```
Research Topic: Global Laundering of Selected Products of Slave Labor

Hypothesis:
Obscured supply chains, conceal the true number of various products being purchased from countries that employ slave labor to produce these products (i.e. at risk countries).

The total 'total imports (in thousands of $US)' and 'total imports (tonnes)' would be higher, if the items in the 'Product' category were tracked from the 'Source country' to the 'Purchasing Country' where they were finally sold as a final product, even if this required tracking them through middle-party purchasing countries which may process or refine the product in some way, before reselling it.
...or, the middle-party purchasing country may simply be part of the supply chain because purchasing from that country tods not carry the same import taxes or possible humanitarian sanctions.

Samples:
Control dataset - Global Slavery Index to be published in 2020.  It would contain the same variables as the 2018 data set that has been imported into this jupyter notebook. Note: This dataset only lists products that have a one transactional point. In other words, they are purchased directly from the 'Source country' by the country listed as the 'Purchasing Country.'

Test dataset - Creation of a new dataframe that uses the same variables as the control dataframe, and the same sample group of products, but includes two additional column indexes, namely: 'total indirect imports (in thousands of $US)' and 'total indirect imports (tonnes)', these columns would include data from imports/purchases of products that originated in the source countries, but were acquired by the country in the "purchasing country" column from a third party country, or countrires.

Outcomes of Interest:
Data points for the following dataset variables-
'total imports (in thousands of $US)'
'total imports (tonnes)'

'total indirect imports (in thousands of $US)'
'total indirect imports (tonnes)'

...in relation to all 'Source country ' and the top three 'Purchasing Country'--in terms of 'total imports (in thousands of $US)' and 'total imports (tonnes)' in the control dataset--for each product.

Analysis:
-Analysis that highlights your experimental hypothesis.
Run t-test on the afforementioned totals variables and get p-values to find out the probability that the null hypothesis, "Tracking supply chain from the 'Source country' through third party countries will now reveal a greater dollar amount and tonage of the products in question being acquired by the 'Purchasing Country.' Adopt a 0.05 level of significance.

```

```
Calculate at the standard error between the totals variables for control and test datasets.

Rollout:
-A rollout plan showing how you would implement and rollout the experiment
Control dataset - data collection rollout: The 2020 rollout for data collection conducted by the Global Slavery Index will proceed as it did in 2018.

Test dataset - data collection (test) rollout: The rollout for data collection for the test dataset will mirror the rollout of data collection for the control dataset, it will look at the same sample group of selected products.  However, it will track to their final destinations, products that are not immediately purchased by the countries that were their 'Purchasing Country' in the 2018 Global Slavery Index dataset.

Evaluation:
-An evaluation plan showing what constitutes success in this experiment
A successful test will:

1. ...produce t-test for each product that shows the possibility that the null hypothesis being true is below the level of significance, i.e. less than 0.05.
(Note: The null hypothesis is: "Tracking supply chain from the 'Source country' through third party countries will now reveal a greater dollar amount and tonage of the products in question being acquired by the 'Purchasing Country.')

2. ...show the additional 'total imports (in thousands of $US)' and 'total imports (tonnes)' that are purchased/imported by from the 'Source country ' by the 'Purchasing Country' that can be found when the supply chain is followed from the 'Sourch counry ' through third party country suppliers to the 'Puchasing country '.
```

In [95]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib inline
import scipy.stats

```

In [96]:
```
sns.set(color_codes=True)
```

download python library called: "nbextensions" for jupyter notebook shortcuts use: conda install nbextensions in the terminal line

In [97]:
```
#Step 1. Show the dataframe.
df = pd.read_excel('FINAL_Global_Slavery_Index_2018_DATA_1549422157.xls
df

```

		risk countries		($US)	country		(tonnes)	cou
72	Peru	10	111352.934290	0.000983	12	114522.048000	0.001054	
73	Total	34328.5	111352.934290	0.308286	52859.93680	114522.048000	0.461570	
74	NaN	Argentina	NaN	NaN	NaN	NaN	NaN	A
75	BRAZIL NUTS/ CHESTNUTS	NaN	NaN	NaN	NaN	NaN	NaN	
76	Bolivia	306.228	308.779779	0.991735	41.96500	43.004000	0.975839	
77	Total	306.228	308.779779	0.991735	41.96500	43.004000	0.975839	
78	NaN	Argentina	NaN	NaN	NaN	NaN	NaN	A
79	COCOA	NaN	NaN	NaN	NaN	NaN	NaN	
80	Cote D'Ivoire	0	177710.971870	0.000000	0.00000	39689.387733	0.000000	
81	Ghana	0	177710.971870	0.000000	0.00000	39689.387733	0.000000	
82	Total	0	177710.971870	0.000000	0.00000	39689.387733	0.000000	
83	NaN	Argentina	NaN	NaN	NaN	NaN	NaN	A
84	DIAMONDS	NaN	NaN	NaN	NaN	NaN	NaN	

In [98]:
```
#Step 2. Show all column names.
df.columns
```

Out[98]:
```
Index(['Source country ',
       'Value (in thousands of $US) of imports from at-risk countries',
       'total imports (in thousands of $US)',
       '% imported from at-risk country ($US)',
       'Quantity (tonnes)  imported from at-risk country',
       'total imports (tonnes)', '%  imported from at-risk country (tonne
s)',
       'Value (in thousands of $US) of imports from at-risk countries.1',
       'total imports (in thousands of $US).1',
       '% imported from at-risk country ($US).1',
       ...
       '% imported from at-risk country ($US).16',
       'Quantity (tonnes)  imported from at-risk country.16',
       'total imports (tonnes).16',
       '%  imported from at-risk country (tonnes).16',
       'Value (in thousands of $US) of imports from at-risk countries.1
7',
       'total imports (in thousands of $US).17',
       '% imported from at-risk country ($US).17',
       'Quantity (tonnes)  imported from at-risk country.17',
       'total imports (tonnes).17',
       '%  imported from at-risk country (tonnes).17'],
      dtype='object', length=109)
```

In [99]:
```
df['Value (in thousands of $US) of imports from at-risk countries'].isnull(
```

```
1                                  COTTON
8                                  BRICKS
15        APPAREL & CLOTHING ACCESSORIES
25                                 CATTLE
32                              SUGARCANE
37                                   GOLD
43                                CARPETS
48                                   COAL
64                                   RICE
69                                 TIMBER
75                 BRAZIL NUTS/ CHESTNUTS
79                                  COCOA
84                               DIAMONDS
88    LAPTOPS, COMPUTERS & MOBILE PHONES
Name: Source country , dtype: object
```

In [100]:
```
1 df['Source country '].value_counts()
```

Out[100]:
```
Total                                  15
India                                   4
Brazil                                  4
China                                   3
North Korea                             3
Pakistan                                3
Bolivia                                 2
Ghana                                   2
Myanmar                                 2
Peru                                    2
Thailand                                2
Malaysia                                2
SUGARCANE                               1
Afghanistan                             1
RICE                                    1
Uzbekistan                              1
Russia                                  1
BRICKS                                  1
Angola                                  1
APPAREL & CLOTHING ACCESSORIES          1
CARPETS                                 1
Tajikistan                              1
South Korea                             1
COCOA                                   1
DIAMONDS                                1
Argentina                               1
Paraguay                                1
COTTON                                  1
Japan                                   1
Dominican Republic                      1
COAL                                    1
GOLD                                    1
LAPTOPS, COMPUTERS & MOBILE PHONES      1
CATTLE                                  1
Cote D'Ivoire                           1
Taiwan                                  1
Democratic Republic of the Congo        1
FISH                                    1
Kazakhstan                              1
TIMBER                                  1
Indonesia                               1
Niger                                   1
BRAZIL NUTS/ CHESTNUTS                  1
Vietnam                                 1
Turkmenistan                            1
Name: Source country , dtype: int64
```

In [101]:
```
df.loc[df['Source country '] == 'FISH']

```

Out[101]:

	Source country	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country	total imports (tonnes)	% imported from at-risk country (tonnes)	Value (in thousands of $US) of imports from at-risk countries.1	total imports (in thousands of $US).1
53	FISH		NaN	NaN	NaN	NaN	NaN	NaN	NaN

1 rows × 109 columns

In [102]:

```
#Step 4. Create a new Category named 'Product',
#and make it display as the second category in the dataframe.


#df.insert(loc, column, value)
#The following code creates a new column, and tells where to put it in
df.insert(1, 'Product', '')

#I can use a loc stateent to assign values.
#Because I am not using a boolean for the rows, I do not need a df in
df.loc[1:7, 'Product'] = 'COTTON'
df.loc[8:14, 'Product'] = 'BRICKS'
df.loc[15:24, 'Product'] = 'APPAREL & CLOTHING ACCESSORIES'
df.loc[25:31, 'Product'] = 'CATTLE'
df.loc[32:36, 'Product'] = 'SUGARCANE'
df.loc[37:42, 'Product'] = 'GOLD'
df.loc[43:47, 'Product'] = 'CARPETS'
df.loc[48:52, 'Product'] = 'COAL'
df.loc[53:63, 'Product'] = 'FISH'
df.loc[64:68, 'Product'] = 'RICE'
df.loc[69:74, 'Product'] = 'TIMBER'
df.loc[75:78, 'Product'] = 'BRAZILE NUTS/ CHESNUTS'
df.loc[79:83, 'Product'] = 'COCOA'
df.loc[84:87, 'Product'] = 'DIAMONDS'
df.loc[88:91, 'Product'] = 'LAPTOPS, COMPUTERS & MOBILE PHONES'

df
```

Out[102]:

	Source country	Product	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country	total imports (tonnes)
0	NaN		Argentina	NaN	NaN	NaN	NaN
1	COTTON	COTTON	NaN	NaN	NaN	NaN	NaN
2	Kazakhstan	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
3	Tajikistan	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
4	Turkmenistan	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
5	Uzbekistan	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
6	Total	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
7	NaN	COTTON	Argentina	NaN	NaN	NaN	NaN

In [103]:
```
#Step 5. Drop all rows in the category named 'Source country ' that ha

drop_rows_with_products_as_source_country = df.loc[df['Value (in thousa
df = df.drop(drop_rows_with_products_as_source_country)
df
```

Out[103]:

	Source country	Product	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country	total imports (tonnes)
0	NaN		Argentina	NaN	NaN	NaN	NaN
2	Kazakhstan	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
3	Tajikistan	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
4	Turkmenistan	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
5	Uzbekistan	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
6	Total	COTTON	0	2481.661895	0.000000	0.00000	1648.641057
7	NaN	COTTON	Argentina	NaN	NaN	NaN	NaN
9	Afghanistan	BRICKS	0	36892.515910	0.000000	0.00000	61926.480756
10	India	BRICKS	69.9216	36892.515910	0.001895	88.84486	61926.480756
11	Myanmar	BRICKS	0	36892.515910	0.000000	0.00000	61926.480756
12	Pakistan	BRICKS	0	36892.515910	0.000000	0.00000	61926.480756
13	Total	BRICKS	69.9216	36892.515910	0.001895	88.84486	61926.480756
14	NaN	BRICKS	Argentina	NaN	NaN	NaN	NaN
16	Argentina	APPAREL & CLOTHING ACCESSORIES	0	325045.932000	0.000000	0.00000	11364.109956
17	Brazil	APPAREL & CLOTHING ACCESSORIES	3315.19	325045.932000	0.010199	136.32080	11364.109956
18	China	APPAREL & CLOTHING ACCESSORIES	157343	325045.932000	0.484063	6933.05500	11364.109956
19	India	APPAREL & CLOTHING ACCESSORIES	21809.2	325045.932000	0.067096	683.58890	11364.109956
20	Malaysia	APPAREL & CLOTHING ACCESSORIES	4397.34	325045.932000	0.013528	219.53770	11364.109956
21	Thailand	APPAREL & CLOTHING ACCESSORIES	5470.28	325045.932000	0.016829	121.81790	11364.109956
22	Vietnam	APPAREL & CLOTHING ACCESSORIES	22792.1	325045.932000	0.070120	534.10650	11364.109956

	Source country	Product	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country	total imports (tonnes)
23	Total	APPAREL & CLOTHING ACCESSORIES	215127	325045.932000	0.661835	8628.42680	11364.109956
24	NaN	APPAREL & CLOTHING ACCESSORIES	Argentina	NaN	NaN	NaN	NaN
26	Bolivia	CATTLE	0	644.605869	0.000000	0.00000	102.115080
27	Brazil	CATTLE	112.135	644.605869	0.173959	22.99500	102.115080
28	Niger	CATTLE	0	644.605869	0.000000	0.00000	102.115080
29	Paraguay	CATTLE	0	644.605869	0.000000	0.00000	102.115080
30	Total	CATTLE	112.135	644.605869	0.173959	22.99500	102.115080
31	NaN	CATTLE	Argentina	NaN	NaN	NaN	NaN
33	Brazil	SUGARCANE	1.20307	354.760642	0.003391	1.00000	277.036428
34	Dominican Republic	SUGARCANE	0	354.760642	0.000000	0.00000	277.036428
...	...	...	...	...	...	...	...
56	Indonesia	FISH	73.546	146663.572510	0.000501	0.20800	40684.243783
57	Japan	FISH	4.437	146663.572510	0.000030	0.15600	40684.243783
58	Russia	FISH	0	146663.572510	0.000000	0.00000	40684.243783
59	South Korea	FISH	6.079	146663.572510	0.000041	1.47400	40684.243783
60	Taiwan	FISH	0	146663.572510	0.000000	0.00000	40684.243783
61	Thailand	FISH	20224.9	146663.572510	0.137900	10174.42000	40684.243783
62	Total	FISH	20309	146663.572510	0.138473	10176.25800	40684.243783
63	NaN	FISH	Argentina	NaN	NaN	NaN	NaN
65	India	RICE	60.8339	4498.253608	0.013524	44.89945	7505.271919
66	Myanmar	RICE	0	4498.253608	0.000000	0.00000	7505.271919
67	Total	RICE	60.8339	4498.253608	0.013524	44.89945	7505.271919
68	NaN	RICE	Argentina	NaN	NaN	NaN	NaN
70	Brazil	TIMBER	34219	111352.934290	0.307302	52739.24000	114522.048000
71	North Korea	TIMBER	0	111352.934290	0.000000	0.00000	114522.048000
72	Peru	TIMBER	109.51	111352.934290	0.000983	120.69680	114522.048000
73	Total	TIMBER	34328.5	111352.934290	0.308286	52859.93680	114522.048000
74	NaN	TIMBER	Argentina	NaN	NaN	NaN	NaN

	Source country	Product	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country	total imports (tonnes)
76	Bolivia	BRAZILE NUTS/ CHESNUTS	306.228	308.779779	0.991735	41.96500	43.004000
77	Total	BRAZILE NUTS/ CHESNUTS	306.228	308.779779	0.991735	41.96500	43.004000
78	NaN	BRAZILE NUTS/ CHESNUTS	Argentina	NaN	NaN	NaN	NaN
80	Cote D'Ivoire	COCOA	0	177710.971870	0.000000	0.00000	39689.387733
81	Ghana	COCOA	0	177710.971870	0.000000	0.00000	39689.387733
82	Total	COCOA	0	177710.971870	0.000000	0.00000	39689.387733
83	NaN	COCOA	Argentina	NaN	NaN	NaN	NaN
85	Angola	DIAMONDS	0	451.591871	0.000000	0.00000	382.234067
86	Total	DIAMONDS	0	451.591871	0.000000	0.00000	382.234067
87	NaN	DIAMONDS	Argentina	NaN	NaN	NaN	NaN
89	China	LAPTOPS, COMPUTERS & MOBILE PHONES	446274	771632.595500	0.578351	4963.12200	6834.794379
90	Malaysia	LAPTOPS, COMPUTERS & MOBILE PHONES	20925.4	771632.595500	0.027118	92.30421	6834.794379
91	Total	LAPTOPS, COMPUTERS & MOBILE PHONES	467200	771632.595500	0.605469	5055.42621	6834.794379

78 rows × 110 columns

In [104]:
```
#Step 6: Create a script that drops all rows in the 'Source country '
#NaN value, except the row at y (top-bottom) axis index 0.  It countai
drop_nan_values_in_source_country = df.loc[df['Source country '].isnul
df = df.drop(drop_nan_values_in_source_country)
df
```

countries

72	Peru	TIMBER	109.51	111352.934290	0.000983	120.69680	1.145220e+05	0
73	Total	TIMBER	34328.5	111352.934290	0.308286	52859.93680	1.145220e+05	0
76	Bolivia	BRAZILE NUTS/ CHESNUTS	306.228	308.779779	0.991735	41.96500	4.300400e+01	0
77	Total	BRAZILE NUTS/ CHESNUTS	306.228	308.779779	0.991735	41.96500	4.300400e+01	0
80	Cote D'Ivoire	COCOA	0	177710.971870	0.000000	0.00000	3.968939e+04	0
81	Ghana	COCOA	0	177710.971870	0.000000	0.00000	3.968939e+04	0
82	Total	COCOA	0	177710.971870	0.000000	0.00000	3.968939e+04	0
85	Angola	DIAMONDS	0	451.591871	0.000000	0.00000	3.822341e+02	0

In [105]:

```
#Step 7: Create a separate dataframes for each country. To see how eac
#look at the documentation in Step 9.

Argentina_df = df.iloc[1:91, 2:8]
Argentina_df.insert(0, 'Purchasing Country', 'Argentina')
Argentina_df.insert(1, 'Product', df[['Product']].copy())
Argentina_df.insert(2, 'Source country ', df[['Source country ']].copy

sns.set(color_codes=True)
Argentina_df
```

57	Argentina	FISH	Japan	4.437	146663.572510	0.000030	0.15600	4.0
58	Argentina	FISH	Russia	0	146663.572510	0.000000	0.00000	4.0
59	Argentina	FISH	South Korea	6.079	146663.572510	0.000041	1.47400	4.0
60	Argentina	FISH	Taiwan	0	146663.572510	0.000000	0.00000	4.0
61	Argentina	FISH	Thailand	20224.9	146663.572510	0.137900	10174.42000	4.0
62	Argentina	FISH	Total	20309	146663.572510	0.138473	10176.25800	4.0
65	Argentina	RICE	India	60.8339	4498.253608	0.013524	44.89945	7.5

In [106]:

```
#Step 8. Show all of the purchasing countries,
#so that I can make a separete dataframe for each.
#Note: Use the iloc index numbers to start and end each dataframe.

#This piece of code will allow me to check any y axis index position
#in a single row (same as x axis or across) in a data frame.
#The first number is the x (across) axis index position.
#The second number is the y (same as top-bottoom) axis index position.

#>>> x = pd.Series([0, 1, 2])
#>>> x.all()   # because one element is zero
#False
#>>> x.any()   # because one (or more) elements are non-zero
#True

purchasing_country_1=pd.Series((df.iloc[0, 2]))
purchasing_country_2=pd.Series(df.iloc[0, 8])
purchasing_country_3=pd.Series(df.iloc[0, 14])
purchasing_country_4=pd.Series(df.iloc[0, 20])
purchasing_country_5=pd.Series(df.iloc[0, 26])
purchasing_country_6=pd.Series(df.iloc[0, 32])
purchasing_country_7=pd.Series(df.iloc[0, 38])
purchasing_country_8=pd.Series(df.iloc[0, 44])
purchasing_country_9=pd.Series(df.iloc[0, 50])
purchasing_country_10=pd.Series(df.iloc[0, 56])
purchasing_country_11=pd.Series(df.iloc[0, 62])
purchasing_country_12=pd.Series(df.iloc[0, 68])
purchasing_country_13=pd.Series(df.iloc[0, 74])
purchasing_country_14=pd.Series(df.iloc[0, 80])
purchasing_country_15=pd.Series(df.iloc[0, 86])
purchasing_country_16=pd.Series(df.iloc[0, 92])
purchasing_country_17=pd.Series(df.iloc[0, 98])
purchasing_country_18=pd.Series(df.iloc[0, 104])


print(purchasing_country_1.any())
print(purchasing_country_2.any())
print(purchasing_country_3.any())
print(purchasing_country_4.any())
print(purchasing_country_5.any())
print(purchasing_country_6.any())
print(purchasing_country_7.any())
print(purchasing_country_8.any())
print(purchasing_country_9.any())
print(purchasing_country_10.any())
print(purchasing_country_11.any())
print(purchasing_country_12.any())
print(purchasing_country_13.any())
print(purchasing_country_14.any())
print(purchasing_country_15.any())
print(purchasing_country_16.any())
print(purchasing_country_17.any())
print(purchasing_country_18.any())

```

```
Argentina
Australia
```

```
Brazil
Canada
China
France
Germany
Indonesia
Italy
Japan
South Korea
Mexico
Russia
Saudi Arabia
India
Turkey
UK
US
```

In [107]:

```
reate a separate dataframe for Australia,
late it with its related variables(columns),
oints(rows).  (Start with Australia, as example code for all other countrys
lf = df.iloc[1:91, 8:14]

Create a column called 'Purchasing Country' in the dataframe Australia_df,
ountry as 'Australia' for index position 0 in all of the rows for that colu
lf.insert(0, 'Purchasing Country', 'Australia')

Insert the 'Product' column and all variables for each row in that column,
riginal df into Australia_df at index position 1.
lf.insert(1, 'Product', df[['Product']].copy())


Insert the 'Source country ' column and all variables for each row in that
riginal df into Australia_df at index position 2.
lf.insert(2, 'Source country ', df[['Source country ']].copy())



Rename the columns in Australia_df so that their names are identical to the
iginal dataframe.
lf = Australia_df.rename(index=str, columns={'Value (in thousands of $US) o
                                             'total imports (in thousands of $US).1' :
                                             '% imported from at-risk country ($US).1'
                                             'Quantity (tonnes)  imported from at-risk
                                             'total imports (tonnes).1' : 'total import
                                             '%  imported from at-risk country (tonnes

,index_codes=True)
df
```

In [108]:
```
#Create Brazil_df.

Brazil_df = df.iloc[1:91, 14:20]
Brazil_df.insert(0, 'Purchasing Country', 'Brazil')
Brazil_df.insert(1, 'Product', df[['Product']].copy())
Brazil_df.insert(2, 'Source country ', df[['Source country ']].copy())
Brazil_df = Brazil_df.rename(index=str, columns={'Value (in thousands of $US) of
                                    'total imports (in thousands of $US).
                                    '% imported from at-risk country ($US
                                    'Quantity (tonnes)  imported from at-
                                    'total imports (tonnes).2' : 'total i
                                    '%  imported from at-risk country (to

sns.set(color_codes=True)
#Brazil_df
```

In [109]:
```
#Step 11. Create Canada_df.

Canada_df = df.iloc[1:91, 20:26]
Canada_df.insert(0, 'Purchasing Country', 'Canada')
Canada_df.insert(1, 'Product', df[['Product']].copy())
Canada_df.insert(2, 'Source country ', df[['Source country ']].copy())
Canada_df = Canada_df.rename(index=str, columns={'Value (in thousands
                                        'total imports (in thousan
                                        '% imported from at-risk c
                                        'Quantity (tonnes)  import
                                        'total imports (tonnes).3'
                                        '%  imported from at-risk

sns.set(color_codes=True)
#Canada_df
```

In [110]:
```
#Step 12. Create China_df.

China_df = df.iloc[1:91, 26:32]
China_df.insert(0, 'Purchasing Country', 'China')
China_df.insert(1, 'Product', df[['Product']].copy())
China_df.insert(2, 'Source country ', df[['Source country ']].copy())
China_df = China_df.rename(index=str, columns={'Value (in thousands of
                                        'total imports (in thousan
                                        '% imported from at-risk c
                                        'Quantity (tonnes)  import
                                        'total imports (tonnes).4'
                                        '%  imported from at-risk

sns.set(color_codes=True)
#China_df
```

In [111]:
```
#Step 13. Create France_df.

France_df = df.iloc[1:91, 32:38]
France_df.insert(0, 'Purchasing Country', 'France')
France_df.insert(1, 'Product', df[['Product']].copy())
France_df.insert(2, 'Source country ', df[['Source country ']].copy())
France_df = France_df.rename(index=str, columns={'Value (in thousands of $US) o
                                        'total imports (in thousands of $US
                                        '% imported from at-risk country ($
                                        'Quantity (tonnes)  imported from a
                                        'total imports (tonnes).5' : 'total
                                        '%  imported from at-risk country (
sns.set(color_codes=True)
France_df
```

In [112]:
```
#Step 14. Create Germany_df.

Germany_df = df.iloc[1:91, 38:44]
Germany_df.insert(0, 'Purchasing Country', 'Germany')
Germany_df.insert(1, 'Product', df[['Product']].copy())
Germany_df.insert(2, 'Source country ', df[['Source country ']].copy()
Germany_df = Germany_df.rename(index=str, columns={'Value (in thousand
                                        'total imports (in thousan
                                        '% imported from at-risk c
                                        'Quantity (tonnes)  import
                                        'total imports (tonnes).6'
                                        '%  imported from at-risk
sns.set(color_codes=True)
#Germany_df
```

In [113]:
```
#Step 15. Create Indonesia_df.

Indonesia_df = df.iloc[1:91, 44:50]
Indonesia_df.insert(0, 'Purchasing Country', 'Indonesia')
Indonesia_df.insert(1, 'Product', df[['Product']].copy())
Indonesia_df.insert(2, 'Source country ', df[['Source country ']].copy
Indonesia_df = Indonesia_df.rename(index=str, columns={'Value (in thou
                                        'total imports (in thousan
                                        '% imported from at-risk c
                                        'Quantity (tonnes)  import
                                        'total imports (tonnes).7'
                                        '%  imported from at-risk
sns.set(color_codes=True)
#Indonesia_df
```

In [114]:
```
#Step 16. Create Italy_df.

Italy_df = df.iloc[1:91, 50:56]
Italy_df.insert(0, 'Purchasing Country', 'Italy')
Italy_df.insert(1, 'Product', df[['Product']].copy())
Italy_df.insert(2, 'Source country ', df[['Source country ']].copy())
Italy_df = Italy_df.rename(index=str, columns={'Value (in thousands of
                                               'total imports (in thousan
                                               '% imported from at-risk c
                                               'Quantity (tonnes)  import
                                               'total imports (tonnes).8'
                                               '%  imported from at-risk
sns.set(color_codes=True)
#Italy_df
```

In [115]:
```
#Step 17. Create Japan_df.

Japan_df = df.iloc[1:91, 56:62]
Japan_df.insert(0, 'Purchasing Country', 'Japan')
Japan_df.insert(1, 'Product', df[['Product']].copy())
Japan_df.insert(2, 'Source country ', df[['Source country ']].copy())
Japan_df = Japan_df.rename(index=str, columns={'Value (in thousands of
                                               'total imports (in thousan
                                               '% imported from at-risk c
                                               'Quantity (tonnes)  import
                                               'total imports (tonnes).9'
                                               '%  imported from at-risk
sns.set(color_codes=True)
#Japan_df

```

In [116]:
```
#Step 18. Create South_Korea_df.

South_Korea_df = df.iloc[1:91, 62:68]
South_Korea_df.insert(0, 'Purchasing Country', 'South Korea')
South_Korea_df.insert(1, 'Product', df[['Product']].copy())
South_Korea_df.insert(2, 'Source country ', df[['Source country ']].co
South_Korea_df = South_Korea_df.rename(index=str, columns={'Value (in
                                               'total imports (in thousan
                                               '% imported from at-risk c
                                               'Quantity (tonnes)  import
                                               'total imports (tonnes).10
                                               '%  imported from at-risk
sns.set(color_codes=True)
#South_Korea_df

```

In [117]:
```
#Step 19. Create Mexico_df.

Mexico_df = df.iloc[1:91, 68:74]
Mexico_df.insert(0, 'Purchasing Country', 'Mexico')
Mexico_df.insert(1, 'Product', df[['Product']].copy())
Mexico_df.insert(2, 'Source country ', df[['Source country ']].copy())
Mexico_df = Mexico_df.rename(index=str, columns={'Value (in thousands
                                                 'total imports (in thousan
                                                 '% imported from at-risk c
                                                 'Quantity (tonnes)  import
                                                 'total imports (tonnes).11
                                                 '%  imported from at-risk

sns.set(color_codes=True)
#Mexico_df

```

In [118]:
```
#Step 20. Create Russia_df.

Russia_df = df.iloc[1:91, 74:80]
Russia_df.insert(0, 'Purchasing Country', 'Russia')
Russia_df.insert(1, 'Product', df[['Product']].copy())
Russia_df.insert(2, 'Source country ', df[['Source country ']].copy())
Russia_df = Russia_df.rename(index=str, columns={'Value (in thousands of $U
                                                 'total imports (in thousands of
                                                 '% imported from at-risk countr
                                                 'Quantity (tonnes)  imported fr
                                                 'total imports (tonnes).12' :
                                                 '%  imported from at-risk count

sns.set(color_codes=True)
#Russia_df

```

In [119]:
```
#Step 21. Create Saudi_Arabia_df.

Saudi_Arabia_df = df.iloc[1:91, 80:86]
Saudi_Arabia_df.insert(0, 'Purchasing Country', 'Saudi Arabia')
Saudi_Arabia_df.insert(1, 'Product', df[['Product']].copy())
Saudi_Arabia_df.insert(2, 'Source country ', df[['Source country ']].copy(
Saudi_Arabia_df = Saudi_Arabia_df.rename(index=str, columns={'Value (in th
                                                 'total imports (in thousands c
                                                 '% imported from at-risk count
                                                 'Quantity (tonnes)  imported f
                                                 'total imports (tonnes).13' :
                                                 '%  imported from at-risk cour

sns.set(color_codes=True)
#Saudi_Arabia_df

```

In [120]:
```
#Step 22. Create India_df.

India_df = df.iloc[1:91, 86:92]
India_df.insert(0, 'Purchasing Country', 'India')
India_df.insert(1, 'Product', df[['Product']].copy())
India_df.insert(2, 'Source country ', df[['Source country ']].copy())
India_df = India_df.rename(index=str, columns={'Value (in thousands of
                                               'total imports (in thousan
                                               '% imported from at-risk c
                                               'Quantity (tonnes)  import
                                               'total imports (tonnes).14
                                               '%  imported from at-risk
sns.set(color_codes=True)
#India_df

```

In [121]:
```
#Step 23. Create Trukey_df.

Turkey_df = df.iloc[1:91, 92:98]
Turkey_df.insert(0, 'Purchasing Country', 'Turkey')
Turkey_df.insert(1, 'Product', df[['Product']].copy())
Turkey_df.insert(2, 'Source country ', df[['Source country ']].copy())
Turkey_df = Turkey_df.rename(index=str, columns={'Value (in thousands of
                                              'total imports (in thousands
                                              '% imported from at-risk cou
                                              'Quantity (tonnes)  imported
                                              'total imports (tonnes).15'
                                              '%  imported from at-risk co
sns.set(color_codes=True)
#Turkey_df


```

In [122]:
```
#Step 24. Create UK_df.


UK_df = df.iloc[1:91, 98:104]
UK_df.insert(0, 'Purchasing Country', 'UK')
UK_df.insert(1, 'Product', df[['Product']].copy())
UK_df.insert(2, 'Source country ', df[['Source country ']].copy())
UK_df = UK_df.rename(index=str, columns={'Value (in thousands of $US)
                                               'total imports (in thousan
                                               '% imported from at-risk c
                                               'Quantity (tonnes)  import
                                               'total imports (tonnes).16
                                               '%  imported from at-risk
sns.set(color_codes=True)
#UK_df
```

In [123]:
```
#Step 25. Create US_df.

US_df = df.iloc[1:91, 104:110]
US_df.insert(0, 'Purchasing Country', 'US')
US_df.insert(1, 'Product', df[['Product']].copy())
US_df.insert(2, 'Source country ', df[['Source country ']].copy())
US_df = US_df.rename(index=str, columns={'Value (in thousands of $US) of
                                         'total imports (in thousands
                                         '% imported from at-risk cou
                                         'Quantity (tonnes)  imported
                                         'total imports (tonnes).17'
                                         '%  imported from at-risk co
sns.set(color_codes=True)
US_df

```

Out[123]:

	Purchasing Country	Product	Source country	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country	t
2	US	COTTON	Kazakhstan	0	1.028213e+04	0.000000	0.000000e+00	6
3	US	COTTON	Tajikistan	0	1.028213e+04	0.000000	0.000000e+00	6
4	US	COTTON	Turkmenistan	0	1.028213e+04	0.000000	0.000000e+00	6
5	US	COTTON	Uzbekistan	0	1.028213e+04	0.000000	0.000000e+00	6
6	US	COTTON	Total	0	1.028213e+04	0.000000	0.000000e+00	6
9	US	BRICKS	Afghanistan	0	3.765840e+05	0.000000	0.000000e+00	7
10	US	BRICKS	India	4657.89	3.765840e+05	0.012369	3.500615e+03	7
11	US	BRICKS	Myanmar	0	3.765840e+05	0.000000	0.000000e+00	7

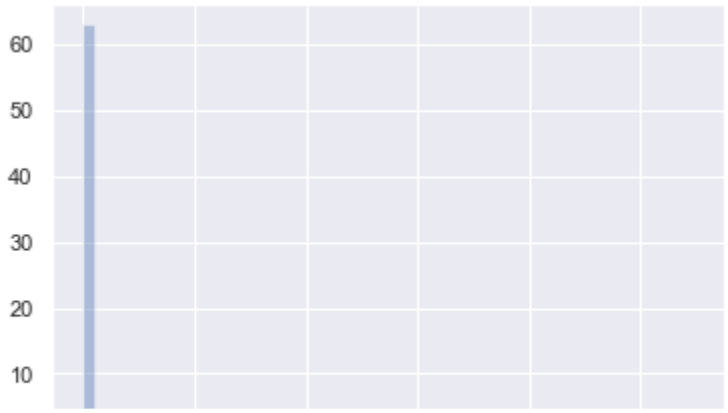
In [124]:
```
#Step 26. Concatenate all dataframes that begin with a country name.

cc_df = pd.concat([Argentina_df, Australia_df, Brazil_df, Canada_df, C
           France_df, Germany_df, Indonesia_df, Italy_df, Japan_df,
           South_Korea_df, Mexico_df, Russia_df, Saudi_Arabia_df,
           India_df, Turkey_df, UK_df, US_df], ignore_index=True, sort

cc_df

```

Out[124]:

	Purchasing Country	Product	Source country	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country
0	Argentina	COTTON	Kazakhstan	0	2.481662e+03	0.000000	0.000000
1	Argentina	COTTON	Tajikistan	0	2.481662e+03	0.000000	0.000000
2	Argentina	COTTON	Turkmenistan	0	2.481662e+03	0.000000	0.000000
3	Argentina	COTTON	Uzbekistan	0	2.481662e+03	0.000000	0.000000
4	Argentina	COTTON	Total	0	2.481662e+03	0.000000	0.000000
5	Argentina	BRICKS	Afghanistan	0	3.689252e+04	0.000000	0.000000
6	Argentina	BRICKS	India	69.9216	3.689252e+04	0.001895	88.844860
7	Argentina	BRICKS	Myanmar	0	3.689252e+04	0.000000	0.000000

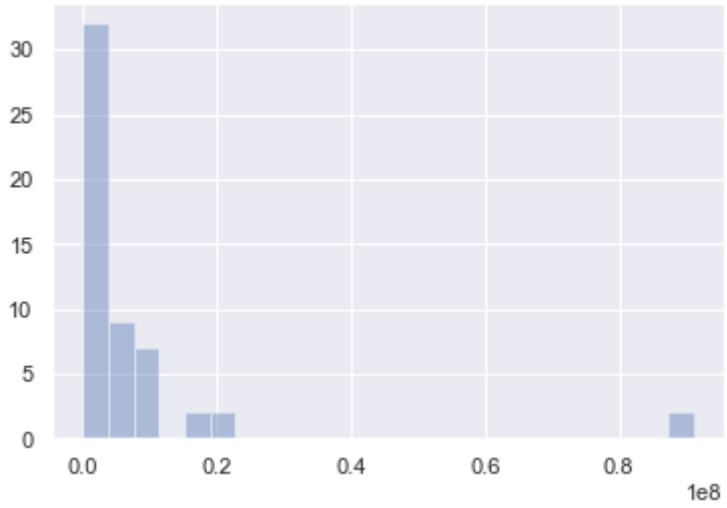
In [125]:
```
#Step 27. Create histograms of categorical variables.

#loc statements deal with rows, then columns.
all_cotton = cc_df.loc[(cc_df['Product'] == 'COTTON') & (cc_df['Source
                       'Value (in thousands of $US) of imports from at

#y axis = the number of rows for which the quantity range in teh x axi

sns.distplot([float (i) for i in all_cotton], kde=False, rug=False, co
plt.show()

```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



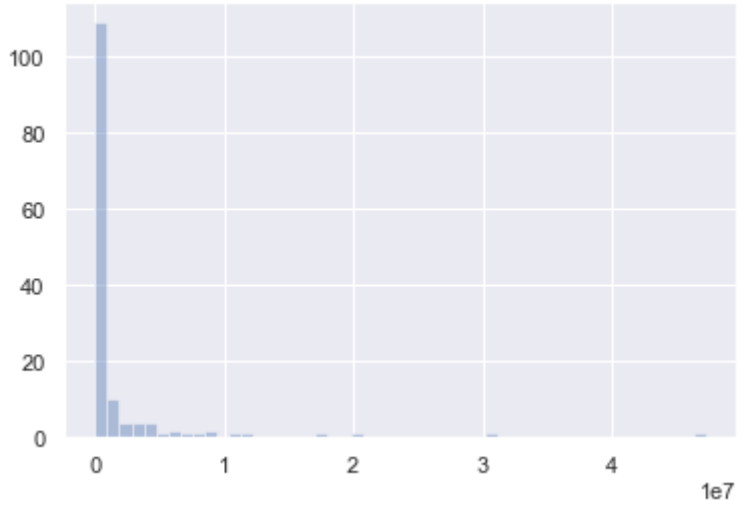
In [127]:
```
all_laptops_computers_mobile_phones = cc_df.loc[(cc_df['Product'] == '
                        'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_laptops_computers_mobile_phones],
plt.show()

```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



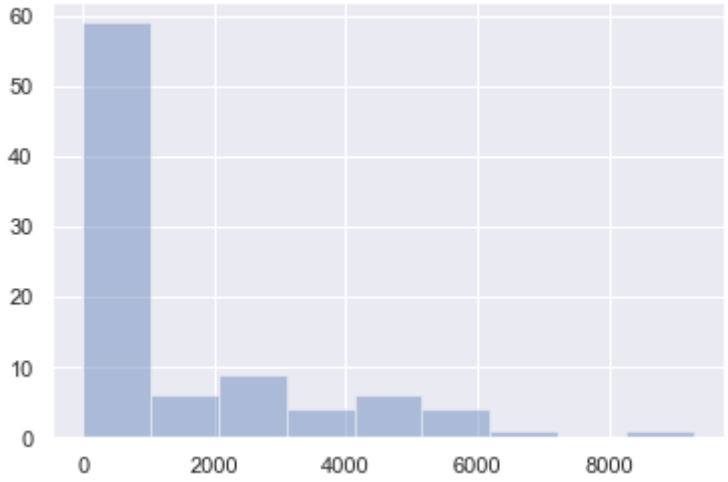
In [128]:
```
all_apparel_and_clothing_accessories = cc_df.loc[(cc_df['Product'] ==
                        'Value (in thousands of $US) of imports from at-

sns.distplot([float (i) for i in all_apparel_and_clothing_accessories],
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



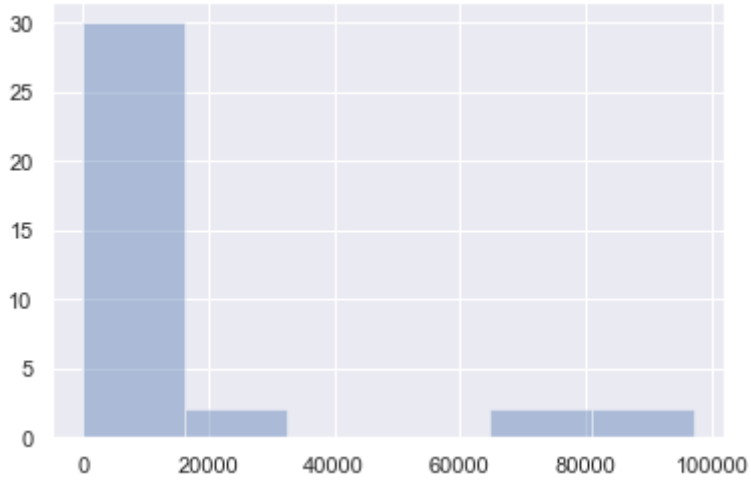
In [129]:
```
all_bricks = cc_df.loc[(cc_df['Product'] == 'BRICKS'),
                       'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_bricks], kde=False, rug=False, co
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



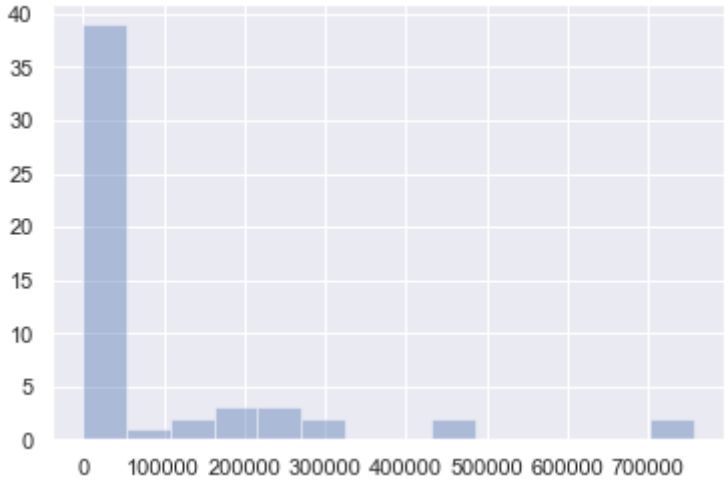
In [130]:
```
all_diamonds = cc_df.loc[(cc_df['Product'] == 'DIAMONDS'),
                         'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_diamonds], kde=False, rug=False,
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



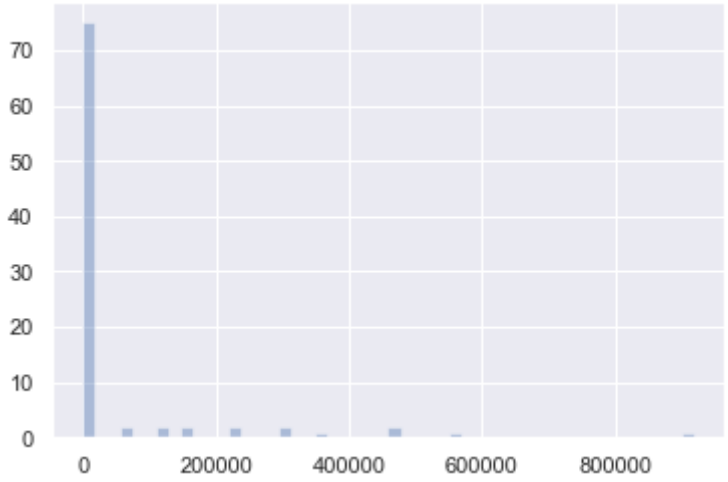
In [131]:
```
all_sugarcane = cc_df.loc[(cc_df['Product'] == 'SUGARCANE'),
                          'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_sugarcane], kde=False, rug=False,
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



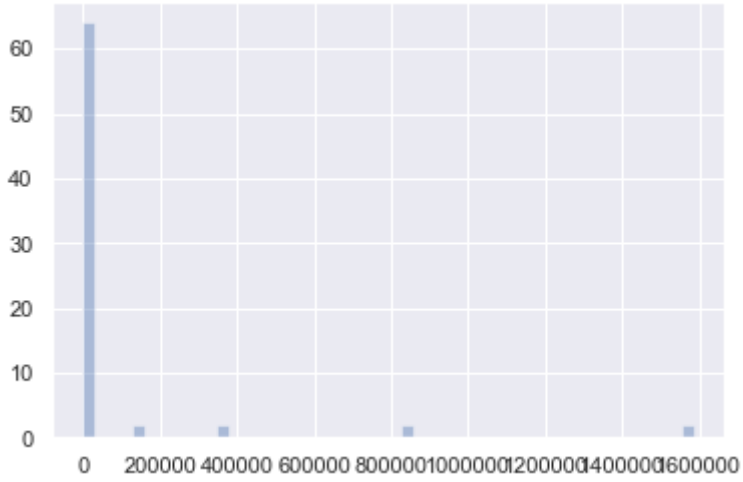
In [132]:
```
all_cattle = cc_df.loc[(cc_df['Product'] == 'CATTLE'),
                       'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_cattle], kde=False, rug=False, co
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [133]:
```
all_gold = cc_df.loc[(cc_df['Product'] == 'GOLD'),
                     'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_gold], kde=False, rug=False, colo
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [ ]:
```
1
```

In [134]:
```
all_coal = cc_df.loc[(cc_df['Product'] == 'COAL'),
                     'Value (in thousands of $US) of imports from at
all_coal.value_counts()
#sns.distplot([float (i) for i in all_coal], kde=False, rug=False, col
#plt.show()
```

Out[134]:
```
0.0000         50
2456.4829       2
954000.0000     2
Name: Value (in thousands of $US) of imports from at-risk countries, dtyp
e: int64
```

In [135]:
```
cc_df.loc[(cc_df['Product'] == 'COAL') &
          (cc_df['Value (in thousands of $US) of imports from at-risk

```

Out[135]:

Purchasing Country	Product	Source country	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country	total imports (tonnes)	% imported from at-risk country (tonnes)

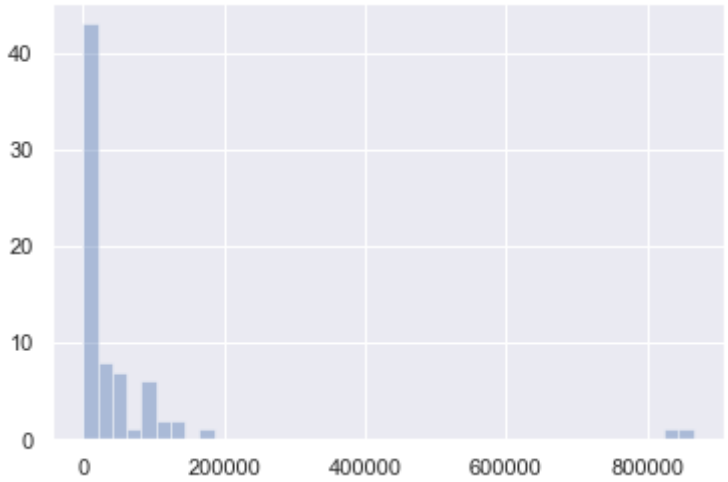
In [136]:
```
#basic row selection.  The first argument is rows.
#If choose to put in a second argument, that will be the column name a
#for example: cc_df.loc[33:38, 'Scource country ']
cc_df.loc[33:38]
```

Out[136]:

	Purchasing Country	Product	Source country	Value (in thousands of $US) of imports from at-risk countries	total imports (in thousands of $US)	% imported from at-risk country ($US)	Quantity (tonnes) imported from at-risk country	total imports (tonnes)	imp fr c (t
33	Argentina	COAL	North Korea	0	132037.60338	0.0	0.0	1.505572e+06	
34	Argentina	COAL	Pakistan	0	132037.60338	0.0	0.0	1.505572e+06	
35	Argentina	COAL	Total	0	132037.60338	0.0	0.0	1.505572e+06	
36	Argentina	FISH	FISH		NaN	NaN	NaN	NaN	
37	Argentina	FISH	China	0	146663.57251	0.0	0.0	4.068424e+04	
38	Argentina	FISH	Ghana	0	146663.57251	0.0	0.0	4.068424e+04	

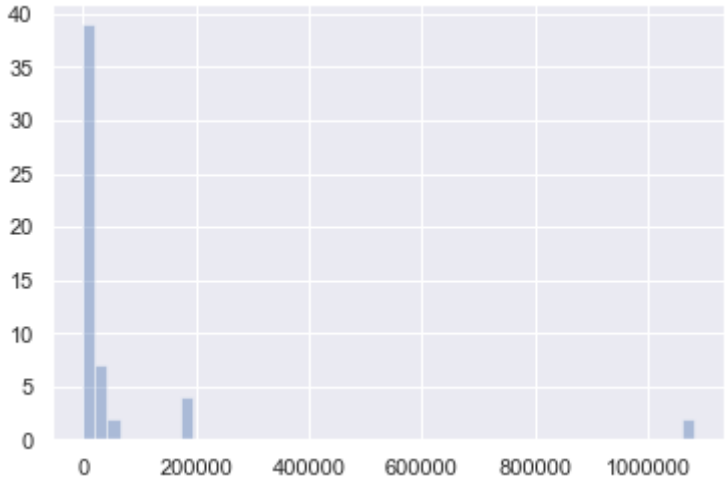
In [137]:
```
all_timber = cc_df.loc[(cc_df['Product'] == 'TIMBER'),
                       'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_timber], kde=False, rug=False, co
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



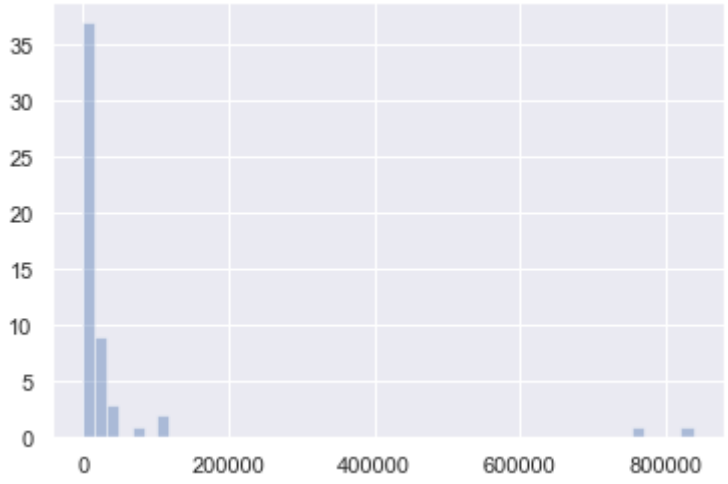
In [138]:
```
all_rice = cc_df.loc[(cc_df['Product'] == 'RICE'),
                         'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_rice], kde=False, rug=False, colo
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



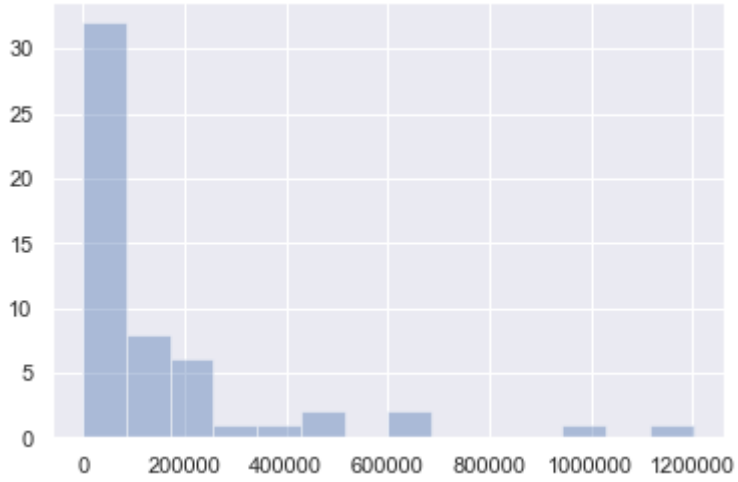
In [139]:
```
all_carpets = cc_df.loc[(cc_df['Product'] == 'CARPETS'),
                        'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_carpets], kde=False, rug=False, c
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [140]:
```
all_cocoa = cc_df.loc[(cc_df['Product'] == 'COCOA'),
                      'Value (in thousands of $US) of imports from at

sns.distplot([float (i) for i in all_cocoa], kde=False, rug=False, col
plt.show()
```

```
/Users/katrinajohnson/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index, `arr[np.array(s
eq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [ ]:
```
#Evaluation - t-test

#Example Code I will use to run t-tests
#on data this proposed control and test datasets will yield.  I will use
#to discover the p-values.

# Generate a test statistic comparing test and control on treatment date
#print('T-Test Results by Date')

#for date in experiment_data.treatment_date.unique():
    #dated_data = experiment_data[experiment_data.treatment_date == date
    #print(stats.ttest_ind(dated_data[dated_data.test == 1].is_signed_up
                          #dated_data[dated_data.test == 0].is_signed_up
```