nan)).
 ↪fillna(0)

# Farm Equipment
data['hhfarmequip'] =0
for i in range(1,15):
    data['hhfarmequip'] += (data['sellfrmeqp_'+str(i)].replace(9999,np.nan)).
 ↪fillna(0)

# Farm Structure
data['hhfarmstruct'] =0
for i in range(1,10):
    data['hhfarmstruct'] += (data['sellfrmstrc_'+str(i)].replace(9999,np.nan)).
 ↪fillna(0)



## farming capital
data['k_farm'] = data['hhfarmequip'].fillna(0)+data['hhfarmstruct'].fillna(0)

print('================================================================')
print('Check: Farm Capital Value (in $)')
print('================================================================')
print((data[['k_farm','hhlivestock','hhfarmequip','hhfarmstruct']]/dollar_MWK).
 ↪describe())


outliers_kfarm = data.loc[(data['k_farm']>200*dollar_MWK) |␣
 ↪(data['hhlivestock']>300*dollar_MWK)]

outl1 = data.iloc[57,:]

outl2 = data.iloc[23,:]

print('I checked the two households with extreme values. The reason of the high␣
 ↪values is because they have cows.')
```

==

```
Check: Farm Capital Value (in $)
================================================================
       k_farm  hhlivestock  hhfarmequip  hhfarmstruct
count  273.00       273.00       273.00        273.00
mean    13.45        47.73         8.66          4.79
std     26.57       142.81         9.41         23.41
min      0.00         0.00         0.00          0.00
25%      2.91         0.00         2.91          0.00
50%      7.28         6.79         6.79          0.00
75%     14.07        42.22        11.16          0.00
max    284.85     2,066.16        90.74        276.60
I checked the two households with extreme values. The reason of the high values
is because they have cows.
```

4 Agricultural Production

[4]:

```
# =============================================================================
#%% Convert agricultural outputs to kgs and MWK. total quantities reported
# =============================================================================

# Import conversion rates
crop_unit = pd.read_csv("conversions/crop_conversions_kg.csv")
crop_unit.set_index('unit', inplace=True)


#Check units
tab_units = []
for crop in list_crops:
    unitscrop = pd.value_counts(data['unitstotal'+crop])
    tab_units.append(unitscrop)

tab_units = pd.DataFrame(tab_units)

print('Reported units in agricultural production ============')
print(tab_units.to_string())
print('We might want to remove some units for a next time.')
## unit 10 is other units: we have 4 cases of other units
data.loc[data['unitstotalpigeonpeas']==10,'otherunitspigeonpeas']
data.loc[data['unitstotalcotton']==10,'otherunitscotton']
data.loc[data['unitstotalcassava']==10,'otherunitscassava']

#Generate empty variables
for crop in list_crops:
    data['total_kg_'+crop] = np.nan
    data['sold_kg_'+crop] = np.nan
    data['sold_insiders_kg_'+crop] = np.nan
```

```
    data['store_kg_'+crop] = np.nan
    data['total2_kg_'+crop] =np.nan
    data['sold_bigger_total_'+crop] = 0
    data['store_bigger_total_'+crop] = 0
    data['soldstore_bigger_total_'+crop] = 0
    data['p_'+crop] = np.nan
    data['y_'+crop] = 0
    data['y_agric'] = 0
    data['sold_MWK_'+crop] = 0
    data['sold_agric'] = 0
    data['sold_insiders_MWK_'+crop] = 0
    data['sold_insiders_agric'] = 0
    data['store_MWK_'+crop] = 0
    data['store_agric'] = 0
    data[['unitstotal'+crop, 'unitssold'+crop, 'unitsstore'+crop]].replace(np.
 ↪nan, 0, inplace=True)


# ==============================================================================
# Main Loop: Conversion to kgs for all crops and questions


### change guys that reported other units and there wasnt question other units␣
 ↪appearing

data.replace(np.nan, 0, inplace=True)
for i in range(len(data)):
    for crop in list_crops:
        data.iloc[i, data.columns.get_loc('total_kg_'+crop)] = data.iloc[i,data.
 ↪columns.get_loc('totalamount'+crop)]*crop_unit.loc[int(data.iloc[i, data.
 ↪columns.get_loc('unitstotal'+crop)]),'conversionkg']
        data.iloc[i, data.columns.get_loc('sold_kg_'+crop)] = data.iloc[i,data.
 ↪columns.get_loc('soldquantity'+crop)]*crop_unit.loc[int(data.iloc[i, data.
 ↪columns.get_loc('unitssold'+crop)]),'conversionkg']
        data.iloc[i, data.columns.get_loc('sold_insiders_kg_'+crop)] = data.
 ↪iloc[i,data.columns.get_loc('soldquantity'+crop+'in')]*crop_unit.loc[int(data.
 ↪iloc[i, data.columns.get_loc('unitssold'+crop+'in')]),'conversionkg']
        data.iloc[i, data.columns.get_loc('store_kg_'+crop)] = data.iloc[i,data.
 ↪columns.get_loc('store'+crop+'quantity')]*crop_unit.loc[int(data.iloc[i, data.
 ↪columns.get_loc('unitsstore'+crop)]),'conversionkg']

for crop in list_crops:
    data['total2_kg_'+crop] = data['sold_kg_'+crop].fillna(0)␣
 ↪+data['store_kg_'+crop].fillna(0)
```

```
#Summary total output kg:
pd.options.display.float_format = '{:,.0f}'.format
sum_kg = (data[['total_kg_maize', 'total_kg_groundnut', 'total_kg_groundbean',␣
 ↪'total_kg_sweetpotatoe', 'total_kg_fingermillet', 'total_kg_sorghum',␣
 ↪'total_kg_pearlmillet', 'total_kg_soyabean', 'total_kg_pigeonpeas',␣
 ↪'total_kg_cotton', 'total_kg_nkhwani', 'total_kg_cassava', ␣
 ↪'total_kg_sugarcane',  'total_kg_tomatoes',  'total_kg_therereokra',␣
 ↪'total_kg_tanaposi']].replace(0,np.nan)).describe(percentiles=percentiles)


## NON-PRODUCED CROPS
# pearl millet

# tomatoes: 1 hh, therereokra: 1 hh, tanaposi: 2 hh. Strange?
print('================================================================')
print('Crop production: Number of households harvested crops')
print('================================================================')
print((data[['total_kg_maize', 'total_kg_groundnut', 'total_kg_groundbean',␣
 ↪'total_kg_sweetpotatoe', 'total_kg_fingermillet', 'total_kg_sorghum',␣
 ↪'total_kg_pearlmillet', 'total_kg_soyabean', 'total_kg_pigeonpeas',␣
 ↪'total_kg_cotton', 'total_kg_nkhwani', 'total_kg_cassava', ␣
 ↪'total_kg_sugarcane',  'total_kg_tomatoes',  'total_kg_therereokra',␣
 ↪'total_kg_tanaposi']].replace(0,np.nan)).count())


print('================================================================')
print(' Distribution of crop production (in kg)')
print('================================================================')
sum_kg = sum_kg.dropna(axis=1, how='any')
N_prodcrops = sum_kg.iloc[0,:]
T_prodcrops = sum_kg.iloc[0,:]*sum_kg.iloc[1,:]
T_prod = T_prodcrops.sum()
print(sum_kg)
## STOP RUN

# who are the top maize and cassava producers?
big_kg = data.loc[(data['total_kg_maize']>2000) |␣
 ↪(data['total_kg_cassava']>1000),[ 'hhid','oldhhid','total_kg_maize',␣
 ↪'total_kg_cassava','hh_area_plots']]

print('biggest agricultural producers. Both households have relatively large␣
 ↪land sizes (5 acres and 7 acres). 3,500 kg maize in 7 acres is not necessary a␣
 ↪lot, so they are not outliers to be removed.')
print(big_kg)

data19_maize = data19[['hhid','total_kg_maize', 'land_area']]
data19_maize.columns = ['oldhhid','total_kg_maize19', 'land_area']
```

```
data19_maize['maizeyield19'] = data19_maize['total_kg_maize19']/
 ↪data19_maize['land_area']

data_maize = data[['hhid','oldhhid','total_kg_maize','hh_area_plots']]

data_maize['maizeyield'] = data['total_kg_maize']/data['hh_area_plots']


panel_maize = data_maize.merge(data19_maize, how='inner', on='oldhhid')
panel_maize['maize_diff'] = panel_maize['total_kg_maize'] -␣
 ↪panel_maize['total_kg_maize19']
panel_maize['maizeyield_diff'] =  panel_maize['maizeyield'] -␣
 ↪panel_maize['maizeyield19']


check_bigdrops = panel_maize.nsmallest(n=5, columns=['maize_diff'])

#print(check_bigdrops[['hhid','oldhhid', 'maize_diff',␣
 ↪'maizeyield_diff','total_kg_maize','hh_area_plots']])

check_bigdrops2 = panel_maize.nsmallest(n=8, columns=['maizeyield_diff'])

#print(check_bigdrops2[['hhid','oldhhid', 'maize_diff',␣
 ↪'maizeyield_diff','total_kg_maize','hh_area_plots']])

check_bigdrops2.hhid


# Summary total sellings kg:
sum_sold_kg= (data[['sold_kg_maize', 'sold_kg_groundnut', 'sold_kg_groundbean',␣
 ↪'sold_kg_sweetpotatoe', 'sold_kg_fingermillet', 'sold_kg_sorghum',␣
 ↪'sold_kg_pearlmillet', 'sold_kg_soyabean', 'sold_kg_pigeonpeas',␣
 ↪'sold_kg_cotton', 'sold_kg_nkhwani', 'sold_kg_cassava',  'sold_kg_sugarcane', ␣
 ↪'sold_kg_tomatoes',  'sold_kg_therereokra', 'sold_kg_tanaposi']].replace(0,np.
 ↪nan)).describe()
print('==============================================================')
print('Check: Distribution of crop Sellings (in kg)')
print('==============================================================')
sum_sold_kg.dropna(axis=1, how='any', inplace=True)
N_sellcrops = sum_sold_kg.iloc[0,:]
T_sellcrops = sum_sold_kg.iloc[0,:]*sum_sold_kg.iloc[1,:]
T_sell = T_sellcrops.sum()
print(sum_sold_kg)
## STOP RUN
```

```
#Summary sellings inside kg:
sum_sold_kg_inside = (data[['sold_insiders_kg_maize',␣
↪'sold_insiders_kg_groundnut', 'sold_insiders_kg_groundbean',␣
↪'sold_insiders_kg_sweetpotatoe', 'sold_insiders_kg_fingermillet',␣
↪'sold_insiders_kg_sorghum', 'sold_insiders_kg_pearlmillet',␣
↪'sold_insiders_kg_soyabean', 'sold_insiders_kg_pigeonpeas',␣
↪'sold_insiders_kg_cotton', 'sold_insiders_kg_nkhwani',␣
↪'sold_insiders_kg_cassava',  'sold_insiders_kg_sugarcane', ␣
↪'sold_insiders_kg_tomatoes',  'sold_insiders_kg_therereokra',␣
↪'sold_insiders_kg_tanaposi']].replace(0,np.nan)).
↪describe(percentiles=percentiles)
print('================================================================')
print('Check: Distribution of crop Sellings to Villagers')
print('================================================================')
sum_sold_kg_inside.dropna(axis=1, how='any', inplace=True)

print(sum_sold_kg_inside)
## STOP RUN


list_cropssell =␣
↪['maize','groundnut','soyabean','pigeonpeas','cotton','cassava','sugarcane','tanaposi']



share_didsell = []
share_sell = []
share_sell.append(T_sell/T_prod)
for crop in list_cropssell:
    share_didsell.append(N_sellcrops['sold_kg_'+crop]/
↪N_prodcrops['total_kg_'+crop])
    share_sell.append(T_sellcrops['sold_kg_'+crop]/T_prodcrops['total_kg_'+crop])

print('Share of sellings across crops')  # dont show it for all the crops. Just␣
↪for the crops that had a decent number of sellers.
print(list_crops)
print(share_didsell)
print(share_sell)
# Sum transportation costs
sum_transport_c = (data[['transcostmaizeout', 'transcostgroundnutout',␣
↪'transcostgroundbeanout', 'transcostsweetpotatoeout',␣
↪'transcostfingermilletout', 'transcostsorghumout', 'transcostpearlmilletout',␣
↪'transcostsoyabeanout', 'transcostpigeonpeasout', 'transcostcottonout',␣
↪'transcostnkhwaniout', 'transcostcassavaout',  'transcostsugarcaneout', ␣
↪'transcosttomatoesout',  'transcostthererekraout', 'transcosttanaposiout']].
↪replace(0,np.nan)).describe(percentiles=percentiles)
```

```
# Summary Store kg:
sum_store_kg= (data[['store_kg_maize', 'store_kg_groundnut',␣
 ↪'store_kg_groundbean', 'store_kg_sweetpotatoe', 'store_kg_fingermillet',␣
 ↪'store_kg_sorghum', 'store_kg_pearlmillet', 'store_kg_soyabean',␣
 ↪'store_kg_pigeonpeas', 'store_kg_cotton', 'store_kg_nkhwani',␣
 ↪'store_kg_cassava',  'store_kg_sugarcane',  'store_kg_tomatoes', ␣
 ↪'store_kg_therereokra', 'store_kg_tanaposi']].replace(0,np.nan)).
 ↪describe(percentiles=percentiles)
print('================================================================')
print('Check: Distribution of crop store (in kg)')
print('================================================================')
sum_store_kg.dropna(axis=1, how='any')
## STOP RUN


# ==============================================================================
# Check quantity sold, store, not larger than total
# ==============================================================================
for crop in list_crops:
    data['sold_bigger_total_'+crop] = 1*(data['sold_kg_'+crop].fillna(0)>␣
 ↪data['total_kg_'+crop].fillna(0)+5)
    data['store_bigger_total_'+crop] = 1*(data['store_kg_'+crop].fillna(0)>␣
 ↪data['total_kg_'+crop].fillna(0)+5)

check_sold_bigger_total = data[['sold_bigger_total_maize',␣
 ↪'sold_bigger_total_groundnut', 'sold_bigger_total_groundbean',␣
 ↪'sold_bigger_total_sweetpotatoe', 'sold_bigger_total_fingermillet',␣
 ↪'sold_bigger_total_sorghum', 'sold_bigger_total_pearlmillet',␣
 ↪'sold_bigger_total_soyabean', 'sold_bigger_total_pigeonpeas',␣
 ↪'sold_bigger_total_cotton', 'sold_bigger_total_nkhwani',␣
 ↪'sold_bigger_total_cassava',  'sold_bigger_total_sugarcane', ␣
 ↪'sold_bigger_total_tomatoes',  'sold_bigger_total_therereokra',␣
 ↪'sold_bigger_total_tanaposi']]


#Get the households that reported larger amounts than total:
list_hh_check_sell = []
list_hh_check_lost = []
list_hh_check_store = []
list_hh_check = []

for crop in list_crops:
    liers_sell = data.loc[data['sold_bigger_total_'+crop]==1, 'hhid'] #␣
 ↪'intervieweename']
    liers_store = data.loc[data['store_bigger_total_'+crop]==1, 'hhid'] #
    liers = data.loc[data['soldstore_bigger_total_'+crop]==1, 'hhid'] #
```

```
    list_hh_check_sell.append(liers_sell)
    list_hh_check_store.append(liers_store)
    list_hh_check.append(liers)

# sellings check:
hh_to_check_sell = pd.concat(list_hh_check_sell, axis=1)
hh_to_check_sell.columns = list_crops
print('')
print('================================================================')
print('Check: Households-crop combination where SELLINGS larger than total␣
 ↪produced')
print('================================================================')
print(hh_to_check_sell.dropna(axis=1,  how='all'))
###STOP RUN

print('2 cases where sellings higher than total: Replace total by quantitiy sold␣
 ↪(if necessary)')

data.
 ↪loc[data['total_kg_soyabean']<data['sold_kg_soyabean'],['total_kg_soyabean']]␣
 ↪= data.
 ↪loc[data['total_kg_soyabean']<data['sold_kg_soyabean'],['sold_kg_soyabean'] ]
data.
 ↪loc[data['total_kg_tomatoes']<data['sold_kg_tomatoes'],['total_kg_tomatoes']]␣
 ↪= data.
 ↪loc[data['total_kg_tomatoes']<data['sold_kg_tomatoes'],['sold_kg_tomatoes'] ]


# Store quantity check:
hh_to_check_store = pd.concat(list_hh_check_store, axis=1)
hh_to_check_store.columns = list_crops
print('')
print('================================================================')
print('Check: Households-crop combination where STORED larger than total␣
 ↪produced')
print('================================================================')
print(hh_to_check_store.dropna(axis=1, how='all'))
### STOP RUN





hh_to_check = pd.concat(list_hh_check, axis=1)
hh_to_check.columns = list_crops
print('')
```

```
print('==================================================================')
print('Check: Households-crop combination where SELL+STORED larger than total␣
↪produced')
print('==================================================================')
hh_to_check.dropna(axis=1).to_string()

### Check each household that reported some amount bigger. look at values, units␣
↪and enumerator.
# Write a note per each household and sent them to the enumerators.
#data_elia = data.loc[data['hhid']==93,]
#data_sell_outliers = data.loc[(data['hhid']==93) | (data['hhid']==56) |␣
↪(data['hhid']==31) | (data['hhid']==89),]
#data_store_outliers = data.loc[(data['hhid']==62) | (data['hhid']==13) |␣
↪(data['hhid']==161) | (data['hhid']==260) | (data['hhid']==21) |␣
↪(data['hhid']==56) | (data['hhid']==250),]



pd.options.display.float_format = '{:,.2f}'.format
# ==============================================================================
#  get selling PRICES per kg
# ==============================================================================
for crop in list_crops:
    data['p_'+crop] = (data['soldvalue'+crop].replace(0,np.nan)).dropna()  /␣
↪(data['sold_kg_'+crop].replace(0,np.nan)).dropna()
    #DF = data[['soldvalue'+crop, 'sold_kg_'+crop]].dropna()
sum_prices = data[['p_maize', 'p_groundnut', 'p_groundbean', 'p_sweetpotatoe',␣
↪'p_fingermillet', 'p_sorghum', 'p_pearlmillet', 'p_soyabean', 'p_pigeonpeas',␣
↪'p_cotton', 'p_nkhwani', 'p_cassava',  'p_sugarcane',  'p_tomatoes', ␣
↪'p_therereokra', 'p_tanaposi']].describe()
print('')
print('==================================================================')
print('Check: Distribution of prices')
print('==================================================================')
print(sum_prices.dropna(axis=1))

list_crops_price =␣
↪['maize','groundnut','sweetpotatoe','pigeonpeas','cotton','tomatoes']
price_data = pd.DataFrame(list_crops_price, columns=['crop'])
price_data['p_sell'] = np.nan

for item in list_crops_price:
    price_data.loc[price_data['crop']==item,'p_sell'] = np.
↪nanmedian(data['p_'+item])

if save==True:
```

```
    price_data.to_csv('prices/village_selling_prices_22.csv', index=False)

### UPLOAD SET OF SELLING AND CONSUMPTION PRICES:
prices = p_22

### For the missing prices I use the ones from ISA-LSMS 2017. I use the maize␣
↪price as the reference for the conversion from Malawi 17 to village 19.

print('WE NEED AN UPDATED ISA-LSMS TO USE PRICES AND KILOGRAMS CONVERSIONS. FOR␣
↪THE MOMENT I USE 2017 WAVE WITH THE MAIZE REFERENCE IN THE VILLAGE')
print('This is only for the few crops we do not have consupmtion price')
maize_isavillage = 297/57
prices.loc[prices['crop']=='groundbean','p_c'] = 147.6*maize_isavillage
prices.loc[prices['crop']=='fingermillet','p_c'] = 290*maize_isavillage
prices.loc[prices['crop']=='sorghum','p_c'] = 63.49*maize_isavillage
prices.loc[prices['crop']=='pearlmillet','p_c'] = 95*maize_isavillage
prices.loc[prices['crop']=='soyabean','p_c'] = 79.36*maize_isavillage
## tanaposi and okra I couldnt find a price in ISA-LSMS (neither on cons nor␣
↪prod). I assume pigeon peas are a similar product and assign that price.
prices.loc[prices['crop']=='therereokra','p_c'] = prices.
↪loc[prices['crop']=='pigeonpeas','p_c']
#no price for there okra in village or ISA-LSMS. Also not in internet. I use␣
↪price of pigeon peas since it seems to be a similar crop.
prices['p_c'].fillna(prices['p_sell'], inplace=True)




#Get monetary value:

# Using consumption prices. To use selling ones replace p_c for p_sell.
for crop in list_crops:
    data['y_'+crop] =  float(prices.loc[prices['crop']==crop,␣
↪'p_c'])*data['total_kg_'+crop].fillna(0)
    data['sold_MWK_'+crop] = float(prices.loc[prices['crop']==crop,␣
↪'p_sell'])*data['sold_kg_'+crop].fillna(0)
    data['sold2_MWK_'+crop] =  data['soldvalue'+crop]
    data['sold_insiders_MWK_'+crop] = data['sold_insiders_kg_'+crop]
    data['store_MWK_'+crop] = float(prices.loc[prices['crop']==crop,␣
↪'p_c'])*data['store_kg_'+crop].fillna(0)
   ### without loss production there is not an easy way to value all the␣
↪production accounting for the sold production
    data['y_agric'] += data['y_'+crop].fillna(0)
    data['sold_agric'] += data['sold_MWK_'+crop].fillna(0)
    data['sold_insiders_agric'] += data['sold_insiders_MWK_'+crop].fillna(0)
    data['store_agric'] += data['store_MWK_'+crop].fillna(0)
    data['y_'+crop].replace(0,np.nan,inplace=True)
```

```
    data['sold_MWK_'+crop].replace(0,np.nan,inplace=True)
    data['store_MWK_'+crop].replace(0,np.nan,inplace=True)


data[['y_agric','sold_agric','sold_insiders_agric','store_agric']] =␣
 ↪data[['y_agric','sold_agric','sold_insiders_agric','store_agric']].replace(0.
 ↪0,np.nan)



sum_y = (data[['y_agric','y_maize', 'y_groundnut', 'y_pigeonpeas']]/dollar_MWK).
 ↪describe(percentiles=percentiles)

print('')
print('===============================================================')
print('Agricultural Output (rainy season) in $')
print('(===============================================================')
print(sum_y)
print('Agricultural Output (rainy season) in Kgs')
print('(===============================================================')
sum_ykg = data[['total_kg_maize', 'total_kg_groundnut','total_kg_pigeonpeas']].
 ↪describe()

print(sum_ykg)
```

```
Reported units in agricultural production ============
                        1.00   2.00   3.00   4.00   5.00   8.00   9.00  10.00
11.00  12.00
unitstotalmaize         5.00   4.00 228.00    nan   7.00    nan   3.00    nan
nan   6.00
unitstotalgroundnut    12.00  35.00   4.00   1.00    nan   6.00  36.00    nan
3.00    nan
unitstotalgroundbean    1.00    nan   2.00    nan    nan   1.00   6.00    nan
1.00   1.00
unitstotalsweetpotatoe   nan   1.00   1.00    nan    nan    nan   3.00    nan
nan    nan
unitstotalfingermillet   nan    nan    nan    nan    nan    nan   1.00    nan
nan    nan
unitstotalsorghum       3.00    nan   5.00    nan    nan   1.00   5.00    nan
nan   2.00
unitstotalpearlmillet    nan    nan    nan    nan    nan    nan    nan    nan
nan    nan
unitstotalsoyabean      1.00    nan    nan    nan    nan    nan   2.00    nan
nan   1.00
unitstotalpigeonpeas   31.00   3.00  22.00    nan    nan   7.00  78.00   2.00
nan   6.00
unitstotalcotton        2.00    nan    nan    nan    nan   1.00    nan   1.00
```

```
nan    nan
unitstotalnkhwani        1.00   1.00    nan    nan    nan   3.00   2.00    nan
nan   1.00
unitstotalcassava         nan   3.00   9.00    nan    nan   1.00   1.00   1.00
nan    nan
unitstotalsugarcane      2.00    nan    nan    nan    nan    nan    nan    nan
nan    nan
unitstotaltomatoes       1.00    nan    nan    nan    nan    nan    nan    nan
nan    nan
unitstotalthererεokra     nan    nan    nan    nan    nan    nan   1.00    nan
nan    nan
unitstotaltanaposi       1.00    nan    nan    nan    nan    nan   1.00   1.00
nan    nan
We might want to remove some units for a next time.
================================================================
Crop production: Number of households harvested crops
================================================================
total_kg_maize          253
total_kg_groundnut       97
total_kg_groundbean      12
total_kg_sweetpotatoe     5
total_kg_fingermillet     1
total_kg_sorghum         16
total_kg_pearlmillet      0
total_kg_soyabean         4
total_kg_pigeonpeas     147
total_kg_cotton           3
total_kg_nkhwani          8
total_kg_cassava         14
total_kg_sugarcane        2
total_kg_tomatoes         1
total_kg_thererεokra      1
total_kg_tanaposi         2
dtype: int64
================================================================
 Distribution of crop production (in kg)
================================================================
       total_kg_maize  total_kg_groundnut  total_kg_groundbean  \
count             253                  97                   12
mean              249                  40                   18
std               313                  56                   20
min                 5                   1                    1
10%                50                   5                    5
25%               100                   5                    5
50%               150                  20                   10
75%               300                  50                   25
90%               500                 100                   25
99%             1,298                 302                   70
```

```
max             3,500                 350                 75

       total_kg_sweetpotatoe  total_kg_sorghum  total_kg_soyabean  \
count                      5                16                  4
mean                      25                42                 14
std                       23                59                  8
min                        5                 5                  5
10%                        7                 5                  7
25%                       10                18                 10
50%                       10                22                 14
75%                       50                50                 18
90%                       50                62                 22
99%                       50               224                 25
max                       50               250                 25

       total_kg_pigeonpeas  total_kg_cotton  total_kg_nkhwani  \
count                  147                3                 8
mean                    18              120                28
std                     19               52                25
min                      2               60                 3
10%                      5               78                 4
25%                      5              105                16
50%                     10              150                20
75%                     22              150                31
90%                     50              150                59
99%                     75              150                78
max                    100              150                80

       total_kg_cassava  total_kg_sugarcane  total_kg_tanaposi
count                14                   2                  2
mean                190                 585                 18
std                 387                 587                 18
min                  10                 170                  5
10%                  22                 253                  8
25%                  50                 378                 11
50%                  50                 585                 18
75%                 138                 792                 24
90%                 290                 917                 28
99%               1,350                 992                 30
max               1,500               1,000                 30
biggest agricultural producers. Both households have relatively large land sizes
(5 acres and 7 acres). 3,500 kg maize in 7 acres is not necessary a lot, so they
are not outliers to be removed.
     hhid  oldhhid  total_kg_maize  total_kg_cassava  hh_area_plots
46   1050      250             450             1,500              5
203  1430      118           3,500                 0              7
==============================================================
Check: Distribution of crop Sellings (in kg)
```

```
================================================================
       sold_kg_maize  sold_kg_groundnut  sold_kg_soyabean  sold_kg_pigeonpeas  \
count             29                  6                 3                  38
mean              86                 79                26                  19
std              124                 71                30                  20
min                4                 10                 5                   1
25%               20                 24                 8                   5
50%               50                 75                12                  10
75%              100                100                36                  25
max              600                200                60                  75

       sold_kg_cotton  sold_kg_cassava  sold_kg_sugarcane  sold_kg_tanaposi
count               3                3                  2                 2
mean              120              520                585                18
std                52              849                587                18
min                60               10                170                 5
25%               105               30                378                11
50%               150               50                585                18
75%               150              775                792                24
max               150            1,500              1,000                30
================================================================
Check: Distribution of crop Sellings to Villagers
================================================================
       sold_insiders_kg_maize  sold_insiders_kg_groundnut  \
count                      16                           2
mean                       41                         108
std                        34                         131
min                        10                          15
10%                        10                          34
25%                        19                          61
50%                        25                         108
75%                        50                         154
90%                        88                         182
99%                       121                         198
max                       125                         200

       sold_insiders_kg_pigeonpeas  sold_insiders_kg_cassava  \
count                           18                         2
mean                            10                       755
std                             11                     1,054
min                              1                        10
10%                              4                       159
25%                              5                       382
50%                              5                       755
75%                             10                     1,128
90%                             22                     1,351
99%                             42                     1,485
max                             45                     1,500
```

```
       sold_insiders_kg_sugarcane
count                           2
mean                          585
std                           587
min                           170
10%                           253
25%                           378
50%                           585
75%                           792
90%                           917
99%                           992
max                         1,000
Share of sellings across crops
['maize', 'groundnut', 'groundbean', 'sweetpotatoe', 'fingermillet', 'sorghum',
'pearlmillet', 'soyabean', 'pigeonpeas', 'cotton', 'nkhwani', 'cassava',
'sugarcane', 'tomatoes', 'therereokra', 'tanaposi']
[0.11462450592885376, 0.061855670103092786, 0.75, 0.2585034013605442, 1.0,
0.21428571428571427, 1.0, 1.0]
[0.09215947729848657, 0.039701326554928904, 0.12251741036884191,
1.3508771929824561, 0.2757780277465317, 1.0, 0.5875706214689266, 1.0, 1.0]
================================================================
Check: Distribution of crop store (in kg)
================================================================

================================================================
Check: Households-crop combination where SELLINGS larger than total produced
================================================================
     soyabean  tomatoes
176       nan     1,348
203     1,430       nan
2 cases where sellings higher than total: Replace total by quantitiy sold (if
necessary)

================================================================
Check: Households-crop combination where STORED larger than total produced
================================================================
Empty DataFrame
Columns: []
Index: []

================================================================
Check: Households-crop combination where SELL+STORED larger than total produced
================================================================

================================================================
Check: Distribution of prices
================================================================
```

	p_maize	p_groundnut	p_soyabean	p_pigeonpeas	p_cotton	p_cassava \
count	29.00	6.00	3.00	38.00	3.00	3.00
mean	232.49	286.67	866.67	356.17	538.89	345.78
std	121.44	110.03	317.98	144.56	41.94	312.84
min	70.00	100.00	566.67	100.00	500.00	107.33
25%	160.00	240.00	700.00	300.00	516.67	168.67
50%	200.00	320.00	833.33	330.00	533.33	230.00
75%	280.00	355.00	1,016.67	400.00	558.33	465.00
max	700.00	400.00	1,200.00	1,000.00	583.33	700.00

	p_sugarcane	p_tanaposi
count	2.00	2.00
mean	98.82	1,133.33
std	26.62	659.97
min	80.00	666.67
25%	89.41	900.00
50%	98.82	1,133.33
75%	108.24	1,366.67
max	117.65	1,600.00

WE NEED AN UPDATED ISA-LSMS TO USE PRICES AND KILOGRAMS CONVERSIONS. FOR THE MOMENT I USE 2017 WAVE WITH THE MAIZE REFERENCE IN THE VILLAGE
This is only for the few crops we do not have consupmtion price

==
Agricultural Output (rainy season) in $
(==

	y_agric	y_maize	y_groundnut	y_pigeonpeas
count	254.00	253.00	97.00	147.00
mean	96.35	71.69	38.79	8.80
std	110.01	90.29	54.33	9.10
min	2.43	1.44	0.97	0.97
10%	16.83	14.41	4.85	2.23
25%	37.36	28.82	4.85	2.43
50%	66.46	43.22	19.41	4.85
75%	114.70	86.45	48.53	10.92
90%	181.24	144.08	97.05	24.26
99%	484.47	374.04	293.10	36.40
max	1,087.89	1,008.57	339.69	48.53

Agricultural Output (rainy season) in Kgs
(==

	total_kg_maize	total_kg_groundnut	total_kg_pigeonpeas
count	273.00	273.00	273.00
mean	230.57	14.20	9.77
std	308.49	38.38	16.46
min	0.00	0.00	0.00
25%	75.00	0.00	0.00
50%	150.00	0.00	3.00
75%	275.00	10.00	10.00

```
max            3,500.00                 350.00                 100.00
```

5 Agricultural inputs and Production (at Plot Level)

```
[5]: # =============================================================================
#  AGRICULTURAL OUTPUT AND INPUTS AT PLOT LEVEL. GENERATE A PLOT LEVEL DATASET.
# COMPUTATION OF AGRIC LABOR
# COMPARISON WITH TOTAL REPORTED VS SUM ACCROSS PLOTS.
# =============================================================================

data = data.stack().apply(pd.to_numeric, errors='ignore').fillna(0).unstack()




# Generate empty dataset
N_p= np.sum(data['total_plots'])
ones = np.ones((int(N_p),2))
data_plots = pd.DataFrame({'hhid':ones[:,0], 'plotid':ones[:,1]})

## Populate dataset with hhid and plotid
i=-1
for hhid in data['hhid']:
    for plot in range(1,int(data.loc[data['hhid']==hhid, 'manyplot'])+1):
        i+=1
        data_plots.iloc[i,0] = hhid
        data_plots.iloc[i,1] = plot


## generate variables:
# List of chosen crops.. If not chosen then the variables associated to␣
 ↪not-chosen crop are unexistent. Update this list
# Everytime we get new data. Check sum_kg for a quick selection.
#list_crops_selected = ['maize', 'groundnut', 'sorghum',  'pigeonpeas']
# Code also works with all the crops. This is just to avoid empty columns.

for crop in list_crops:
    data_plots[crop+'_kg'] = np.nan


data_plots['area_cultivated'] = np.nan  #area is already converted in acres
data_plots['rentoutplot'] = np.nan
data_plots['valueplot'] = np.nan
data_plots['kg_fertilizer'] = np.nan

#### Loop for plot characteristics
i=-1
```

```
for hhid in data['hhid']:
    for plot in range(1,int(data.loc[data['hhid']==hhid, 'manyplot'])+1):
        i+=1
        data_plots.iloc[i, data_plots.columns.get_loc('area_cultivated')] =␣
 ↪float(data.loc[data['hhid']==hhid, 'area_plot_acr_'+str(plot)])
        ## problem: area of rented-in plots. In this case the one with area=0
        data_plots.iloc[i, data_plots.columns.get_loc('rentoutplot')] =␣
 ↪float(data.loc[data['hhid']==hhid, 'rentoutplot_'+str(plot)])
        data_plots.iloc[i, data_plots.columns.get_loc('valueplot')] = float(data.
 ↪loc[data['hhid']==hhid, 'valueplot_'+str(plot)])

#### Looop for fertilizer
i=-1
for hhid in data['hhid']:
    for plot in range(1,int(data.loc[data['hhid']==hhid,␣
 ↪'repeatplotsfertilizer_count'])+1):
        i+=1
        data_plots.loc[(data_plots['hhid']==hhid) & (data_plots['plotid']==␣
 ↪int(float(data.
 ↪loc[data['hhid']==hhid,'fertilizerplotsselected_'+str(plot)]))),'kg_fertilizer']␣
 ↪= float(data.loc[data['hhid']==hhid, 'plotkgfertilizer_'+str(plot)])


#### Loop for crop production
for hhid in data['hhid']:
        for crop in list_crops:
            for plot in range(1,int(data.loc[data['hhid']==hhid,␣
 ↪'repeatplots'+crop+'_count'])+1):
                #print(data.
 ↪loc[data['hhid']==hhid,crop+'perplot_'+str(plot)]*crop_unit.loc[int(data.
 ↪loc[data['hhid']==hhid, 'unitsplot'+crop+'_'+str(plot)]),'conversionkg'])
                data_plots.loc[(data_plots['hhid']==hhid) &␣
 ↪(data_plots['plotid']== int(float(data.
 ↪loc[data['hhid']==hhid,crop+'plotsselected_'+str(plot)]))), crop+'_kg'] =␣
 ↪float(data.loc[data['hhid']==hhid,crop+'perplot_'+str(plot)]*crop_unit.
 ↪loc[int(data.loc[data['hhid']==hhid,␣
 ↪'unitsplot'+crop+'_'+str(plot)]),'conversionkg'])
                #data_plots.loc[(data_plots['hhid']==hhid) &␣
 ↪(data_plots['plotid']== int(float(data.
 ↪loc[data['hhid']==hhid,crop+'plotsselected_'+str(plot)]))), crop+'_kg'] = i


#### Loop for labor input
for member in range(1,int(np.max(data['manyhhlaborplot'])+1)):
    data_plots['months_member_'+str(member)] = np.nan
    data_plots['weeks_member_'+str(member)] = np.nan
```

```
    data_plots['days_member_'+str(member)] = np.nan
    data_plots['hours_member_'+str(member)] = np.nan
    data_plots['hours_member_'+str(member)] = np.nan
for hhid in data['hhid']:
    for member in range(1,int(data.loc[data['hhid']==hhid,␣
 ↪'manyhhlaborplot'])+1):
        for plot in range(1,int(data.loc[data['hhid']==hhid,␣
 ↪'hhlaborperplotrepeat_count_'+str(member)])+1):

            #print(data.
 ↪loc[data['hhid']==hhid,crop+'perplot_'+str(plot)]*crop_unit.loc[int(data.
 ↪loc[data['hhid']==hhid, 'unitsplot'+crop+'_'+str(plot)]),'conversionkg'])
                data_plots.loc[(data_plots['hhid']==hhid) &␣
 ↪(data_plots['plotid']== int(float(data.
 ↪loc[data['hhid']==hhid,'hhlaborplotsselected_'+str(member)+'_'+str(plot)]))),␣
 ↪'months_member_'+str(member)] = float(data.
 ↪loc[data['hhid']==hhid,'monthshhplot_'+str(member)+'_'+str(plot)])
                data_plots.loc[(data_plots['hhid']==hhid) &␣
 ↪(data_plots['plotid']== int(float(data.
 ↪loc[data['hhid']==hhid,'hhlaborplotsselected_'+str(member)+'_'+str(plot)]))),␣
 ↪'weeks_member_'+str(member)] = float(data.
 ↪loc[data['hhid']==hhid,'weekshhplot_'+str(member)+'_'+str(plot)])
                data_plots.loc[(data_plots['hhid']==hhid) &␣
 ↪(data_plots['plotid']== int(float(data.
 ↪loc[data['hhid']==hhid,'hhlaborplotsselected_'+str(member)+'_'+str(plot)]))),␣
 ↪'days_member_'+str(member)] = float(data.
 ↪loc[data['hhid']==hhid,'dayshhplot_'+str(member)+'_'+str(plot)])
                data_plots.loc[(data_plots['hhid']==hhid) &␣
 ↪(data_plots['plotid']== int(float(data.
 ↪loc[data['hhid']==hhid,'hhlaborplotsselected_'+str(member)+'_'+str(plot)]))),␣
 ↪'hours_member_'+str(member)] = float(data.
 ↪loc[data['hhid']==hhid,'hourshhplot_'+str(member)+'_'+str(plot)])

sum_member1 = data_plots[['months_member_1', 'weeks_member_1', 'days_member_1',␣
 ↪'hours_member_1']].describe(percentiles=percentiles)
print('===============================================================')
print('Agriculture hh labor member 1')
print('===============================================================')
print(sum_member1)
### STOP RUN

sum_member2 = data_plots[['months_member_2', 'weeks_member_2', 'days_member_2',␣
 ↪'hours_member_2']].describe(percentiles=percentiles)
print('===============================================================')
print('Agriculture hh labor member 2')
print('===============================================================')
```

```
print(sum_member2)

sum_member3 = data_plots[['months_member_3', 'weeks_member_3', 'days_member_3',␣
 ↪'hours_member_3']].describe(percentiles=percentiles)
print('================================================================')
print('Agriculture hh labor member 3')
print('================================================================')
print(sum_member3)


data_plots['hh_labor_days'] = 0
data_plots['hh_labor_hours'] = 0

for member in range(1,int(np.max(data['manyhhlaborplot'])+1)):
    data_plots['member_'+str(member)+'_labor_days'] = ␣
 ↪(data_plots['months_member_'+str(member)].
 ↪multiply(data_plots['weeks_member_'+str(member)],axis=0, fill_value=0)).
 ↪multiply(data_plots['days_member_'+str(member)],axis=0, fill_value=0)
    data_plots['member_'+str(member)+'_labor_hours'] =␣
 ↪data_plots['member_'+str(member)+'_labor_days'].
 ↪multiply(data_plots['hours_member_'+str(member)],axis=0, fill_value=0)

for member in range(1,int(np.max(data['manyhhlaborplot'])+1)):

    data_plots['hh_labor_days'] +=␣
 ↪data_plots['member_'+str(member)+'_labor_days'].fillna(0)
    data_plots['hh_labor_hours'] +=␣
 ↪data_plots['member_'+str(member)+'_labor_hours'].fillna(0)


print('================================================================')
print('Distribution Agric Household Labor in days')
print('================================================================')
sum_labor_days = data_plots[['hh_labor_days', 'member_1_labor_days',␣
 ↪'member_2_labor_days', 'member_3_labor_days']].
 ↪describe(percentiles=percentiles)
print(sum_labor_days)


### check the households that reported more labor



print('================================================================')
print('Distribution Agric Household Labor in hours')
print('================================================================')
```

```
sum_labor_hours = data_plots[['hh_labor_hours', 'member_1_labor_hours',␣
 ↪'member_2_labor_hours', 'member_3_labor_hours']].
 ↪describe(percentiles=percentiles)
print(sum_labor_hours)




print('IN FILE data_plotlevel.csv there is hhid-plotid long format dataset.␣
 ↪CHECK IT TO SEE IF VALUES MAKE SENSE. ESPECIALLY FERTILIZER!')
#if save==True:
    #data_plots.to_csv('data_plotlevel.csv')


## Systematic check no plot has input investment (fertilizer) with 0 output␣
 ↪================

## Compare reported aggregate quantity vs summing across plots:
data_plots_agg = data_plots.groupby(by='hhid').sum()
data_plots_agg.reset_index(inplace=True)

data_plots_agg = data_plots_agg[['hhid','maize_kg', 'groundnut_kg',␣
 ↪'groundbean_kg',
       'sweetpotatoe_kg', 'fingermillet_kg', 'sorghum_kg', 'pearlmillet_kg',
       'soyabean_kg', 'pigeonpeas_kg', 'cotton_kg', 'nkhwani_kg', 'cassava_kg',
       'sugarcane_kg', 'tomatoes_kg', 'therereokra_kg', 'tanaposi_kg',␣
 ↪'area_cultivated', 'kg_fertilizer','hh_labor_hours']]


data = data.merge(data_plots_agg, on='hhid', how='left')


data[['kg_fertilizer', 'hh_labor_hours']] = data_plots_agg[['kg_fertilizer',␣
 ↪'hh_labor_hours']]
data_kg_check = data[['hhid','hh_area_plots','total_kg_maize',␣
 ↪'total_kg_groundnut','total_kg_sorghum', 'total_kg_pigeonpeas', 'fertilizerkg'␣
 ↪]]
data_kg_check = data_kg_check.astype('float64')

data_kg_check = data_kg_check.merge(data_plots_agg, on='hhid')


### create difference. report those households with big differences
list_crops_check = ['maize', 'groundnut', 'pigeonpeas']
for crop in list_crops_check:
```

```
    data_kg_check['check_diff_'+crop] = data_kg_check['total_kg_'+crop].
 ↪fillna(0) - data_kg_check[crop+'_kg'].fillna(0)

data_kg_check['check_diff_fertilizer'] = data_kg_check['fertilizerkg'].fillna(0)␣
 ↪- data_kg_check['kg_fertilizer'].fillna(0)

data_diff =␣
 ↪data_kg_check[['hhid','check_diff_maize','check_diff_groundnut','check_diff_pigeonpeas','chec

print('================================================================')
print('Differences total - sum across plots per crop summary')
print('================================================================')
print(data_diff.describe())

print('COMMENTS')
print('1. High data quality: the difference between total reported and when then␣
 ↪I sum across plots are not that big! People do know well what they produce.')
print('2. As a general rule, for measurign agricultural production, lets use␣
 ↪total quantity report instead of sum accross plots.')
print('3. total quantity tends to be larger than across plots.')
print('4. We investigated the hhs with big difference in reporting (next table)␣
 ↪and asked Augustine for corrections')


data_diff.replace([0,0.0], np.nan, inplace=True)
data_diff.
 ↪dropna(subset=['check_diff_maize','check_diff_groundnut','check_diff_pigeonpeas','check_diff_
 ↪axis=0, how='all',inplace=True)


### These are the households to check:
print('')
print('================================================================')
print('Check: Households that aggregate vs sum(plots) variables do not coincide')
print('================================================================')
print(data_diff)


### implement Augustine corrections --------------
'''
7.        Comparison reported total vs sum across plots of crops production and␣
 ↪fertilizer. Ask enumerators for the differences and whether the correct value␣
 ↪is the total reported quantity or the sum across plots.
-        Hhid=1003 reported125kg of maize production more when asked in total␣
 ↪than when we do the sum across plots. For groundnuts the difference is 25kg,␣
 ↪for pigeonpeas is 12.5, and for fertilizer is 100kg.
```

```
-        Hhid=1323, mentioned 100 kg more of maize when asked in total than when␣
↪asked across plots. Also 50kg more of groundnut when asked in total than␣
↪across plots.
-        Hhid=1416: reported 300kg more of maize when asked in total than when␣
↪asked across plots.
-        Hhid= 1506: reported 150kg more of maize when asked in total than when␣
↪asked across plots. 200 vs 50 kg.
-        Hhid=1521: reported 200kg more of maize when asked in total than when␣
↪asked across plots.
-        Hhid=1416: reported total kg fertilizer of 100 but sum across plots is␣
↪25kg.
-        Hhid= 1519: reported total kg fertilizer of 50kg but sum across plots␣
↪is 100kg.
-        Hhid= 1531: reported total kg of fertilizer of 100kg but sum across␣
↪plots Is 50kg.

Augustine corrections are in word file: REPLY-SIEG_2022 CALLBACKS␣
↪(ADDRESSED)_Albert comments.doc

'''
# 1003 missing
# 1323

print('I apply Augustine corrections from word file: REPLY-SIEG_2022 CALLBACKS␣
↪(ADDRESSED)_Albert comments.doc')

# If we were to to do per plot analysis some of these measures need to be␣
↪adressed in each plot.
data.loc[data['hhid']==1323,␣
↪['total_kg_maize','maize_kg','total_kg_groundnut','groundnut_kg']] # 1323␣
↪reported the correct amount in maize summing across plots
data.loc[data['hhid']==1323, ['total_kg_maize']] = data.loc[data['hhid']==1323,␣
↪['maize_kg']]
data.loc[data['hhid']==1323, ['total_kg_groundnut','groundnut_kg']] = 100␣
↪#100kgs unshelled
data.loc[data['hhid']==1416, ['kg_fertilizer']] = 100
data.loc[data['hhid']==1506, ['total_kg_maize','maize_kg']] = 250
data.loc[data['hhid']==1521, ['total_kg_maize','maize_kg']] = 300
data.loc[data['hhid']==1519, ['fertilizerkg','kg_fertilizer']] = 100
data.loc[data['hhid']==1531, ['kg_fertilizer']] = 100


#data.to_csv('income_data_preliminary.csv')
```

```
================================================================
Agriculture hh labor member 1
================================================================
```

	months_member_1	weeks_member_1	days_member_1	hours_member_1
count	356.00	356.00	356.00	356.00
mean	5.11	3.60	5.45	3.39
std	1.70	0.73	1.24	1.30
min	1.00	1.00	1.00	1.00
10%	3.00	2.00	3.00	2.00
25%	4.00	3.00	5.00	2.00
50%	6.00	4.00	6.00	3.00
75%	6.00	4.00	6.00	4.00
90%	7.00	4.00	6.00	5.00
99%	7.00	4.00	7.00	8.00
max	7.00	4.00	7.00	8.00

==

Agriculture hh labor member 2

==

	months_member_2	weeks_member_2	days_member_2	hours_member_2
count	289.00	289.00	289.00	289.00
mean	5.10	3.63	4.85	3.13
std	1.62	0.66	1.73	1.20
min	1.00	2.00	1.00	1.00
10%	3.00	2.80	2.00	2.00
25%	4.00	3.00	3.00	2.00
50%	5.00	4.00	6.00	3.00
75%	6.00	4.00	6.00	4.00
90%	7.00	4.00	6.00	4.20
99%	7.00	4.00	7.00	7.00
max	7.00	4.00	7.00	8.00

==

Agriculture hh labor member 3

==

	months_member_3	weeks_member_3	days_member_3	hours_member_3
count	158.00	158.00	158.00	158.00
mean	5.20	3.63	3.41	2.63
std	1.78	0.77	1.93	1.06
min	0.00	0.00	0.00	0.00
10%	3.00	2.00	2.00	2.00
25%	4.00	4.00	2.00	2.00
50%	6.00	4.00	2.00	3.00
75%	7.00	4.00	6.00	3.00
90%	7.00	4.00	6.00	4.00
99%	7.00	4.00	7.00	6.43
max	7.00	4.00	7.00	7.00

==

Distribution Agric Household Labor in days

==

	hh_labor_days	member_1_labor_days	member_2_labor_days	\
count	470.00	356.00	289.00	
mean	182.11	105.29	93.95	

```
std          178.85                 49.75                 52.99
min            0.00                  2.00                  8.00
10%            0.00                 30.00                 24.00
25%           19.50                 72.00                 48.00
50%          148.00                112.00                 90.00
75%          288.00                144.00                144.00
90%          432.00                168.00                168.00
99%          708.68                168.00                196.00
max        1,176.00                196.00                196.00

       member_3_labor_days
count               158.00
mean                 69.15
std                  52.03
min                   0.00
10%                  14.80
25%                  32.00
50%                  56.00
75%                  96.00
90%                 168.00
99%                 196.00
max                 196.00
=================================================================
Distribution Agric Household Labor in hours
=================================================================
       hh_labor_hours  member_1_labor_hours  member_2_labor_hours  \
count          470.00                356.00                289.00
mean           587.47                370.57                310.32
std            616.21                249.37                237.64
min              0.00                  8.00                 16.00
10%              0.00                 87.00                 59.20
25%             39.00                166.50                120.00
50%            433.00                336.00                224.00
75%            877.50                504.00                480.00
90%          1,370.40                720.00                672.00
99%          2,821.08              1,008.00              1,008.00
max          3,024.00              1,568.00              1,152.00

       member_3_labor_hours
count                158.00
mean                 181.23
std                  164.26
min                    0.00
10%                   36.00
25%                   72.00
50%                  144.00
75%                  224.00
90%                  432.00
```

```
99%                  804.44
max                  980.00
IN FILE data_plotlevel.csv there is hhid-plotid long format dataset. CHECK IT TO
SEE IF VALUES MAKE SENSE. ESPECIALLY FERTILIZER!
================================================================
Differences total - sum across plots per crop summary
================================================================
          hhid  check_diff_maize  check_diff_groundnut  check_diff_pigeonpeas  \
count   268.00            268.00                268.00                 268.00
mean  1,277.59             15.05                  2.61                   0.47
std     173.96             70.67                 15.01                   5.75
min   1,001.00            -25.00                -10.00                 -40.00
25%   1,123.75              0.00                  0.00                   0.00
50%   1,304.50              0.00                  0.00                   0.00
75%   1,429.25              0.00                  0.00                   0.00
max   1,550.00            675.00                150.00                  50.00

       check_diff_fertilizer
count                 268.00
mean                  513.53
std                 6,187.47
min                -7,450.00
25%                     0.00
50%                     0.00
75%                     0.00
max                94,810.00
COMMENTS
1. High data quality: the difference between total reported and when then I sum
across plots are not that big! People do know well what they produce.
2. As a general rule, for measurign agricultural production, lets use total
quantity report instead of sum accross plots.
3. total quantity tends to be larger than across plots.
4. We investigated the hhs with big difference in reporting (next table) and
asked Augustine for corrections
```

I apply Augustine corrections from word file: REPLY-SIEG_2022 CALLBACKS (ADDRESSED)_Albert comments.doc

6 Cash-Transfer Subsidy

[6]:
```
#%% Check cashtransfer subsidy

## Need to reupload dataset since now was in string format.
data = pd.read_csv('income_data_preliminary.csv')
```

```
data['cashtrans_yes'] = data['other_sour_income_3'].replace(2,0)
data['cashtrans_value'] = data['other_sour_income_4'].replace(0,np.nan)

sum_subsidy = data[['cashtrans_yes','cashtrans_value']].describe()
print('================================================================')
print('Conditional Cash Transfer Program Implementation in the Village.')
print('================================================================')
print(sum_subsidy)
```

7 Coupons and Fertilizer

```
================================================================
Conditional Cash Transfer Program Implementation in the Village.
================================================================
       cashtrans_yes  cashtrans_value
count         273.00            13.00
mean            0.05        31,050.46
std             0.21        12,954.17
min             0.00           656.00
25%             0.00        36,000.00
50%             0.00        36,000.00
75%             0.00        36,000.00
max             1.00        47,000.00
```

[7]:
```
#%% Check  fertilizer ============================

(data[['fertilizerbuymarketkg', 'buyfertilizierpay']].replace(0,np.nan)).
 ↪describe()

data[['fertilizerbuymarketkg','buyfertilizierpay']] =␣
 ↪remove_outliers(data[['fertilizerbuymarketkg','buyfertilizierpay']],lq=0.05,␣
 ↪hq=0.95)

data['hh_p_fert_kg'] = data['buyfertilizierpay'].replace(0,np.nan)/
 ↪data['fertilizerbuymarketkg'].replace(0,np.nan)

p_fertmean =np.nanmean(data['hh_p_fert_kg'])
p_fertmed =np.nanmedian(data['hh_p_fert_kg'])
x0 = data.columns.get_loc("govcoupon")

x1 = data.columns.get_loc("lobor_inc1")

data_fert = data[['hhid', 'cashtrans_yes', 'cashtrans_value']]
data_fert = pd.concat([data_fert, data.iloc[:,x0:x1]], axis=1)
```

```
### replace 0s by nans. replace 2 by 0.

data_fert['p_fert_mean'] = p_fertmean
data_fert['p_fert_median'] = p_fertmed

# median price of a 50kg bag
print('mean price 50kg fertilizer bag', p_fertmean*50)
print('med price 50kg fertilizer bag', p_fertmed*50)

# AUGUSTINE: 50kg bag of fertilizer costs 35000MWK.

# Given this, I'll use the mean price 15000MWK

data[['govcoupon',␣
↪'fertilizeryes','fertilizerbuymarketyes','recevfertilizeryes','fertoutyes','paybackfertkg_1',
↪ 'chiefinvolvedfert_1', 'fertvillageryes_1', 'recevbackfer_1', ␣
↪'chiefproposefertout_1', 'chiefbargainfertout_1','recevbackfer_2',␣
↪'chiefproposefertout_2', 'chiefbargainfertout_2']] = (data[['govcoupon',␣
↪'fertilizeryes','fertilizerbuymarketyes','recevfertilizeryes','fertoutyes','paybackfertkg_1',
↪ 'chiefinvolvedfert_1', 'fertvillageryes_1', 'recevbackfer_1', ␣
↪'chiefproposefertout_1', 'chiefbargainfertout_1','recevbackfer_2',␣
↪'chiefproposefertout_2', 'chiefbargainfertout_2']].replace([0,0.0],np.nan)).
↪replace([2,2.0],0)

## Coupons summary
print('=================================================================')
print('Summary of Coupons')
print('=================================================================')
print((data[['govcoupon', 'govcouponmany' ]].describe()))

sum_fertilizer =␣
↪data[['fertilizeryes','fertilizerkg','kg_fertilizer','fertilizerbuymarketyes','fertilizerbuym
↪describe(percentiles=percentiles)

print('Summary fertilizer =========================')
print(sum_fertilizer)

### extreme values in fertlizier. 85% of housheolds used fertilizer!

print('Top extreme values fertilizer ===================')
print(data.loc[data['fertilizerkg']>200,['hhid','fertilizerkg','kg_fertilizer']])
print(data.
↪loc[data['kg_fertilizer']>200,['hhid','fertilizerkg','kg_fertilizer']])



# replace extreme values in aggregate reported by sum across plots and viceversa
```

```
data.loc[data['fertilizerkg']>300,['fertilizerkg']] = data.
 ↪loc[data['fertilizerkg']>300,['kg_fertilizer']]
data.loc[data['kg_fertilizer']>300,['kg_fertilizer']] = data.
 ↪loc[data['kg_fertilizer']>300,['fertilizerkg']]

# also it seems big discrepancies btw total vs sum across plots. Some even 0.␣
 ↪replace 0 or smaller values but value on other variable:
# FOR A NEXT TIME: SEEMS TO BIG DIFFERENCE> I ALSO THINK SOME HH ANSWERED 2 BAGS␣
 ↪INSTEAD OF 100KG
# in the following code I try to correct for these problems
print('bottom extreme values discrepancies fertilizer ===================')
print(data.loc[data['fertilizerkg']<5,['hhid','fertilizerkg','kg_fertilizer']])
print(data.loc[data['kg_fertilizer']<5,['hhid','fertilizerkg','kg_fertilizer']])

data.loc[data['fertilizerkg']<5,['fertilizerkg']] = data.
 ↪loc[data['fertilizerkg']<5,['kg_fertilizer']]
data.loc[data['kg_fertilizer']<5,['kg_fertilizer']] = data.
 ↪loc[data['kg_fertilizer']<5,['fertilizerkg']]

sum_fertilizer =␣
 ↪data[['fertilizeryes','fertilizerkg','kg_fertilizer','fertilizerbuymarketyes','fertilizerbuym
 ↪describe(percentiles=percentiles)
print('================================================================')
print('Summary fertilizer final---after cleaning and applying corrections.')
print('================================================================')
print(sum_fertilizer)

# also many discrepancies between fertilizer at aggregate level and per plot.␣
 ↪Sometimes aggregate is 0 sum across plots is 250
```

```
mean price 50kg fertilizer bag 14620.366563891745
med price 50kg fertilizer bag 7500.0
================================================================
Summary of Coupons
================================================================
       govcoupon  govcouponmany
count     254.00         273.00
mean        0.72           1.32
std         0.45           6.72
min         0.00           0.00
25%         0.00           0.00
50%         1.00           1.00
75%         1.00           1.00
max         1.00         100.00
Summary fertilizer ========================
       fertilizeryes  fertilizerkg  kg_fertilizer  fertilizerbuymarketyes  \
count         254.00        273.00         269.00                  217.00
```

```
mean            0.85       581.19         78.17                      0.76
std             0.35     6,118.98        458.16                      0.43
min             0.00         0.00          0.00                      0.00
10%             0.00         0.00          0.00                      0.00
25%             1.00         5.00          5.00                      1.00
50%             1.00        50.00         45.00                      1.00
75%             1.00       100.00         95.00                      1.00
90%             1.00       100.00        100.00                      1.00
99%             1.00     7,216.00        266.00                      1.00
max             1.00    95,000.00      7,500.00                      1.00

       fertilizerbuymarketkg  buyfertilizierpay
count                 247.00             247.00
mean                   35.18           6,295.75
std                    40.00           6,730.43
min                     0.00               0.00
10%                     0.00               0.00
25%                     0.00               0.00
50%                    20.00           6,000.00
75%                    50.00          14,000.00
90%                   100.00          15,000.00
99%                   100.00          23,310.00
max                   150.00          25,000.00
Top extreme values fertilizer ==================
     hhid  fertilizerkg  kg_fertilizer
100  1211        250.00         250.00
116  1229     95,000.00           5.00
134  1303        250.00          50.00
181  1403        300.00          20.00
231  1506     25,000.00          50.00
264  1539     25,000.00          50.00
     hhid  fertilizerkg  kg_fertilizer
100  1211        250.00         250.00
133  1302          0.00         250.00
180  1402         50.00         300.00
202  1429        200.00         550.00
263  1538         50.00       7,500.00

================================================================
Summary fertilizer final---after cleaning and applying corrections.
================================================================
       fertilizeryes  fertilizerkg  kg_fertilizer  fertilizerbuymarketyes  \
count         254.00        209.00         201.00                  217.00
mean            0.85         65.33          64.52                    0.76
std             0.35         50.66          50.46                    0.43
min             0.00          5.00           5.00                    0.00
10%             0.00         15.00          15.00                    0.00
25%             1.00         25.00          25.00                    1.00
```

```
50%               1.00          50.00          50.00                    1.00
75%               1.00         100.00         100.00                    1.00
90%               1.00         100.00         100.00                    1.00
99%               1.00         246.00         250.00                    1.00
max               1.00         300.00         300.00                    1.00

       fertilizerbuymarketkg  buyfertilizierpay
count                 247.00             247.00
mean                   35.18           6,295.75
std                    40.00           6,730.43
min                     0.00               0.00
10%                     0.00               0.00
25%                     0.00               0.00
50%                    20.00           6,000.00
75%                    50.00          14,000.00
90%                   100.00          15,000.00
99%                   100.00          23,310.00
max                   150.00          25,000.00
```

8 Agricultural Inputs: Labor, Fertilizer, and other Intermediates

[8]:

```
#%% ============================================================================
#  AGRICULTURAL INPUTS
# ============================================================================
# labor
# fertilizer
# others

# total labor number persons


# Hired labor
list_persons = ['men','women','kids']
data['w_men']=np.nan
data['w_women']=np.nan
data['w_kids']=np.nan
data['hired_N'] = 0

for person in list_persons:
    data['hired_N'] += data['manyhired'+str(person)].fillna(0)
    data['hired_'+str(person)+'_avg_hours'] =␣
 ↪(data['hireplotmotnhs'+str(person)]*data['hireplotweeks'+str(person)]*data['hireplotdays'+str
 ↪replace(0,np.nan)
    data['hired_'+str(person)+'_L'] =␣
 ↪data['manyhired'+str(person)]*data['hired_'+str(person)+'_avg_hours']
```

```
    data['w_'+str(person)] = (data['hireplotwage'+str(person)].replace(0,np.nan)␣
↪/ data['hired_'+str(person)+'_avg_hours'])
    data['weight_'+str(person)] = np.nanmedian(data['w_'+str(person)])

sum_hiredlabor = data[['w_men','w_women','w_kids' , 'hired_men_avg_hours',␣
↪'hired_women_avg_hours','hired_kids_avg_hours' , 'hireplotwagemen',␣
↪'hireplotwagewomen', 'hireplotwagekids']].describe()
print('=========== Summary Hired Labor ===============')
print(sum_hiredlabor)


data['hhlabor_N'] = data['manyhhlaborplot']
data['labor_N'] = (data['hhlabor_N'].fillna(0) +data['hired_N'].fillna(0))
data['labor_h'] = (data['hh_labor_hours'].fillna(0) +data['hired_men_L'].
↪fillna(0) +data['hired_women_L'].fillna(0) +data['hired_kids_L'].fillna(0))


sum_agriclabor = data[['labor_N', 'labor_h', 'hired_N',␣
↪'hh_labor_hours','hired_men_L','hired_women_L', 'hired_kids_L']].describe()
print('=========== Summary Household + Hired Agricultural Labor input␣
↪===============')
print(sum_agriclabor)
print('Where _N denotes in supply number of persons, _h or _L in total hours')


### NEEED TO CLEAN YING VARIABLES

# obtain value non-bought fertilizer. Use median price
data['p_fert']  = pd.to_numeric(data['buyfertilizierpay'].
↪divide(data['fertilizerbuymarketkg'].replace([0,0.0], np.nan)))

## Use kg of fertilizer by total report
data['value_fertilizer'] = p_fertmed*data['fertilizerkg']

#intermediates
data['interm'] = (data['spendseeds'].fillna(0) +data['buyfertilizierpay'].
↪fillna(0) +data['spendpesticides'].fillna(0)).replace(0,np.nan)

sum_interm = data[['interm','value_fertilizer','kg_fertilizer','spendseeds',␣
↪'spendpesticides']].describe(percentiles=percentiles)
print('===============================================================')
print(' Summary Intermediate inputs')
print('===============================================================')
print('All variables in MWK except kg_fertilizer.')
print(sum_interm)
```

```
datalab= data[['hhid','hired_N', 'y_agric', 'total_kg_maize']]



### ===== SUMMARY AGRICULTURAL INPUTS ================
data_inp = data[['hh_area_plots','hh_value_plots','k_farm','interm','labor_N',␣
 ↪'labor_h','hired_N', 'hh_labor_hours','hired_men_L','hired_women_L',␣
 ↪'hired_kids_L','value_fertilizer','kg_fertilizer','spendseeds',␣
 ↪'spendpesticides']]

data_inp[['hh_value_plots','k_farm','interm','value_fertilizer','spendseeds',␣
 ↪'spendpesticides']] =␣
 ↪data_inp[['hh_value_plots','k_farm','interm','value_fertilizer','spendseeds',␣
 ↪'spendpesticides']]/dollar_MWK
sum_inp = data_inp[['hh_area_plots','hh_value_plots','k_farm','labor_N',␣
 ↪'labor_h', 'hh_labor_hours','hired_men_L', 'hired_women_L', ]].
 ↪describe(percentiles=percentiles)

print('  ')
print('  ')
print('  ')
print('=============================================================')
print('    SUMMARY AGRICULTURAL INPUTS ')
print('=============================================================')
print(sum_inp)

sum_inp2 = data_inp[[ 'hired_kids_L',␣
 ↪'interm','value_fertilizer','kg_fertilizer','spendseeds', 'spendpesticides']].
 ↪describe(percentiles=percentiles)
print(sum_inp2)
```

```
=========== Summary Hired Labor ===============
          w_men    w_women    w_kids  hired_men_avg_hours  \
count     35.00      19.00     19.00                35.00
mean     648.80     833.03    287.19               181.74
std    2,034.71   2,474.57    289.96               231.77
min       17.36      31.25      3.47                 1.00
25%       69.44     100.73     81.25                22.00
50%      121.53     200.00    214.29                84.00
75%      385.42     386.90    428.82               288.00
max   12,000.00  11,000.00  1,200.00              720.00

       hired_women_avg_hours  hired_kids_avg_hours  hireplotwagemen  \
count                  19.00                 19.00           273.00
mean                  102.95                 65.26         1,968.74
std                   146.93                128.87         6,860.69
min                     1.00                  1.00             0.00
```

25%	13.00	12.00	0.00
50%	35.00	24.00	0.00
75%	108.00	57.00	0.00
max	480.00	576.00	50,000.00

	hireplotwagewomen	hireplotwagekids
count	273.00	273.00
mean	842.49	380.59
std	4,949.65	1,940.08
min	0.00	0.00
25%	0.00	0.00
50%	0.00	0.00
75%	0.00	0.00
max	69,000.00	20,000.00

=========== Summary Household + Hired Agricultural Labor input ===============

	labor_N	labor_h	hired_N	hh_labor_hours	hired_men_L	hired_women_L \
count	273.00	273.00	273.00	269.00	35.00	19.00
mean	3.19	1,158.46	0.86	1,026.43	560.00	298.05
std	2.73	1,311.86	2.50	1,050.29	1,068.76	518.31
min	0.00	0.00	0.00	0.00	1.00	1.00
25%	2.00	378.00	0.00	344.00	22.00	26.00
50%	3.00	768.00	0.00	702.00	160.00	48.00
75%	4.00	1,536.00	0.00	1,344.00	638.00	244.00
max	20.00	12,192.00	20.00	5,880.00	5,760.00	1,920.00

	hired_kids_L
count	19.00
mean	783.53
std	2,608.50
min	1.00
25%	26.00
50%	120.00
75%	280.00
max	11,520.00

Where _N denotes in supply number of persons, _h or _L in total hours

===

Summary Intermediate inputs

===

All variables in MWK except kg_fertilizer.

	interm	value_fertilizer	kg_fertilizer	spendseeds	spendpesticides
count	218.00	209.00	201.00	273.00	273.00
mean	12,649.93	9,799.52	64.52	4,062.82	342.44
std	9,926.87	7,598.33	50.46	5,625.22	3,333.13
min	100.00	750.00	5.00	0.00	0.00
10%	2,000.00	2,250.00	15.00	0.00	0.00
25%	5,000.00	3,750.00	25.00	0.00	0.00
50%	10,450.00	7,500.00	50.00	2,000.00	0.00
75%	17,500.00	15,000.00	100.00	5,600.00	0.00

90%	24,650.00	15,000.00	100.00	11,080.00	0.00
99%	45,484.00	36,900.00	250.00	25,368.00	6,831.80
max	62,000.00	45,000.00	300.00	31,000.00	50,000.00

===

SUMMARY AGRICULTURAL INPUTS

===

	hh_area_plots	hh_value_plots	k_farm	labor_N	labor_h	\
count	273.00	273.00	273.00	273.00	273.00	
mean	2.22	445.80	13.45	3.19	1,158.46	
std	2.15	621.68	26.57	2.73	1,311.86	
min	0.00	0.00	0.00	0.00	0.00	
10%	0.50	93.17	1.46	1.00	38.60	
25%	1.00	145.58	2.91	2.00	378.00	
50%	1.50	291.16	7.28	3.00	768.00	
75%	3.00	485.27	14.07	4.00	1,536.00	
90%	4.00	776.43	23.58	5.00	2,668.80	
99%	8.78	3,253.23	143.08	15.56	5,784.96	
max	21.00	5,726.15	284.85	20.00	12,192.00	

	hh_labor_hours	hired_men_L	hired_women_L
count	269.00	35.00	19.00
mean	1,026.43	560.00	298.05
std	1,050.29	1,068.76	518.31
min	0.00	1.00	1.00
10%	24.80	8.80	8.60
25%	344.00	22.00	26.00
50%	702.00	160.00	48.00
75%	1,344.00	638.00	244.00
90%	2,308.80	1,324.80	832.00
99%	5,559.68	4,536.00	1,804.80
max	5,880.00	5,760.00	1,920.00

	hired_kids_L	interm	value_fertilizer	kg_fertilizer	spendseeds	\
count	19.00	218.00	209.00	201.00	273.00	
mean	783.53	12.28	9.51	64.52	3.94	
std	2,608.50	9.63	7.37	50.46	5.46	
min	1.00	0.10	0.73	5.00	0.00	
10%	17.60	1.94	2.18	15.00	0.00	
25%	26.00	4.85	3.64	25.00	0.00	
50%	120.00	10.14	7.28	50.00	1.94	
75%	280.00	16.98	14.56	100.00	5.43	
90%	624.00	23.92	14.56	100.00	10.75	
99%	9,576.00	44.14	35.81	250.00	24.62	
max	11,520.00	60.17	43.67	300.00	30.09	

spendpesticides

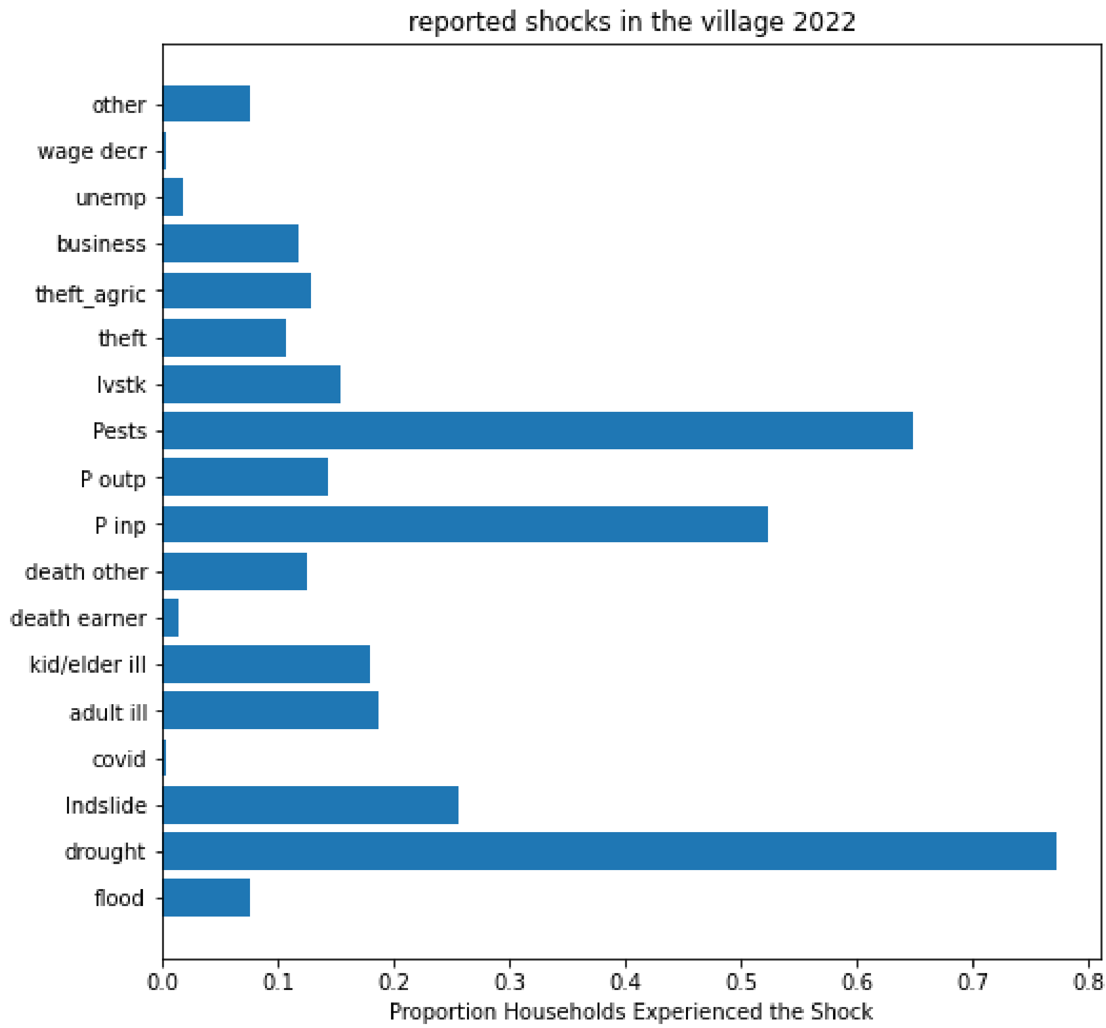
```
count           273.00
mean              0.33
std               3.23
min               0.00
10%               0.00
25%               0.00
50%               0.00
75%               0.00
90%               0.00
99%               6.63
max              48.53
```

9 Shocks

[9]:

```
#%% ============================================================================
#  Shocks
# ============================================================================

list_abcd =␣
 ↪['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r']

data['shocks'] = 0
for a in list_abcd:
    data['shocks'] += (data['shocks_'+a+'1'].replace(2,0)).fillna(0)

data['shock_flood'] = data['shocks_a1'].replace(2,0)
data['shock_drought'] = data['shocks_b1'].replace(2,0)
data['shock_lndslide'] = data['shocks_c1'].replace(2,0)
data['shock_covid'] = data['shocks_d1'].replace(2,0)
data['shock_adultill'] = data['shocks_e1'].replace(2,0)
data['shock_kidill'] = data['shocks_f1'].replace(2,0)
data['shock_death_earner'] = data['shocks_g1'].replace(2,0)
data['shock_death_othermemb'] = data['shocks_h1'].replace(2,0)
data['shock_inp_p'] = data['shocks_i1'].replace(2,0)
data['shock_out_p'] = data['shocks_j1'].replace(2,0)
data['shock_pests'] = data['shocks_k1'].replace(2,0)
data['shock_lvstk'] = data['shocks_l1'].replace(2,0)
data['shock_theft'] = data['shocks_m1'].replace(2,0)
data['shock_theft_agric'] = data['shocks_n1'].replace(2,0)
data['shock_business'] = data['shocks_o1'].replace(2,0)
data['shock_unemp'] = data['shocks_p1'].replace(2,0)
data['shock_wage_decr'] = data['shocks_q1'].replace(2,0)
data['shock_other'] = data['shocks_r1'].replace(2,0)



shocks = data['shocks'].value_counts()/len(data)
#Proportion of individuals that reported each shock
```

```
shocks_avg= np.array(np.
 ↪mean(data[['shock_flood','shock_drought','shock_lndslide','shock_covid','shock_adultill','sho
]],axis=0))

p_shocks = np.sum(shocks_avg)


labels = ['flood', 'drought', 'lndslide', 'covid', 'adult ill', 'kid/elder ill',␣
 ↪'death earner', 'death other', 'P inp', 'P outp', 'Pests', 'lvstk', 'theft',␣
 ↪'theft_agric', 'business', 'unemp', 'wage decr', 'other']

save=False
#Bar Plot
fig, ax = plt.subplots(figsize=(8,8))
ax.barh(np.arange(len(shocks_avg)), shocks_avg, tick_label=labels)
plt.title('reported shocks in the village 2022')
plt.xlabel('Proportion Households Experienced the Shock')
plt.show()
if save==True:
    fig.savefig(folder_fig+'village_shocks.png', bbox_inches='tight')
```



10 Formal Labor and Ganyu

[10]:
```
### LABOR INCOME: SALARY LABOR (last rainy season) ----------------------

## propotion households getting sallary work:
data['lobor_inc1'].replace(2,0, inplace=True)
pd.value_counts(data['lobor_inc1'])/len(data)

data['wlabor'] = 0
## labor supply in hours
data['wlabor_supply'] = ((data['lobor_inc4']).
↪multiply(data['lobor_inc5'],axis=0, fill_value=0)).
↪multiply(data['lobor_inc6'],axis=0, fill_value=0)
```

```
## Construct wlabor income for the rainy season
pd.value_counts(data['lobor_inc8'])

data.loc[data['lobor_inc8']==1,'wlabor_inc'] = data.loc[data['lobor_inc8']==1,␣
↪'lobor_inc7']
data.loc[data['lobor_inc8']==2, 'wlabor_inc'] = data.loc[data['lobor_inc8']==2,␣
↪'lobor_inc7'].multiply(data['lobor_inc4'],axis=0, fill_value=0)

data['wlabor_supply'] = pd.to_numeric(data['wlabor_supply'])
data['wlabor_inc_dollar'] = data['wlabor_inc']/dollar_MWK

## To rainy season
data['wlabor_inc'] = pd.to_numeric(data['wlabor_inc'])*7


### NEEED TO CLEAN YING VARIABLES

print('========== Wage Labor earnings (1 month, in dollars) ==============')
print(data['wlabor_inc_dollar'].describe())


## LABOR INCOME: GANYU (last  month) -----------------------------------
data['lobor_inc9'].replace(2,0, inplace=True)
pd.value_counts(data['lobor_inc9'])/len(data)
# 44% households involved in ganyu

# summary ganyu labor weeks, days, hours

data[['lobor_inc10','lobor_inc11','lobor_inc12']].describe()


## more than 4 weeks per month, 7 days, 10 hours, replacement
data.loc[data['lobor_inc10']>4.00,'lobor_inc10'] = 4
data.loc[data['lobor_inc11']>7.00,'lobor_inc11'] = 7
data.loc[data['lobor_inc12']>10.00,'lobor_inc12'] = 10


## labor supply in hours
data['ganyu_supply'] = (data['lobor_inc10'].multiply(data['lobor_inc11'],axis=0,␣
↪fill_value=0)).multiply(data['lobor_inc12'],axis=0, fill_value=0)

(data[['lobor_inc10','lobor_inc11','lobor_inc12','ganyu_supply']].replace(0, np.
↪nan)).describe()

#data[['lobor_inc10','lobor_inc11','lobor_inc12','ganyu_supply']] =␣
↪remove_outliers(data[['lobor_inc10','lobor_inc11','lobor_inc12','ganyu_supply']],hq=0.
↪975)
```

```
# for some reason they could put more weeks days and hours than possible

## average weeks per 3 months
data['lobor_inc10'].replace(0,np.nan).describe()
## Construct wage per hour
pd.value_counts(data['lobor_inc13_period']) ## most people reported in ganyu per␣
↪task. followed by weekly and daily

### need to know how many ganyus...
# daily
data.loc[data['lobor_inc13_period']==2, 'ganyu_inc'] = (data.
↪loc[data['lobor_inc13_period']==2, 'lobor_inc13'].
↪multiply(data['lobor_inc11'],axis=0, fill_value=0)).
↪multiply(data['lobor_inc10'],axis=0, fill_value=0)
#weekly
data.loc[data['lobor_inc13_period']==3, 'ganyu_inc'] = data.
↪loc[data['lobor_inc13_period']==3, 'lobor_inc13'].
↪multiply(data['lobor_inc10'],axis=0, fill_value=0)
# number ganyu. Since we didn't ask but households reported sallary per␣
↪ganyu,I'll use the median number of ganyus from February 2023.
#data.loc[data['lobor_inc13_period']==5, 'ganyu_inc'] = data.
↪loc[data['lobor_inc13_period']==5, 'lobor_inc13']*

## to those reported by task, assume it is the total of the whole month. Numbers␣
↪too large

#Median of 14$
data.loc[data['lobor_inc13_period']==5,'ganyu_inc'] = data.
↪loc[data['lobor_inc13_period']==5, 'lobor_inc13']  ## I assume 1 ganyu..
data.loc[data['lobor_inc13_period']==4,'ganyu_inc'] = data.
↪loc[data['lobor_inc13_period']==4, 'lobor_inc13']  ## I assume 1 ganyu..


data[['ganyu_inc','lobor_inc13','lobor_inc13_period']]

## outliers in hours work in ganyu and income received
data['ganyu_inc'] = pd.to_numeric(data['ganyu_inc'])
data['ganyu_supply'] = pd.to_numeric(data['ganyu_supply'])
data[['ganyu_inc','ganyu_supply']] =␣
↪remove_outliers(data[['ganyu_inc','ganyu_supply']], hq=0.95)


## to rainy season:
data['ganyu_inc'] = data['ganyu_inc']*(7)
```

```
data['ganyu_supply'] = data['ganyu_supply']*(7)
data['ganyu_inc_dollar'] = data['ganyu_inc']/dollar_MWK


sum_nonagri_labor =␣
 ↪data[['wlabor_inc_dollar','wlabor_supply','ganyu_inc_dollar','ganyu_supply']].
 ↪describe(percentiles=percentiles)
print('===============================================================')
print('Salary and Ganyu labor income and supply (at rainy season, 7 months)')
print('===============================================================')
print('income in dollars')
print('at household level')
print(sum_nonagri_labor)
```

```
========== Wage Labor earnings (1 month, in dollars) ==============
count     8.00
mean     45.31
std      37.14
min       0.00
25%      19.90
50%      37.37
75%      69.76
max      97.05
Name: wlabor_inc_dollar, dtype: float64
===============================================================
Salary and Ganyu labor income and supply (at rainy season, 7 months)
===============================================================
income in dollars
at household level
       wlabor_inc_dollar  wlabor_supply  ganyu_inc_dollar  ganyu_supply
count               8.00         273.00            103.00        255.00
mean               45.31           4.95            189.36        176.02
std                37.14          31.71            241.12        312.22
min                 0.00           0.00              0.01          0.00
10%                 6.79           0.00             24.46          0.00
25%                19.90           0.00             50.95          0.00
50%                37.37           0.00            101.91          0.00
75%                69.76           0.00            203.81        217.00
90%                97.05           0.00            500.02        672.00
99%                97.05         181.44          1,085.91      1,298.64
max                97.05         288.00          1,087.00      1,344.00
```

11 Business Income

[11]:
```
### BUSINESS INCOME
data['busin_income_1'].replace(2,0)
pd.value_counts(data['busin_income_1'])
type_business = pd.value_counts(data['busin_income_2'])
data['business_type'] = data['busin_income_2']
data['business_months'] = data['busin_income_3']

pd.value_counts(data['business_months'])
data['business_profits1'] = data['busin_income_4']
data['business_revenue'] =  data['busin_income_5']
data['business_costs'] = data['busin_income_6'] + data['busin_income_7']
data['business_profits2'] = data['business_revenue'] - data['business_costs']

business_data = data.loc[data['busin_income_1']==1, ['hhid', 'business_type',␣
↪'business_revenue', 'business_costs', 'business_profits1','business_profits2']]
business_data[['business_revenue', 'business_costs',␣
↪'business_profits1','business_profits2']] = business_data[['business_revenue',␣
↪'business_costs', 'business_profits1','business_profits2']]/dollar_MWK

print(business_data)


data['business_months'] =  pd.to_numeric(data['business_months'],errors='coerce')
data['business_revenue'] =  pd.
↪to_numeric(data['business_revenue'],errors='coerce')
data['business_costs'] =  pd.to_numeric(data['business_costs'],errors='coerce')
data['business_profits1'] =  pd.
↪to_numeric(data['business_profits1'],errors='coerce')
data['business_profits2'] =  pd.
↪to_numeric(data['business_profits2'],errors='coerce')

### to rainy season level:
data[['business_revenue']] =data['business_revenue']*data['business_months']*7/12
data[['business_costs']] =data['business_costs']*data['business_months']*7/12
data[['business_profits1']] =data['business_profits1']*data['business_months']*7/
↪12
data[['business_profits2']] =data['business_profits2']*data['business_months']*7/
↪12

sum_business = (data[['business_revenue', 'business_costs',␣
↪'business_profits1','business_profits2']].replace(0,np.nan)/dollar_MWK).
↪describe(percentiles=percentiles)
print('===============================================================')
print('Summary Business income at rainy season')
print('===============================================================')
```

```
print('values in dollars')
print(sum_business)
```

```
==============================================================
Summary Business income at rainy season
==============================================================
values in dollars
       business_revenue  business_costs  business_profits1  business_profits2
count             62.00           54.00              63.00              63.00
mean             411.12          227.61             161.35             209.50
std              857.55          607.48             355.95             404.07
min                3.96            1.13               2.83            -101.91
10%               19.59            5.60               8.49               8.04
25%               48.41           16.98              16.98              16.98
50%              127.38           35.78              67.94              67.94
75%              331.19           99.08             161.35             217.12
90%              712.07          432.76             280.81             339.69
99%            4,183.81        2,964.79           1,788.16           1,908.05
max            4,529.16        3,396.87           2,488.77           2,488.77
```

12 Transfers: Government, NGO, and Remittances

[12]:

```
### OTHER SOURCES OF INCOME -------------------------------------------------
data[['NGO_yes', 'cashtrans_yes', 'gov_yes', 'remittances_yes']] =␣
 ↪data[['other_sour_income_1', 'other_sour_income_3', 'other_sour_income_5',␣
 ↪'other_sour_income_7']]
data[['NGO_yes', 'cashtrans_yes', 'gov_yes', 'remittances_yes']] =␣
 ↪data[['NGO_yes', 'cashtrans_yes', 'gov_yes', 'remittances_yes']].replace(2,0)
sum_other_prop = np.mean(data[['NGO_yes', 'cashtrans_yes', 'gov_yes',␣
 ↪'remittances_yes']], axis=0)

data['cashtrans_value'] = data['cashtrans_value']
data['NGO_trans'] = data['other_sour_income_2']
data['gov_trans'] = data['other_sour_income_6']
data['remittances'] = data['other_sour_income_8']
data['other_inc'] = data[['cashtrans_value', 'NGO_trans', 'gov_trans',␣
 ↪'remittances']].sum(axis=1)
sum_other = (data[['cashtrans_value', 'NGO_trans', 'gov_trans', 'remittances']].
 ↪replace(0,np.nan)/dollar_MWK).describe()
print('==============================================================')
print('Other sources of income: government, NGO and remittances transfers')
print('==============================================================')
print(sum_other)
```

```
==============================================================
Other sources of income: government, NGO and remittances transfers
```

```
==============================================================
       cashtrans_value  NGO_trans  gov_trans  remittances
count            13.00      17.00       5.00       111.00
mean             30.14      56.07      30.21        79.78
std              12.57      62.10      21.02       141.69
min               0.64       0.53       0.61         0.65
25%              34.94      24.26      22.32         9.71
50%              34.94      34.94      34.94        24.26
75%              34.94      69.59      34.94        72.79
max              45.62     242.63      58.23       776.43
```

13 Aggregate Income

[13]:

```
### AGGREGATE INCOME -----------------------------------------------------------

### FOR THE MOMENT I DO NOT SUBSTRACT FOR INTERMEDIATES COSTS. NEED TO BE SURE␣
↪HOW WE MEASURE COST FERTILIZERS.

data['y_net'] = data['y_agric'].fillna(0) -data['hireplotwagemen'].fillna(0)␣
↪-data['hireplotwagewomen'].fillna(0) -data['hireplotwagekids'].fillna(0)


## inctotal using agric revenues not profits
data['inctotal'] = data[['y_agric' ,'wlabor_inc', 'ganyu_inc',␣
↪'business_profits1']].sum(axis=1)
data['inctotal_trans'] = data[['y_agric' ,'wlabor_inc', 'ganyu_inc',␣
↪'business_profits1', 'other_inc']].sum(axis=1)



income = data[['hhid','inctotal','inctotal_trans','y_net','y_agric','y_maize',␣
↪'y_groundnut', 'wlabor_inc', 'ganyu_inc', 'business_profits1', 'other_inc']].
↪replace(0,np.nan)
sum_inc = (income.loc[:, income.columns != 'hhid']/dollar_MWK).
↪describe(percentiles=[0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 0.99])

var_list = ['inctotal','inctotal_trans', 'y_net', 'y_agric','y_maize',␣
↪'y_groundnut', 'wlabor_inc', 'ganyu_inc', 'business_profits1', 'other_inc']
gini_stat= np.empty((1, len(var_list)))

for i,state in enumerate(var_list):
    gini_stat[:,i] = gini(income[state].dropna().values)

data_gini = pd.DataFrame(gini_stat, columns=var_list)
data_gini.reset_index(inplace=True)
data_gini['index'] = 'gini'
```

```
sum_inc.reset_index(inplace=True)
sum_inc = sum_inc.append(data_gini, ignore_index=True)

print('================================================================')
print('Summary total Income (rainy season)')
print('================================================================')
print('values in $')
print(sum_inc)

print('Comment')
print('Very low income. Consistent with the bad 2022 harvest in Malawi (& other␣
 ↪countries) and what villagers and other people explained us in July 22.')
```

```
================================================================
Summary total Income (rainy season)
================================================================
values in $
    index  inctotal  inctotal_trans    y_net  y_agric  y_maize  y_groundnut  \
0   count    268.00          271.00   254.00   254.00   253.00        97.00
1    mean    211.48          247.34    93.02    96.35    71.69        38.79
2     std    268.55          283.42   106.75   110.01    90.29        54.33
3     min      5.29            5.29   -29.65     2.43     1.44         0.97
4      1%      9.48           13.67     2.77     4.16     3.63         0.97
5     10%     28.82           43.22    14.84    16.83    14.41         4.85
6     25%     59.76           77.12    36.10    37.36    28.82         4.85
7     50%    119.54          148.93    62.95    66.46    43.22        19.41
8     75%    286.78          314.11   108.88   114.70    86.45        48.53
9     90%    431.59          551.90   178.48   181.24   144.08        97.05
10    99%  1,135.80        1,142.25   476.94   484.47   374.04       293.10
11    max  2,495.98        2,495.98 1,039.36 1,087.89 1,008.57       339.69
12   gini      0.54            0.51     0.72     0.48     0.48         0.58

    wlabor_inc  ganyu_inc  business_profits1  other_inc
0         8.00     103.00              63.00     131.00
1       317.18     189.36             161.35      79.02
2       260.01     241.12             355.95     132.83
3         0.01       0.01               2.83       1.94
4         4.76       0.72               3.88       1.94
5        47.56      24.46               8.49       5.82
6       139.27      50.95              16.98       9.71
7       261.56     101.91              67.94      33.97
8       488.30     203.81             161.35      79.10
9       679.37     500.02             280.81     194.11
10      679.37   1,085.91           1,788.16     669.67
11      679.37   1,087.00           2,488.77     776.43
12        0.43       0.57               0.68       0.66
Comment
Very low income. Consistent with the bad 2022 harvest in Malawi (& other
```

countries) and what villagers and other people explained us in July 22.

14 Wealth

[14]:
```
#%% =============================================================================
#   WEALTH
# =============================================================================

data['housing'] = data['selldwell']

data['hh_assets'] = 0
for i in range(1,12):
    data['hh_assets'] += data['sellhhasset_'+str(i)]


sum_assets = data[['housing','hh_assets']].describe(percentiles=percentiles)



#STOP RUN



data['wtotal'] =␣
↪data[['housing','hh_assets','hh_value_plots','k_farm','hhlivestock']].
↪sum(axis=1)

print('======== Summary Wealth ==========')
print((data[['wtotal','housing','hh_assets','hh_value_plots','k_farm','hhlivestock']]/
↪dollar_MWK).describe())
```

```
======== Summary Wealth ==========
          wtotal    housing  hh_assets  hh_value_plots  k_farm  hhlivestock
count     273.00     273.00     273.00          273.00  273.00       273.00
mean    1,170.33     611.26      52.08          445.80   13.45        47.73
std     1,646.93   1,338.46     100.17          621.68   26.57       142.81
min        67.94       0.00       0.00            0.00    0.00         0.00
25%       492.06     145.58       0.00          145.58    2.91         0.00
50%       870.08     291.16      19.41          291.16    7.28         6.79
75%     1,232.58     727.90      52.41          485.27   14.07        42.22
max    18,872.04 14,558.02     732.75        5,726.15  284.85     2,066.16
```

[15]:
```
#%% Save dataset

### let's do some checks (before we get augustine corrections). Remove␣
↪observations with extreme/weird values
data[['y_agric', 'interm','labor_h', 'k_farm', 'hh_area_plots']].
↪describe(percentiles)
```

```
data['wave'] = 2022



data_short =  data[['hhid',  'wave','rightsellland', 'chiefpreventsell',␣
↪'chiefpreventbequeat', 'cashtrans_yes', 'govcoupon' ,
                     'inctotal', 'inctotal_trans','y_net','y_agric', 'y_maize',␣
↪'y_groundnut', 'y_pigeonpeas', 'total_kg_maize', 'total_kg_groundnut', ␣
↪'total_kg_pigeonpeas',
                     'wlabor_inc', 'ganyu_inc','business_revenue',␣
↪'business_profits1','business_profits2', 'other_inc', 'cashtrans_value',␣
↪'NGO_trans', 'gov_trans', 'remittances',
                    'hh_area_plots', 'hh_rent_per_acre', 'hh_value_plots',␣
↪'hh_rentout_plots' ,'labor_N','hhlabor_N','hired_N', 'labor_h',
                    'hh_labor_hours',␣
↪'interm','value_fertilizer','kg_fertilizer', 'fertilizerkg', 'shocks',␣
↪'shock_flood','shock_drought','shock_lndslide','shock_covid',
                   ␣
↪'shock_adultill','shock_kidill','shock_death_earner','shock_death_othermemb','shock_inp_p','s
                    'y_cassava',  'y_soyabean', 'y_sorghum',  'y_fingermillet',␣
↪'y_cotton', 'y_tanaposi', 'y_groundbean', 'y_nkhwani', 'y_sugarcane',␣
↪'y_sweetpotatoe',
                   ␣
↪'wtotal','housing','hh_assets','hh_value_plots','k_farm','hhlivestock']]


## data with income at the year level ----------------------------------

data_short_year =  data_short
data_short_year[['wlabor_inc', 'ganyu_inc','business_revenue',␣
↪'business_profits1','business_profits2', 'other_inc', 'cashtrans_value',␣
↪'NGO_trans', 'gov_trans', 'remittances']] =  data_short_year[['wlabor_inc',␣
↪'ganyu_inc','business_revenue', 'business_profits1','business_profits2', ␣
↪'other_inc', 'cashtrans_value', 'NGO_trans', 'gov_trans', 'remittances']]*(12/
↪7)


## inctotal using agric revenues not profits
data_short_year['inctotal'] = data_short_year[['y_agric' ,'wlabor_inc',␣
↪'ganyu_inc', 'business_profits1']].sum(axis=1)
data_short_year['inctotal_trans'] = data_short_year[['y_agric' ,'wlabor_inc',␣
↪'ganyu_inc', 'business_profits1', 'other_inc']].sum(axis=1)
```

```
# summary

income =␣
 ↪data_short_year[['hhid','inctotal','inctotal_trans','y_net','y_agric','y_maize',␣
 ↪'y_groundnut', 'wlabor_inc', 'ganyu_inc', 'business_profits1', 'other_inc']].
 ↪replace(0,np.nan)
sum_inc = (income.loc[:, income.columns != 'hhid']/dollar_MWK).
 ↪describe(percentiles=[0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 0.99])
var_list = ['inctotal','inctotal_trans', 'y_net', 'y_agric','y_maize',␣
 ↪'y_groundnut', 'wlabor_inc', 'ganyu_inc', 'business_profits1', 'other_inc']
gini_stat= np.empty((1, len(var_list)))

for i,state in enumerate(var_list):
    gini_stat[:,i] = gini(income[state].dropna().values)

data_gini = pd.DataFrame(gini_stat, columns=var_list)
data_gini.reset_index(inplace=True)
data_gini['index'] = 'gini'
sum_inc.reset_index(inplace=True)
sum_inc = sum_inc.append(data_gini, ignore_index=True)

print('================================================================')
print('Summary total Income (year level)')
print('================================================================')
print('values in $')
print(sum_inc)


'''

data_weird = data_short.loc[data_short['y_net']<0, ['y_net','y_agric','y_maize',␣
 ↪'total_kg_maize' ,'hh_area_plots','interm','value_fertilizer','kg_fertilizer',␣
 ↪'fertilizerkg']]


data_crops = data[['hhid','hh_area_plots','inctotal','y_agric','total_kg_maize',␣
 ↪'total_kg_groundnut', 'total_kg_groundbean', 'total_kg_sweetpotatoe',␣
 ↪'total_kg_fingermillet', 'total_kg_sorghum', 'total_kg_pearlmillet',␣
 ↪'total_kg_soyabean', 'total_kg_pigeonpeas', 'total_kg_cotton',␣
 ↪'total_kg_nkhwani', 'total_kg_cassava',  'total_kg_sugarcane', ␣
 ↪'total_kg_tomatoes',  'total_kg_therereokra', 'total_kg_tanaposi',
             'y_maize', 'y_groundnut', 'y_groundbean', 'y_sweetpotatoe', ␣
 ↪'y_fingermillet', 'y_sorghum', 'y_soyabean',  'y_pigeonpeas', 'y_cotton',␣
 ↪'y_nkhwani', 'y_cassava',  'y_sugarcane',  'y_therereokra', 'y_tanaposi', ␣
 ↪'wlabor_inc', 'ganyu_inc']]
```

```
#data_crops.to_csv('C:/Users/rodri/Dropbox/Chied_Field_June_19/Data/Finished␣
↪Dataframes/income_sources.csv', index=False)

#data_weird.to_csv('outputs/neg_netoutput_19.csv')

'''
if save==True:
    data.to_csv('income_wealth_22_LONG_rainseas.csv')
    data_short.to_csv('income_wealth_22_rainseas.csv', index=False)
    data_short_year.to_csv('income_wealth_22_year.csv', index=False)



### y_net is agricultucal net income (minus intermediates). (MWK)
### y_agric is gross agricultural income (MWK)
### Labor variables: N denotes unit is number of persons. labor_h, denotes total␣
↪labor input (hh+hired) in hours.
### Shock variables: whether households reported the shock or not.
### I trim the variables: 'interm','labor_h', 'k_farm', 'hh_area_plots' at the␣
↪1% both tails to remove outliers. For example, in land area the 99% was around␣
↪20 acres. The maximum more than 100 acres...
```

```
================================================================
Summary total Income (year level)
================================================================
values in $
    index  inctotal  inctotal_trans    y_net  y_agric  y_maize  y_groundnut  \
0   count    268.00          271.00   254.00   254.00   253.00        97.00
1    mean    297.32          359.51    93.02    96.35    71.69        38.79
2     std    438.09          460.06   106.75   110.01    90.29        54.33
3     min      5.29            5.29   -29.65     2.43     1.44         0.97
4      1%      9.48           14.41     2.77     4.16     3.63         0.97
5     10%     32.21           45.80    14.84    16.83    14.41         4.85
6     25%     64.74           96.00    36.10    37.36    28.82         4.85
7     50%    146.79          198.35    62.95    66.46    43.22        19.41
8     75%    364.19          452.24   108.88   114.70    86.45        48.53
9     90%    619.18          851.23   178.48   181.24   144.08        97.05
10    99%  1,891.40        1,895.60   476.94   484.47   374.04       293.10
11    max  4,273.67        4,273.67 1,039.36 1,087.89 1,008.57       339.69
12   gini      0.59            0.55     0.72     0.48     0.48         0.58

    wlabor_inc  ganyu_inc  business_profits1  other_inc
0         8.00     103.00              63.00     131.00
1       543.74     324.61             276.59     135.46
2       445.73     413.35             610.21     227.71
3         0.01       0.01               4.85       3.33
```

4	8.16	1.23	6.66	3.33
5	81.53	41.93	14.56	9.98
6	238.75	87.35	29.12	16.64
7	448.39	174.70	116.46	58.23
8	837.09	349.39	276.60	135.60
9	1,164.64	857.18	481.39	332.75
10	1,164.64	1,861.56	3,065.41	1,148.00
11	1,164.64	1,863.43	4,266.47	1,331.02
12	0.43	0.57	0.68	0.66

[ ]:

```
```