In [ ]:

```
# ==============================================================================
# CONSUMPTION 18: MALAWI VILLAGE
# ==============================================================================

import numpy as np
import pandas as pd
import os
os.chdir('C:/Users/rodri/Dropbox/JMP/python')
from data_functions_albert import remove_outliers, gini
os.chdir('C:/Users/rodri/Dropbox/Malawi/SIEG2021 (1)/2018 July/Data/consumption/')
percentiles = [0.05, 0.1, .25, .5, .75, 0.8, 0.9, 0.95, 0.99]
dollar_MWK = 745.54


import warnings
warnings.filterwarnings("ignore")

###  I NEEED TO RECHECK WITH USING SAME LOOPING AS IN THE 2019 WAVE.

#Import data
data = pd.read_csv("raw/data_SIEG_clean.csv")

x0 = data.columns.get_loc("consume_1")
x1 = data.columns.get_loc("salt_unit9")
datacon = data

list_items = ['maizemgaiwa', 'maizerefined', 'maizemadeya', 'maizegrain', 'greenmaiz
, 'ipotatoes', 'potatocrisps', 'bbean', 'pigeonpea', 'groundnut', 'groundnutf', 'oni
'driedfish', 'fleshfish', 'goat', 'chicken', 'otherpoultry', 'smockedfish', 'mango',
'wildfruits',  'sugar', 'sugarcane', 'cookingoil', 'softdrinks',
'thobwa', 'locallybrewed', 'salt']


#Obtain the names of the variables per each question of item.
#Question c is monetary question so not conversion to kgs needed
a_var = []
b_var = []
d_var = []
e_var = []
f_var = []
g_var = []
h_var = []
i_var = []
j_var = []

#Generate variable lables in a list
for item in list_items :
    a = item+'_a'
    b= item+'_b'
    d = item+'_d'
    e = item+'_e'
    f = item+'_f'
    g = item+'_g'
    h = item+'_h'
    i = item+'_i'
    j = item+'_j'
    a_var.append(a)
    b_var.append(b)
    d_var.append(d)
    e_var.append(e)
    f_var.append(f)
    g_var.append(g)
```

```
    h_var.append(h)
    i_var.append(i)
    j_var.append(j)


list_questions = ['a','b','d','e','f','g','h','i','j']

# Generate kg variables empty
for item in list_items:
    for q in list_questions:
        datacon[item+'_'+q+'kg']= np.nan

# IN UNITS QUESTIONS: Drop nan observations. Also drop unit 25 (number not in our ch
for var in list_items:
    for i in range(1,9):
        datacon[[var+'_unit'+str(i)]] = datacon[[var+'_unit'+str(i)]].replace(np.nan
        datacon[[var+'_unit'+str(i)]] = datacon[[var+'_unit'+str(i)]].replace(25, 99
        datacon[[var+'_unit'+str(i)]] = datacon[[var+'_unit'+str(i)]].replace(24, 99
        datacon[[var+'_unit'+str(i)]] = datacon[[var+'_unit'+str(i)]].replace(23, 99
## For salt we don't have question of units for given-out ganyu:
for var in list_items[0:-1]:
    datacon[[var+'_unit9']] = datacon[[var+'_unit9']].replace(np.nan, 99)
    datacon[[var+'_unit9']] = datacon[[var+'_unit9']].replace([23,24,25], 99)



datacon[['salt_unit9']] =99

'''
##### THIS CODE IS TO GENERATE THE MATRIX OF ITEMS/UNITS COMBINATIONS TO CONVERT TO
data_conversion = pd.read_excel('conversionkgrates_consumption_malawi_short.xls')

#del data, datasets, data0, data1, data2, data3, data4, x0, x1

items_conv = data_conversion.item_label.unique()


list_items_conv = ['Maize Mgaiwa', 'Maize refined', 'Maize Madeya', 'Maize grain',
       'Greenmaize', 'Rice', 'Cassava tubers',
       'White sweet potatoes', 'Orange sweet potatoes', 'Irish Potatoe',
       'Potatoe Chips', 'Bean, brown', 'Pigeon peas (ndolo)', 'Groundnut',
       'Groundnut flour', 'Onion', 'Cabbage', 'Tanaposi/Rape',
       'Other cultivated green leafy vegetables', 'Tomatoe', 'Eggs',
       'Dried Fish', 'Fresh Fish', 'Goat', 'Chicken',
       'Other poultry-guinea fowl, doves, etc', 'Smoked Fish', 'Mango',
       'Banana', 'Guava', 'Wild fruit (Masau, Malambe, etc)', 'Sugar',
       'Sugar Cane', 'Cooking oil',
       'Soft drinks(coca-cola, fanta, sprite etc)', 'Thobwa',
       'Locally brewed liquor(kachasu)', 'Salt']


#Obtain the names of the variables per each question of item. Question c is monetary
a_var = []
b_var = []
d_var = []
e_var = []
f_var = []
g_var = []
h_var = []
i_var = []
j_var = []
#Generate variable lables in a list
for item in list_items :
    a = item+'_a'
```

```
    b= item+'_b'
    d = item+'_d'
    e = item+'_e'
    f = item+'_f'
    g = item+'_g'
    h = item+'_h'
    i = item+'_i'
    j = item+'_j'
    a_var.append(a)
    b_var.append(b)
    d_var.append(d)
    e_var.append(e)
    f_var.append(f)
    g_var.append(g)
    h_var.append(h)
    i_var.append(i)
    j_var.append(j)


list_questions = ['a','b','d','e','f','g','h','i','j']

# Generate kg variables empty
for item in list_items:
    for q in list_questions:
        datacon[item+'_'+q+'kg']= np.nan

# Drop nan observations. Also drop unit 25 (number not in our choices). Also drop 24
for var in list_items:
    for i in range(1,9):
        datacon[[var+'_unit'+str(i)]] = datacon[[var+'_unit'+str(i)]].replace(np.nan
        datacon[[var+'_unit'+str(i)]] = datacon[[var+'_unit'+str(i)]].replace(25, 99
        datacon[[var+'_unit'+str(i)]] = datacon[[var+'_unit'+str(i)]].replace(24, 99

#
# Import conversion files
conversionkg = data_conversion[['unit','item_label','conversion_kgs_country']]


# Reshape as: rows:units, columns:crops

conversionkg = conversionkg.replace(list_items_conv, list_items)

conversionkg_pivot = conversionkg.pivot_table(values='conversion_kgs_country',
                                index='unit',
                                columns='item_label')


conversionkg_pivot.loc[99,:] = np.nan

conversionkg_pivot.to_csv('conversionkg_isaprices_matrix.csv')
'''

### 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('conversionkg_final_matrix.csv',  index_col=0)

#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)

#%% CONVERT TO KGS FOR ALL QUESTIONS FOR ALL ITEMS
```

```
print('a: Total Consumption')
for var in a_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c
    print(datacon[[var+'kg']].describe())

print('b: Bought')
for var in b_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c

print('d: Own-produced')
for var in d_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c

print('e: Gift-in')
for var in e_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c

print('f: Gift-in for free')
for var in f_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c

print('g: Gift-in for ganyu')
for var in g_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c

print('h: Gift-out')
for var in h_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c

print('i: Food giften out for free')
for var in i_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c

print('j: Food giften out for ganyu')
for var in j_var:
    item = var[:-2]
    for i in range(len(datacon)):
        datacon.iloc[i,datacon.columns.get_loc(var+'kg')] = datacon.iloc[i,datacon.c




# Replace extreme values for median (let's be careful with this)
# WE MIGHT WANT TO CHANGE THESE CORRECTIONS OF EXTREME VALUES
data.loc[data['sugar_akg']>5,['sugar_akg']] = data['sugar_akg'].median()
data.loc[data['thobwa_akg']>20,['thobwa_akg']] = data['thobwa_akg'].median()
data.loc[data['rice_akg']>20,['rice_akg']] = data['rice_akg'].median()
data.loc[data['tomato_akg']>10,['tomato_akg']] = data['tomato_akg'].median()
```

```
data.loc[data['tanaposi_akg']>10,['tanaposi_akg']] = data['tomato_akg'].median()
data.loc[data['mango_akg']>10,['mango_akg']] = data['mango_akg'].median()
data.loc[data['eggs_akg']>10,['eggs_akg']] = data['eggs_akg'].median()

data.loc[data['maizemgaiwa_akg']>60,['maizemgaiwa_akg']] = data['maizemgaiwa_akg'].m
data.loc[data['maizerefined_akg']>60,['maizerefined_akg']] = data['maizerefined_akg'
data.loc[data['maizegrain_akg']>30,['maizegrain_akg']] = data['maizegrain_akg'].medi
data.loc[data['wsweetpotatoes_akg']>30,['wsweetpotatoes_akg']] = data['wsweetpotatoe
data.loc[data['bbean_akg']>10,['bbean_akg']] = data['bbean_akg'].median()
data.loc[data['fleshfish_akg']>10,['fleshfish_akg']] = data['fleshfish_akg'].median(
data.loc[data['banana_akg']>10,['banana_akg']] = data['banana_akg'].median()
data.loc[data['sugarcane_akg']>10,['sugarcane_akg']] = data['sugarcane_akg'].median(
data.loc[data['cookingoil_akg']>10,['cookingoil_akg']] = data['cookingoil_akg'].medi


data.loc[data['sugar_bkg']>5,['sugar_bkg']] = data['sugar_bkg'].median()
data.loc[data['thobwa_bkg']>20,['thobwa_bkg']] = data['thobwa_bkg'].median()
data.loc[data['rice_bkg']>20,['rice_bkg']] = data['rice_bkg'].median()
data.loc[data['tomato_bkg']>10,['tomato_bkg']] = data['tomato_bkg'].median()
data.loc[data['tanaposi_bkg']>10,['tanaposi_bkg']] = data['tomato_bkg'].median()
data.loc[data['mango_bkg']>10,['mango_bkg']] = data['mango_bkg'].median()
data.loc[data['eggs_bkg']>10,['eggs_bkg']] = data['eggs_bkg'].median()

data.loc[data['maizemgaiwa_bkg']>60,['maizemgaiwa_bkg']] = data['maizemgaiwa_bkg'].m
data.loc[data['maizerefined_bkg']>60,['maizerefined_bkg']] = data['maizerefined_bkg'
data.loc[data['maizegrain_bkg']>30,['maizegrain_bkg']] = data['maizegrain_bkg'].medi
data.loc[data['wsweetpotatoes_bkg']>30,['wsweetpotatoes_bkg']] = data['wsweetpotatoe
data.loc[data['bbean_bkg']>10,['bbean_bkg']] = data['bbean_bkg'].median()
data.loc[data['fleshfish_bkg']>10,['fleshfish_bkg']] = data['fleshfish_bkg'].median(
data.loc[data['banana_bkg']>10,['banana_bkg']] = data['banana_bkg'].median()
data.loc[data['sugarcane_bkg']>10,['sugarcane_bkg']] = data['sugarcane_bkg'].median(
data.loc[data['cookingoil_bkg']>10,['cookingoil_bkg']] = data['cookingoil_bkg'].medi



print('================================================================')
print('a: Total Consumption (in kg) after corrections (cleaned)')
print('================================================================')

for var in a_var:
    item = var[:-2]
    print(data[[var+'kg']].describe())




#%% Check conversions for beans (as example)
print(datacon[['bbean_a', 'bbean_unit1', 'bbean_akg','bbean_b', 'bbean_bkg', 'bbean_


#%% Checks

# Sum the subset questions
for item in list_items:
    datacon[item+'_akg_check'] = datacon[item+'_bkg'] + datacon[item+'_dkg'] + datac
    datacon[item+'_ekg_check'] = datacon[item+'_fkg'] + datacon[item+'_gkg']
    datacon[item+'_hkg_check'] = datacon[item+'_ikg'] + datacon[item+'_jkg']


# Check the correlation of totals wrt sum of subset.
corr_total = []
corr_gift = []
corr_given = []
for item in list_items:
```

```
    a = datacon[item+'_akg'].corr(datacon[item+'_akg_check'])
    b = datacon[item+'_ekg'].corr(datacon[item+'_ekg_check'])
    c = datacon[item+'_hkg'].corr(datacon[item+'_hkg_check'])
    corr_total.append(a)
    corr_gift.append(b)
    corr_given.append(c)
    #print(datacon[[item+'_akg',item+'_akg_check']])
    #print(datacon[[item+'_ekg',item+'_ekg_check']])
    #print(datacon[[item+'_hkg',item+'_hkg_check']])




#%% Checks

# Sum the subset questions


#check total consumption
datacon['total_foodkg'] = 0
datacon['purchased_kg'] = 0
datacon['ownproduced_kg'] = 0
datacon['giftin_kg'] = 0
datacon['giftin_free_kg'] = 0
datacon['giftin_ganyu_kg'] = 0
datacon['giftout_kg'] = 0
datacon['giftout_free_kg'] = 0
datacon['giftout_ganyu_kg'] = 0


for item in list_items:
    datacon['total_foodkg'] += datacon[item+'_akg'].replace(np.nan, 0)
    datacon['purchased_kg'] += datacon[item+'_bkg'].replace(np.nan, 0)
    datacon['ownproduced_kg'] += datacon[item+'_dkg'].replace(np.nan, 0)
    datacon['giftin_kg'] += datacon[item+'_ekg'].replace(np.nan, 0)
    datacon['giftin_free_kg'] += datacon[item+'_fkg'].replace(np.nan, 0)
    datacon['giftin_ganyu_kg'] += datacon[item+'_gkg'].replace(np.nan, 0)
    datacon['giftout_kg'] += datacon[item+'_hkg'].replace(np.nan, 0)
    datacon['giftout_free_kg'] += datacon[item+'_ikg'].replace(np.nan, 0)
    datacon['giftout_ganyu_kg'] += datacon[item+'_jkg'].replace(np.nan, 0)

sumtotalfoodkg = datacon[['total_foodkg', 'purchased_kg','ownproduced_kg', 'giftin_k


print('==== Summary Food Consumption last 7 days in kgs aggregated across items ====
print('')
print(sumtotalfoodkg)
#%%  CONVERT TO MONETARY VALUE   ==========================================

# Get prices -------------------------
for item in list_items:
        datacon[item+'_price']= np.nan

## Compute price paid per each household each item
for item in list_items:
    datacon[item+'_price'] = datacon[item+'_c'] / datacon[item+'_bkg'].replace(0,np.

price_data = pd.DataFrame(list_items, columns=['good'])
price_data['p_c'] = np.nan

## Get median prices per item
for item in list_items:
    print('Median Price 1 kg of '+item)
    datacon['med_price_'+item] = datacon[item+'_price'].median()
```

```
    print(datacon['med_price_'+item].mean())
    price_data.loc[price_data['good']==item,'p_c'] = data['med_price_'+item].mean()

### mango doesnt have a price. Put the guava one:
price_data.loc[price_data['good']=='mango', 'p_c'] = 292.8

price_data.to_csv('prices/village_c_prices_18.csv', index=False)


## compute consumption in MWK -----------------
for item in list_items:
    for q in list_questions:
        datacon[item+'_'+q+'MWK']= np.nan

print('a: Total Consumption')
for item in list_items:
    datacon[item+'_aMWK'] = datacon[item+'_akg']*datacon['med_price_'+item]
    #print('Food Consumption in MWK during last 7 days item: '+item)
    #print(datacon[item+'_aMWK'].describe(percentiles=percentiles))

print('b: Bought')
for item in list_items:
    datacon[item+'_bMWK'] = datacon[item+'_bkg']*datacon['med_price_'+item]


print('d: Own-produced')
for item in list_items:
    datacon[item+'_dMWK'] = datacon[item+'_dkg']*datacon['med_price_'+item]


print('e: Gift-in')
for item in list_items:
    datacon[item+'_dMWK'] = datacon[item+'_dkg']*datacon['med_price_'+item]

print('f: Gift-in for free')
for item in list_items:
    datacon[item+'_fMWK'] = datacon[item+'_fkg']*datacon['med_price_'+item]

print('g: Gift-in for ganyu')
for item in list_items:
    datacon[item+'_gMWK'] = datacon[item+'_gkg']*datacon['med_price_'+item]


print('h: Gift-out')
for item in list_items:
    datacon[item+'_hMWK'] = datacon[item+'_hkg']*datacon['med_price_'+item]


print('i: Food giften out for free')
for item in list_items:
    datacon[item+'_iMWK'] = datacon[item+'_ikg']*datacon['med_price_'+item]


print('j: Food giften out for ganyu')
for item in list_items:
    datacon[item+'_jMWK'] = datacon[item+'_jkg']*datacon['med_price_'+item]

#check total consumption
datacon['c_food'] = 0
datacon['c_food_purch'] = 0
datacon['c_food_ownprod'] = 0
datacon['c_food_giftin'] = 0
datacon['c_food_giftin_free'] = 0
datacon['c_food_giftin_ganyu'] = 0
```

```
datacon['c_food_giftout'] = 0
datacon['c_food_giftout_free'] = 0
datacon['c_food_giftout_ganyu'] = 0

for item in list_items:
    datacon['c_food'] += datacon[item+'_aMWK'].replace(np.nan, 0)
    datacon['c_food_purch'] += datacon[item+'_bMWK'].replace(np.nan, 0)
    datacon['c_food_ownprod'] += datacon[item+'_dMWK'].replace(np.nan, 0)
    datacon['c_food_giftin'] += datacon[item+'_eMWK'].replace(np.nan, 0)
    datacon['c_food_giftin_free'] += datacon[item+'_fMWK'].replace(np.nan, 0)
    datacon['c_food_giftin_ganyu'] += datacon[item+'_gMWK'].replace(np.nan, 0)
    datacon['c_food_giftout'] += datacon[item+'_hMWK'].replace(np.nan, 0)
    datacon['c_food_giftout_free'] += datacon[item+'_iMWK'].replace(np.nan, 0)
    datacon['c_food_giftout_ganyu'] += datacon[item+'_jMWK'].replace(np.nan, 0)

datacon[['c_food', 'c_food_purch' , 'c_food_ownprod', 'c_food_giftin', 'c_food_gifti


sumcfood= ((datacon[['c_food', 'c_food_purch' , 'c_food_ownprod', 'c_food_giftin', '

print('==== Summary Food Consumption at Month level in Euros =======')
print(sumcfood)


#%% non-food consumption


datacon['c_housing'] = datacon['bills']
datacon['c_clothes'] = datacon['clothesothers']
datacon['c_education'] = datacon['educationothers']
datacon['c_health'] = datacon['healthothers']
datacon['c_funeralout'] = datacon['funerals1']
datacon['c_funeralin'] = datacon['funerals2']
datacon['c_weddingout'] = datacon['wedding1']
datacon['c_weddingin'] = datacon['wedding2']




#%%  Export dataset

## Convert to rainy season (7 months)

datacon[['c_food','c_food_purch', 'c_housing']] = remove_outliers(datacon[['c_food',

datacon['c_nonfood'] = datacon[['c_housing', 'c_clothes', 'c_education', 'c_health']

sum_cnonfood = ((data[['c_nonfood','c_housing', 'c_clothes', 'c_education', 'c_healt
print('======== SUMMARY  NON-FOOD CONSUMPTION (MONTH LEVEL)')
print('summary in MWK')
sum_cnonfood


for item in list_items:
    data['c_'+item+'_kg_7days'] = data[item+'_akg']
    data['c_'+item+'_MWK_7days'] = data[item+'_aMWK']




datacon_short = datacon[[ 'hhid','c_food','total_foodkg', 'c_food_purch' , 'c_food_o


## Food at monthly level
```

```
datacon_short[['c_food','total_foodkg', 'c_food_purch' , 'c_food_ownprod', 'c_food_g

datacon_short['ctotal'] = datacon_short[['c_nonfood', 'c_food']].sum(axis = 1, skipn

## Consumption at year level
datacon_short[['ctotal','c_food','total_foodkg','c_food_purch','c_food_ownprod',  'c

c_summary =  ((datacon_short[['ctotal','c_food','c_food_purch','c_food_ownprod', 'c_

print('======== SUMMARY CONSUMPTION (YEAR LEVEL)')
print('summary in dollars')
print(c_summary)


datacon_short.to_csv('cons_short_18.csv', index=False)

## Not at rainy season level
datacon.to_csv('cons_long_18.csv')
```

```
a: Total Consumption
       maizemgaiwa_akg
count       210.000000
mean          9.412158
std           7.539566
min           0.600000
25%           3.333333
50%           8.000000
75%          14.133334
max          38.400002
       maizerefined_akg
count        133.000000
mean           8.011927
std            6.962351
min            0.600000
25%            2.941176
50%            5.882353
75%           10.000000
max           50.000000
       maizemadeya_akg
count        79.000000
mean          2.629570
std           2.726511
min           0.353571
25%           1.000000
50%           1.800000
75%           2.863636
max          17.142857
       maizegrain_akg
count       63.000000
mean         0.853244
std          0.858336
min          0.176786
25%          0.353571
50%          0.500000
75%          0.750000
max          5.000000
       greenmaize_akg
count       13.000000
mean         0.597167
std          0.458442
min          0.000000
25%          0.336800
50%          0.353571
75%          0.673600
max          1.684000
        rice_akg
count  82.000000
mean    4.411643
```

```
std     6.756272
min     0.357143
25%     1.000000
50%     2.000000
75%     3.000000
max    40.000000
       cassavatubers_akg
count         157.000000
mean            1.201696
std             1.135047
min             0.157580
25%             0.315160
50%             0.945480
75%             1.575800
max             9.454801
       wsweetpotatoes_akg
count           81.000000
mean             3.169979
std              2.950510
min              0.252740
25%              1.263700
50%              2.527400
75%              3.791100
max             17.333334
       osweetpotatoes_akg
count           37.000000
mean             1.923323
std              1.365008
min              0.465150
25%              1.162875
50%              1.162875
75%              2.325750
max              6.977251
       ipotatoes_akg
count      24.000000
mean        1.593330
std         1.354183
min         0.000000
25%         0.625000
50%         1.333333
75%         3.000000
max         4.333334
       potatocrisps_akg
count         10.000000
mean           1.361646
std            1.524095
min            0.050000
25%            0.385872
50%            0.550000
75%            1.892157
max            4.000000
       bbean_akg
count  92.000000
mean    1.080443
std     0.922198
min     0.000000
25%     0.600000
50%     0.869565
75%     1.250000
max     6.521739
       pigeonpea_akg
count     241.000000
mean        2.170856
std         2.360676
min         0.250000
25%         0.800000
50%         1.200000
75%         3.000000
max        17.142857
```

```
       groundnut_akg
count     143.000000
mean        1.707350
std         2.542678
min         0.000000
25%         0.500000
50%         0.600000
75%         2.200000
max        25.000000
       groundnutf_akg
count      117.000000
mean         1.029778
std          1.914057
min          0.024000
25%          0.150376
50%          0.285600
75%          0.856800
max         13.000001
        onion_akg
count  172.000000
mean     0.332643
std      0.278675
min      0.081136
25%      0.162273
50%      0.256932
75%      0.405682
max      1.622727
       cabbage_akg
count    58.000000
mean      1.710956
std       1.054610
min       0.500000
25%       1.220316
50%       1.220316
75%       2.135553
max       6.101579
       tanaposi_akg
count    180.000000
mean       0.907577
std        0.632565
min        0.000000
25%        0.463778
50%        0.695667
75%        1.159444
max        4.869667
       leafyvegetables_akg
count            99.000000
mean              0.434929
std               0.251353
min               0.000000
25%               0.288500
50%               0.288500
75%               0.577000
max               1.442500
       tomato_akg
count  250.000000
mean     3.156306
std     19.365243
min      0.000000
25%      0.878581
50%      1.757161
75%      2.635742
max    307.503217
        eggs_akg
count  71.000000
mean    0.977256
std     5.645191
min     0.059815
25%     0.179444
```

```
50%     0.299074
75%     0.358889
max    47.851851
       driedfish_akg
count     168.000000
mean        1.007323
std         0.815433
min         0.000000
25%         0.671745
50%         0.695659
75%         1.391319
max         7.304424
       fleshfish_akg
count      95.000000
mean        1.159564
std         1.843168
min         0.347830
25%         0.647830
50%         0.851449
75%         1.043489
max        17.333334
        goat_akg
count  51.000000
mean    0.937132
std     0.789584
min     0.062500
25%     0.500000
50%     1.000000
75%     1.000000
max     4.000000
       chicken_akg
count    19.000000
mean      2.815351
std       1.599282
min       0.000000
25%       1.637500
50%       3.275000
75%       3.820833
max       6.550000
       otherpoultry_akg
count          8.000000
mean           4.007076
std            2.533552
min            1.187282
25%            2.374564
50%            3.561845
75%            4.749127
max            9.498254
       smockedfish_akg
count       123.000000
mean          0.887463
std           0.767238
min           0.000000
25%           0.333000
50%           0.666000
75%           0.999000
max           6.660000
        mango_akg
count  150.000000
mean     2.244868
std      4.343328
min      0.203314
25%      0.609941
50%      1.423196
75%      2.033137
max     50.000000
       banana_akg
count  109.000000
mean     2.291540
```

```
std      4.869619
min      0.168880
25%      1.144292
50%      1.144292
75%      2.288583
max     50.000000
       guava_akg
count   9.000000
mean    0.642674
std     0.953259
min     0.085381
25%     0.170762
50%     0.426905
75%     0.426905
max     3.137255
       wildfruits_akg
count       10.000000
mean         1.481358
std          1.366210
min          0.249833
25%          0.524750
50%          0.874417
75%          1.874750
max          4.000000
        sugar_akg
count  119.000000
mean    13.112549
std     59.880954
min      0.024000
25%      0.175000
50%      0.500000
75%      1.000000
max    297.826080
       sugarcane_akg
count      23.000000
mean        3.261506
std         2.427099
min         1.154071
25%         1.731107
50%         2.308143
75%         3.462214
max        11.540715
       cookingoil_akg
count      214.000000
mean         0.472218
std          0.367376
min          0.000000
25%          0.250000
50%          0.466667
75%          0.500000
max          3.000000
       softdrinks_akg
count       17.000000
mean         0.520798
std          0.454085
min          0.300000
25%          0.300000
50%          0.350000
75%          0.600000
max          2.200000
       thobwa_akg
count  184.000000
mean     5.479964
std      6.831654
min      0.300000
25%      1.000000
50%      2.914286
75%      5.000000
max     43.333335
```

```
       locallybrewed_akg
count           6.000000
mean            1.073958
std             1.068735
min             0.093750
25%             0.387500
50%             0.750000
75%             1.375000
max             3.000000
         salt_akg
count  266.000000
mean     0.312588
std      0.356112
min      0.000000
25%      0.110000
50%      0.220000
75%      0.400000
max      4.333334
b: Bought
d: Own-produced
e: Gift-in
f: Gift-in for free
```

In [ ]: