The Latin American Agriculture Sector as Related With a Country's Macroeconomy ¶

Daniel Fridman (df1647) and Samantha Ruilova Vallejo (srv261)

21 December 2018

Outline:

The World Bank has an immense compendium of economic data for over 300 nations, regions, and disputed states. There's a lot to be done with such data, and economists, data scientists, and hedge funds comb through the over 26,000,000 pieces of data. We have decided to join them in their pursuits.

This project analyzes the agricultural industry of thirty two countries in Latin America. Agriculture is of great importance to the economy overall. Especially in developing countries, agriculture has a strategic impotance in social and economic welfare. Oftentimes, the sector can contribute up to fifty percent to Gross National Income. However, the agricultural sector is often not given enough analytical attention. Thus, we decided to analyze this sector by assesing the relevant macroeconomic trends and country-specific factors that affect this industry across the region.

Specifically, we aim to evaluate any potential correlations between pure macroeconomic factors, such as interest rates, and agricultural indicators, such as percentage velue added. We start by finding these correlations on a country-level basis (e.g. what's the correlation between interest rates and value added in Antigua and Barbuda?), and then generalize our programming to be able to find the average of all correlation pairs in all countries. We will thus be able to compare all correlations at once, instead of fishing blindly for factors we believe *might* be correlated.

Generalization is a big theme here. The World Bank has so much data, and we aimed to create a program that doesn't just work for LatAm, or macro indicators, or agricultural indicators. We seek scalability, so that adding another country or another indicator constitututes nothing more than changing a couple of lines.

The effect of this is that we avoided hardcoding like the plague, and thus have *lots* of seemingly complicated lines of code. We tried to explain these as best as possible. Overall, we have less lines of code than comparable projects due to a liberal usage of loops. Most of our time spent on this project regarded figuring out the logic behind the methods.

Let's start off with some simple imports and definitions

In [2361]:
```
import sys
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import datetime as dt
import numpy as np
import seaborn as sns
import math as math
from scipy.stats.stats import pearsonr

%matplotlib inline
```

Import WDI data as dataframe:

In [2362]:
```
file = '/Users/danielfridman/Downloads/WDI_csv/WDIData.csv'
WDI = pd.read_csv(file)
```

Create dictionary of countries/keys and convert to dataframe. *Note: We have not included Venezuela due to lack of data.*

In [2363]:
```
countries_dict = {
    'country': {1: 'Antigua and Barbuda',
                2: 'Argentina',
                3: 'Bahamas',
                4: 'Barbados',
                5: 'Belize',
                6: 'Bolivia',
                7: 'Brazil',
                8: 'Chile',
                9: 'Colombia',
                10: 'Costa Rica',
                11: 'Cuba',
                12: 'Dominica',
                13: 'Dominican Republic',
                14: 'Ecuador',
                15: 'El Salvador',
                16: 'Grenada',
                17: 'Guatemala',
                18: 'Guyana',
                19: 'Haiti',
                20: 'Honduras',
                21: 'Jamaica',
                22: 'Mexico',
                23: 'Nicaragua',
                24: 'Panama',
                25: 'Paraguay',
                26: 'Peru',
                27: 'Saint Kitts & Nevis',
                28: 'Saint Lucia',
                29: 'Saint Vincent & Grenadines',
                30: 'Suriname',
                31: 'Trinidad and Tobago',
                32: 'Uruguay',
               },
    'code': {1: 'ATG', 2: 'ARG', 3: 'BHS', 4: 'BRB', 5: 'BLZ', 6: 'BO
L', 7: 'BRA', 8: 'CHL', 9: 'COL', 10: 'CRI',
             11: 'CUB', 12: 'DMA', 13: 'DOM', 14: 'ECU', 15: 'SLV', 1
6: 'GRD', 17: 'GTM', 18: 'GUY', 19: 'HTI',
             20: 'HND', 21: 'JAM', 22: 'MEX', 23: 'NIC', 24: 'PAN', 2
5: 'PRY', 26: 'PER', 27: 'KNA', 28: 'LCA',
             29: 'VCT', 30: 'SUR', 31: 'TTO', 32: 'URY'
            }
}
countries = pd.DataFrame(countries_dict)
```

Select relevant indicators:

In [2364]:
```
indicators_macro = {
              'NY.GNP.PCAP.CD':'GNI per Capita',
              'NY.GDP.MKTP.KD.ZG':'GDP Growth (%)',
              'BX.KLT.DINV.CD.WD': 'Foreign direct investment, net in
flows (BoP, current US$)',
              'FP.CPI.TOTL.ZG': 'Inflation, consumer prices (annual %
)',
              'FR.INR.RINR': 'Real interest rate (%)',
              'SM.POP.NETM': 'Net migration',
              'PA.NUS.FCRF': 'Official exchange rate (LCU per US$, pe
riod average)',
              'SL.UEM.TOTL.ZS': 'Unemployment, total (% of total labo
r force) (modeled ILO estimate)',
              'IC.PRP.PROC': 'Procedures to register property (number
)',
              'FR.INR.RISK': 'Risk premium on lending (lending rate m
inus treasury bill rate, %)',
             }

indicators_agro = { 'NV.AGR.TOTL.ZS' : 'Agriculture, forestry, and fi
shing, value added (% of GDP)',
                    'NV.AGR.TOTL.KD.ZG' : 'Agriculture, forestry, and
fishing, value added (annual % growth)',
                    'NV.AGR.TOTL.KD' : 'Agriculture, forestry, and fi
shing, value added (constant 2010 US$)',
                    'NV.AGR.TOTL.KN' : 'Agriculture, forestry, and fi
shing, value added (constant LCU)',
                    'NV.AGR.TOTL.CN' : 'Agriculture, forestry, and fi
shing, value added (current LCU)',
                    'NV.AGR.TOTL.CD' : 'Agriculture, forestry, and fi
shing, value added (current US$)',
                    'NV.AGR.EMPL.KD' : 'Agriculture, forestry, and fi
shing, value added per worker (constant 2010 US$)',
                    'ER.H2O.FWAG.ZS' : 'Annual freshwater withdrawals
, agriculture (% of total freshwater withdrawal)',
                    'SL.AGR.0714.ZS' : 'Child employment in agricultu
re (% of economically active children ages 7-14)',
                    'SL.AGR.0714.FE.ZS' : 'Child employment in agricu
lture, female (% of female economically active children ages 7-14)',
                    'SL.AGR.0714.MA.ZS' : 'Child employment in agricu
lture, male (% of male economically active children ages 7-14)',
                    'SL.AGR.EMPL.ZS' : 'Employment in agriculture (%
of total employment) (modeled ILO estimate)',
                    'SL.AGR.EMPL.FE.ZS' : 'Employment in agriculture,
female (% of female employment) (modeled ILO estimate)',
                    'SL.AGR.EMPL.MA.ZS' : 'Employment in agriculture,
male (% of female employment) (modeled ILO estimate)',
}
```

Now let's take a look at our data and clean it up a bit

In [2365]:
```
WDI.tail(10)
```
Out[2365]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963
422390	Zimbabwe	ZWE	Women participating in the three decisions (ow...	SG.DMK.ALLD.FN.ZS	NaN	NaN	NaN	NaN
422391	Zimbabwe	ZWE	Women who believe a husband is justified in be...	SG.VAW.REAS.ZS	NaN	NaN	NaN	NaN
422392	Zimbabwe	ZWE	Women who believe a husband is justified in be...	SG.VAW.ARGU.ZS	NaN	NaN	NaN	NaN
422393	Zimbabwe	ZWE	Women who believe a husband is justified in be...	SG.VAW.BURN.ZS	NaN	NaN	NaN	NaN
422394	Zimbabwe	ZWE	Women who believe a husband is justified in be...	SG.VAW.GOES.ZS	NaN	NaN	NaN	NaN
422395	Zimbabwe	ZWE	Women who believe a husband is justified in be...	SG.VAW.NEGL.ZS	NaN	NaN	NaN	NaN
422396	Zimbabwe	ZWE	Women who believe a husband is justified in be...	SG.VAW.REFU.ZS	NaN	NaN	NaN	NaN
422397			Women who were first					

	Zimbabwe	ZWE	married by age 15 (% of w...	SP.M15.2024.FE.ZS	NaN	NaN	NaN	NaN
422398	Zimbabwe	ZWE	Women who were first married by age 18 (% of w...	SP.M18.2024.FE.ZS	NaN	NaN	NaN	NaN
422399	Zimbabwe	ZWE	Women's share of population ages 15+ living wi...	SH.DYN.AIDS.FE.ZS	NaN	NaN	NaN	NaN

10 rows × 63 columns

In [2366]:
```
WDI.shape
```

Out[2366]:
```
(422400, 63)
```

In [2367]:
```
WDI.dtypes
```

Out[2367]:
```
Country Name       object
Country Code       object
Indicator Name     object
Indicator Code     object
1960              float64
1961              float64
1962              float64
1963              float64
1964              float64
1965              float64
1966              float64
1967              float64
1968              float64
1969              float64
1970              float64
1971              float64
1972              float64
1973              float64
1974              float64
1975              float64
1976              float64
1977              float64
1978              float64
1979              float64
1980              float64
1981              float64
1982              float64
1983              float64
```

```
1984              float64
1985              float64
                    ...
1989              float64
1990              float64
1991              float64
1992              float64
1993              float64
1994              float64
1995              float64
1996              float64
1997              float64
1998              float64
1999              float64
2000              float64
2001              float64
2002              float64
2003              float64
2004              float64
2005              float64
2006              float64
2007              float64
2008              float64
2009              float64
2010              float64
2011              float64
2012              float64
2013              float64
2014              float64
2015              float64
2016              float64
2017              float64
Unnamed: 62       float64
Length: 63, dtype: object
```

Set all columns lowercase, set index as country code, drop any unsorted columns, and remove all rows with less than 10 non-NaN values for more reliable data

In [2368]:
```
WDI.columns = [i.lower() for i in WDI.columns]
WDI = WDI.set_index('country code')
WDI = WDI.drop('unnamed: 62', 1)
WDI = WDI.dropna(how = 'all', subset = [[str(i) for i in list(range(1
960, 2018, 1))]], thresh = 10) #removes all rows with less than 10 no
n-NaN values
```

In [2369]:
```
WDI.shape
```

Out[2369]:
```
(225787, 61)
```

Looks like we got rid of 150,000 lines!

Now let's create our LatAm dataframe from the list of countries we created a dictionary of earlier – this is the only part that is *not* scalable, e.g. if we were to create a WDI_Europe DataFrame, we'd have to change some of the code.

It's possible to get around this by adding countries to countries_dict, instead of creating a separate dictionary for different countries.

In [2370]:
```
WDI_LatAm = WDI.loc[(WDI['country name'].isin(countries['country'])),
:]
WDI_LatAm.shape
```

Out[2370]: (28953, 61)

In [2371]:
```
WDI_LatAm_Macro = WDI_LatAm.loc[(WDI_LatAm['indicator code'].isin(lis
t(indicators_macro))),:]
WDI_LatAm_Macro = WDI_LatAm_Macro.reset_index()
WDI_LatAm_Macro.shape
```

Out[2371]: (250, 62)

In [2372]:
```
WDI_LatAm_Agro = WDI_LatAm.loc[(WDI_LatAm['indicator code'].isin(list
(indicators_agro))),:]
WDI_LatAm_Agro = WDI_LatAm_Agro.reset_index()
WDI_LatAm_Agro.shape
```

Out[2372]: (268, 62)

And now, from the LatAm dataframe, let's create a sample country dataframe – obviously not scalable, but for now. Our first country is Antigua and Barbuda, so let's check out its Macroeconomic indicators!

In [2373]:
```
WDI_LatAm_Macro_ATG = WDI_LatAm_Macro.loc[(WDI_LatAm_Macro['country c
ode'] == 'ATG')][[str(i) for i in list(range(1960, 2018, 1))]]
WDI_LatAm_Macro_ATG
```

Out[2373]:

	1960	1961	1962	1963	1964	1965	1966	1967	196
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
4	NaN	NaN	-1703.00000	NaN	NaN	NaN	NaN	-1625.000000	Nal
5	1.71429	1.71429	1.71429	1.71429	1.71429	1.71429	1.71429	1.761908	2.0
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal

9 rows × 58 columns

Clean it up a bit and set an index...

In [2374]:
```
WDI_LatAm_Macro_ATG['indicator code'] = list(WDI_LatAm_Macro.loc[(WDI
_LatAm_Macro['country code'] == 'ATG')]['indicator code'])
WDI_LatAm_Macro_ATG = WDI_LatAm_Macro_ATG.set_index('indicator code')
WDI_LatAm_Macro_ATG
```

Out[2374]:

	1960	1961	1962	1963	1964	1965	1966
indicator code							
BX.KLT.DINV.CD.WD	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NY.GDP.MKTP.KD.ZG	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NY.GNP.PCAP.CD	NaN	NaN	NaN	NaN	NaN	NaN	NaN
FP.CPI.TOTL.ZG	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SM.POP.NETM	NaN	NaN	-1703.00000	NaN	NaN	NaN	NaN
PA.NUS.FCRF	1.71429	1.71429	1.71429	1.71429	1.71429	1.71429	1.7142
IC.PRP.PROC	NaN	NaN	NaN	NaN	NaN	NaN	NaN
FR.INR.RINR	NaN	NaN	NaN	NaN	NaN	NaN	NaN
FR.INR.RISK	NaN	NaN	NaN	NaN	NaN	NaN	NaN

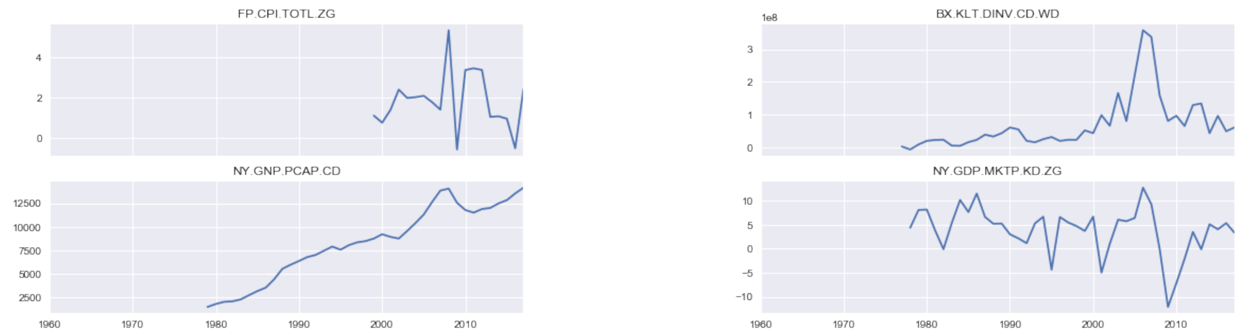
9 rows × 58 columns

Now let's plot the data!

In [2375]:
```
fig, ax = plt.subplots(nrows = 2, ncols = 2, sharex = True)
fig.subplots_adjust(wspace = 0.5)

WDI_LatAm_Macro_ATG.T[WDI_LatAm_Macro_ATG.T.columns[0]].plot(ax = ax[
0, 1], title = WDI_LatAm_Macro_ATG.T.columns[0], figsize = (20, 5))
WDI_LatAm_Macro_ATG.T[WDI_LatAm_Macro_ATG.T.columns[1]].plot(ax = ax[
1, 1], title = WDI_LatAm_Macro_ATG.T.columns[1])
WDI_LatAm_Macro_ATG.T[WDI_LatAm_Macro_ATG.T.columns[2]].plot(ax = ax[
1, 0], title = WDI_LatAm_Macro_ATG.T.columns[2])
WDI_LatAm_Macro_ATG.T[WDI_LatAm_Macro_ATG.T.columns[3]].plot(ax = ax[
0, 0], title = WDI_LatAm_Macro_ATG.T.columns[3])
```

Out[2375]: <matplotlib.axes._subplots.AxesSubplot at 0x167764a90>



It's taking too long to do all the graphs one by one...why not automate it with a loop?

We will be using this logic throughout the project.

This is also not scalable due to hardcoding the number of indicators (nrows * ncols) and due to the title. We leave it as is since it only exists for demonstration purposes.

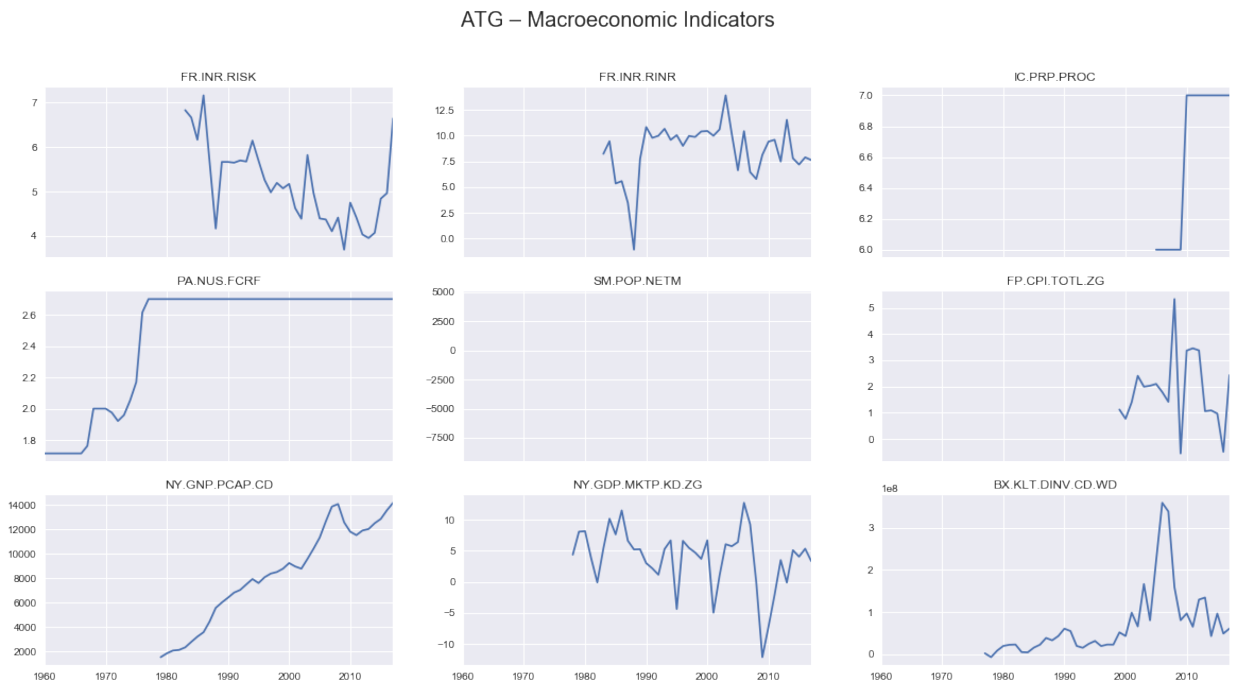
In [2376]:
```
ncols = len(WDI_LatAm_Macro_ATG.index) - 1

fig, ax = plt.subplots(nrows = 3, ncols = 3, sharex = True)

while ncols > -1:

    for i in range(3):
        for j in range(3):
            WDI_LatAm_Macro_ATG.T[WDI_LatAm_Macro_ATG.T.columns[ncols
]].plot(ax = ax[i, j], title = WDI_LatAm_Macro_ATG.T.columns[ncols],
figsize = (20, 5))
            ncols-=1

fig.set_figwidth(20)
fig.set_figheight(10)
fig = fig.suptitle('ATG – Macroeconomic Indicators', fontsize = 20)
```



Interesting! A couple of conclusions:

– SM.POP.NETM seems to not exist, but looking up the dataframe shows that there's only a value every five years. Perhaps that is messing with it.

– FR.INR.RISK and FR.INR.RINR seem to be directly correlated. This is not a surprise since .RISR is the risk-free rate, while .RISK is the risk-free rate.

– PA.NUS.FCRF evens out. This is the exchange rate. Antigua and Barbuda fixed its currency to the dollar in 1976, so reliability is not compromised.

– Rest of the data has at least 20 datapoints – looking good!

Same thing for the Agricultural indicators...

In [2377]:
```
WDI_LatAm_Agro_ATG['indicator code'] = list(WDI_LatAm_Agro.loc[(WDI_L
atAm_Agro['country code'] == 'ATG')]['indicator code'])
WDI_LatAm_Agro_ATG = WDI_LatAm_Agro_ATG.set_index('indicator code')
WDI_LatAm_Agro_ATG
```

Out[2377]:

	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	...	2(
indicator code												
NV.AGR.TOTL.ZS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.
NV.AGR.TOTL.KD.ZG	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	-6
NV.AGR.TOTL.KD	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2.
NV.AGR.TOTL.KN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	4.
NV.AGR.TOTL.CN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	5.
NV.AGR.TOTL.CD	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2.

6 rows × 58 columns

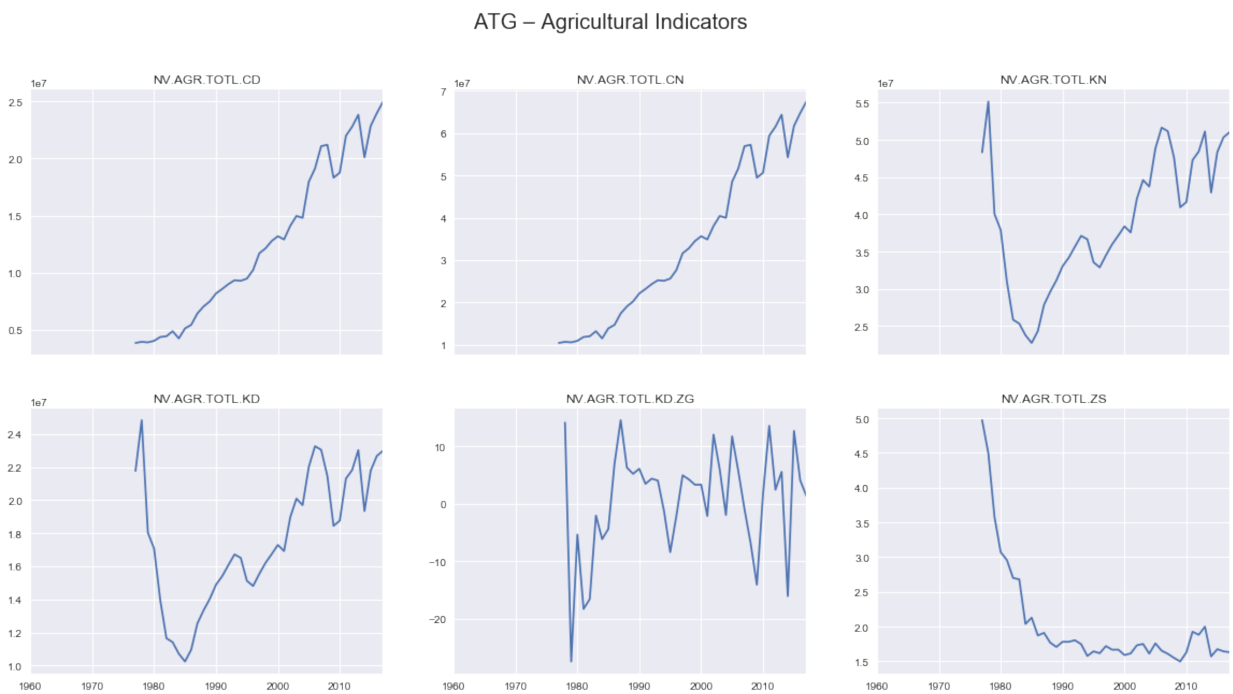
```
In [2378]: ncols = len(WDI_LatAm_Agro_ATG.index) - 1

fig, ax = plt.subplots(nrows = 2, ncols = 3, sharex = True)

while ncols > -1:

    for i in range(2):
        for j in range(3):
            WDI_LatAm_Agro_ATG.T[WDI_LatAm_Agro_ATG.T.columns[ncols]]
.plot(ax = ax[i, j], title = WDI_LatAm_Agro_ATG.T.columns[ncols], fig
size = (20, 5))
            ncols-=1

fig.set_figwidth(20)
fig.set_figheight(10)
fig = fig.suptitle('ATG – Agricultural Indicators', fontsize = 20)
```



Some conclusions:

– At a first glance, NV.AGR.TOTL.CD and NY.GNP.PCAP.CD seem to have a positive correlation – will be looking out for this later.

– NV_AGR.TOTL.KD and FR.INR.RINR also seem to be correlated.

– Data is quite reliable, going back further than 1980 in all cases.

And now, for the main event: finding the correlations between all graphs.

Many of the other projects find the correlations one by one, drawing conclusions one by one.

We want to do it all at once.

Lots going on below.

First: Set up the correlation statement

This is tricky because we need the correlations between two *time series*. We are actually working with a 3D data set here: we don't just want the correlation between WDI_LatAm_Agro_ATG and WDI_LatAm_Macro_ATG – we want the correlation between those two *across the time series*.

Hence, a simple .corr() doesn't work – there is no use to finding the correlation between just two datapoints – Macro_ATG in 2018 and Agro_ATG in 2018. Thus, pearsonr, which returns (r, p-value). We will only be looking at the r here since we want to average it later on across countries.

Additionally, we want to make this scalable, so we can add and remove indicators without breaking the project. Different indicators have different amounts of data. Thus, we generalize the correlations to make sure the time series lines up across rows, setting it up through .tail(len(WDI_LatAm_Macro_ATG.iloc[j].dropna()))

Finally, we drop NaN values so the correlation actually works.

In [2379]:
```
pearsonr(WDI_LatAm_Agro_ATG.iloc[i].dropna().tail(len(WDI_LatAm_Macro_ATG.iloc[j].dropna())), WDI_LatAm_Macro_ATG.iloc[j].dropna())
```

Out[2379]:
```
(0.31561217591734325, 0.050321743155617528)
```

Second: Use if statement to account for differing NA values

Additionally, earlier on, we dropped any rows that had less than 10 data points across time. We need to account for this:

If the **length of the list of non-NA values in Agro_ATG** is greater than **the length of the list of non-NA values in Macro_ATG**, then:

Find the correlation

between

the last x values in Agro_ATG, with *x* being equal to *the length of the list of non-NA values in Macro_ATG*

and

the list of non-NA values in Macro_ATG

else:

the reverse

This makes sure that we are actually lined up across time series.

In [2380]:
```
i = WDI_LatAm_Agro_ATG.shape[0] - 1
j = WDI_LatAm_Macro_ATG.shape[0] - 1

if len(WDI_LatAm_Agro_ATG.iloc[i].dropna()) > len(WDI_LatAm_Macro_ATG
.iloc[j].dropna()):
    print(pearsonr(WDI_LatAm_Agro_ATG.iloc[i].dropna().tail(len(WDI_L
atAm_Macro_ATG.iloc[j].dropna())), WDI_LatAm_Macro_ATG.iloc[j].dropna
()))
else:
    print(pearsonr(WDI_LatAm_Agro_ATG.iloc[i].dropna(), WDI_LatAm_Mac
ro_ATG.iloc[j].dropna().tail(len(WDI_LatAm_Agro_ATG.iloc[i].dropna())
)))
```

```
(-0.63013520161223313, 4.9789575746451641e-05)
```

Third: use nested loops to loop through master country dataframe

We have our master country dataframe WDI_LatAm_Macro_ATG. We now want to loop through every row and find its correlation with every column.

All we have to do is use nested loops.

The rest of the code is the same – the text exists to verify that this strategy works.

In [2381]:

```
i = WDI_LatAm_Agro_ATG.shape[0] - 1

while i > -1:

    j = WDI_LatAm_Macro_ATG.shape[0] - 1

    while j > -1:

        print('i = ' + str(i))
        print('j = ' + str(j))

        if len(WDI_LatAm_Agro_ATG.iloc[i].dropna()) > len(WDI_LatAm_M
acro_ATG.iloc[j].dropna()):
            print('Correlation between ' + WDI_LatAm_Macro_ATG.index[
j] + ' and ' + WDI_LatAm_Agro_ATG.index[i])
            print(pearsonr(WDI_LatAm_Agro_ATG.iloc[i].dropna().tail(l
en(WDI_LatAm_Macro_ATG.iloc[j].dropna())), WDI_LatAm_Macro_ATG.iloc[j
].dropna()))
            print('n = ' + str(len(WDI_LatAm_Agro_ATG.iloc[i].dropna(
).tail(len(WDI_LatAm_Macro_ATG.iloc[j].dropna())))))
        else:
            print('Correlation between ' + WDI_LatAm_Macro_ATG.index[
j] + ' and ' + WDI_LatAm_Agro_ATG.index[i])
            print(pearsonr(WDI_LatAm_Agro_ATG.iloc[i].dropna(), WDI_L
atAm_Macro_ATG.iloc[j].dropna().tail(len(WDI_LatAm_Agro_ATG.iloc[i].d
ropna()))))
            print('n = ' + str(len(WDI_LatAm_Agro_ATG.iloc[i].dropna(
).tail(len(WDI_LatAm_Macro_ATG.iloc[j].dropna())))))
        print('*****************************************************
*******************')
        j-=1

    i-=1
```

```
i = 5
j = 8
Correlation between FR.INR.RISK and NV.AGR.TOTL.CD
(-0.6301352016122313, 4.978957574645164le-05)
n = 35
*********************************************************************
********
i = 5
j = 7
Correlation between FR.INR.RINR and NV.AGR.TOTL.CD
(0.11764303182267173, 0.50091399843320095)
n = 35
*********************************************************************
********
i = 5
j = 6
Correlation between IC.PRP.PROC and NV.AGR.TOTL.CD
(0.62511382899079038, 0.022337001502883161)
n = 13
```

```
****************************************************************
********
i = 5
j = 5
Correlation between PA.NUS.FCRF and NV.AGR.TOTL.CD
(nan, 1.0)
n = 41
****************************************************************
********
i = 5
j = 4
Correlation between SM.POP.NETM and NV.AGR.TOTL.CD
(0.57656315358815857, 0.049715865512860599)
n = 12
****************************************************************
********
i = 5
j = 3
Correlation between FP.CPI.TOTL.ZG and NV.AGR.TOTL.CD
(0.1118100512875487 4, 0.64859470202147884)
n = 19
****************************************************************
********
i = 5
j = 2
Correlation between NY.GNP.PCAP.CD and NV.AGR.TOTL.CD
(0.95901238341293593, 7.3939402314023583e-22)
n = 39
****************************************************************
********
i = 5
j = 1
Correlation between NY.GDP.MKTP.KD.ZG and NV.AGR.TOTL.CD
(-0.26935061393672621, 0.092816025551934897)
n = 40
****************************************************************
********
i = 5
j = 0
Correlation between BX.KLT.DINV.CD.WD and NV.AGR.TOTL.CD
(0.59938390183932899, 3.4589956773934924e-05)
n = 41
****************************************************************
********
i = 4
j = 8
Correlation between FR.INR.RISK and NV.AGR.TOTL.CN
(-0.63013520161223324, 4.9789575746451478e-05)
n = 35
****************************************************************
********
i = 4
j = 7
```

```
Correlation between FR.INR.RINR and NV.AGR.TOTL.CN
(0.11764303182267266, 0.50091399843319739)
n = 35
**********************************************************************
i = 4
j = 6
Correlation between IC.PRP.PROC and NV.AGR.TOTL.CN
(0.62511382899079215, 0.022337001502882668)
n = 13
**********************************************************************

/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:3021: Ru
ntimeWarning: invalid value encountered in double_scalars
  r = r_num / r_den

i = 4
j = 5
Correlation between PA.NUS.FCRF and NV.AGR.TOTL.CN
(nan, 1.0)
n = 41
**********************************************************************
i = 4
j = 4
Correlation between SM.POP.NETM and NV.AGR.TOTL.CN
(0.57656315358816013, 0.049715865512859898)
n = 12
**********************************************************************
i = 4
j = 3
Correlation between FP.CPI.TOTL.ZG and NV.AGR.TOTL.CN
(0.11181005128754817, 0.64859470202148173)
n = 19
**********************************************************************
i = 4
j = 2
Correlation between NY.GNP.PCAP.CD and NV.AGR.TOTL.CN
(0.95901238341293604, 7.3939402314019793e-22)
n = 39
**********************************************************************
i = 4
j = 1
Correlation between NY.GDP.MKTP.KD.ZG and NV.AGR.TOTL.CN
(-0.26935061393672677, 0.092816025551934078)
n = 40
**********************************************************************
i = 4
j = 0
```

```
Correlation between BX.KLT.DINV.CD.WD and NV.AGR.TOTL.CN
(0.5993839018393285 5, 3.4589956773935534e-05)
n = 41
********************************************************************
********
i = 3
j = 8
Correlation between FR.INR.RISK and NV.AGR.TOTL.KN
(-0.6302398726839103 5, 4.9602881682317532e-05)
n = 35
********************************************************************
********
i = 3
j = 7
Correlation between FR.INR.RINR and NV.AGR.TOTL.KN
(0.23200541546286474, 0.17988866455240193)
n = 35
********************************************************************
********
i = 3
j = 6
Correlation between IC.PRP.PROC and NV.AGR.TOTL.KN
(-0.055516817863305049, 0.85704539673990932)
n = 13
********************************************************************
********
i = 3
j = 5
Correlation between PA.NUS.FCRF and NV.AGR.TOTL.KN
(nan, 1.0)
n = 41
********************************************************************
********
i = 3
j = 4
Correlation between SM.POP.NETM and NV.AGR.TOTL.KN
(0.53946228679731401, 0.07027017403089468)
n = 12
********************************************************************
********
i = 3
j = 3
Correlation between FP.CPI.TOTL.ZG and NV.AGR.TOTL.KN
(0.15925716966359885, 0.51489475227584691)
n = 19
********************************************************************
********
i = 3
j = 2
Correlation between NY.GNP.PCAP.CD and NV.AGR.TOTL.KN
(0.8556141519571735, 3.9524448694945589e-12)
n = 39
********************************************************************
```

```
********
i = 3
j = 1
Correlation between NY.GDP.MKTP.KD.ZG and NV.AGR.TOTL.KN
(-0.11993308645940895, 0.46103864893261548)
n = 40
**********************************************************************
********
i = 3
j = 0
Correlation between BX.KLT.DINV.CD.WD and NV.AGR.TOTL.KN
(0.54796391573872616, 0.00020848827719326327)
n = 41
**********************************************************************
********
i = 2
j = 8
Correlation between FR.INR.RISK and NV.AGR.TOTL.KD
(-0.63023987268390924, 4.9602881682319409e-05)
n = 35
**********************************************************************
********
i = 2
j = 7
Correlation between FR.INR.RINR and NV.AGR.TOTL.KD
(0.23200541546286502, 0.17988866455240157)
n = 35
**********************************************************************
********
i = 2
j = 6
Correlation between IC.PRP.PROC and NV.AGR.TOTL.KD
(-0.055516817863307866, 0.85704539673990165)
n = 13
**********************************************************************
********
i = 2
j = 5
Correlation between PA.NUS.FCRF and NV.AGR.TOTL.KD
(nan, 1.0)
n = 41
**********************************************************************
********
i = 2
j = 4
Correlation between SM.POP.NETM and NV.AGR.TOTL.KD
(0.53946228679731811, 0.070270174030892182)
n = 12
**********************************************************************
********
i = 2
j = 3
Correlation between FP.CPI.TOTL.ZG and NV.AGR.TOTL.KD
```

```
(0.1592571696635961, 0.51489475227585346)
n = 19
***********************************************************************
********
i = 2
j = 2
Correlation between NY.GNP.PCAP.CD and NV.AGR.TOTL.KD
(0.85561415195717394, 3.9524448694943602e-12)
n = 39
***********************************************************************
********
i = 2
j = 1
Correlation between NY.GDP.MKTP.KD.ZG and NV.AGR.TOTL.KD
(-0.11993308645940919, 0.46103864893261548)
n = 40
***********************************************************************
********
i = 2
j = 0
Correlation between BX.KLT.DINV.CD.WD and NV.AGR.TOTL.KD
(0.54796391573872638, 0.00020848827719326197)
n = 41
***********************************************************************
********
i = 1
j = 8
Correlation between FR.INR.RISK and NV.AGR.TOTL.KD.ZG
(0.027707197492120757, 0.87446067805800853)
n = 35
***********************************************************************
********
i = 1
j = 7
Correlation between FR.INR.RINR and NV.AGR.TOTL.KD.ZG
(-0.034930969229406493, 0.84209856717490372)
n = 35
***********************************************************************
********
i = 1
j = 6
Correlation between IC.PRP.PROC and NV.AGR.TOTL.KD.ZG
(0.22043296052157405, 0.46926443160783449)
n = 13
***********************************************************************
********
i = 1
j = 5
Correlation between PA.NUS.FCRF and NV.AGR.TOTL.KD.ZG
(nan, 1.0)
n = 40
***********************************************************************
********
```

```
i = 1
j = 4
Correlation between SM.POP.NETM and NV.AGR.TOTL.KD.ZG
(-0.022967140660174742, 0.9435190355176227З)
n = 12
********************************************************************
********
i = 1
j = 3
Correlation between FP.CPI.TOTL.ZG and NV.AGR.TOTL.KD.ZG
(0.15093208740837352, 0.5373774906130652 8)
n = 19
********************************************************************
********
i = 1
j = 2
Correlation between NY.GNP.PCAP.CD and NV.AGR.TOTL.KD.ZG
(0.31561217591734325, 0.05032174315561752 8)
n = 39
********************************************************************
********
i = 1
j = 1
Correlation between NY.GDP.MKTP.KD.ZG and NV.AGR.TOTL.KD.ZG
(0.13119509034292895, 0.41970342691662865)
n = 40
********************************************************************
********
i = 1
j = 0
Correlation between BX.KLT.DINV.CD.WD and NV.AGR.TOTL.KD.ZG
(0.18314364854967696, 0.25798265269444709)
n = 40
********************************************************************
********
i = 0
j = 8
Correlation between FR.INR.RISK and NV.AGR.TOTL.ZS
(0.4338714271651064, 0.009214969680483121 2)
n = 35
********************************************************************
********
i = 0
j = 7
Correlation between FR.INR.RINR and NV.AGR.TOTL.ZS
(-0.10217975563430168, 0.55916526378801856)
n = 35
********************************************************************
********
i = 0
j = 6
Correlation between IC.PRP.PROC and NV.AGR.TOTL.ZS
(0.43323339769265018, 0.13918582865424206)
```

```
n = 13
**********************************************************************
i = 0
j = 5
Correlation between PA.NUS.FCRF and NV.AGR.TOTL.ZS
(nan, 1.0)
n = 41
**********************************************************************
i = 0
j = 4
Correlation between SM.POP.NETM and NV.AGR.TOTL.ZS
(0.288118591621017, 0.36380188442880129)
n = 12
**********************************************************************
i = 0
j = 3
Correlation between FP.CPI.TOTL.ZG and NV.AGR.TOTL.ZS
(0.21692421840040058, 0.37236744589638338)
n = 19
**********************************************************************
i = 0
j = 2
Correlation between NY.GNP.PCAP.CD and NV.AGR.TOTL.ZS
(-0.7110045143682624, 3.9296363253435808e-07)
n = 39
**********************************************************************
i = 0
j = 1
Correlation between NY.GDP.MKTP.KD.ZG and NV.AGR.TOTL.ZS
(0.165609876565047, 0.30712686752779705)
n = 40
**********************************************************************
i = 0
j = 0
Correlation between BX.KLT.DINV.CD.WD and NV.AGR.TOTL.ZS
(-0.34068845389342467, 0.02928119920160072)
n = 41
**********************************************************************
```

Now, we generalize away from WDI_LatAm_Agro_ATG to enable loop usage across countries:

We start off by locating the time series in the dataframe WDI_Lat_Am_Agro, generalized to *country_code*

We then set the index equal to the indicator code, so that we know what we're actually looking at!

In [2382]:
```
current_country_agro = WDI_LatAm_Agro.loc[(WDI_LatAm_Agro['country co
de'] == country_code)][[str(i) for i in list(range(1960, 2018, 1))]]
current_country_agro['indicator code'] = list(WDI_LatAm_Agro.loc[(WDI
_LatAm_Agro['country code'] == country_code)]['indicator code'])
current_country_agro = current_country_agro.set_index('indicator code
')
current_country_agro
```

Out[2382]:

	1960	1961	1962	1963	196
indicator code					
NV.AGR.TOTL.ZS	9.909312e+00	9.545033e+00	8.727907e+00	7.575173e+00	8.1(
NV.AGR.TOTL.KD.ZG	NaN	-1.836386e+00	-4.556404e+00	6.202341e+00	1.2(
NV.AGR.TOTL.KD	1.117711e+09	1.097186e+09	1.047193e+09	1.112144e+09	1.1:
NV.AGR.TOTL.KN	6.192410e+11	6.078694e+11	5.801724e+11	6.161567e+11	6.2
NV.AGR.TOTL.CN	4.480000e+05	4.840000e+05	5.200000e+05	6.870000e+05	1.1:
NV.AGR.TOTL.CD	4.072727e+08	4.400000e+08	4.727273e+08	4.293750e+08	4.8
NV.AGR.EMPL.KD	NaN	NaN	NaN	NaN	Nal
SL.AGR.EMPL.ZS	NaN	NaN	NaN	NaN	Nal
SL.AGR.EMPL.FE.ZS	NaN	NaN	NaN	NaN	Nal
SL.AGR.EMPL.MA.ZS	NaN	NaN	NaN	NaN	Nal

10 rows × 58 columns

Fourth: implement generalization and define as method

The method doesn't return anything yet. For readability reasons, we will add that in the next step. All we did here was copy-paste the above code inside the loop, and put the whole thing inside a method.

In [2383]:
```
def crosscorrs(country_code):

    current_country_macro = WDI_LatAm_Macro.loc[(WDI_LatAm_Macro['cou
ntry code'] == country_code)][[str(i) for i in list(range(1960, 2018,
1))]]
```

```
    current_country_macro['indicator code'] = list(WDI_LatAm_Macro.loc[(WDI_LatAm_Macro['country code'] == country_code)]['indicator code'])
    current_country_macro = current_country_macro.set_index('indicator code')

    current_country_agro = WDI_LatAm_Agro.loc[(WDI_LatAm_Agro['country code'] == country_code)][[str(i) for i in list(range(1960, 2018, 1))]]
    current_country_agro['indicator code'] = list(WDI_LatAm_Agro.loc[(WDI_LatAm_Agro['country code'] == country_code)]['indicator code'])
    current_country_agro = current_country_agro.set_index('indicator code')

    i = current_country_agro.shape[0] - 1

    while i > -1:

        j = current_country_macro.shape[0] - 1

        while j > -1:

            print('i = ' + str(i))
            print('j = ' + str(j))

            if len(current_country_agro.iloc[i].dropna()) > len(current_country_macro.iloc[j].dropna()):
                print('Correlation between ' + current_country_macro.index[j] + ' and ' + current_country_agro.index[i])
                print(pearsonr(current_country_agro.iloc[i].dropna().tail(len(current_country_macro.iloc[j].dropna())), current_country_macro.iloc[j].dropna()))
                print('n = ' + str(len(current_country_agro.iloc[i].dropna().tail(len(current_country_macro.iloc[j].dropna())))))
            else:
                print('Correlation between ' + current_country_macro.index[j] + ' and ' + current_country_agro.index[i])
                print(pearsonr(current_country_agro.iloc[i].dropna(), current_country_macro.iloc[j].dropna().tail(len(current_country_agro.iloc[i].dropna()))))
                print('n = ' + str(len(current_country_agro.iloc[i].dropna().tail(len(current_country_macro.iloc[j].dropna())))))
            print('****************************************************************************')
            j-=1

        i-=1

# crosscorrs(countries.iloc[7][0])
# We won't run this because the output is the same as the output in Step 3. Everything that goes on is behind-the-scenes.
```

Fifth: store correlation results in lists

We create two lists here. The first list, *list1*, stores the correlations for pairs across one single row. *list1* is cleared every time we move to the next row, with its contents added to a master list *list_crosscorr*.

We return *list_crosscorr*, depicting all correlation pairs in a very unreadable format.

In [2384]:
```
list_crosscorr = []

def crosscorrs(country_code):

    current_country_macro = WDI_LatAm_Macro.loc[(WDI_LatAm_Macro['cou
ntry code'] == country_code)][[str(i) for i in list(range(1960, 2018,
1))]]
    current_country_macro['indicator code'] = list(WDI_LatAm_Macro.lo
c[(WDI_LatAm_Macro['country code'] == country_code)]['indicator code'
])
    current_country_macro = current_country_macro.set_index('indicato
r code')

    current_country_agro = WDI_LatAm_Agro.loc[(WDI_LatAm_Agro['countr
y code'] == country_code)][[str(i) for i in list(range(1960, 2018, 1)
)]]
    current_country_agro['indicator code'] = list(WDI_LatAm_Agro.loc[
(WDI_LatAm_Agro['country code'] == country_code)]['indicator code'])
    current_country_agro = current_country_agro.set_index('indicator
code')

    i = current_country_agro.shape[0] - 1

    while i > -1:

        j = current_country_macro.shape[0] - 1
        list1.clear()

        while j > -1:

            if len(current_country_agro.iloc[i].dropna()) > len(curre
nt_country_macro.iloc[j].dropna()):
                corr = pearsonr(current_country_agro.iloc[i].dropna()
.tail(len(current_country_macro.iloc[j].dropna())), current_country_m
acro.iloc[j].dropna())

            else:
                corr = pearsonr(current_country_agro.iloc[i].dropna()
, current_country_macro.iloc[j].dropna().tail(len(current_country_agr
o.iloc[i].dropna())))

            list1.append(corr[0])

            j-=1

        list_crosscorr.append(list1.copy())
        i-=1

    print(list_crosscorr)

crosscorrs(countries.iloc[7][0])
```

```
[[-0.30599489555335296, 0.41777456716679445, nan, -0.613001071827234
18, -0.83176680422827287, 0.72772656920708323, -0.93033521089255711,
0.54279265740158167, -0.80779767824250837], [-0.20131601992979045, -
0.19554572401650358, nan, -0.38320872609977275, -0.75430054999572049
, 0.42988193345456688, -0.3170895192437917, 0.29470688634263331, -0.
25147258466291134], [-0.3130483011107415, 0.39723415555400193, nan,
-0.61989920086586037, -0.8259287213547708, 0.73466617411901014, -0.9
2784615453478925, 0.55220586192878351, -0.80572994138865872], [0.282
78717835592876, -0.46888822126060964, nan, 0.57820690109178785, 0.85
511511815663899, -0.64349382926958265, 0.93556059940677994, -0.50887
189875035943, 0.82086925336309369], [-0.052262720737316928, -0.26994
232466679818, nan, 0.82622404952853157, 0.87185837903104257, -0.4061
2368071238697, 0.98148506761712195, 0.044571529288059457, 0.86499122
355289504], [0.090645017397428265, -0.35400286884191812, nan, 0.8505
9531936155086, 0.93325138453494128, -0.3756083265531206, 0.965654671
69224106, -0.012078437397979012, 0.80690190364396375], [0.2776139055
9670462, -0.36594579925497811, nan, 0.90012504138750637, 0.850566393
90032041, -0.42675840346489241, 0.95251488598621914, 0.0324594241343
16954, 0.85847622830789239], [0.27761390559670474, -0.36594579925497
789, nan, 0.90012504138750649, 0.85056639390031985, -0.4267584034648
9236, 0.95251488598621903, 0.032459424134316878, 0.8584762283078925]
, [-0.15243064787541694, -0.032699361168367753, nan, 0.0950019532688
14193, -0.10526931127960384, 0.15579123671279982, -0.085671149403876
587, 0.39142136274598099, -0.24176572459688822], [-0.564248173950461
12, 0.35812636350471971, nan, -0.76771101235261041, 0.23194377302300
087, 0.18722799337813953, -0.82800769230501636, 0.22556244375356679,
-0.76270285383531622]]
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:3021: Ru
ntimeWarning: invalid value encountered in double_scalars
  r = r_num / r_den
```

Sixth: create a DataFrame depicting the correlation matrix stored in lists

This vastly improves the readability. Thanks to employing the nested list as our data structure, converting it to a dataframe is very easy.

We now have a beautiful dataframe return from the crosscorrs() method:

In [2385]:
```
def crosscorrs(country_code):

    list_crosscorr = []

    current_country_macro = WDI_LatAm_Macro.loc[(WDI_LatAm_Macro['cou
ntry code'] == country_code)][[str(i) for i in list(range(1960, 2018,
1))]]
    current_country_macro['indicator code'] = list(WDI_LatAm_Macro.lo
c[(WDI_LatAm_Macro['country code'] == country_code)]['indicator code'
])
    current_country_macro = current_country_macro.set_index('indicato
r code')
```

```
    current_country_agro = WDI_LatAm_Agro.loc[(WDI_LatAm_Agro['country code'] == country_code)][[str(i) for i in list(range(1960, 2018, 1))]]
    current_country_agro['indicator code'] = list(WDI_LatAm_Agro.loc[(WDI_LatAm_Agro['country code'] == country_code)]['indicator code'])
    current_country_agro = current_country_agro.set_index('indicator code')

    i = current_country_agro.shape[0] - 1

    while i > -1:

        j = current_country_macro.shape[0] - 1
        list1.clear()

        while j > -1:

            if len(current_country_agro.iloc[i].dropna()) > len(current_country_macro.iloc[j].dropna()):
                corr = pearsonr(current_country_agro.iloc[i].dropna().tail(len(current_country_macro.iloc[j].dropna())), current_country_macro.iloc[j].dropna())

            else:
                corr = pearsonr(current_country_agro.iloc[i].dropna(), current_country_macro.iloc[j].dropna().tail(len(current_country_agro.iloc[i].dropna())))

            list1.append(corr[0])

            j-=1

        list_crosscorr.append(list1.copy())
        i-=1

    df = pd.DataFrame(list_crosscorr, columns = list(reversed(current_country_macro.index)))
    df[''] = list(reversed(current_country_agro.index))
    df = df.set_index('')
    df.index.name = country_code

    return df

crosscorrs(countries.iloc[7][0])
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:3021: Ru
ntimeWarning: invalid value encountered in double_scalars
  r = r_num / r_den
```

Out[2385]:

	SL.UEM.TOTL.ZS	FR.INR.RINR	IC.PRP.PROC	PA.NUS.FCRF	S
CHL					
SL.AGR.EMPL.MA.ZS	-0.305995	0.417775	NaN	-0.613001	-(
SL.AGR.EMPL.FE.ZS	-0.201316	-0.195546	NaN	-0.383209	-(
SL.AGR.EMPL.ZS	-0.313048	0.397234	NaN	-0.619899	-(
NV.AGR.EMPL.KD	0.282787	-0.468888	NaN	0.578207	0.
NV.AGR.TOTL.CD	-0.052263	-0.269942	NaN	0.826224	0.
NV.AGR.TOTL.CN	0.090645	-0.354003	NaN	0.850595	0.
NV.AGR.TOTL.KN	0.277614	-0.365946	NaN	0.900125	0.
NV.AGR.TOTL.KD	0.277614	-0.365946	NaN	0.900125	0.
NV.AGR.TOTL.KD.ZG	-0.152431	-0.032699	NaN	0.095002	-(
NV.AGR.TOTL.ZS	-0.564248	0.358126	NaN	-0.767711	0.

Seventh: change World Bank indicator codes into human-readable labels

Just increasing visibility here by moving away from the World Bank jargon to the actual keys stored in our _indicators dictionaries at the very begining.

In [2386]:

```
def crosscorrs(country_code):

    list_crosscorr = []

    current_country_macro = WDI_LatAm_Macro.loc[(WDI_LatAm_Macro['cou
ntry code'] == country_code)][[str(i) for i in list(range(1960, 2018,
1))]]
    current_country_macro['indicator code'] = list(WDI_LatAm_Macro.lo
c[(WDI_LatAm_Macro['country code'] == country_code)]['indicator code'
])
    current_country_macro = current_country_macro.set_index('indicato
r code')

    current_country_agro = WDI_LatAm_Agro.loc[(WDI_LatAm_Agro['countr
y code'] == country_code)][[str(i) for i in list(range(1960, 2018, 1)
)]]
    current_country_agro['indicator code'] = list(WDI_LatAm_Agro.loc[
(WDI_LatAm_Agro['country code'] == country_code)]['indicator code'])
```

```
    current_country_agro = current_country_agro.set_index('indicator
code')

    list_indicators_macro = [indicators_macro.get(i) for i in list(re
versed(current_country_macro.index))]
    list_indicators_agro = [indicators_agro.get(i) for i in list(reve
rsed(current_country_agro.index))]

    i = current_country_agro.shape[0] - 1

    while i > -1:

        j = current_country_macro.shape[0] - 1
        list1.clear()

        while j > -1:

            if len(current_country_agro.iloc[i].dropna()) > len(curre
nt_country_macro.iloc[j].dropna()):
                corr = pearsonr(current_country_agro.iloc[i].dropna()
.tail(len(current_country_macro.iloc[j].dropna())), current_country_m
acro.iloc[j].dropna())

            else:
                corr = pearsonr(current_country_agro.iloc[i].dropna()
, current_country_macro.iloc[j].dropna().tail(len(current_country_agr
o.iloc[i].dropna())))

            list1.append(corr[0])

            j-=1

        list_crosscorr.append(list1.copy())
        i-=1

    df = pd.DataFrame(list_crosscorr, columns = list_indicators_macro
)
    df[''] = list_indicators_agro
    df = df.set_index('')
    df.index.name = country_code

    return df
```

In [2387]:

```
crosscorrs(countries.iloc[8][0])
```

Out[2387]:

	Unemployment, total (% of total labor force) (modeled ILO estimate)	Real interest rate (%)	Procedures to register property (number)	Official exchange rate (LCU per US$, period average)	Net migration	Inflation, consumer prices (annual %)

COL						
Employment in agriculture, male (% of female employment) (modeled ILO estimate)	0.285841	0.277638	0.677011	-0.734466	-0.938903	0.839712
Employment in agriculture, female (% of female employment) (modeled ILO estimate)	-0.470581	0.044500	-0.140641	0.165110	0.497582	-0.056026
Employment in agriculture (% of total employment) (modeled ILO estimate)	0.224292	0.272378	0.705420	-0.755619	-0.919964	0.886783
Agriculture, forestry, and fishing, value added per worker (constant 2010 US$)	-0.409550	0.122252	-0.382249	-0.386178	0.846857	0.558723
Agriculture, forestry, and fishing, value added (current US$)	-0.469452	-0.141325	-0.704225	0.762896	0.476606	-0.487035
Agriculture,						

Agriculture, forestry, and fishing, value added (current LCU)	-0.282814	-0.254634	-0.742499	0.906470	0.863307	-0.689767
Agriculture, forestry, and fishing, value added (constant LCU)	-0.671990	-0.068325	-0.655452	0.710960	0.941742	-0.171702
Agriculture, forestry, and fishing, value added (constant 2010 US$)	-0.671990	-0.068325	-0.655452	0.710960	0.941742	-0.171702
Agriculture, forestry, and fishing, value added (annual % growth)	-0.043542	0.453642	0.132834	-0.151097	0.515477	-0.059868
Agriculture, forestry, and fishing, value added (% of GDP)	0.035389	0.399312	0.641019	-0.899517	-0.548702	0.479190

Done for now!

We can now create 32 DataFrames, one for each country. We won't bore and show you every one, but just to prove it:

In [2388]:
```
crosscorrs(countries.iloc[0][0])
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:3021: Ru
ntimeWarning: invalid value encountered in double_scalars
  r = r_num / r_den
```

Out[2388]:

	Risk						

	premium on lending (lending rate minus treasury bill rate, %)	Real interest rate (%)	Procedures to register property (number)	Official exchange rate (LCU per US$, period average)	Net migration	Inflation, consumer prices (annual %)	GNI p Capit
ATG							
Agriculture, forestry, and fishing, value added (current US$)	-0.630135	0.117643	0.625114	NaN	0.576563	0.111810	0.959C
Agriculture, forestry, and fishing, value added (current LCU)	-0.630135	0.117643	0.625114	NaN	0.576563	0.111810	0.959C
Agriculture, forestry, and fishing, value added (constant LCU)	-0.630240	0.232005	-0.055517	NaN	0.539462	0.159257	0.855€
Agriculture, forestry, and fishing, value added (constant 2010 US$)	-0.630240	0.232005	-0.055517	NaN	0.539462	0.159257	0.855€
Agriculture							

Agriculture, forestry, and fishing, value added (annual % growth)	0.027707	-0.034931	0.220433	NaN	-0.022967	0.150932	0.3156
Agriculture, forestry, and fishing, value added (% of GDP)	0.433871	-0.102180	0.433233	NaN	0.288119	0.216924	-0.711

In [2389]:
```
crosscorrs(countries.iloc[31][0])
```

Out[2389]:

	Unemployment, total (% of total labor force) (modeled ILO estimate)	Risk premium on lending (lending rate minus treasury bill rate, %)	Real interest rate (%)	Procedures to register property (number)	Official exchange rate (LCU per US$, period average)	Net migration
URY						
Employment in agriculture, male (% of female employment) (modeled ILO estimate)	-0.694933	-0.860384	-0.781335	0.183202	0.587684	-0.232654
Employment in agriculture, female (% of female	-0.682706	-0.834117	-0.748602	0.306379	0.627484	-0.083539

employment) (modeled ILO estimate)						
Employment in agriculture (% of total employment) (modeled ILO estimate)	-0.700200	-0.857225	-0.778738	0.200060	0.587321	-0.210253
Agriculture, forestry, and fishing, value added per worker (constant 2010 US$)	0.728176	0.899030	0.797361	-0.205348	-0.516942	0.365506
Agriculture, forestry, and fishing, value added (current US$)	-0.726483	-0.755183	-0.521755	0.014636	0.654171	-0.037265
Agriculture, forestry, and fishing, value added (current LCU)	-0.567714	-0.803147	-0.545992	0.407907	0.844811	0.035789
Agriculture, forestry, and fishing, value added (constant LCU)	-0.421893	-0.811507	-0.331617	0.363436	0.879131	0.291430
Agriculture, forestry, and fishing, value added (constant 2010 US$)	-0.421893	-0.811507	-0.331617	0.363436	0.879131	0.291430

Agriculture, forestry, and fishing, value added (annual % growth)	-0.016572	-0.105236	0.045224	-0.083325	0.011021	0.022838
Agriculture, forestry, and fishing, value added (% of GDP)	0.147084	0.000097	-0.174184	-0.667720	-0.542962	-0.182804

We will deal with the NaN values later.

Let's make it pretty! Thanks to seaborn for the heatmap:

[https://seaborn.pydata.org/generated/seaborn.heatmap.html (https://seaborn.pydata.org/generated/seaborn.heatmap.html)](https://seaborn.pydata.org/generated/seaborn.heatmap.html)

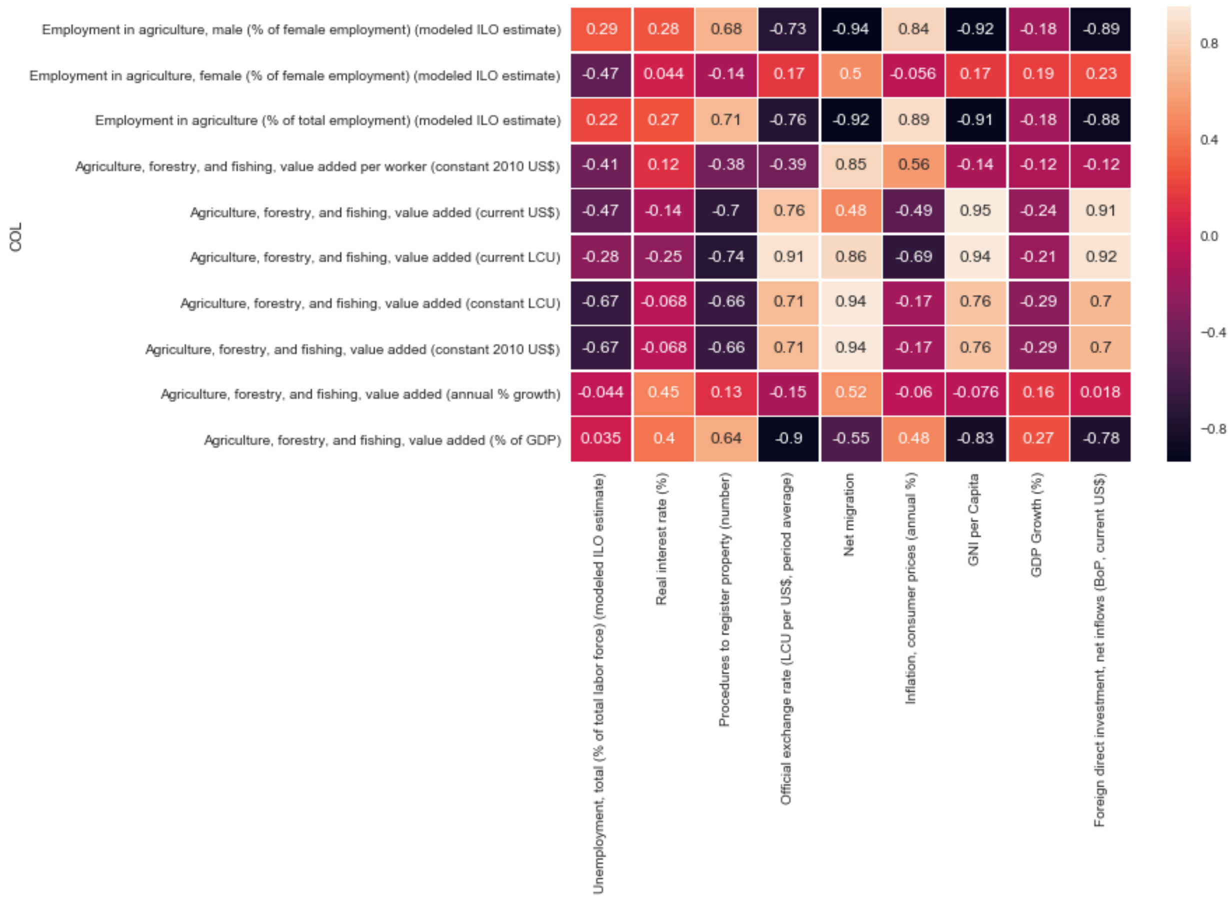
In [2390]:

```
def visualize(df):

    sns.set()

    # Draw a heatmap with the numeric values in each cell
    f, ax = plt.subplots(figsize=(9, 6))
    sns.heatmap(df, annot=True, linewidths=.5, ax=ax)
```

In [2391]:
```
visualize(crosscorrs(countries.iloc[8][0]))
```



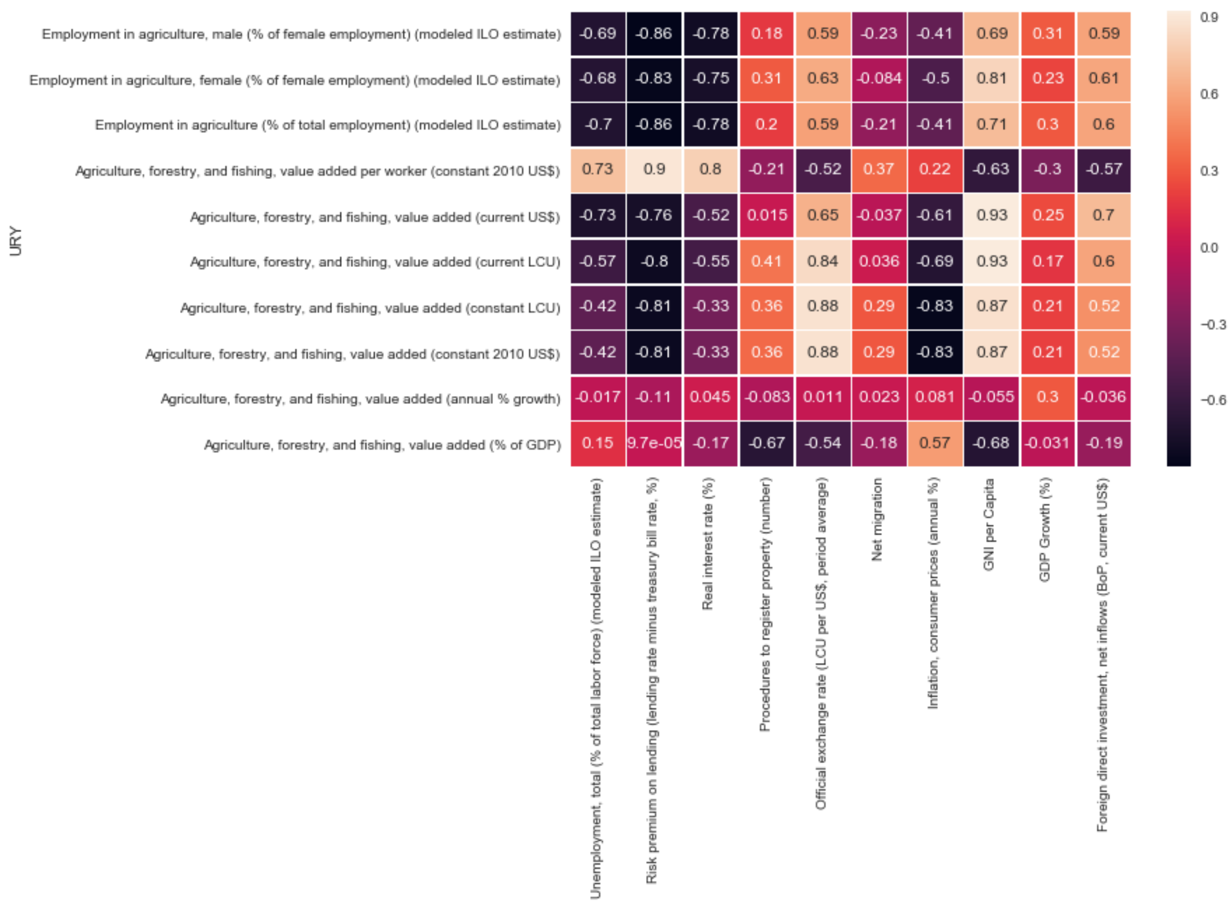
Just like there's potentially 32 DataFrames, we can also create 32 of these.

Again, proof:

In [2392]:
```
visualize(crosscorrs(countries.iloc[0][0]))
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:3021: Ru
ntimeWarning: invalid value encountered in double_scalars
  r = r_num / r_den
```

```
In [2393]: visualize(crosscorrs(countries.iloc[31][0]))
```



As mentioned, ATG pegged their currency to the U.S. Dollar. Guess that broke the correlation statement. No worries – we will deal with it later.

Overall, it seems like Agriculture Employment is negatively correlated with Unemployment and Interest Rates. This makes macroeconomic theoretical sense: when the economy is going better, it usually diversifies away from primary sectors such as agriculture!

Let's find the average of each indicator pair so that we can conclude which factors are most correlated.

Start by creating an empty Master DataFrame (master_df) with row labels = macro_indicators and columns labels = agro_indicators. We do this using some funky list magic:

We create a nested list master_list holding all possible correlation pairs:

In [2394]:
```
master_list = []

for i in list(indicators_macro.values()):
    master_list.append([i])
    idx = master_list.index([i])

    for j in list(indicators_agro.values()):
        master_list[idx].append(j)

master_list
```

Out[2394]:
```
[['GNI per Capita',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
 ['GDP Growth (%)',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
```

```
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
 ['Foreign direct investment, net inflows (BoP, current US$)',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
 ['Inflation, consumer prices (annual %)',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
```

```
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
 ['Real interest rate (%)',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
 ['Net migration',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
```

```
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
 ['Official exchange rate (LCU per US$, period average)',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
 ['Unemployment, total (% of total labor force) (modeled ILO estimat
e)',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
```

```
 ['Procedures to register property (number)',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)'],
 ['Risk premium on lending (lending rate minus treasury bill rate, %
)',
  'Agriculture, forestry, and fishing, value added (% of GDP)',
  'Agriculture, forestry, and fishing, value added (annual % growth)
',
  'Agriculture, forestry, and fishing, value added (constant 2010 US
$)',
  'Agriculture, forestry, and fishing, value added (constant LCU)',
  'Agriculture, forestry, and fishing, value added (current LCU)',
  'Agriculture, forestry, and fishing, value added (current US$)',
  'Agriculture, forestry, and fishing, value added per worker (const
ant 2010 US$)',
  'Annual freshwater withdrawals, agriculture (% of total freshwater
withdrawal)',
  'Child employment in agriculture (% of economically active childre
n ages 7-14)',
  'Child employment in agriculture, female (% of female economically
active children ages 7-14)',
  'Child employment in agriculture, male (% of male economically act
ive children ages 7-14)',
  'Employment in agriculture (% of total employment) (modeled ILO es
timate)',
  'Employment in agriculture, female (% of female employment) (model
ed ILO estimate)',
  'Employment in agriculture, male (% of female employment) (modeled
ILO estimate)']]
```

And now we turn the nested list into a DataFrame. We also make the cells empty – or, well, fully populate them by zeroes:

In [2395]:
```
master_df = pd.DataFrame(master_list)
master_df = dfc.set_index(master_df.pop(0))
master_df.columns = (list(master_df.iloc[0]))

idx = len(indicators_macro) - 1

while idx > -1:
    master_df.iloc[idx] = 0
    idx -= 1

master_df
```

Out[2395]:

	Agriculture, forestry, and fishing, value added (% of GDP)	Agriculture, forestry, and fishing, value added (annual % growth)	Agriculture, forestry, and fishing, value added (constant 2010 US$)	Agriculture, forestry, and fishing, value added (constant LCU)	Agriculture, forestry, and fishing, value added (current LCU)	Agric fores and fishir value adde (curr US$)
0						
GNI per Capita	0	0	0	0	0	0
GDP Growth (%)	0	0	0	0	0	0
Foreign direct investment, net inflows (BoP, current US$)	0	0	0	0	0	0
Inflation, consumer prices (annual %)	0	0	0	0	0	0
Real interest rate (%)	0	0	0	0	0	0
Net migration	0	0	0	0	0	0
Official						

exchange rate (LCU per US$, period average)	0	0	0	0	0	0
Unemployment, total (% of total labor force) (modeled ILO estimate)	0	0	0	0	0	0
Procedures to register property (number)	0	0	0	0	0	0
Risk premium on lending (lending rate minus treasury bill rate, %)	0	0	0	0	0	0

Now comes the meat of filling in the correlations for each [macro, agro] pair.

Essentially, we create a DataFrame of each country's correlations, and then add the cells from those DataFrames ('current') to the corresponding cell in the Master DataFrame (master_df) we just created.

The end goal is to find the average, and to find the average, we need to keep track of the number of times we added something to the cell ('n'). *n* is not the same for each cell because some countries have more data than others. Instead of creating yet another variable to keep track of this, we cheat by adding 1000 to the correlations, essentially tracking *n* in the first digit or two of the cells. On the way we also remove any NaN values.

Note: this takes a while to run and is wasteful computationally, but it was the best way we found to do this given our data structures.

In [2396]:

```
jdx = countries.shape[0] - 1

while jdx > -1:
    current = crosscorrs(countries.iloc[jdx][0])
    current = current.fillna(0, downcast='infer')
    print('country ' + str(jdx)) #since it takes so long, we want to
make sure it's actually running!

    for row in current.iterrows():
        idx = 0

        while idx < len(row[1]):
            if np.isnan(master_df.loc[list(row[1].index)[idx]][row[1]
.name]) == False:
                master_df.loc[list(row[1].index)[idx]][row[1].name] =
master_df.loc[list(row[1].index)[idx]][row[1].name] + row[1][idx] + 1
000
            else:
                print('NaN')
            idx += 1

    jdx -= 1
```

```
country 31
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:3021: Ru
ntimeWarning: invalid value encountered in double_scalars
  r = r_num / r_den
```

```
country 30
country 29
country 28
country 27
country 26
country 25
country 24
country 23
country 22
country 21
country 20
country 19
country 18
country 17
country 16
country 15
country 14
country 13
country 12
country 11
country 10
country 9
country 8
country 7
country 6
country 5
country 4
country 3
country 2
country 1
country 0
```

And here's the DataFrame of correlations...plus the temporary math stuff:

In [2397]:
```
master_df
```

Out[2397]:

	Agriculture, forestry, and fishing, value added (%	Agriculture, forestry, and fishing, value added	Agriculture, forestry, and fishing, value added	Agriculture, forestry, and fishing, value added	Agriculture, forestry, and fishing, value added	Agric fores and fishir value adde

	of GDP)	(annual % growth)	(constant 2010 US$)	(constant LCU)	(current LCU)	(curr US$)
0						
GNI per Capita	27980.9	27999.8	28015.2	28015.2	28023.7	2802
GDP Growth (%)	28003	28010.2	28000.4	28000.4	27999.1	2799
Foreign direct investment, net inflows (BoP, current US$)	26985.6	27000.8	27013.6	27013.6	27017.6	2701
Inflation, consumer prices (annual %)	26006.7	25999.1	25996.6	25996.6	25991.9	2599
Real interest rate (%)	26000.9	26000.5	25997.3	25997.3	25995.8	2599
Net migration	27003.3	27000.4	27002.2	27002.2	27002.7	2700
Official exchange rate (LCU per US$, period average)	27988.6	27999.7	28014.1	28014.1	28016	2801
Unemployment, total (% of total labor force) (modeled ILO estimate)	25005.9	25000.1	24997.9	24997.9	24994.9	2499
Procedures to register property (number)	24002.3	24000.7	23999.7	23999.7	24000.4	2399
Risk premium on lending (lending rate minus treasury bill rate, %)	11001.1	10999.1	10998.2	10998.2	10998.2	1099

Due to the funky list magic earlier, the dtype of the cells is 'object'. Let's change this to numeric so we can use math to convert the sum of correlations + (number of correlations * 1000) into the average.

In [2398]:
```
master_df.dtypes
```

Out[2398]:
```
Agriculture, forestry, and fishing, value added (% of GDP)
object
Agriculture, forestry, and fishing, value added (annual % growth)
object
Agriculture, forestry, and fishing, value added (constant 2010 US$)
object
Agriculture, forestry, and fishing, value added (constant LCU)
object
Agriculture, forestry, and fishing, value added (current LCU)
object
Agriculture, forestry, and fishing, value added (current US$)
object
Agriculture, forestry, and fishing, value added per worker (constant
2010 US$)                   object
Annual freshwater withdrawals, agriculture (% of total freshwater wi
thdrawal)                    object
Child employment in agriculture (% of economically active children a
ges 7-14)                    object
Child employment in agriculture, female (% of female economically ac
tive children ages 7-14)     object
Child employment in agriculture, male (% of male economically active
children ages 7-14)         object
Employment in agriculture (% of total employment) (modeled ILO estim
ate)                         object
Employment in agriculture, female (% of female employment) (modeled
ILO estimate)                object
Employment in agriculture, male (% of female employment) (modeled IL
O estimate)                  object
dtype: object
```

In [2399]:
```
cols = master_df.columns[master_df.dtypes.eq(object)]
master_df[cols] = master_df[cols].apply(pd.to_numeric, errors='coerce
', axis=0)
master_df
```

Out[2399]:

	Agriculture, forestry, and fishing, value added (% of GDP)	Agriculture, forestry, and fishing, value added (annual % growth)	Agriculture, forestry, and fishing, value added (constant 2010 US$)	Agriculture, forestry, and fishing, value added (constant LCU)	Agriculture forestry, ar fishing, val added (current LC
0					
GNI per Capita	27980.897705	27999.812038	28015.246316	28015.246316	28023.7043

GDP Growth (%)	28002.997914	28010.156111	28000.421151	28000.421151	27999.0907
Foreign direct investment, net inflows (BoP, current US$)	26985.589688	27000.773057	27013.641153	27013.641153	27017.6467
Inflation, consumer prices (annual %)	26006.749178	25999.051251	25996.643712	25996.643712	25991.9191
Real interest rate (%)	26000.894821	26000.528003	25997.278847	25997.278847	25995.7775
Net migration	27003.272516	27000.350580	27002.226590	27002.226590	27002.7257
Official exchange rate (LCU per US$, period average)	27988.596030	27999.676339	28014.143081	28014.143081	28016.0183
Unemployment, total (% of total labor force) (modeled ILO estimate)	25005.906204	25000.093049	24997.881130	24997.881130	24994.9398
Procedures to register property (number)	24002.324599	24000.653400	23999.727340	23999.727340	24000.3804
Risk premium on lending (lending rate minus treasury bill rate, %)	11001.104976	10999.109459	10998.157857	10998.157857	10998.1981

An explanation of the below math.

The first two digits 'store' the number of times we added a value to a cell. We access these through (round(x / 1000)), so e.g. round(27980.9) / 1000 returns 28, meaning we added 28 correlations to that cell.

All we have to do is subtract 28,000 from the total and then divide that by 28 to get the average!

In [2400]:
```
x = 27980.9
x = (x - (round(x / 1000) * 1000)) / (round(x / 1000))
x
```

Out[2400]: -0.6821428571428052

Let's implement this:

In [2401]:
```
idx = len(indicators_macro) - 1

while idx > -1:
    master_df.iloc[idx] = (master_df.iloc[idx] - (round(master_df.ilo
c[idx] / 1000) * 1000)) / (round(master_df.iloc[idx] / 1000))
    idx -= 1
```

In [2402]:
```
master_df
```

Out[2402]:

	Agriculture, forestry, and fishing, value added (% of GDP)	Agriculture, forestry, and fishing, value added (annual % growth)	Agriculture, forestry, and fishing, value added (constant 2010 US$)	Agriculture, forestry, and fishing, value added (constant LCU)	Agriculture, forestry, and fishing, value added (current LCU)	Agric fores and fishir value adde (curr US$)
0						
GNI per Capita	-0.682225	-0.006713	0.544511	0.544511	0.846585	0.831
GDP Growth (%)	0.107068	0.362718	0.015041	0.015041	-0.032475	-0.05
Foreign direct investment, net inflows (BoP, current US$)	-0.533715	0.028632	0.505228	0.505228	0.653583	0.644
Inflation, consumer prices (annual %)	0.259584	-0.036490	-0.129088	-0.129088	-0.310800	-0.24
Real interest rate (%)	0.034416	0.020308	-0.104660	-0.104660	-0.162402	-0.14
Net migration	0.121204	0.012984	0.082466	0.082466	0.100954	0.038

Official exchange rate (LCU per US$, period average)	-0.407285	-0.011559	0.505110	0.505110	0.572084	0.471
Unemployment, total (% of total labor force) (modeled ILO estimate)	0.236248	0.003722	-0.084755	-0.084755	-0.202405	-0.21
Procedures to register property (number)	0.096858	0.027225	-0.011361	-0.011361	0.015852	-0.00
Risk premium on lending (lending rate minus treasury bill rate, %)	0.100452	-0.080958	-0.167468	-0.167468	-0.163803	-0.11

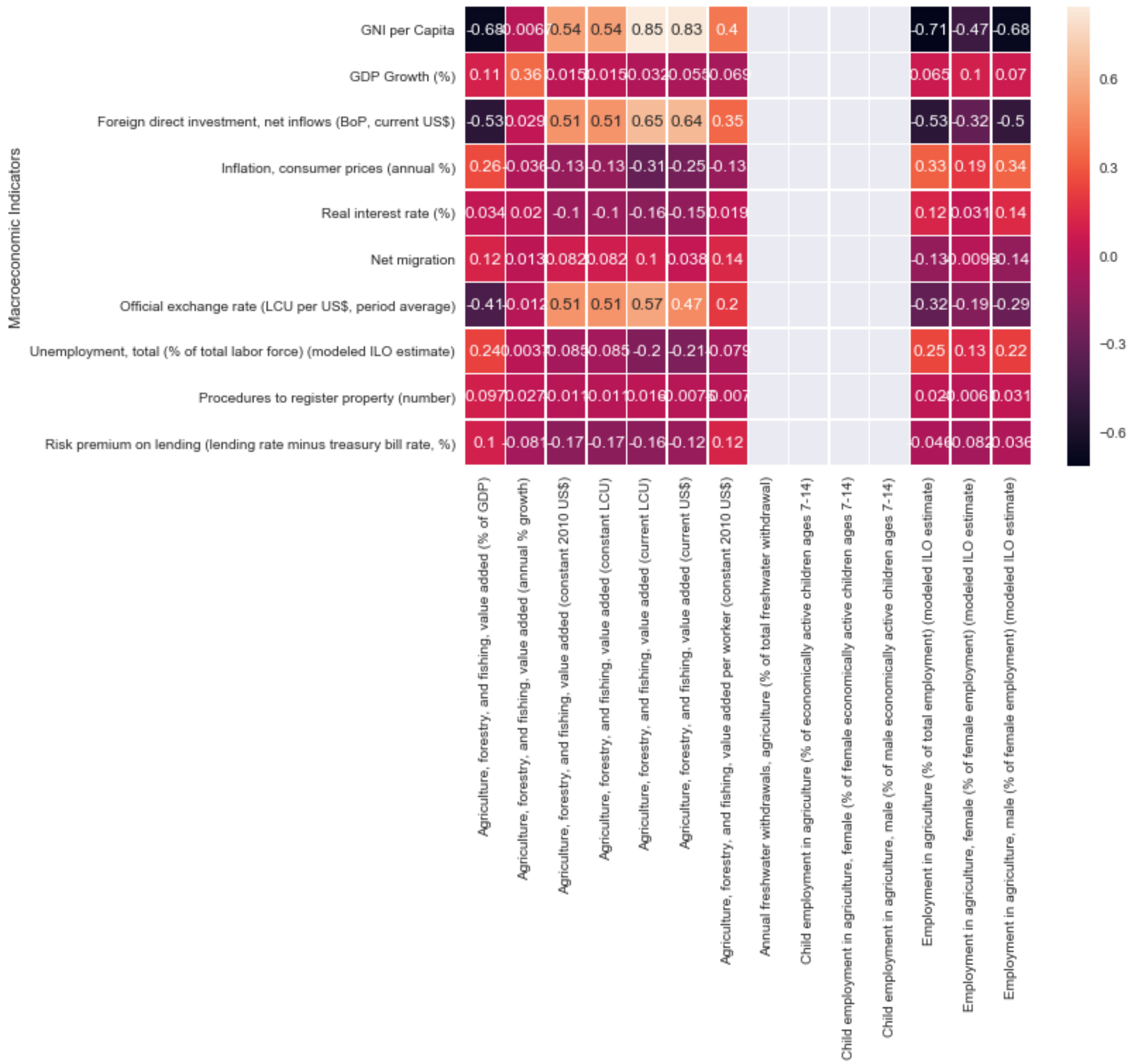
Finally here is what we were ultimately looking for; the end goal of the project: the correlations of each [Agro, Macro] pair.

Now we can draw tons of conclusions at once, instead of fishing in the dark for correlations!

In [2403]:
```
master_df.index.name = 'Macroeconomic Indicators'
```

In [2404]:

```
visualize(master_df)
```



Finally, this last graph summarizes our conclusions by illustrating the total average corralations for all countries.

As we see in the graph, on the vertical axis we have different macroeconomics topics while in our horizontal axis we have the agriculture factors we wanted to analyze.

This graph demonstrates which economic factors have a positive or negative impact on the different agriculture factors. For example, the Gross National Income (GNI) per capita is highly negative corralated to the percentage of total employement in agriculture while it is highly correlated to the total amount of current USD.

This makes sense by simple logic, as the current amount of USD in the economy rises it should positively impact Gross National Income per capita making it rise as well.