Data_Coverage

August 16, 2017

```
In [1]: from IPython.display import HTML
        HTML('''<script>
        code_show=true;
        function code_toggle() {
         if (code_show){
         $('div.input').hide();
         } else {
         $('div.input').show();
         }
         code_show = !code_show
        }
        $( document ).ready(code_toggle);
        </script>''')

Out[1]: <IPython.core.display.HTML object>

In [2]: from IPython.display import HTML
        HTML('''
        <style>
            .yourDiv {position: fixed;top: 100px; right: 0px;
                      background: white;
                      height: 100%;
                      width: 175px;
                      padding: 10px;
                      z-index: 10000}
        </style>
        <script>
        function showthis(url) {
                window.open(url, "pres",
                        "toolbar=yes,scrollbars=yes,resizable=yes,top=10,left=400,width=500,heig
                return(false);
        }
        </script>

        <div class=yourDiv>
            <h4>MENU</h4><br>
            <a href=#Data>1.Data</a><br>
            <a href=#SpatialCoverage>2. Spatial Coverage</a><br><br>

            <a href=#Top>Top</a><br>
            <a href="javascript:code_toggle()">Toggle Code On/Off</a><br>
            <a href=#LeftOff>Left Off Here</a><br>
            <a href='https://vinnyricciardi.github.io/farmsize_site/'>Site Index</a><br>
        </div>
        ''')
```

```
Out[2]: <IPython.core.display.HTML object>
```

Data Coverage Overview

```
In [1]: # Import dependencies
        import warnings
        warnings.filterwarnings('ignore')
        import pandas as pd
        import geopandas as gpd
        import seaborn as sns
        from matplotlib import pyplot as plt
        import matplotlib.pyplot as plt
        from matplotlib.path import Path
        import matplotlib.patches as patches
        from matplotlib.pyplot import cm
        import matplotlib as mpl
        import numpy as np
        import re
        import geopy
        import mpld3
        import plotly.plotly as py
        import cmocean

        pd.set_option('display.max_columns', 500)
        %matplotlib inline
```

```
In [2]: PATH1 = '/Users/Vinny_Ricciardi/Documents/Data_Library_Big/Survey/Global/Farm_Size/Data/
        df = pd.read_csv(PATH1, low_memory=False)
        PATH2 = '/Users/Vinny_Ricciardi/Documents/Data_Library_Big/Survey/Global/FaoStat/FAOSTAT
        fao = pd.read_csv(PATH2)
```

```
In [3]: def perc_global(data, how='area'):

            if how is 'area':
                element = 'Area harvested'

            elif how is 'production':
                element = 'Production'

            tmp = fao.copy()
            tmp1 = tmp[(tmp['Element'] == element)]
            tmp1 = tmp1.sort_values(['Country', 'Item Code', 'Year'])

            multi_index = pd.MultiIndex.from_product([tmp1['Country'].unique(),
                                                      tmp1['Item Code'].unique(),
                                                      tmp1['Year'].unique()],
                                                     names=['Country', 'Item Code', 'Year'])

            tmp1 = tmp1.set_index(['Country', 'Item Code', 'Year']).reindex(multi_index).reset_i
            tmp1 = tmp1.set_index(['Country', 'Item Code', 'Year'])
            tmp1 = tmp1.interpolate(method='linear',
                                    axis=0,
                                    limit_direction='both')
            tmp1 = tmp1.reset_index()
            tmp1 = tmp1[(tmp1['Year'] == float(avg_year))]
            tmp1['overlap1'] = np.in1d(tmp1['Country'], df.NAME_0.unique())
            tmp1['overlap2'] = np.in1d(tmp1['Item Code'], df.Item_Code.unique())

            data_in = tmp1.query("overlap1 == True & overlap2 == True")['Value'].sum()
            data_out = tmp1['Value'].sum()
            perc_rep = round(100 * (data_in / data_out), 2)

            return perc_rep
```

```
In [4]: num_countries = len(df.NAME_0.unique())
        num_crops = len(df.Crop.unique())
        num_crops_fao = len(df.query("production_Food_kcal == production_Food_kcal").Crop.unique
        num_admin = len(df.shpID.unique())
        # num_obs = len(df)
        num_micro = len(df.query("microdata == 1").NAME_0.unique())
        num_tab = len(df.query("microdata == 0").NAME_0.unique())
        num_sur = len(df.query("cen_sur == 'sur'").NAME_0.unique())
        num_cen = len(df.query("cen_sur == 'cen'").NAME_0.unique())
        avg_year = int(round(df.year.mean(), 0))
        min_year = df.year.min().astype(int)
        max_year = df.year.max().astype(int)
        perc_area = perc_global(fao, how='area')
        perc_prod = perc_global(fao, how='production')
```

Data

General - Our dataset caputres the amount crop production by farms size - 58 countries are represented at either the national or subnational level. - In total, there are 2804 national or subnational units. - This captures 16.35% of global harvest area and 20.73% of global crop production - There are 151 crops, of which we were able to match 114 with the FAO's database to calculate the amount of crops produced by farm size class for food, feed, waste, seed, proccessing, and other in terms of kcal. - We used 37 tabulated datasets, and 21 microdatasets (i.e., data at the household record level) - 41 agricultural censuses were used. Where census data was not used, nationally or subnationally representative household surveys were used (17 in total). - On average the data was from 2011, with the oldest datasets from 2001 and the newest from 2013 - We used the World Census of Agriculture's (WCA) farm size categories to be consistent with other studies.

Click here to see our main findings

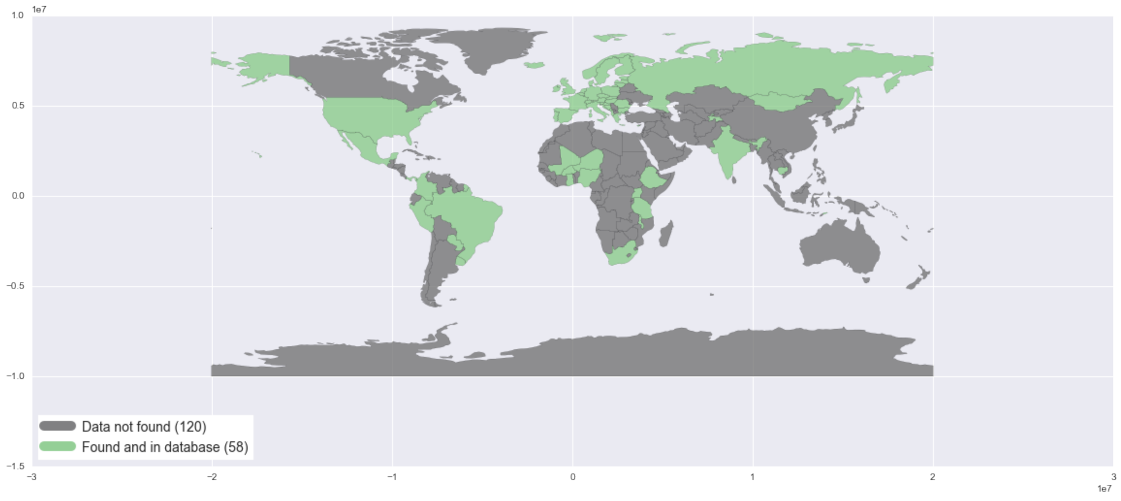
Spatial Coverage

```
In [7]: df['NAME_0'].replace(['United States of America'], ['United States'], inplace=True)
        df['NAME_0'].replace(['Bosnia and Herzegovina'], ['Bosnia and Herz.'], inplace=True)
        df['NAME_0'].replace(['United Republic of Tanzania'], ['Tanzania'], inplace=True)
        df['NAME_0'].replace(['Russian Federation'], ['Russia'], inplace=True)
        df['NAME_0'].replace(['Czech Republic'], ['Czech Rep.'], inplace=True)
        df['NAME_0'].replace(['Czech Republic'], ['Czech Rep.'], inplace=True)
        df['NAME_0'].replace(['Czech Republic'], ['Czech Rep.'], inplace=True)
```

To do: - Map to be replaced with map of sub-national units (and in a better projection!) after we spatially match all admin units

```
In [8]: # Set all plotting params:
        # title_sz = 20
        # x_lab_tick_sz = 18
        # y_lab_tick_sz = 18
        # x_lab_label_sz = 18
        # y_lab_label_sz = 18
        # lengend_sz = 16

        pivoted = pd.pivot_table(df,
                                 index='NAME_0',
                                 values='Crop',
                                 aggfunc=lambda x: len(x.unique()))
        pivoted = pivoted.reset_index()
        pivoted = pivoted.sort_values('Crop', ascending=False)
        pivoted['Data_Available'] = pivoted['Crop'].astype(int)

        world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

        world = world.to_crs(epsg=3786)

        world = pd.merge(world, pivoted,
                         how='outer',
                         left_on='name',
                         right_on='NAME_0')

        world['Orig_crop'] = world['Crop'].fillna(0)
        world['coverage'] = np.where(world['Crop'] > 0,
                                     'Found and downloaded',
                                     np.where(world['Crop'] == -1,
                                              'Found not downloaded',
                                              'No data found'))
```

```
warnings.filterwarnings('ignore')

x = len(pivoted.NAME_0.unique())

try:
    fig, ax = plt.subplots(figsize=(20, 10))
    ax.set_aspect('equal')
    world.plot(column='coverage', cmap='Accent', ax=ax, alpha=0.7, linewidth=0.1)  #cmoc
except:
    pass

ndf, fad = world.coverage.value_counts()
cmap_ = cmocean.tools.get_dict(cmocean.cm.deep, N=4)

p1 = mpl.lines.Line2D([], [],
                             color=[x / 255. for x in [128, 128, 130]],
                             linewidth=10,
                             label='Data not found ({})'.format(ndf))
p2 = mpl.lines.Line2D([], [],
                             color=[x / 255. for x in [148, 207, 150]],
                             linewidth=10,
                             label='Found and in database ({})'.format(fad))

handles = [p1, p2]
labels = [h.get_label() for h in handles]

legend = ax.legend(handles=handles, labels=labels, frameon=True,
                   fontsize=14, loc='lower left')

legend.get_frame().set_facecolor('#ffffff')

plt.show()
```



```
In [106]: # <a name="LeftOff"></a>
          # <h3>Left Off</h3>
```