In [1]:

```
# ============================================================================
#  transfers July 2023
# ============================================================================

import numpy as np
import pandas as pd
import os
os.chdir('C:/Users/rodri/Dropbox/Malawi/SIEG2021 (1)/2023 July/Data/Clean data/Phase
percentiles = [0.05, 0.1, .25, .5, .75, 0.8, 0.9, 0.95, 0.99]
#July 14th 2022 MWK vs US dollar
dollar_MWK = 1030.36
pd.options.display.float_format = '{:,.2f}'.format

import warnings
warnings.filterwarnings("ignore")

# ============================================================================
# Import data: Data from the field and conversion rates (ISA-LSMS price conversions)
# ============================================================================


#  I followed Pau's code in 2019 to generate a dataset containing all the food trans
data = pd.read_stata("transfers.dta")


# Now, let's use this data to
# (1) convert the transfers to kgs and monetary units.
# (2) Provide summary stats of the transfers.
# (3) Create food transfers datasets
# (4) Create measures of total transfers_in transfers_out at household level to appe
#inreturn_sum = pd.value_counts(data['in.return'])



# (1) convert the transfers to kgs and monetary units. ----------------------------
data.replace('cassava', 'cassavatubers', inplace=True)
data.replace('potatocrips', 'potatocrisps', inplace=True)
#data.replace('cowpea', 'cowpeas', inplace=True)

list_items = list(data['good'].unique())
list_units = list(data['units'].unique())

## Other units
# hands/hand/handful conversion rate
data.loc[data['unit.other'].isin(['Hand','Hands']), 'transf_kg'] = 0.113 ##0.5 cup/
# Dove
data.loc[data['unit.other'].isin(['Dove']), 'transf_kg'] = 0.3  ## average weight of
# Pot
data.loc[data['unit.other'].isin(['Pot']), 'transf_kg'] = 0.5  ##from previous year
data_other = data.loc[data['units']=='other']

print(' ')
print(' TRANSFERS DATA')
print(' ')

# Leandro-Raul kgs through price method

### NOTE ON CONVERSIONS ================================
# Using ISA-LSMS 17 I didnt have crop-units conversions for several units. What I ma
# 1. Check if missing units are the ones from upper numbers (above 25)
# 2. Use conversion units from the production side for the crop-units possible: pail
# 3. Use conversion units from the consumption side of an older ISA-LSMS (15): bale,
conversionkg_pivot = pd.read_csv('C:/Users/rodri/Dropbox/Malawi/Chied_Field_June_19/

#4.  All units have at least one crop conversion. To fill the whole matrix I use the
conversionkg_pivot = conversionkg_pivot.apply(lambda x: x.fillna(x.median()),axis=1)

## import median price of goods
prices = pd.read_csv('C:/Users/rodri/Dropbox/Malawi/SIEG2021 (1)/2023 July/Data/Clea

# there was not consumption of samosas in the village (not possible to compute price
# so I'll use the price for mandazidou
conversionkg_pivot['samosa'] = conversionkg_pivot['mandazidou']
prices.loc[40] =['samosa',1651.98]
prices.loc[41] =['chips',       1174.4]

# I don't have conversion rates and  price for cowpeas. I will use pigeon peas which
conversionkg_pivot['cowpea'] = conversionkg_pivot['pigeonpea']
prices.loc[42] =['cowpea', 575]


# chips/crisps
conversionkg_pivot['chips'] = conversionkg_pivot['potatocrisps']

data['units'] = data['units'].replace('other',100)
data['units'] = pd.to_numeric(data['units'])
data['transf_kg'] = np.nan
data[['units']] = data[['units']].replace([np.nan, 23, 25, 0], 99)


## Convert to kgs
for i in range(0,len(data)):
    good = data['good'][i]
    if (good != 'fert') & (good != 'ganyu'):
        data['transf_kg'][i] = data['quant'][i]*conversionkg_pivot.loc[int(data['uni


data.loc[data['good']=='fert','transf_kg'] = data.loc[data['good']=='fert','kg']

## Convert to MWK

# get a price from fertilizer for those that had to pay
print('for the food items I use the median consumption prices in the village')

print('for fertilizer I used the median price reported in the agricultural productio

p_fert =  55000/50   # mean price 50kg fertilizer bag 14620.36656 this is 2022 price
new_row = {'good': 'fert', 'p_c': p_fert}
prices = prices.append(new_row, ignore_index=True)

data = data.merge(prices, on='good', how='left')
data['transf_MWK'] = data['transf_kg']*data['p_c']
data['transf_dollar'] = data['transf_MWK']/dollar_MWK

data['ganyu.cash_dollar'] = data['ganyu.cash']/dollar_MWK



# for the moment dont remove outliers. Now we have fertilizer so outliers not so cle
'''
def fun(x):
    q_99 = x.quantile(0.95)
    q_1 = x.quantile(0.00)
    return (x>q_99) | (x<q_1)

print('Summary transfers in the village')
print(data['transf_dollar'].describe(percentiles=[0.25, .5, .75, 0.95, 0.99]))

data['outlier'] = 1*(data['transf_dollar']>data['transf_dollar'].quantile(0.995))
data = data.loc[data['outlier']==0]

data = data.drop(columns='outlier')
'''
print('===============================================================')
print('Summary transfers in the village')
print('===============================================================')
print(data['transf_dollar'].describe(percentiles=[0.1,.5,0.9]))


N_fert = len(data.loc[data['good']=='fert','good'])
N_ganyu = len(data.loc[data['good']=='ganyu','good'])
N_food = len(data['good'])-N_fert-N_ganyu

print('  ')
print('===============================================================')
print(' Number of transfers')
print('===============================================================')
print('# food transfers:',N_food)
print('# fertilizer transfers:',N_fert)
print('# ganyu transfers:',N_ganyu)
print('===============================================================')
print('Summary only FOOD transfers')
print('===============================================================')
print(data.loc[data['good']!='fert', ['transf_kg','transf_dollar']].describe(percent
print(pd.value_counts(data['good']))

print('===============================================================')
print('Summary only fertilizer')
print('===============================================================')
print(data.loc[data['good']=='fert', ['transf_kg','transf_dollar','recipro.cashback'


N_payfert = sum(data['recipro.cashback']>0)
print('Proportion fert transfers household had to pay back: ',round(N_payfert/N_fert
print('  ')
print('Comparison value given median price vs payback for those hhs that reported to
print(data.loc[(data['good']=='fert') & (data['recipro.cashback']>0), ['transf_kg','
print('NEED TO UPDATE FERTILIZER PRICE')

print('===============================================================')
print('Summary only ganyu')
print('===============================================================')

ganyu = data.loc[data['good']=='ganyu']
print(pd.value_counts(ganyu['ganyu.task']))
print(ganyu[['ganyu.cash','ganyu.cash_dollar','ganyu.days']].describe())

del data['kg']
# food transfers data in long format with values to kgs and MWK.
data.to_csv('transfers_in_kg_MWK_jul23.csv')

print('===============================================================')
print('SAVED transfers datasets pair-level with conversions to kgs and MWK: hhtransf

#%% Create household level transfers dataset =======================================


print('   ')
print(' TRANSFERS AGGREGATED AT HOUSEHOLD LEVEL (ONLY FOOD)')
```

```
all_count_bydirection = data[['direction','quant']].groupby(by='direction').count()
all_avg_bydirection = data[['direction','transf_kg','transf_MWK']].groupby(by='direc
print('===============================================================')
print('transfers in vs out: number and average value')
print(all_count_bydirection)
print(all_avg_bydirection)


# Only transfers that we could match in the village
data_match = data.loc[data['id']!=0]
count_bydirection = data_match[['direction','quant']].groupby(by='direction').count(
avg_bydirection = data_match[['direction','quant']].groupby(by='direction').mean()
print('===============================================================')
print('transfers in vs out MATCHED in the village:')
print(count_bydirection )

print('Almost all transfers were matched!')

data_match_out = data_match.loc[data_match['direction']=='out']
data_match_in = data_match.loc[data_match['direction']=='in']
data_match_in.rename(columns={'hhid':'id','id':'hhid'},inplace=True)


print('   ')
### Only transfers that we can cross-validate directions
data_directions_match = data_match_in.merge(data_match_out, on=['hhid','id'], how='i

### Only transfers that we can match cross-validate directions and the item
data_item_match = data_match_in.merge(data_match_out, on=['hhid','id','good'], how='

print('===============================================================')
print('Number of transfers cross-validated based on direction: coinciding hhid givin
print('num transf:',len(data_directions_match))
### Only transfers that we can match cross-validate directions and the item
data_item_match = data_match_in.merge(data_match_out, on=['hhid','id','good'], how='

print('===============================================================')
print('Number of transfers cross-validated based on direction and food item')
print('num transf:',len(data_item_match))
print('low number of matched transfers. Partly due to 7 days recall period on food')




### Import consumption dataset to create base ids
c23 = pd.read_csv('C:/Users/rodri/Dropbox/Malawi/SIEG2021 (1)/2023 July/Data/Clean d
data_final = c23[['hhid']]


# household level net transfers variables =========================

print('   ')
print(' TRANSFERS AGGREGATED AT HOUSEHOLD LEVEL (ONLY FOOD)')
print('   ')

# -----------

print('===============================================================')
print('household level net transfers variables')
print('===============================================================')
print('net transfers only includes food transfers. Reasons: (1) most fertlizer trans

data = data.loc[data['good']!='fert']
data_given = data.loc[data['direction']=='out',['hhid','transf_MWK']].groupby(by='hh
data_given.columns = ['transfers1_out']

data_received = data.loc[data['direction']=='in',['hhid','transf_MWK']].groupby(by='
data_received.columns = ['transfers1_in']

transfers1 = data_received.merge(data_given,on='hhid', how='outer')

transfers1['transfers1_net'] = transfers1['transfers1_in'].fillna(0) - transfers1['t
data_final = data_final.merge(transfers1, on='hhid', how='left')

data_item_match[['id','hhid','good','transf_kg_x','transf_MWK_x','transf_kg_y','tran

data_item_match['transf_MWK_avg'] = data_item_match[['transf_MWK_x','transf_MWK_y']]

data_given = data_item_match[['id','transf_MWK_avg']].groupby(by='id').sum()
data_given.reset_index(inplace=True)
data_given.columns = ['hhid','transfers2_in']

data_received = data_item_match[['hhid','transf_MWK_avg']].groupby(by='hhid').sum()
data_received.reset_index(inplace=True)
data_received.columns = ['hhid','transfers2_out']

transfers2 = data_given.merge(data_received,on='hhid', how='outer')
transfers2['transfers2_net'] = transfers2['transfers2_in'].fillna(0) - transfers2['t


data_final = data_final.merge(transfers2, on='hhid', how='left')

transfers3 = data.merge(data_item_match[['hhid','id','good','transf_MWK_avg']],on=['
transfers3 = transfers3[transfers3['transf_MWK_avg'].isnull()]
# we lose 130 observations. Exactly the number of transfers we chould cross-validate

data_given1 = transfers3.loc[transfers3['direction']=='out',['hhid','transf_MWK']]
data_given2 = transfers3.loc[transfers3['direction']=='in',['id','transf_MWK']]
data_given2.columns = ['hhid','transf_MWK']

data_given= data_given1.append(data_given2)
data_given= data_given.groupby(by='hhid').sum()
data_given.columns = ['transfers3_out']

data_received1 = transfers3.loc[transfers3['direction']=='in',['hhid','transf_MWK']]
data_received2 = transfers3.loc[transfers3['direction']=='out',['id','transf_MWK']]
data_received2.columns = ['hhid','transf_MWK']

data_received= data_received1.append(data_received2)
data_received= data_received.groupby(by='hhid').sum()
data_received.columns = ['transfers3_in']

transfers3 = data_received.merge(data_given,on='hhid', how='outer')
transfers3['transfers3_net'] = transfers3['transfers3_in'].fillna(0) - transfers3['t



data_final = data_final.merge(transfers3, on='hhid', how='left')
data_final.set_index('hhid', inplace=True)
data_final = data_final[['transfers1_net','transfers2_net','transfers3_net']]
data_final_year = data_final*4*12


data_final.to_csv('hhtransfers_week_jul23.csv')
data_final_year.to_csv('hhtransfers_year_jul23.csv')
```

```
data_final_dollars = data_final/dollar_MWK
print('===============================================================')
print(data_final_dollars.describe(percentiles=[0.05,0.25,0.5,0.75,0.9,0.95]))

print('     ')
print('variable 1: tranfers1_net (no exploit network). Take only what households rep

print('variable 2: tranfers2_net ----- (restrictive). Only those transfers that we c

print('given the problem of the big span of time across surveys, I would not use thi

print('variable 3: tranfers3_net ---- (extensive). what household X reports to recei

print('===============================================================')
print('saved datasets net food transfers hh level: hhtransfers_week.csv, hhtransfers
```

```
 TRANSFERS DATA

for the food items I use the median consumption prices in the village
for fertilizer I used the median price reported in the agricultural production secti
on. One could also use the payback from the transfer.
===============================================================
Summary transfers in the village
===============================================================
count   2,206.00
mean        2.00
std        16.26
min         0.00
10%         0.07
50%         0.33
90%         1.94
max       330.34
Name: transf_dollar, dtype: float64

===============================================================
 Number of transfers
===============================================================
# food transfers: 2114
# fertilizer transfers: 93
# ganyu transfers: 94
===============================================================
Summary only FOOD transfers
===============================================================
       transf_kg  transf_dollar
count   2,113.00       2,113.00
mean        1.90           1.40
std        14.61          16.07
min         0.00           0.00
10%         0.10           0.07
50%         0.50           0.29
90%         2.53           1.20
max       297.83         330.34
wsweetpotatoes     206
salt               195
thobwa             161
groundnut          150
pigeonpea          127
maizemgaiwa        101
tomato              99
ganyu               94
fert                93
maizerefined        83
leafyvegetables     82
tanaposi            75
cowpea              67
maizemadeya         63
sugarcane           63
```

```
groundnutf          61
osweetpotatoes      56
banana              51
goat                42
smockedfish         42
guava               41
chicken             37
rice                35
driedfish           29
cassavatubers       27
sugar               24
onion               22
fleshfish           19
wildfruits          19
maizegrain          18
cookingoil          17
cabbage             17
mandazidou          15
bbean               15
eggs                13
greenmaize          12
fingermillet         9
otherpoultry         6
samosa               5
ipotatoes            4
softdrinks           4
potatocrisps         1
chips                1
Name: good, dtype: int64
===============================================================
Summary only fertilizer
===============================================================
       transf_kg  transf_dollar  recipro.cashback
count      93.00          93.00             45.00
mean       14.67          15.66         14,733.33
std        13.61          14.53         14,810.78
min         1.00           1.07          3,000.00
10%         5.00           5.34          5,800.00
50%        10.00          10.68          9,000.00
90%        25.00          26.69         28,200.00
max       100.00         106.76         90,000.00
Proportion fert transfers household had to pay back:  0.48

Comparison value given median price vs payback for those hhs that reported to pay ba
ck for fertilizer transfer
       transf_kg  transf_MWK  recipro.cashback
count      45.00       45.00             45.00
mean       20.16   22,171.11         14,733.33
std        16.56   18,217.23         14,810.78
min         5.00    5,500.00          3,000.00
10%         5.80    6,380.00          5,800.00
50%        15.00   16,500.00          9,000.00
90%        28.00   30,800.00         28,200.00
max       100.00  110,000.00         90,000.00
NEED TO UPDATE FERTILIZER PRICE
===============================================================
Summary only ganyu
===============================================================
Farm ganyu.                                                  52
Other ganyu specify.Brick laying                              6
Other ganyu specify.Moulding bricks                           4
Other ganyu specify.Cleaning the house                        2
Other ganyu specify.Digging a pit latrine                     2
Other ganyu specify.Building a house                          1
Other ganyu specify.Carrying  dambo sand                      1
Other ganyu specify.Building of the house                     1
Other ganyu specify.Building of a house                       1
Other ganyu specify.Building a kitchen                        1
Other ganyu specify.Building  a bathroom                      1
Other ganyu specify.Building a bathroom                      1
Other ganyu specify.Carrying Sand                            1
Other ganyu specify.Bricklaying                              1
Other ganyu specify.Bicycle taxi                             1
Other ganyu specify.Bathroom construction                    1
Other ganyu specify.Carrying sand                            1
Other ganyu specify.Uvuni                                    1
Other ganyu specify.Carrying sand to a house construction    1
Other ganyu specify.Supplying water for moulding bricks      1
Other ganyu specify.Cutting firewood                         1
Other ganyu specify.Cutting glass                            1
Other ganyu specify.Drawing water                            1
Other ganyu specify.Fetching water                           1
Other ganyu specify.Harvesting groundnut                     1
Other ganyu specify.House painting                           1
Other ganyu specify.House smearing                           1
Other ganyu specify.MASAF                                    1
Other ganyu specify.Motor maintanance                        1
Other ganyu specify.Painting a house                         1
Other ganyu specify.Providing sand to a construction         1
Other ganyu specify.Providing water for brick moulding       1
                                                             1
Name: ganyu.task, dtype: int64
       ganyu.cash  ganyu.cash_dollar  ganyu.days
count       94.00              94.00       94.00
mean    13,201.81              12.81        7.41
std     16,679.62              16.19        8.48
min         20.00               0.02        1.00
25%      2,500.00               2.43        2.00
50%      5,000.00               4.85        3.00
75%     19,000.00              18.44        8.00
max     80,000.00              77.64       30.00
===============================================================
SAVED transfers datasets pair-level with conversions to kgs and MWK: hhtransfers_wee
k.csv, hhtransfers_year.csv

 TRANSFERS AGGREGATED AT HOUSEHOLD LEVEL (ONLY FOOD)
===============================================================
transfers in vs out: number and average value
           quant
direction
in           790
out         1324
           transf_kg  transf_MWK
direction
in              3.00    2,491.88
out             2.08    1,797.10
===============================================================
transfers in vs out MATCHED in the village:
           quant
direction
in           769
out         1261
Almost all transfers were matched!

===============================================================
Number of transfers cross-validated based on direction: coinciding hhid giving with
hhid from who you received---and equvilently for receiving.
num transf: 968
===============================================================
Number of transfers cross-validated based on direction and food item
num transf: 105
low number of matched transfers. Partly due to 7 days recall period on food

 TRANSFERS AGGREGATED AT HOUSEHOLD LEVEL (ONLY FOOD)

===============================================================
household level net transfers variables
===============================================================
net transfers only includes food transfers. Reasons: (1) most fertlizer transfers hh
actually payed (2) consistent with previous data (3) timing is different
===============================================================
       transfers1_net  transfers2_net  transfers3_net
count          270.00          111.00          281.00
mean            -1.71           -0.00           -1.33
std             45.34            5.59           59.65
min           -333.60          -25.34         -333.86
5%             -11.75          -12.15          -11.80
25%             -1.71           -0.39           -2.20
50%             -0.09            0.07            0.14
75%              1.41            0.62            2.24
90%              3.81            5.34            5.60
95%              7.22            8.61           10.27
max            331.02           18.68          332.04

variable 1: tranfers1_net (no exploit network). Take only what households report to
give and receive (not info about what other households say)
variable 2: tranfers2_net ----- (restrictive). Only those transfers that we could cr
oss-validate---X reports giving food item to Y, coincides with Y receiving food item
from X. _x variables denote from direction in. _y variables denote from direction ou
t.
given the problem of the big span of time across surveys, I would not use this measu
re to measure the transfers.
variable 3: tranfers3_net ---- (extensive). what household X reports to receive plus
what rest of households report to give to X minus what household X reports to give p
lus what rest of households report to receive from X. Extensive method.,  First, to
avoid double-counting, eliminate the transfers that we could cross-validate
===============================================================
saved datasets net food transfers hh level: hhtransfers_week.csv, hhtransfers_year.c
sv
```

In [ ]: