Visuals1

August 15, 2017

In [18]:
```
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[18]:
```
<IPython.core.display.HTML object>
```

In [21]:
```
from IPython.display import HTML
HTML('''
<style>
    .yourDiv {position: fixed;top: 100px; right: 0px;
              background: white;
              height: 100%;
              width: 300px;
              padding: 20px;
              z-index: 10000}
</style>
<script>
function showthis(url) {
        window.open(url, "pres",
                "toolbar=yes,scrollbars=yes,resizable=yes,top=10,left=400,width=500,hei
        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>
```

```
            <a href=#TemporalCoverage>3. Temporal Coverage</a><br>
            <a href=#ClassOverlaps>4. Farm size class overlaps</a><br>
            <a href=#YieldLookUpTable>5. Yield look-up table</a><br><br>

            <a href="javascript:code_toggle()">Toggle Code On/Off</a><br>
            <a href=#Top>Top</a><br>
            <a href=#LeftOff>Left Off Here</a><br>
        </div>
        ''')
```

Out[21]: <IPython.core.display.HTML object>

```
In [22]: import seaborn as sns
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from matplotlib import cm
         import copy
         import matplotlib.gridspec as gridspec
         from collections import OrderedDict
         from pivottablejs import pivot_ui  # python setup.py install --user
         %matplotlib inline
```

```
In [23]: def read_data(path):

             data = pd.read_csv(path, low_memory=False)
             data['Farm_Sizes'] = pd.cut(data['fs_class_max'],
                                         bins=[0, 1, 2, 5, 10, 20, 50,
                                               100, 200, 500, 1000, 100000])

             global variables
             variables = OrderedDict([('Farm_Sizes', 'Farm_Sizes'),
                                      ('production_Food_kcal', 'Food'),
                                      ('production_Feed_kcal', 'Feed'),
                                      ('production_Seed_kcal', 'Seed'),
                                      ('production_Waste_kcal', 'Waste'),
                                      ('production_Processing_kcal', 'Processing'),
                                      ('production_Other_kcal', 'Other')])

             data = data.loc[:, variables.keys()]
             data.columns = variables.values()

             return data
```

```
In [4]: def piv(data, func=np.nansum):

            pivot = pd.pivot_table(data,
                                   index=['Farm_Sizes'],
                                   values=variables.values()[1:],
```

```
                                    aggfunc=func)

            return pivot
```

```
In [5]: def perc(data, how='within'):

            if how is 'within':

                pivot = piv(data)
                pivot = pivot.transpose()

                for variable in pivot.columns:
                    pivot[variable] = pivot[variable] / pivot[variable].sum()

                return pivot.transpose()


            elif how is 'cumsum':

                pivot = piv(data)

                for variable in pivot.columns:
                    pivot[variable] = pivot[variable] / pivot[variable].sum()

                pivot = pivot.cumsum(axis=0)

                return pivot


            elif how is 'across':

                pivot = piv(data)

                for variable in pivot.columns:
                    pivot[variable] = pivot[variable] / pivot[variable].sum()

                return pivot.transpose()

            else:

                print 'Require how argument'
```

```
In [6]: def plot_stacked_bar(data, how='within', fig_=True, ax=None):

            txt1 = ['Food', 'Feed', 'Seed', 'Waste', 'Processing', 'Other']
            txt2 = ['< 1', '1 to 2', '2 to 5', '5 to 10', '10 to 20',
                    '20 to 50', '50 to 100', '100 to 200', '200 to 500',
                    '500 to 1000', '> 1000']
```

```
txt3 = ['< 1', '2 to 5', '10 to 20', '50 to 100', '200 to 500', '> 1000']

if how is 'within':

    legend_txts = copy.copy(txt1)
    labels_txts = copy.copy(txt2)
    cmap = cm.get_cmap('Set3')
    kind = 'bar'

elif how is 'across':

    legend_txts = copy.copy(txt2)
    labels_txts = copy.copy(txt1)
    cmap = cm.get_cmap('YlGnBu')
    kind = 'bar'

elif how is 'cumsum':

    legend_txts = copy.copy(txt1)
    labels_txts = copy.copy(txt3)
    cmap = cm.get_cmap('Set3')
    kind = 'area'

else:
    pass

if fig_ is True:

    fig = plt.figure(figsize=[10, 5], facecolor='white')
    ax = fig.add_subplot(111)

else:
    pass

data.plot(kind=kind,
          stacked=True,
          cmap=cmap,
          alpha=0.9,
          linewidth=0,
          grid=False,
          ax=ax)

# Axis main
ax.set_axis_bgcolor("#d6d7e5")
ax.set_clip_on(False)
box = ax.get_position()
ax.set_position([box.x0, box.y0, box.width * 0.8, box.height])
```

```
        # Legend
        legend_txts_r = copy.deepcopy(legend_txts)
        legend_txts_r.reverse()
        handles, labels = ax.get_legend_handles_labels()
        legend = ax.legend(handles[::-1], labels[::-1],
                           loc='center left',
                           frameon=1,
                           bbox_to_anchor=(1, 0.5))

        for i in xrange(len(legend_txts_r)):
            legend.get_texts()[i].set_text(legend_txts_r[i])

        frame = legend.get_frame()
        frame.set_color('white')

        # Axis particulars
        ax.set_xticklabels(labels_txts)
        ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)

        if how is 'within':
            ax.set_xlabel('Farm Sizes (ha)')
            ax.set_ylabel('Percentage\n')
            ax.set_ylim([0, 1])
            ax.set_title('Type of production per farm size\n', fontsize=12)

        elif how is 'across':
            ax.set_xlabel('Category')
            ax.set_ylabel('Percentage\n')
            ax.set_ylim([0, 1])
            ax.set_title('Type of production across farm size\n', fontsize=12)

        elif how is 'cumsum':
            ax.set_xlabel('Farm Sizes (ha)')
            ax.set_ylabel('Percentage\n')
            ax.set_title('Type of production per farm size: Cumulative\n', fontsize=12)

        if fig_ is True:
            return plt.show()

        else:
            return ax
```

In [7]:
```
PATH = '/Users/Vinny_Ricciardi/Documents/Data_Library_Big/Survey/Global/Farm_Size/Data/F
df = read_data(PATH)
```

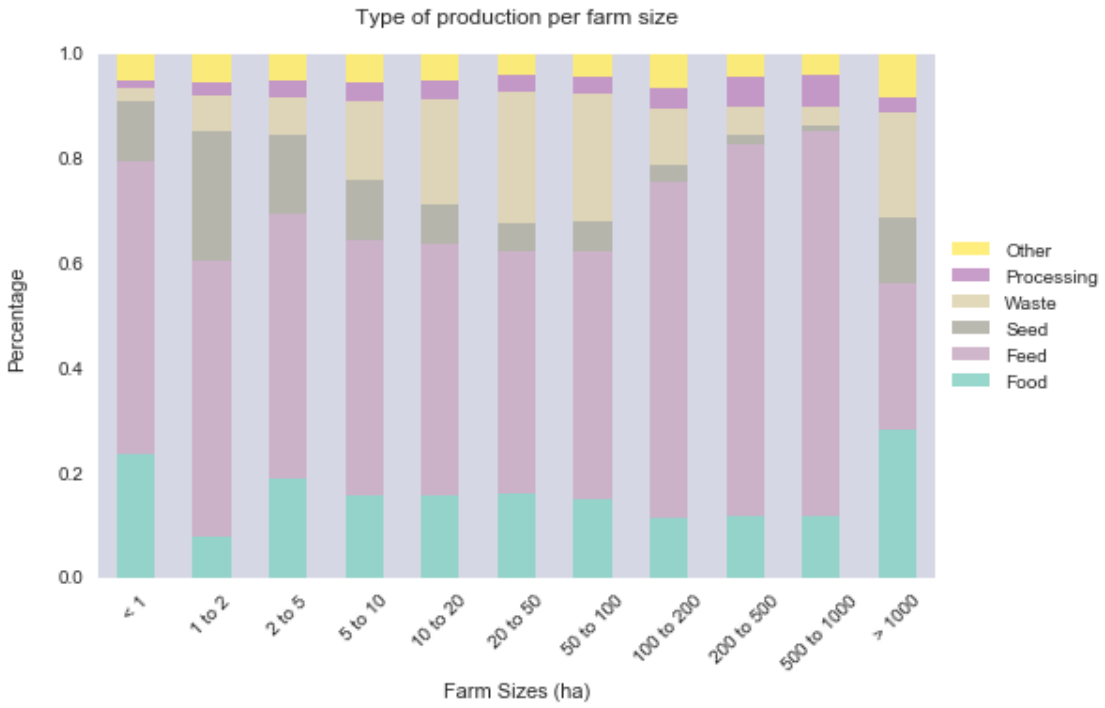
In [8]:
```
df_within = perc(df, how='within')
df_across = perc(df, how='across')
df_cumsum = perc(df, how='cumsum')
df_raw = piv(df)
```

```
In [9]: tmp1 = df_within.copy()
        tmp1['Type'] = 'Within'

        tmp2 = df_across.copy()
        tmp2 = tmp2.transpose()
        tmp2['Type'] = 'Across'

        tmp3 = df_cumsum.copy()
        tmp3['Type'] = 'Cumsum'

        tmp4 = df_raw.copy()
        tmp4['Type'] = 'Raw'

        tmp = pd.concat([tmp1, tmp2, tmp3, tmp4])
        tmp = tmp.reset_index()
        tmp['Farm_Sizes'] = tmp['Farm_Sizes'].str.replace('(', '')
        tmp['Farm_Sizes'] = tmp['Farm_Sizes'].str.replace(']', '')
```

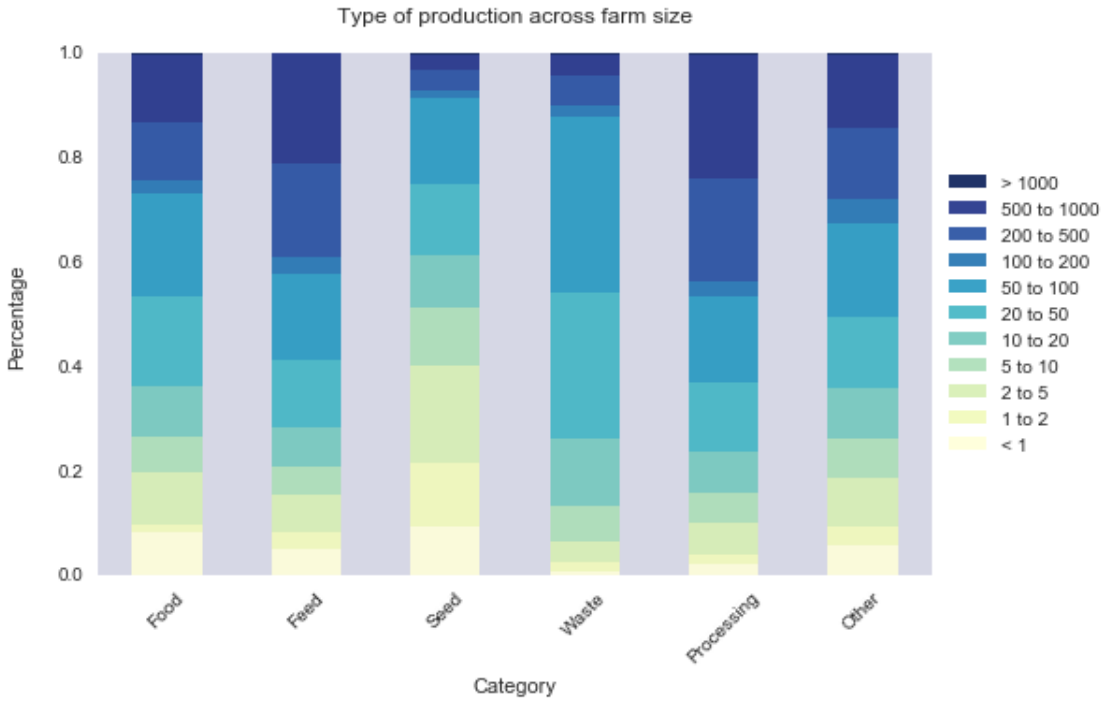
Below is an interactive pivot table. Use 'type' to change whether you are looking at the percentage of crop produced for each category 'within' a farm size group, 'across' groups, via a 'cumulative' percentage across groups, or raw kcal produced. Use the dropdown labeled 'Food' to recalculate based on another production category.

The default setting shows a heatmap for which farm size classes have the highest food production (kcal/person) per category. The 500 to 1000 farm size class makes up 52% of all food production in our dataset, which accounts for 73% of this group's total crop production.

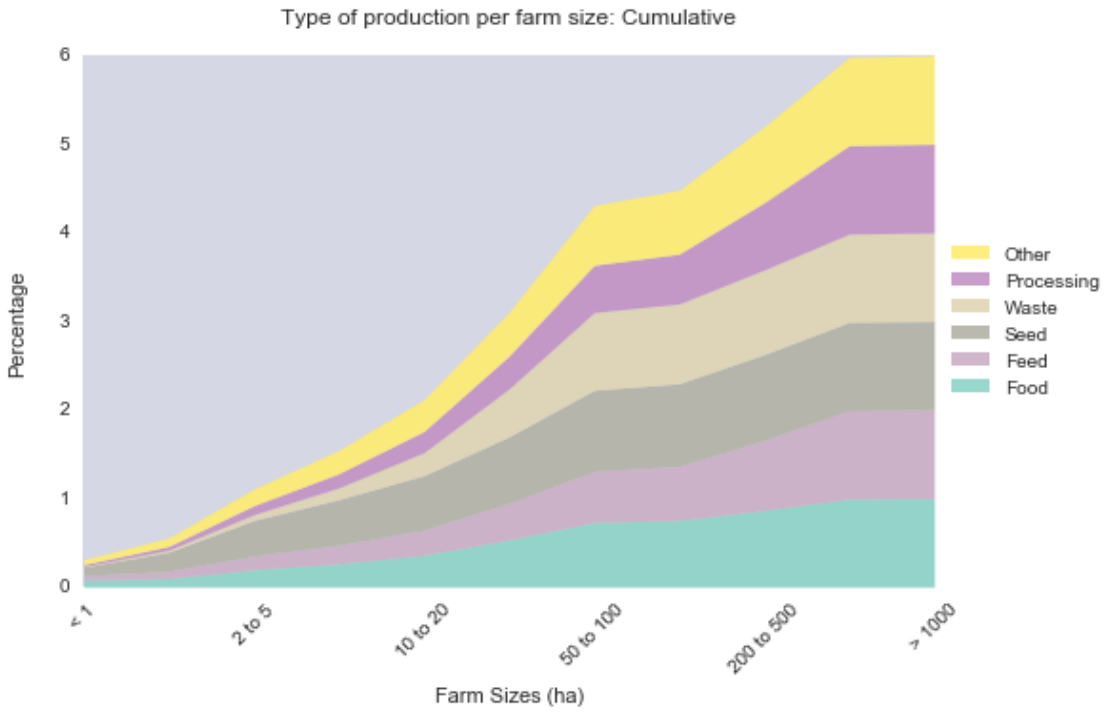
```
In [10]: pivot_ui(tmp)

Out[10]: <IPython.lib.display.IFrame at 0x108b53e90>

In [11]: plot_stacked_bar(df_within, how='within', fig_=True)
```



In [12]: `plot_stacked_bar(df_across, how='across', fig_=True)`



```
In [13]: plot_stacked_bar(df_cumsum, how='cumsum', fig_=True)
```

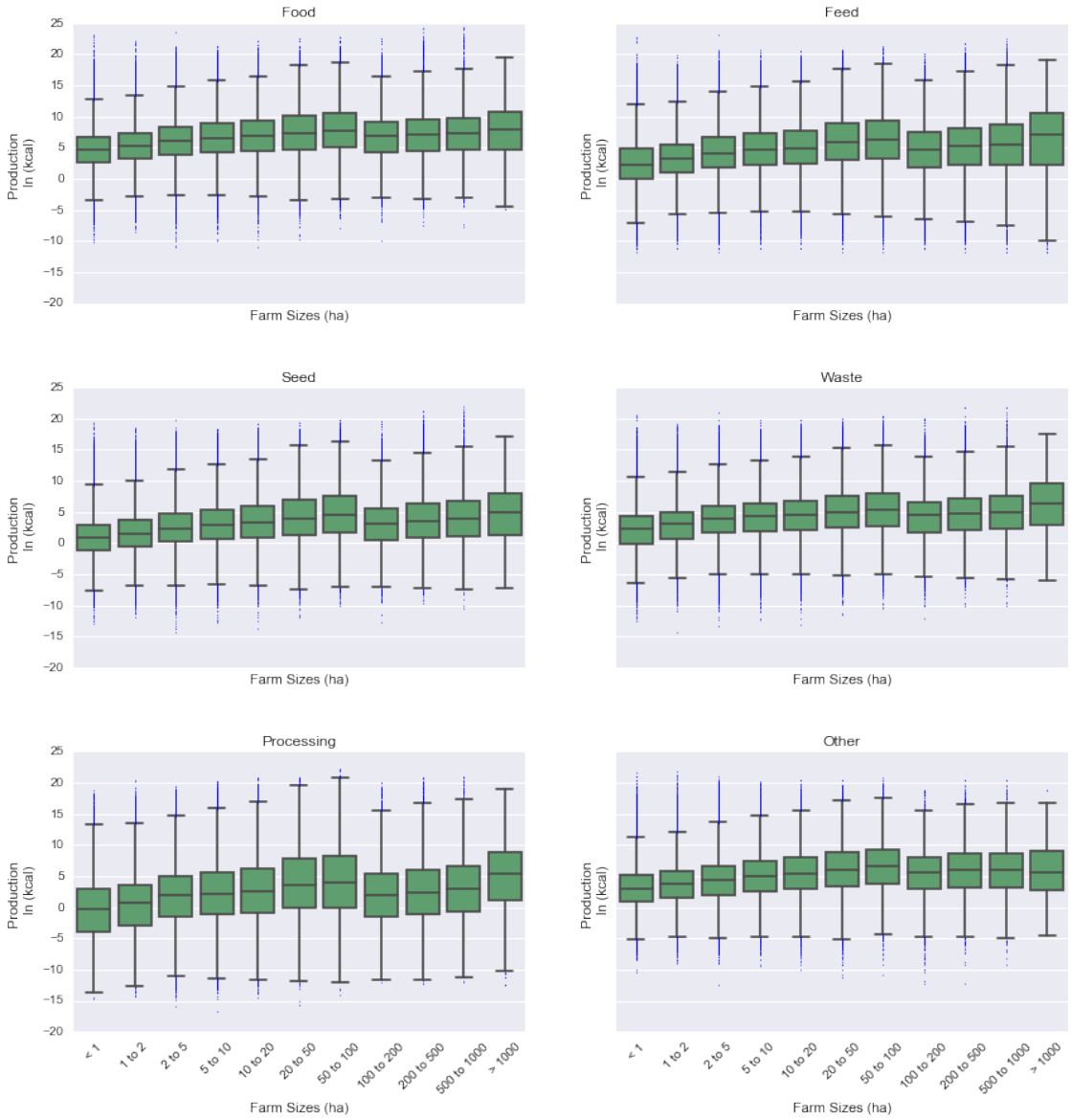



```
In [15]: tmp = df.copy()
         tmp = pd.melt(tmp, id_vars='Farm_Sizes')
         tmp = tmp.loc[tmp['value'] > 0.0]
         tmp['log'] = np.log(tmp['value'])

In [16]: def factor_plot(data):

             g = sns.factorplot(x="Farm_Sizes", y="log",
                                col="variable",
                                data=data,
                                kind='box',
                                col_wrap=2,
                                color='#55a868',
                                fliersize=1,
                                aspect = 1.5,
                                order=['(0, 1]', '(1, 2]', '(2, 5]', '(5, 10]',
                                       '(10, 20]', '(20, 50]', '(50, 100]', '(100, 200]', '(200,
                                       '(500, 1000]', '(1000, 100000]'])

             g.fig.subplots_adjust(wspace=0.2, hspace=0.3)

             titles = data.variable.unique()
```

```
    for ax, title in zip(g.axes.flat, titles):
        ax.set_title(title)
        ax.set_xticklabels(['< 1', '1 to 2', '2 to 5', '5 to 10', '10 to 20',
                '20 to 50', '50 to 100', '100 to 200', '200 to 500',
                '500 to 1000', '> 1000'])
        ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)
        ax.set_xlabel('Farm Sizes (ha)')
        ax.set_ylabel('Production \n ln (kcal)')
```

In [17]:
```
factor_plot(tmp)
```

```
/usr/local/lib/python2.7/site-packages/matplotlib/__init__.py:898: UserWarning: axes.color_cycle
  warnings.warn(self.msg_depr % (key, alt_key))
```



In [116]:
```
tmp = df.copy()
tmp = pd.melt(tmp, id_vars='Farm_Sizes')
tmp = tmp.loc[tmp['value'] > 0.0]
tmp['log'] = np.log(tmp['value'])
tmp['Farm_Sizes'] = tmp['Farm_Sizes'].str.replace('(', '')
tmp['Farm_Sizes'] = tmp['Farm_Sizes'].str.replace(']', '')
tmp = tmp.join(tmp['Farm_Sizes'].str.split(',', 1,
                                           expand=True).rename(columns={0:'Farm_Size_M
                                                                        1:'Farm_Size_M
tmp['Farm_Size_Min'] = tmp['Farm_Size_Min'].astype(float)
tmp['Farm_Size_Max'] = tmp['Farm_Size_Max'].astype(float)
```

In [ ]: