General_Results

August 17, 2017

```
In [15]: 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[15]: <IPython.core.display.HTML object>
```

```
In [16]: 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,hei
                 return(false);
         }
         </script>

         <div class=yourDiv>
             <h4>MENU</h4><br>
             <a href=#Data>1. Data</a><br>
             <a href=#FoodFeedOtherAcross>2. Across Farm Sizes</a><br>
             <a href=#FoodFeedOtherWithin>3. Within Farm Sizes</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[16]: <IPython.core.display.HTML object>
```

General Results
This page overviews our general results:

- The main findings are that in our sample of 58 countries, farms < 2 ha produce 1% of the total food supply, while farms < 200 ha produce 72% of the total food supply. The largest differences (in terms of effect size) in the amount of food produced were between the very large farm and the smallest farm categories in terms of how much food each produced.

- This study also measures what percentage of each farm size class's crop producion goes towards food, feed, processing, seed, waste, or other. The farm size category with the largest percentage of food produced compared to other cateogories within thier farm size group, are from 100 to 1000 ha, where 70% of thier crop production goes towards food.

- 55% of crop production on farms < 2 ha goes towards food, while 16% goes towards feed, and the remainder either goes towards other, waste, processing, or seed (respectively).

- The farm size contributing to the largest amount of food waste (post-harvest loss, not consumer based food waste) are farms between 50-100 ha.

```
In [17]: import warnings
         warnings.filterwarnings('ignore')
         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
         plt.style.use('seaborn-muted')
         %matplotlib inline
```

```
In [18]: def read_data(path):

             data = pd.read_csv(path, low_memory=False)
             data = data.loc[data['data_unit'] == 't']
             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()

             data = data.replace(0.0, np.nan)  # there were many zero values

             return data

In [19]: def data_stats(path):

             path = PATH
             data = pd.read_csv(path, low_memory=False)
             data = data.loc[data['data_unit'] == 't']
             number_records = len(data)
             number_countries = len(data.NAME_0.unique())

             return number_records, number_countries

In [20]: def piv(data, func=np.nansum):

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

             return pivot

In [21]: 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 [22]: def plot_stacked_bar(data, how='within', fig_=True, ax=None):

             txt1 = ['Feed', 'Food', 'Other', 'Processing', 'Seed', 'Waste']
             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 [23]: PATH = '/Users/Vinny_Ricciardi/Documents/Data_Library_Big/Survey/Global/Farm_Size/Data/
         df = read_data(PATH)

In [24]: df_within = perc(df, how='within')
         df_across = perc(df, how='across')
         df_cumsum = perc(df, how='cumsum')
         df_raw = piv(df)

In [25]: 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(']', '')
```

Data

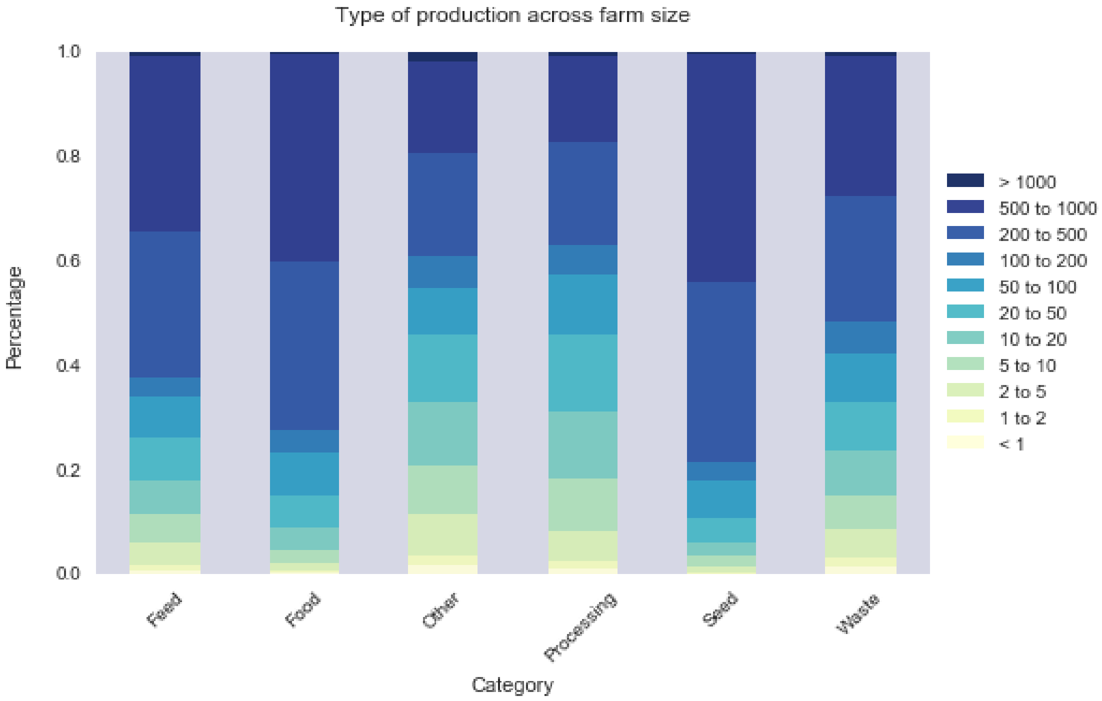
For an overview of the data, see the data coverage notebook. Here is a link to an interactive pivot table so you can explore the data. Use 'type' to change whether you are looking at the percentage of crop produced for each category 'within' a farm size group, 'across' groups, via a 'cumulative' percentage across groups, or raw kcal produced. Use the dropdown labeled 'Food' to recalculate based on another production category.

```
In [84]: # pivot_ui(tmp,
         #          rows=['Farm_Sizes'],
         #          cols=['Type'],
         #          aggregatorName='Sum',
         #          vals=['Food'],
         #          rendererName='Col Heatmap',
         #          colOrder= "value_a_to_z",)

         # Enable for pivot table interactivitity within this Jupyter notebook
         # Here's a nice example of how to change the html output file for prettier output:
         # https://pivottable.js.org/examples/montreal_2014.html
```

Food Feed Other by Farm Size

This plot shows the percentage that each farm size contributes to each category (e.g., food, feed, other, etc.) For example, farms between 500 to 1000 ha produce 40.0% of the total food supply in our sample.

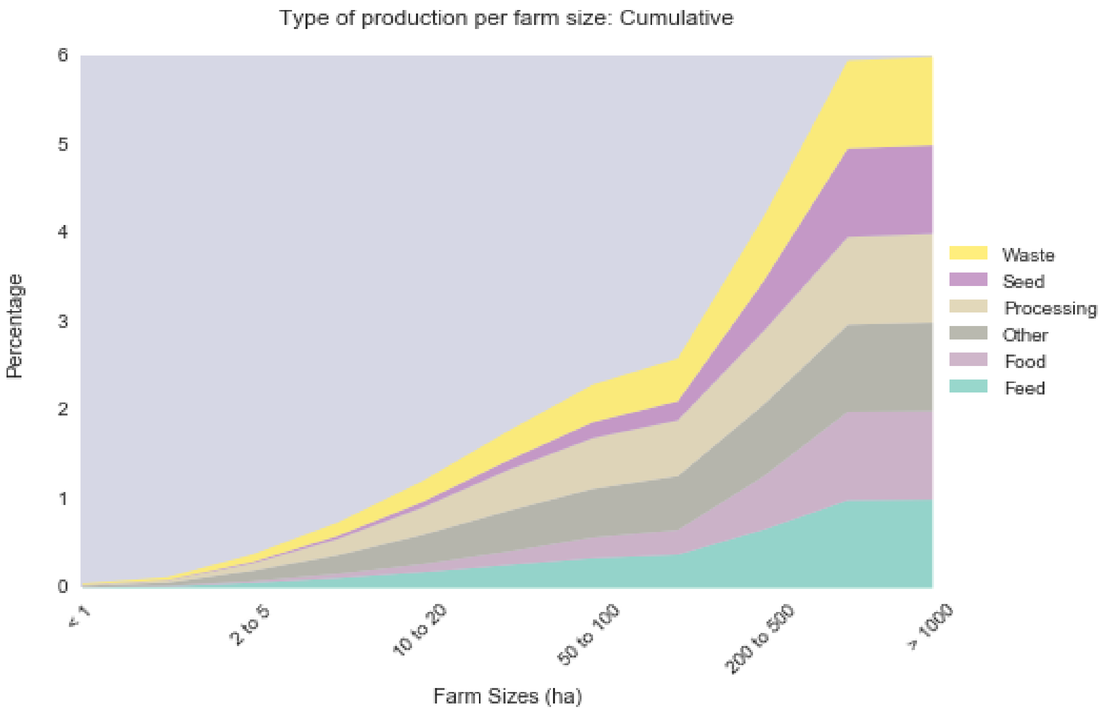
```
In [128]: plot_stacked_bar(df_across, how='across', fig_=True)
```



```
In [129]: under_two = round(100 * df_cumsum.iloc[2,1], 1)
          under_fifty = round(100 * df_cumsum.iloc[5,1], 1)
```

This plot also shows the percentage that each farm size contributes to each category but in cumulative percentages. For example, farms under 2 ha produce 2.1% of the total food supply in our sample and farms 50 ha and under produce 15.2% of the total food supply in our sample. Again, here is a link to an interactive pivot table so you can explore the data in more detail.

```
In [130]: plot_stacked_bar(df_cumsum, how='cumsum', fig_=True)
```



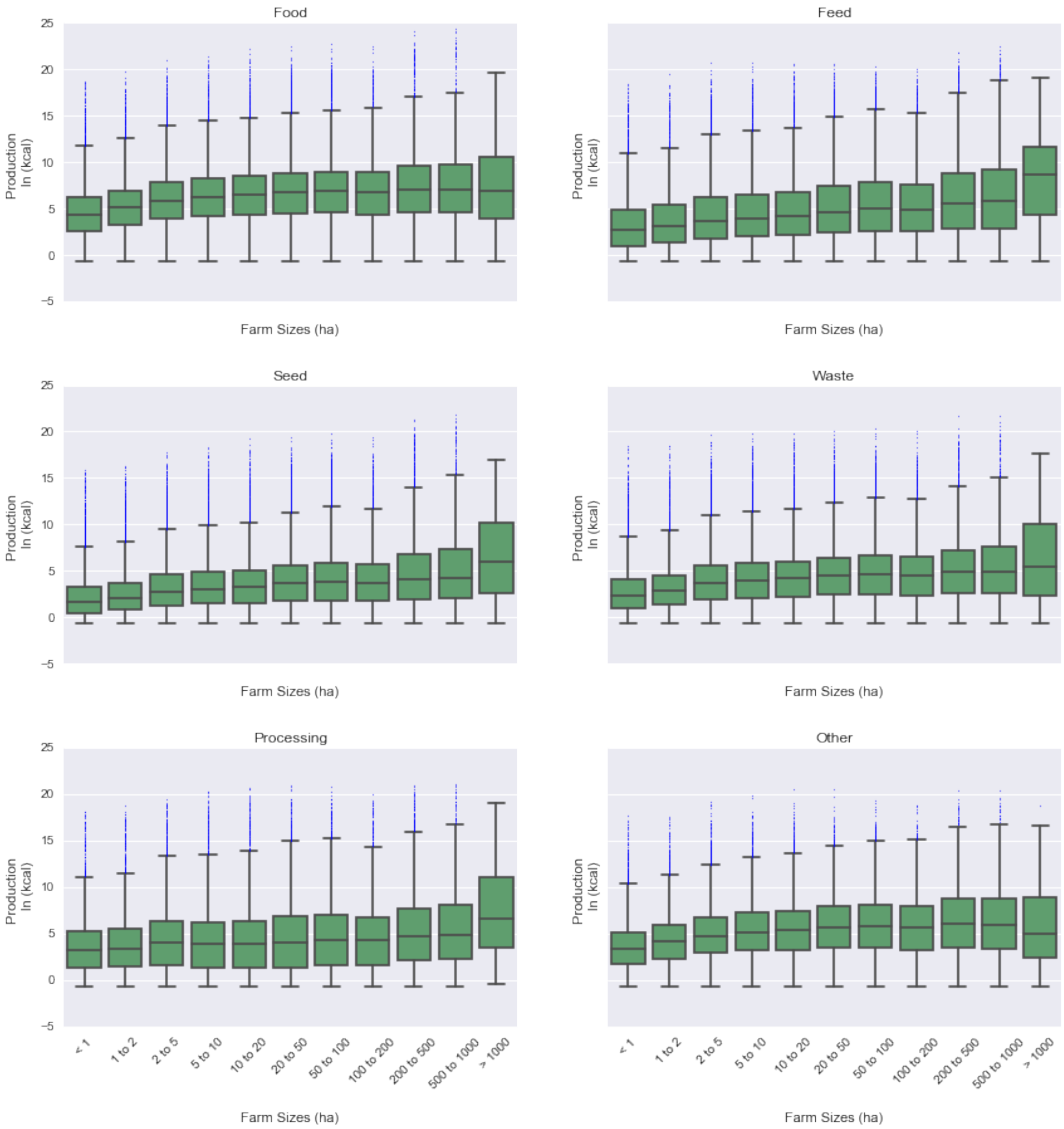
```
In [131]: def factor_plot(data, id_var='Farm_Sizes'):

              data = pd.melt(data, id_vars='Farm_Sizes')
              data = data.loc[data['value'] > 0.5]
              data['log'] = np.log(data['value'])

              fs_order = ['(0, 1]', '(1, 2]', '(2, 5]', '(5, 10]',
                          '(10, 20]', '(20, 50]', '(50, 100]', '(100, 200]', '(200, 500]',
                          '(500, 1000]', '(1000, 100000]']

              if id_var is 'Farm_Sizes':
```

```
    col = 'variable'
    x = 'Farm_Sizes'
    y = 'log'
    order = fs_order

else:

    data = data.sort('Farm_Sizes')
    data['Farm_Sizes'] = pd.Categorical(data['Farm_Sizes'], fs_order)
    data = data.sort('Farm_Sizes')
    col = 'Farm_Sizes'
    x = 'variable'
    y = 'log'
    order = None

g = sns.factorplot(x=x, y=y,
                   col=col,
                   data=data,
                   kind='box',
                   col_wrap=2,
                   color='#55a868',
                   fliersize=1,
                   aspect = 1.5,
                   order=order)

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

titles = data.variable.unique()

fs_txt = ['< 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']

if id_var is 'Farm_Sizes':

    for ax, title in zip(g.axes.flat, titles):

        ax.set_title(title)
        ax.set_ylabel('Production \n ln (kcal)')
        ax.set_xticklabels(fs_txt)
        ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)
        ax.set_xlabel('\nFarm Sizes (ha)')

else:

    for ax, title in zip(g.axes.flat, fs_txt):

        ax.set_title(title)
```

```
            ax.set_ylabel('Production \n ln (kcal)')
            ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)
            ax.set_xlabel('\nPoduction Category')
```

In order to better understand whether there are differences between the farm size groups, these boxplots show the total amount of crops produced (in logged kcal for standardization). Across all production categories (e.g., food, feed, other, etc.) there are no visual differences in the means and confidence intervals. But, there are many of outliers in each plot that indicate high variance.

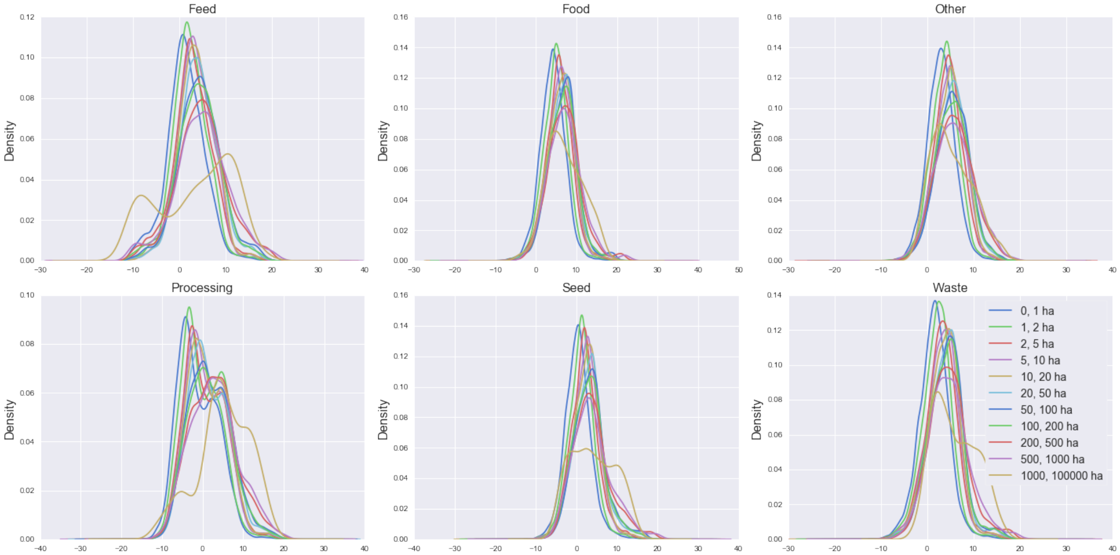
```
In [132]: factor_plot(df, id_var='Farm_Sizes')
```



Since many of the plots have large variance and skewness, here are density plots comparing farm size classes per each category. We can see that the largest farm size group has different distributions from the remaining groups.

To do: - Fix color map to diverging

```
In [392]: tmp = df.copy()
          tmp = pd.melt(tmp, id_vars='Farm_Sizes')
          tmp = tmp.dropna()
          cols = np.sort(tmp['variable'].unique())
          fs = ['(0, 1]', '(1, 2]', '(2, 5]', '(5, 10]',
                '(10, 20]', '(20, 50]', '(50, 100]',
                '(100, 200]', '(200, 500]', '(500, 1000]',
                '(1000, 100000]']

          fig = plt.figure(figsize=[20,10])

          for j in xrange(len(cols)):

              ax = fig.add_subplot(2, 3, j+1)
              tmp1 = tmp.loc[tmp['variable'] == cols[j]]

              for i in xrange(len(fs)):

                  tmp2 = tmp1.loc[tmp1['Farm_Sizes'] == fs[i]]
                  np.log(tmp2['value']).plot(kind='kde',
                                             label=fs[i][1:-1] + ' ha',
                                             ax=ax)

              ax.set_title(cols[j], fontsize=16)
              ax.set_ylabel('Density', fontsize=16)

          legend = ax.legend(frameon=True, fontsize=14)
          plt.tight_layout()
          plt.show()
```



```
In [133]: number_records, number_countries = data_stats(PATH)
```

Since our sample is very large (12 countries which comprise of 130760 total records due to subnational units and crop types) we cannot use p-values to determine significance between the amount of production by varying farm sizes. To circumvent the large dataset issue, the effect size was used to determine differences between farm sizes rather than statistical significance. Cohen's d was calculated via taking the (mean of farmsize 1 - mean of farmsize 2) / (standard deviation of farmsize 1) then taking the absolute value. The relative significance was calculated by thresholds according to Sullivan and Feinn 2012.

Where the Cohen d's effect size values correspond to percent non-overlapping observations, as in:

Relative Size	Effect Size	Percentile	% of Non-overlap
	0	50	0
Small	0.2	58	15
Medium	0.5	69	33
Large	0.8	79	47
	1.0	84	55
	1.5	93	71
	2.0	97	81

```
In [134]: def cohens_d(data1, data2, how='within'):

              cols = data1.columns
              check = []

              for i in xrange(0, len(cols)):
                  for j in xrange(1, len(cols)):

                      col_name = str(cols[i]) + '_' + str(cols[j])
                      col_name_r = str(cols[j]) + '_' + str(cols[i])

                      if col_name in check:

                          pass

                      elif cols[i] is cols[j]:

                          pass

                      else:

                          data1[col_name] = ((data1[cols[i]] - data1[cols[j]]) / data2[cols[i]])
                          check.append(col_name)
                          check.append(col_name_r)

              data = data1.iloc[:, len(cols):]

              data = data.reset_index()

              if how is 'within':

                  data = pd.melt(data, id_vars='Farm_Sizes', value_name='cohens_d')

              else:

                  data = pd.melt(data, id_vars='index', value_name='cohens_d')

              data['cohens_d_level'] = np.where(data['cohens_d'] <= 0.2, 'small',
                                                np.where(data['cohens_d'] >= 0.8, 'large',
                                                         'medium'))

              return data

In [464]: def cohens_interp(func=np.mean, index='index'):

              tmp = pd.pivot_table(out,
                                   index=index,
                                   columns='cohens_d_level',
                                   values='cohens_d',
                                   aggfunc=func)

              tmp = tmp.reset_index()

              try:
```

```
        tmp = tmp.sort('large', ascending=False)
        tmp.columns = ['Category', 'large d', 'medium d', 'small d']

    except:
        tmp = tmp.sort('medium', ascending=False)
        tmp.columns = ['Farm Sizes', 'medium d', 'small d']

    return tmp
```

Here are the biggest differences between farm size groups. As expected, there are large coefficients (effect size) between small and large farm classes.

```
In [437]: data = df.copy()
          data['Farm_Sizes'] = data['Farm_Sizes'].astype(str)
          means = pd.pivot_table(data, columns='Farm_Sizes', aggfunc=np.nanmean)
          sds = pd.pivot_table(data, columns='Farm_Sizes', aggfunc=np.nanstd)
          out = cohens_d(means, sds, how='across')

          tmp = out.copy()
          tmp['Farm_Sizes'] = tmp['Farm_Sizes'].str.replace(']', ')')
          tmp['Farm_Sizes'] = tmp['Farm_Sizes'].str.replace('_', ' and ')
          tmp.columns = ['Category', 'Farm sizes compared', 'Cohens d', 'Relative effect size']
          tmp = tmp.sort('Cohens d', ascending=False)
          tmp.head(10)
```

```
Out[437]:     Category     Farm sizes compared   Cohens d Relative effect size
          58      Seed  (0, 1) and (500, 1000)  48.488213                large
          55      Food  (0, 1) and (500, 1000)  31.978579                large
          112     Seed  (1, 2) and (500, 1000)  31.014690                large
          40      Seed   (0, 1) and (200, 500)  20.790542                large
          109     Food  (1, 2) and (500, 1000)  17.098681                large
          37      Food   (0, 1) and (200, 500)  14.102206                large
          94      Seed   (1, 2) and (200, 500)  13.287843                large
          268     Seed  (2, 5) and (500, 1000)   9.832608                large
          54      Feed  (0, 1) and (500, 1000)   8.674037                large
          91      Food   (1, 2) and (200, 500)   7.530880                large
```

On average, the largest effect size were in the seed and food categories.

```
In [439]: cohens_interp(func=np.mean)
```

```
Out[439]:      Category   large d  medium d   small d
          4        Seed  7.540147  0.474176  0.067408
          1        Food  6.551549  0.483279  0.052100
          0        Feed  2.248732  0.418498  0.060634
          3  Processing  1.957038  0.392405  0.056883
          5       Waste  1.542108  0.382661  0.059074
          2       Other  1.182823  0.277370  0.057448
```

The seed and food categories also had the highest frequency of large effect sizes between farm size groups.

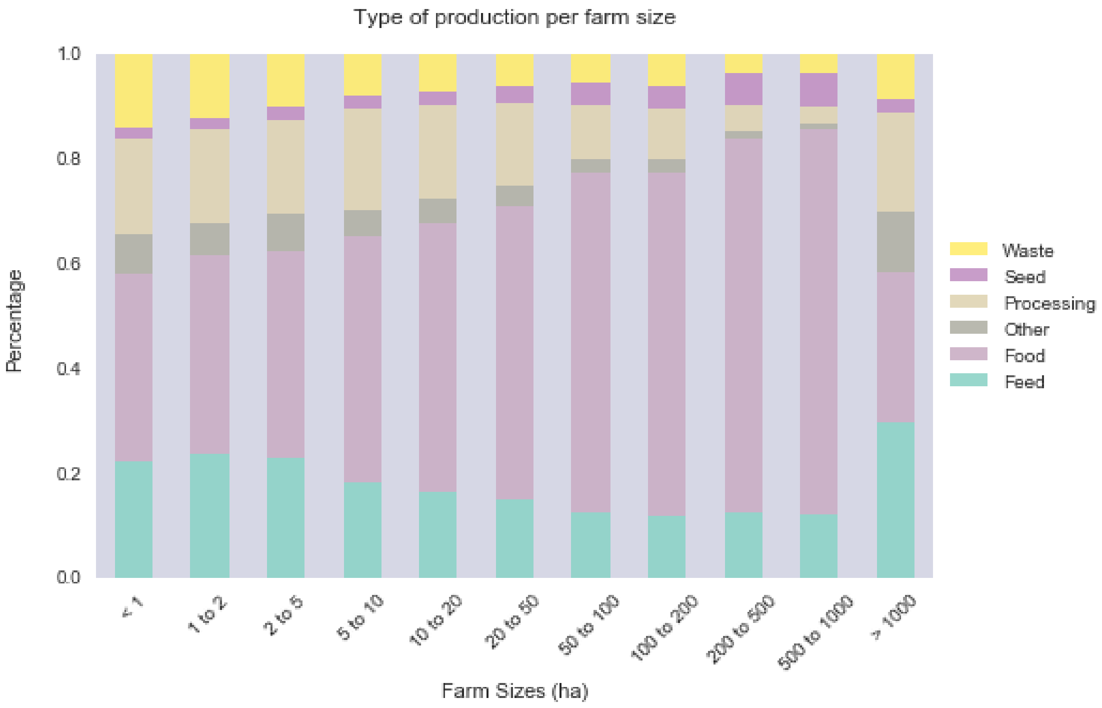
```
In [430]: cohens_interp(func=len)
```

```
Out[430]:       Category  large d  medium d  small d
          4        Seed     21.0       8.0     26.0
          1        Food     15.0      15.0     25.0
          0        Feed     12.0      12.0     31.0
          5       Waste      6.0      13.0     36.0
          2       Other      5.0      12.0     38.0
          3  Processing      4.0      19.0     32.0
```

Food Feed Other within Farm Size Groups

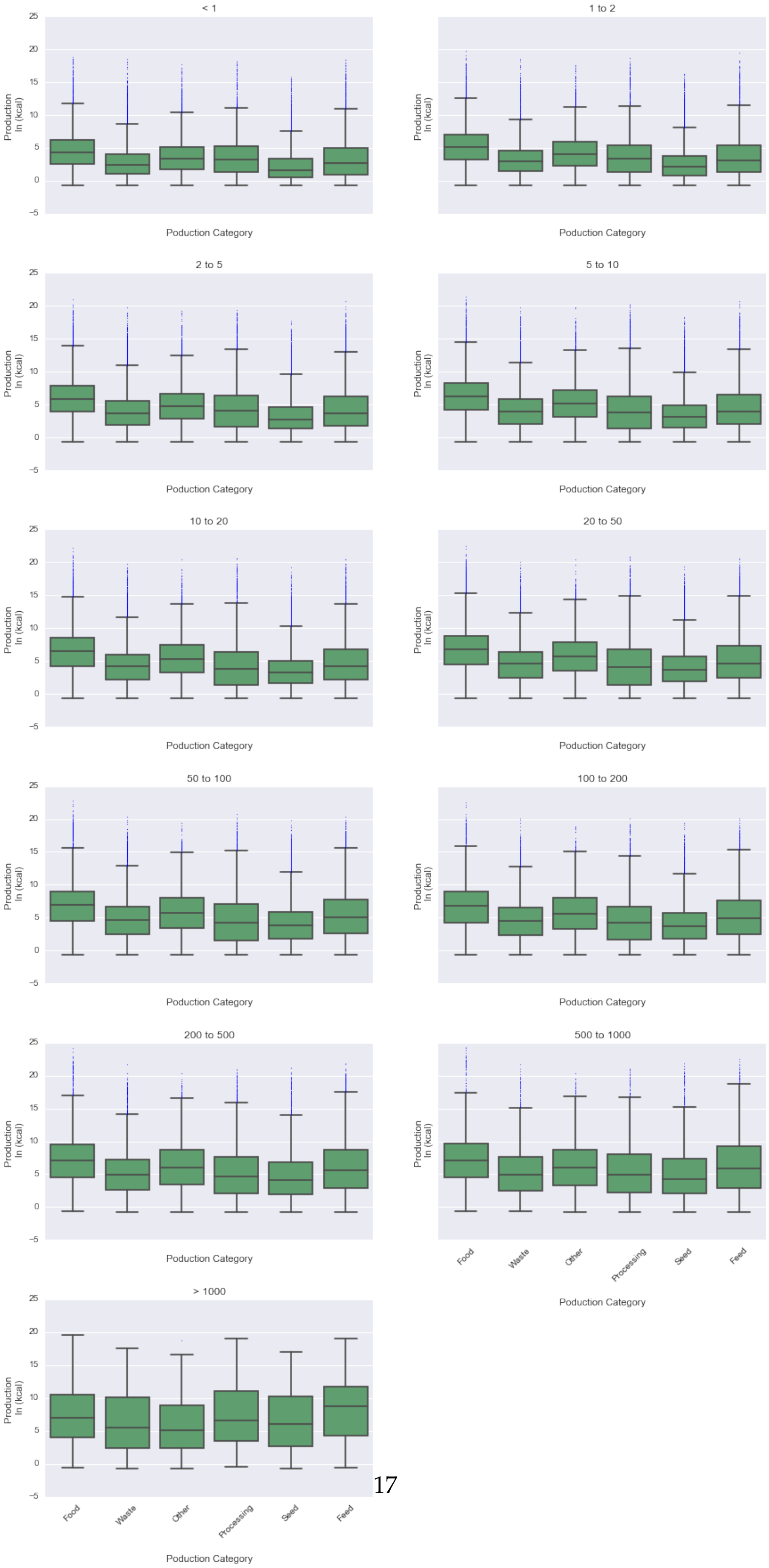
This plot shows the percentage of Food, Feed, Seed, Waste, Processing, and Other for each farm size category. For example, {{one_food}} % of crop production for farms under 1 ha is food, while 6% of their crop production is waste.

```
In [385]: plot_stacked_bar(df_within, how='within', fig_=True)
```



In order to better understand whether there are differences within how a given farm size group allocates their crop production, these boxplots show the total amount of crops produced (in logged kcal for standardization). As in the previous analysis, across all farm size categories there are no visual differences in the means and confidence intervals. But, there are many of outliers in each plot that indicate high variance.

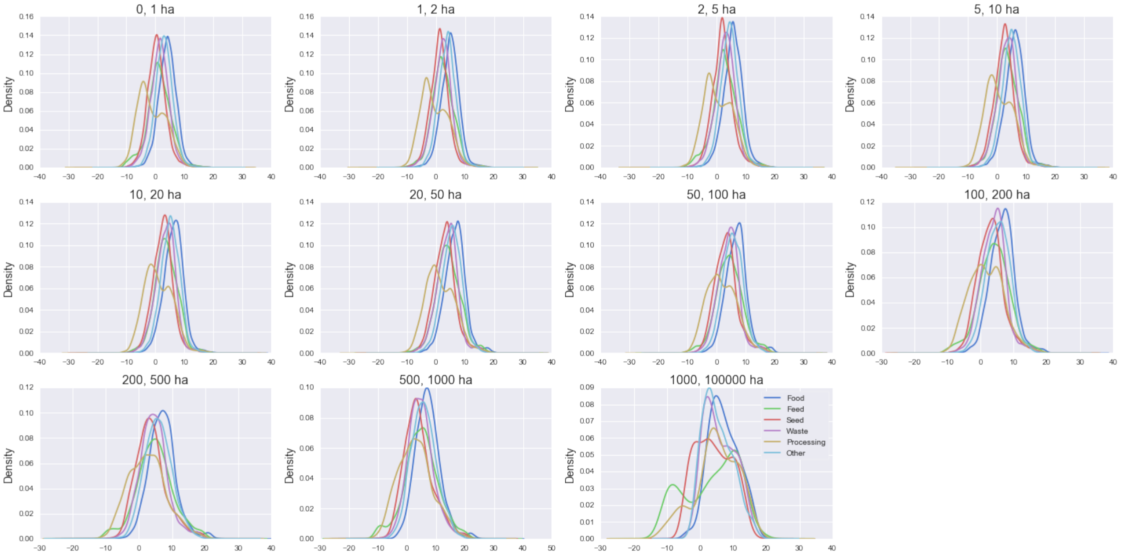
```
In [386]: factor_plot(df, id_var='variable')
```


< 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
Production
ln (kcal)
25
20
15
10
5
0
−5
Poduction Category
Food
Waste
Other
Processing
Seed
Feed

Since many of the plots have large variance and skewness, here are density plots comparing farm size classes per each category. We can see that the largest farm size group has different distributions from the remaining groups.

To do: - Fix color map to diverging

```
In [384]: tmp = df.copy()
          fs = ['(0, 1]', '(1, 2]', '(2, 5]', '(5, 10]',
                '(10, 20]', '(20, 50]', '(50, 100]',
                '(100, 200]', '(200, 500]', '(500, 1000]',
                '(1000, 100000]']
          tmp = tmp.dropna()
          cols = tmp.columns

          fig = plt.figure(figsize=[20,10])

          for j in xrange(len(fs)):

              tmp1 = tmp.loc[tmp['Farm_Sizes'] == fs[j]]

              ax = fig.add_subplot(3, 4, j+1)

              for i in xrange(1, len(cols)):
                  np.log(tmp1[cols[i]]).plot(kind='kde',
                                             label=cols[i],
                                             ax=ax)

              ax.set_title(fs[j][1:-1] + ' ha', fontsize=16)
              ax.set_ylabel('Density', fontsize=14)

          legend = ax.legend(frameon=True)
          plt.tight_layout()
          plt.show()
```



We calculated the effect size for the differences within each farm size group, but across production cateogries (e.g., food, feed, other). There were only medium differences in effect size when looking at how each farm size group allocated their production.

```
In [445]: means = pd.pivot_table(df, index='Farm_Sizes', aggfunc=np.nanmean)
          sds =  pd.pivot_table(df, index='Farm_Sizes', aggfunc=np.nanstd)
          out = cohens_d(means, sds, how='within')

          tmp = out.copy()
          tmp = tmp.loc[out['cohens_d_level'] != 'small']
          tmp['Farm_Sizes'] = tmp['Farm_Sizes'].str.replace(']', ')')
          tmp['variable'] = tmp['variable'].str.replace('_', ' and ')
          tmp.columns = ['Farm sizes', 'Categories commpared', 'Cohens d', 'Relative effect size
          tmp = tmp.sort('Cohens d', ascending=False)
          out = out.reset_index()
          tmp.head(10)
```

```
Out[445]:          Farm sizes Categories commpared  Cohens d Relative effect size
          9       (500, 1000)        Feed and Food  0.399468               medium
          8        (200, 500)        Feed and Food  0.395566               medium
          119     (500, 1000)       Other and Seed  0.349541               medium
          6         (50, 100)        Feed and Food  0.314126               medium
          7        (100, 200)        Feed and Food  0.300623               medium
          164  (1000, 100000)       Seed and Waste  0.237525               medium
          154          (0, 1)       Seed and Waste  0.217207               medium
          155          (1, 2)       Seed and Waste  0.204633               medium
```

On average, the largest effect size were in the larger farm size groups.

```
In [470]: cohens_interp(func=np.mean, index='Farm_Sizes').head()
```

```
Out[470]:          Farm Sizes  medium d   small d
          7         (200, 500]  0.395566  0.068049
          10       (500, 1000]  0.374505  0.070297
          9          (50, 100]  0.314126  0.047100
          3         (100, 200]  0.300623  0.039264
          4     (1000, 100000]  0.237525  0.062113
```

The most frequent differences between categories within a farm size group was for the 500 to 1000 ha group.

```
In [472]: cohens_interp(func=len, index='Farm_Sizes').head()
```

```
Out[472]:          Farm Sizes  medium d  small d
          10       (500, 1000]       2.0     13.0
          0             (0, 1]       1.0     14.0
          1             (1, 2]       1.0     14.0
          3         (100, 200]       1.0     14.0
          4     (1000, 100000]       1.0     14.0
```

Global Estimates

```
In [ ]: data = df.copy()
```

```
In [ ]: data.head()
```

Left Off
To Do: - Need to calculate bootstraps and jackknife estimates

```
In [ ]:
```