DAPP_analysis

December 8, 2015

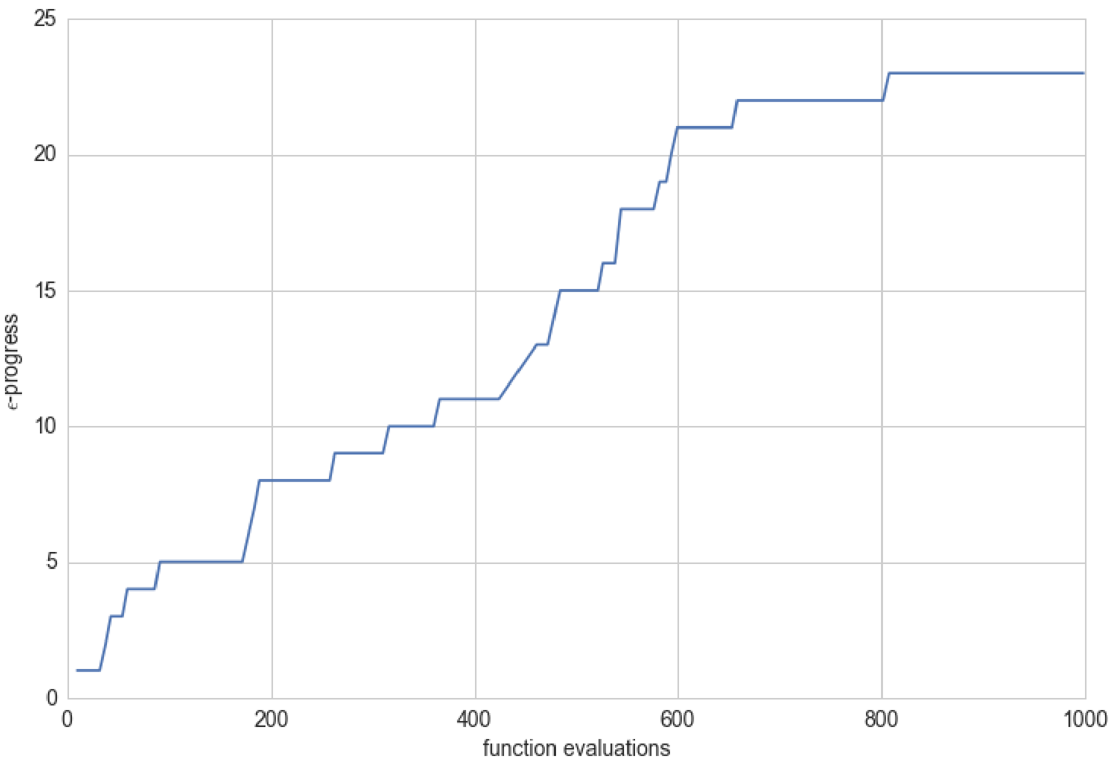
```
In [6]: def save_fig(fig, i):
            plt.savefig('./figs/fig{}_lowres.png'.format(i), dpi=75, bbox_inches='tight')
            plt.savefig('./figs/fig{}_highres.png'.format(i), dpi=300, bbox_inches='tight')

        def change_fontsize(fig, fs=14):

            for ax in fig.axes:
                for item in ([ax.title, ax.xaxis.label, ax.yaxis.label] +
                             ax.get_xticklabels() + ax.get_yticklabels()):
                    item.set_fontsize(fs)
```

```
In [2]: archive = pd.DataFrame.from_csv('./data/archive501.csv')
        stats = pd.DataFrame.from_csv('./data/statistics501.csv')
```

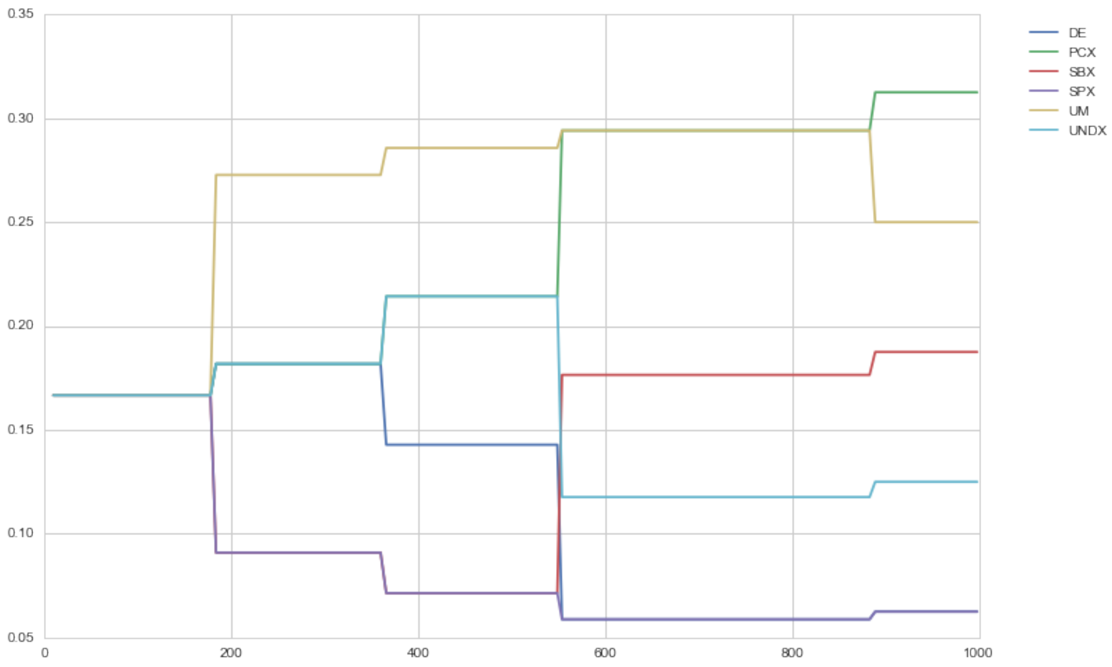
```
In [3]: stats.columns
```

```
Out[3]: Index([u'ArchiveSize', u'DE', u'ElapsedTime', u'Improvements', u'NFE', u'PCX',
               u'PopulationSize', u'Restarts', u'SBX', u'SPX', u'UM', u'UNDX'],
              dtype='object')
```

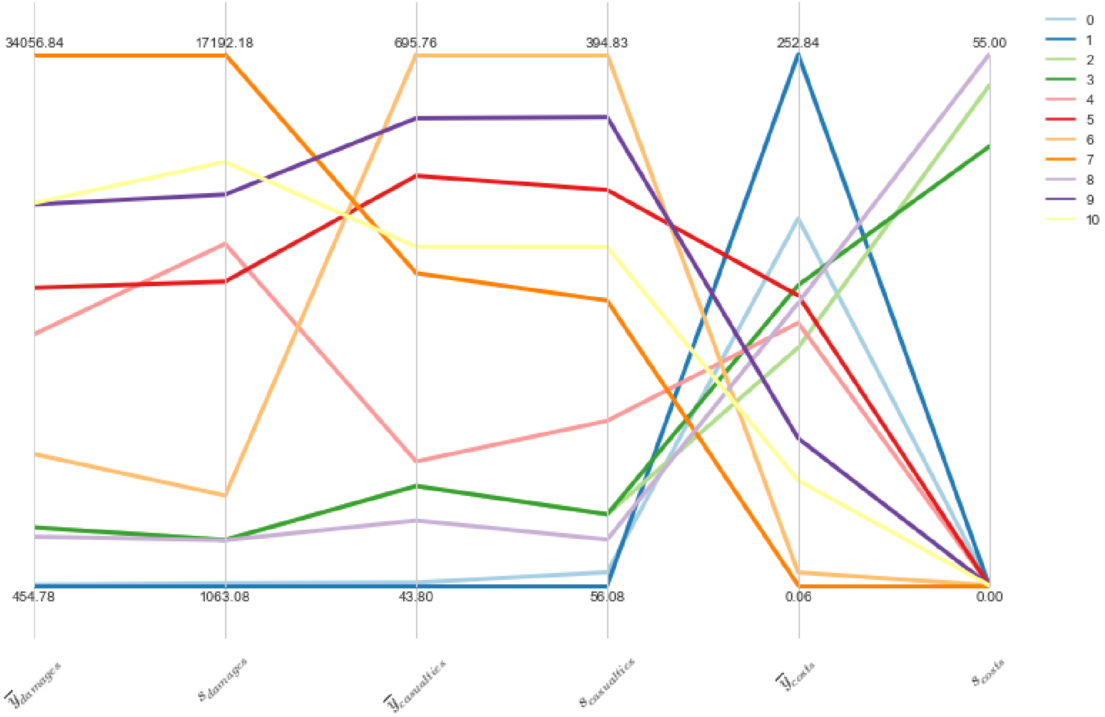
```
In [4]: fig = plt.figure()
        ax = fig.add_subplot(111)
        ax.plot(stats.NFE, stats['Improvements'])
        ax.set_ylabel('$\epsilon$-progress')
        ax.set_xlabel('function evaluations')

        change_fontsize(fig)
        save_fig(fig, 9)

        plt.show()
```



```
In [5]: fig = plt.figure()
        ax = fig.add_subplot(111)

        operators = ['DE', 'PCX', 'SBX','SPX','UM','UNDX']
        for operator in operators:
            ax.plot(stats.NFE, stats[operator], label=operator)
        ax.legend(loc='upper right', bbox_to_anchor=(1.15, 1))
        plt.show()
```



```
In [6]: archive.columns

Out[6]: Index([u’var 0’, u’var 1’, u’var 2’, u’var 3’, u’var 4’, u’obj 0’, u’obj 1’,
               u’obj 2’, u’obj 3’, u’obj 4’, u’obj 5’],
             dtype=’object’)

In [7]: variabels = [’var 0’, ’var 1’, ’var 2’, ’var 3’, ’var 4’]
        pathways = np.floor(archive[variabels])
        pathways

Out[7]:     var 0  var 1  var 2  var 3  var 4
        0       5     20     10      2      2
        1       7     12      9      0      1
        2       4     19     16      2      0
        3       4      5     16      2      0
        4      10     19     18      0      2
        5       1     20     19      1      0
        6      17     20     19      2      0
        7      18     20     19      1      0
        8       6      4     19      2      0
        9       0     20     19      1      0
        10      3     19     19      0      0

In [8]: policies = [’RfR Small Scale’,
                    ’RfR Medium Scale’,
                    ’RfR Large Scale’,
                    ’RfR Side channel’,
                    ’Dike 1:500 +0.5m’,
                    ’Dike 1:500 extr.’,
                    ’Dike 1:1000’,
                    'Dike 1:1000 extr.',
                    'Dike 2nd Q x 1.5',
                    'Dike Climate dikes',
                    'Dike Wave resistant',
                    'Coop Small',
                    'Coop Medium',
                    'Coop Large',
                    'DC Elevated',
                    'DC Dikes',
                    'DC Mounts',
                    'DC Floating',
                    'Alarm Early',
                    'no policy',
                    'Alarm Education'
                    ]

        rules = ['SMALL', 'LARGE', 'XLARGE']

        pathways['var 0'] = [policies[int(i)] for i in pathways['var 0']]
        pathways['var 1'] = [policies[int(i)] for i in pathways['var 1']]
        pathways['var 2'] = [policies[int(i)] for i in pathways['var 2']]

In [9]: pathways

Out[9]:                   var 0             var 1                var 2  var 3  var 4
        0      Dike 1:500 extr.   Alarm Education  Dike Wave resistant      2      2
        1     Dike 1:1000 extr.       Coop Medium   Dike Climate dikes      0      1
        2      Dike 1:500 +0.5m         no policy            DC Mounts      2      0
        3      Dike 1:500 +0.5m  Dike 1:500 extr.            DC Mounts      2      0
        4   Dike Wave resistant         no policy          Alarm Early      0      2
        5      RfR Medium Scale   Alarm Education            no policy      1      0
        6           DC Floating   Alarm Education            no policy      2      0
        7           Alarm Early   Alarm Education            no policy      1      0
        8           Dike 1:1000  Dike 1:500 +0.5m            no policy      2      0
        9       RfR Small Scale   Alarm Education            no policy      1      0
        10     RfR Side channel         no policy            no policy      0      0

In [10]: pathways.to_clipboard()

In [11]: obj_scores = archive[[u'obj 0', u'obj 1',
                      u'obj 2', u'obj 3', u'obj 4', u'obj 5']]

In [12]: import pandas as pd
         import matplotlib.pyplot as plt
         import matplotlib.ticker as ticker
         def make_parallel_plot(nr_columns, labels, maxima, minima):

             fig = plt.figure()
             axes = []

             # we need one axes less than the shape
             for i in range(1, nr_columns):
                 ax = fig.add_subplot(1,nr_columns-1,i,  ylim=(-0.1,1.1))
                 axes.append(ax)
                 ax.set_xlim([i,i+1])
                 ax.xaxis.set_major_locator(ticker.FixedLocator([i]))
                 ax.xaxis.set_ticklabels([labels[i-1]], rotation=45)
                 ax.xaxis.set_tick_params(bottom=False, top=False)

                 #let's put our own tick labels
                 ax.yaxis.set_ticks([])
                 ax.text(i, 1.01, "{:.2f}".format(maxima[i-1]), va="bottom", ha="center")
                 ax.text(i, -0.01,"{:.2f}".format(minima[i-1]), va="top", ha="center")

                 ax.spines['left'].set_bounds(0, 1)
                 ax.spines['right'].set_bounds(0, 1)
                 ax.spines['top'].set_visible(False)
                 ax.spines['bottom'].set_visible(False)


             # for the last axis, we need 2 ticks (also for the right hand side
             ax.spines['right'].set_bounds(0, 1)
             ax.xaxis.set_major_locator(ticker.FixedLocator([i, i+1]))
             ax.xaxis.set_ticklabels(labels[i-1:i+1])
             ax.text(i+1, 1.01, "{:.2f}".format(maxima[i]), va="bottom", ha="center")
             ax.text(i+1, -0.01,"{:.2f}".format(minima[i]), va="top", ha="center")

             # add the tick labels to the rightmost spine
             for tick in ax.yaxis.get_major_ticks():
                 tick.label2On=True

             # stack the subplots together
             plt.subplots_adjust(wspace=0)

             return axes


         def parallel_pareto_front(data, labels):

             def normalize(data, data_to_norm):
                 minima = np.min(data, axis=0)
                 maxima = np.max(data, axis=0)
                 d = maxima - minima

                 d[d==0] = 1

                 norm_data = data_to_norm/d - minima/d
                 return norm_data, minima, maxima


             hof_norm, minima, maxima = normalize(data, data)

             axes = make_parallel_plot(hof_norm.shape[1], labels, maxima, minima)

             # visualize hof
             for i,j in zip(range(hof_norm.shape[1]-1),range(1,hof_norm.shape[1])):
                 ax = axes[i]
                 y = hof_norm[:, i:j+1]
                 x = np.tile([i+1,j+1], (hof_norm.shape[0], 1))
                 ax.plot(x.T, y.T, lw=3)


         normalization_data = np.array([34608.543, 17107.6871672,
                                        756.9814067, 388.78367684,
                                        1111.92459913, 52.2242060867])

         obj_labels = ['$\overline{y}_{damages}$', '$s_{damages}$',
                       '$\overline{y}_{casualties}$', '$s_{casualties}$',
                       '$\overline{y}_{costs}$', '$s_{costs}$']

         sns.set_palette("Paired", 11)
         colors = sns.color_palette("Paired", 11)
         parallel_pareto_front(obj_scores.values*normalization_data, obj_labels)


         artists = []
         labels = []
         for i in range(11):
             artist = plt.Line2D([0,1], [0,1], color=colors[i])
             artists.append(artist)
             labels.append(str(i))
         fig = plt.gcf()
         fig.legend(artists, labels, bbox_to_anchor=(1, 0.9), bbox_transform=plt.gcf().transFigure)

         change_fontsize(fig)
         save_fig(fig, 10)
         plt.show()
```



1 analysis of timing

```
In [1]: from util.util import load_results
        from util import ema_logging

        ema_logging.log_to_stderr(ema_logging.INFO)

        experiments, outcomes = load_results('./data/pathways with timing.tar.gz')
```

```
[INFO] results loaded succesfully from ./data/pathways with timing.tar.gz
```

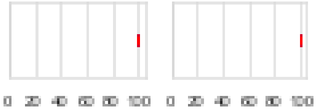
```
In [2]: policy = experiments['policy']
        timing = outcomes['Timing']
```

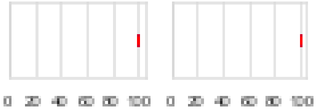
```
In [3]: df = pd.DataFrame([policy, timing[:,0], timing[:,1], timing[:,2]],
                          index=['policy', 'timing 0', 'timing 1', 'timing 2']).T
```

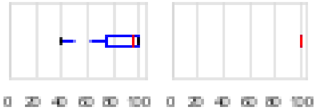
```
In [4]: grouped = df.groupby('policy')
```

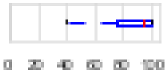
```
In [5]: sns.set(style='whitegrid', rc={'lines.linewidth':0.5, 'grid.linewidth':0.5,
                                       'axes.linewidth':0.5, 'xtick.labelsize':6})

        group_names = [0, 1, 3]

        for name in group_names:
            group = grouped.get_group(name)

            fig = plt.figure(figsize=(2,0.5))

            ax1 = fig.add_subplot(121)
            ax2 = fig.add_subplot(122)

            ax1.boxplot(group['timing 1'].values, vert=False)
            ax2.boxplot(group['timing 2'].values, vert=False)

            for ax in [ax1, ax2]:
                ax.set_xlim(xmin=0, xmax=105)
                ax.set_yticks([])

            plt.savefig('./figs/timing pathway {}.png'.format(int(name)),
                        bbox_inches='tight', dpi=300)


        group_names = [2,4,5,6,7,9]

        for name in group_names:
            group = grouped.get_group(name)

            fig = plt.figure(figsize=(1,.25))

            ax = fig.add_subplot(111)
            ax.boxplot(group['timing 1'].values, vert=False)

            ax.set_xlim(xmin=0, xmax=105)
            ax.set_yticks([])
```

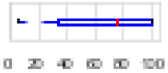
```
plt.savefig('./figs/timing pathway {}.png'.format(int(name)),
            bbox_inches='tight', dpi=300)
```

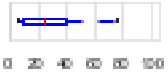


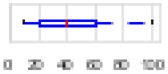


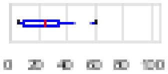


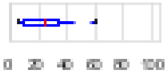












```
In [6]: ema_logging.log_to_stderr(ema_logging.INFO)

        experiments, outcomes = load_results(’./data/pathways with timing.tar.gz’)
```

```
[INFO] results loaded succesfully from ./data/pathways with timing.tar.gz
```

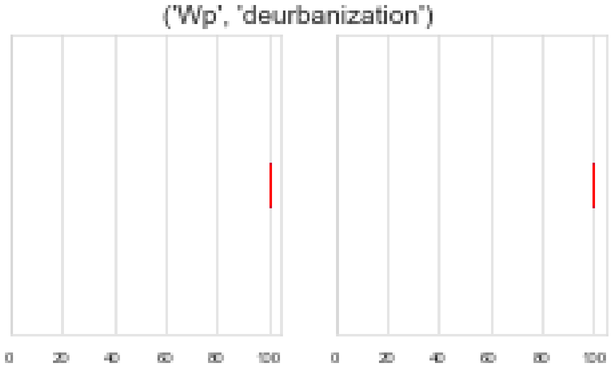
```
In [7]: policy = experiments[’policy’]
        climate = experiments[’climate scenarios’]
        land_use = experiments[’land use scenarios’]
        timing = outcomes[’Timing’]


        df = pd.DataFrame([policy, climate, land_use, timing[:,0], timing[:,1], timing[:,2]],
                          index=[’policy’, ’climate’, ’land use’ ,  ’timing 0’, ’timing 1’, ’timing 2’]
```

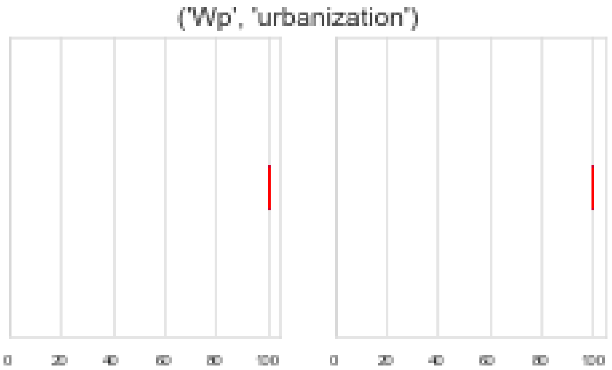
```
In [8]: for i, label in enumerate((’no change/G’, ’no change/G’, ’Wp’)):
            for j in range(i*10+1,  i*10+11):
                df[’climate’].replace(to_replace=j, value=label, inplace=True)
```

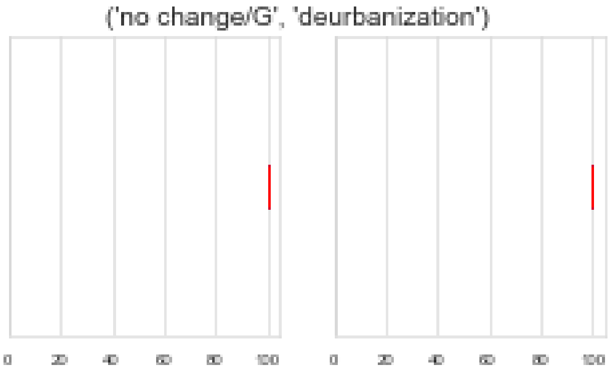
```
In [9]: urbanization = [’sustainableGrowth’, ’urbanizationLargeSteady’,
                         ’urbanizationLargeAndFast’, ’urbanizationDeurbanization’]
        deurbanization = [’Deurbanization’, ’moreNature’, ’NoChange’]

        for entry in urbanization:
            df[’land use’].replace(to_replace=entry, value=’urbanization’, inplace=True)

        for entry in deurbanization:
            df[’land use’].replace(to_replace=entry, value=’deurbanization’, inplace=True)
```

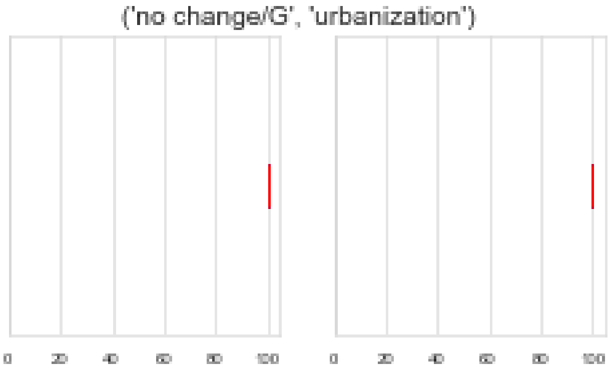
```
In [10]: grouped = df.groupby(’policy’)

         group_names = [0, 1, 3]

         for name in group_names:
             group = grouped.get_group(name)

             for subgroup in group.groupby([’climate’, ’land use’]):
                 label, subgroup = subgroup
```

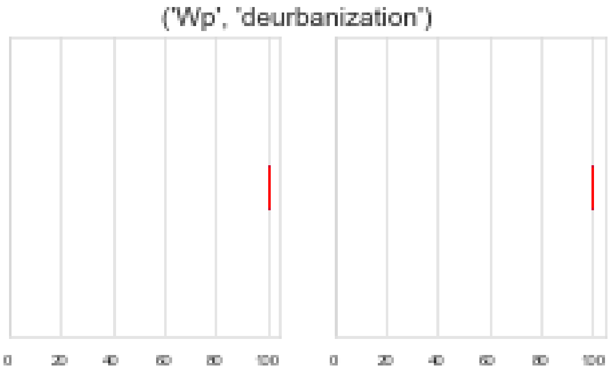
```
        fig = plt.figure(figsize=(4,2))
        fig.suptitle(label)

        ax1 = fig.add_subplot(121)
        ax2 = fig.add_subplot(122)

        ax1.boxplot(subgroup['timing 1'].values, vert=False)
        ax2.boxplot(subgroup['timing 2'].values, vert=False)

        for ax in [ax1, ax2]:
            ax.set_xlim(xmin=0, xmax=105)
            ax.set_yticks([])


group_names = [5]

for name in group_names:


    group = grouped.get_group(name)

    for subgroup in group.groupby(['climate', 'land use']):
        label, subgroup = subgroup

        fig = plt.figure(figsize=(2,2))
        fig.suptitle(label)

        ax = fig.add_subplot(111)
        ax.boxplot(subgroup['timing 1'].values, vert=False)

        ax.set_xlim(xmin=0, xmax=105)
        ax.set_yticks([])


plt.show()
```

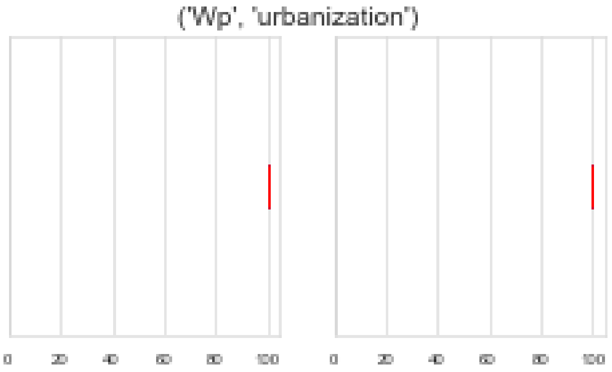


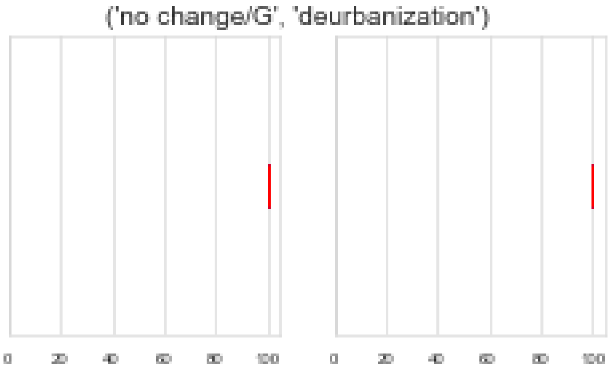

('Wp', 'urbanization')
0
20
40
60
80
100
0
20
40
60
80
100

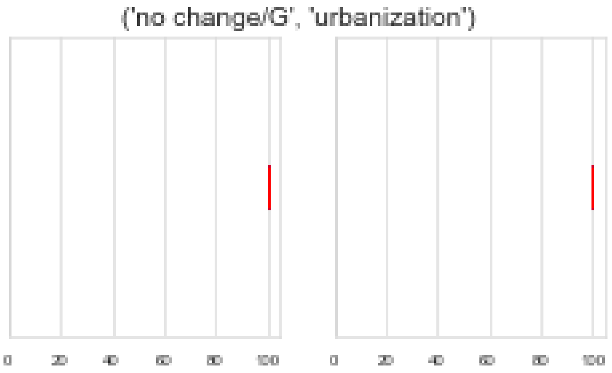

('no change/G', 'deurbanization')
0
20
40
60
80
100
0
20
40
60
80
100

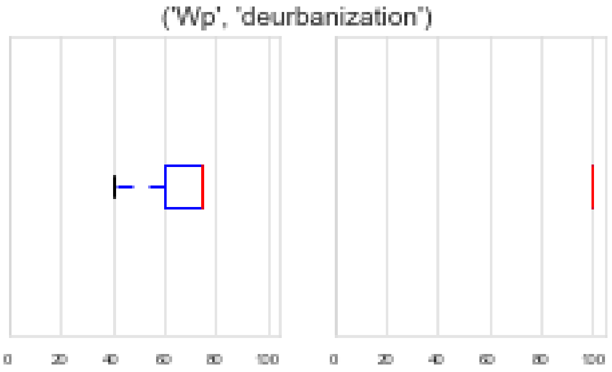

('no change/G', 'urbanization')
0
20
40
60
80
100
0
20
40
60
80
100

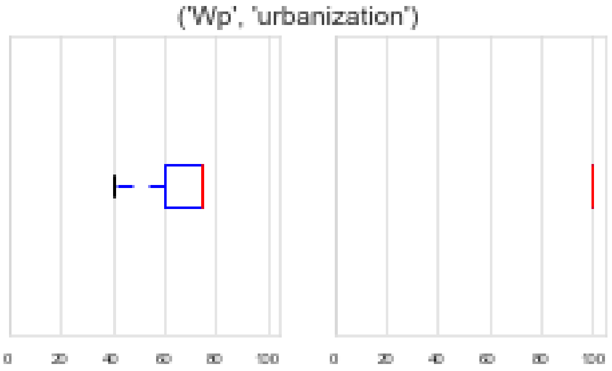

('Wp', 'deurbanization')

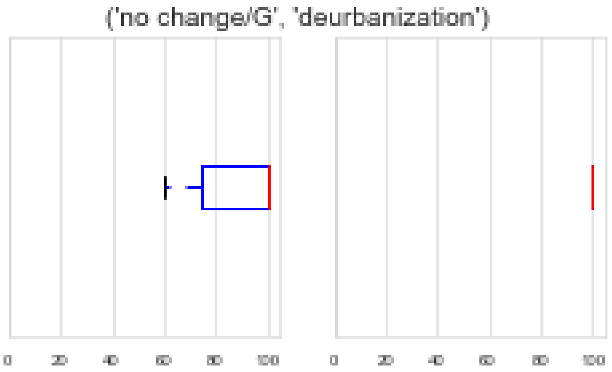

('Wp', 'urbanization')

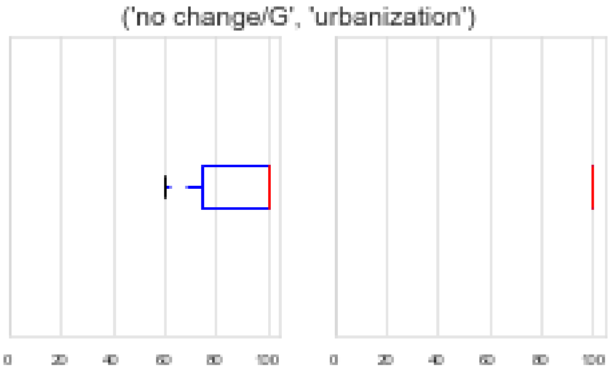

('no change/G', 'deurbanization')

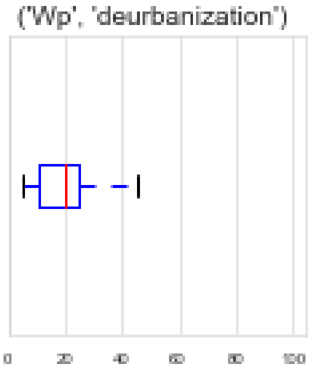

('no change/G', 'urbanization')

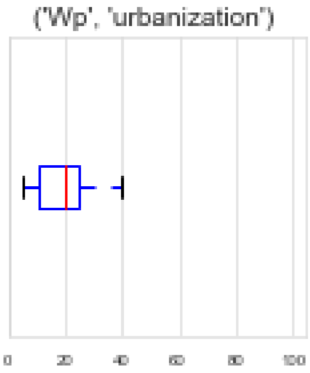

('Wp', 'deurbanization')

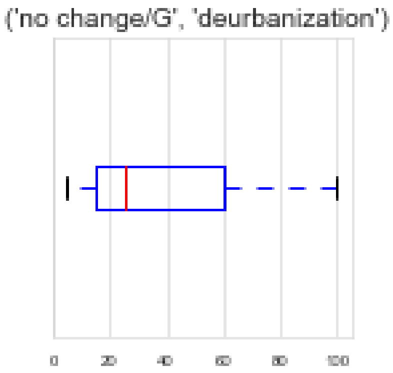

('Wp', 'urbanization')

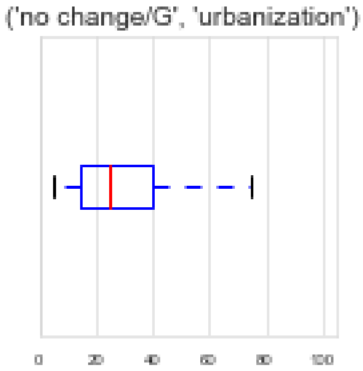

('no change/G', 'deurbanization')
0
20
40
60
80
100
0
20
40
60
80
100


('no change/G', 'urbanization')
0
20
40
60
80
100
0
20
40
60
80
100


('Wp', 'deurbanization')
0
20
40
60
80
100


('Wp', 'urbanization')


('no change/G', 'deurbanization')


('no change/G', 'urbanization')

```
In [68]: # df.groupby(['policy', 'climate', 'land use']).count()
         columns =[ u'timing 0', u'timing 1',u'timing 2']
         df[columns] = df[columns].astype(int)
         def percentile(group):
             percentiles = [25, 50, 75]

             timing_2 = group[’timing 1’]
             timing_3 = group[’timing 2’]

             a = pd.DataFrame({’timing 2’: np.percentile(timing_2.values, percentiles),
                                ’timing 3’: np.percentile(timing_3.values, percentiles)},
                               index=percentiles)
             return a

         tps = df.groupby([’policy’, ’climate’, ’land use’]).apply(percentile)
```

```
In [71]: tps = tps.unstack().unstack().unstack()
```

```
In [72]: tps.to_clipboard()
```

```
In [ ]:
```