Real Micro Crystals - Data Engineering & Exploration 2

playing with different statistics

Michael Janus, June 2018

Use more functions on a real (small) data set.

For explanation and how to usage functions, see the notebook **imgutils_test_and_explain.ipynb**

1. Import the used modules, including the one with test functions:

In [59]:

```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

import matplotlib.pyplot as plt

import imgutils
import imgutils_test as tst
```

In [60]:

```
# Re-run this cell if you altered imgutils or imgutils_test
import importlib
importlib.reload(imgutils)
importlib.reload(tst)
```

```
Out[60]: <module 'imgutils_test' from 'C:\\JADS\\SW\\Grad Proj\\realxtals1\\sources\\imgutils_test.py'>
```

1. Get image files

In [61]:

```
df_imgfiles = imgutils.scanimgdir('../data/Crystals_Apr_12/Tileset7', '.tif')
print(df_imgfiles)
```

```
                                            filename
0  ..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...
1  ..\data\Crystals_Apr_12\Tileset7\Tile_001-002-...
2  ..\data\Crystals_Apr_12\Tileset7\Tile_001-003-...
3  ..\data\Crystals_Apr_12\Tileset7\Tile_002-001-...
4  ..\data\Crystals_Apr_12\Tileset7\Tile_002-002-...
5  ..\data\Crystals_Apr_12\Tileset7\Tile_002-003-...
```

2. Get Image Slice Statistics

This set contains 6 images. Let's slice those up in 4 by 4; this will give total of 6 x 4 x 4 = 96 slices. And also apply the statistics on each slice.

In [62]:

```
statfuncs = imgutils.statfuncs_common_ext()
stat_names = imgutils.stat_names(statfuncs)
print(stat_names)
```

```
['img_min', 'img_max', 'img_mean', 'img_std', 'img_median', 'img_range', 'img_blacktail', 'img_whitetail', 'img_refinte
rval']
```

In [63]:

```
df = imgutils.slicestats(list(df_imgfiles['filename']), 4, 4, statfuncs)
print("records: ", df.shape[0])
df.head()
```

```
records:  96
```

Out[63]:

	filename	s_y	s_x	n_y	n_x	alias	img_min	img_max	img_mean	img_std	img_median	img_
0	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	0	4	4	img0_0-0	5419.0	12927.0	8955.557637	489.754848	8960.0	7508.
1	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	1	4	4	img0_0-1	5248.0	12854.0	8883.137305	501.739963	8893.0	7606.
2	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	2	4	4	img0_0-2	6084.0	10737.0	8786.996070	327.512136	8786.0	4653.
3	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	3	4	4	img0_0-3	7105.0	12208.0	8679.430512	273.673569	8679.0	5103.
4	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	1	0	4	4	img0_1-0	4534.0	10926.0	8982.867158	380.410977	8980.0	6392.

Normalize the statistics using 'standarization'

In [64]:

```
imgutils.normalize(df, stat_names)
df.head()
```

Out[64]:

	filename	s_y	s_x	n_y	n_x	alias	img_min	img_max	img_mean	img_std	...	img_refinterval
0	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	0	4	4	img0_0-0	5419.0	12927.0	8955.557637	489.754848	...	1158.00025
1	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	1	4	4	img0_0-1	5248.0	12854.0	8883.137305	501.739963	...	1334.00000
2	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	2	4	4	img0_0-2	6084.0	10737.0	8786.996070	327.512136	...	1157.00000
3	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	3	4	4	img0_0-3	7105.0	12208.0	8679.430512	273.673569	...	343.00000
4	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	1	0	4	4	img0_1-0	4534.0	10926.0	8982.867158	380.410977	...	545.00025

5 rows × 24 columns

In [65]:

```
stat_normnames = imgutils.normalized_names(stat_names)
print(stat_normnames)
```

```
['|img_min|', '|img_max|', '|img_mean|', '|img_std|', '|img_median|', '|img_range|', '|img_blacktail|', '|img_whitetail
|', '|img_refinterval|']
```

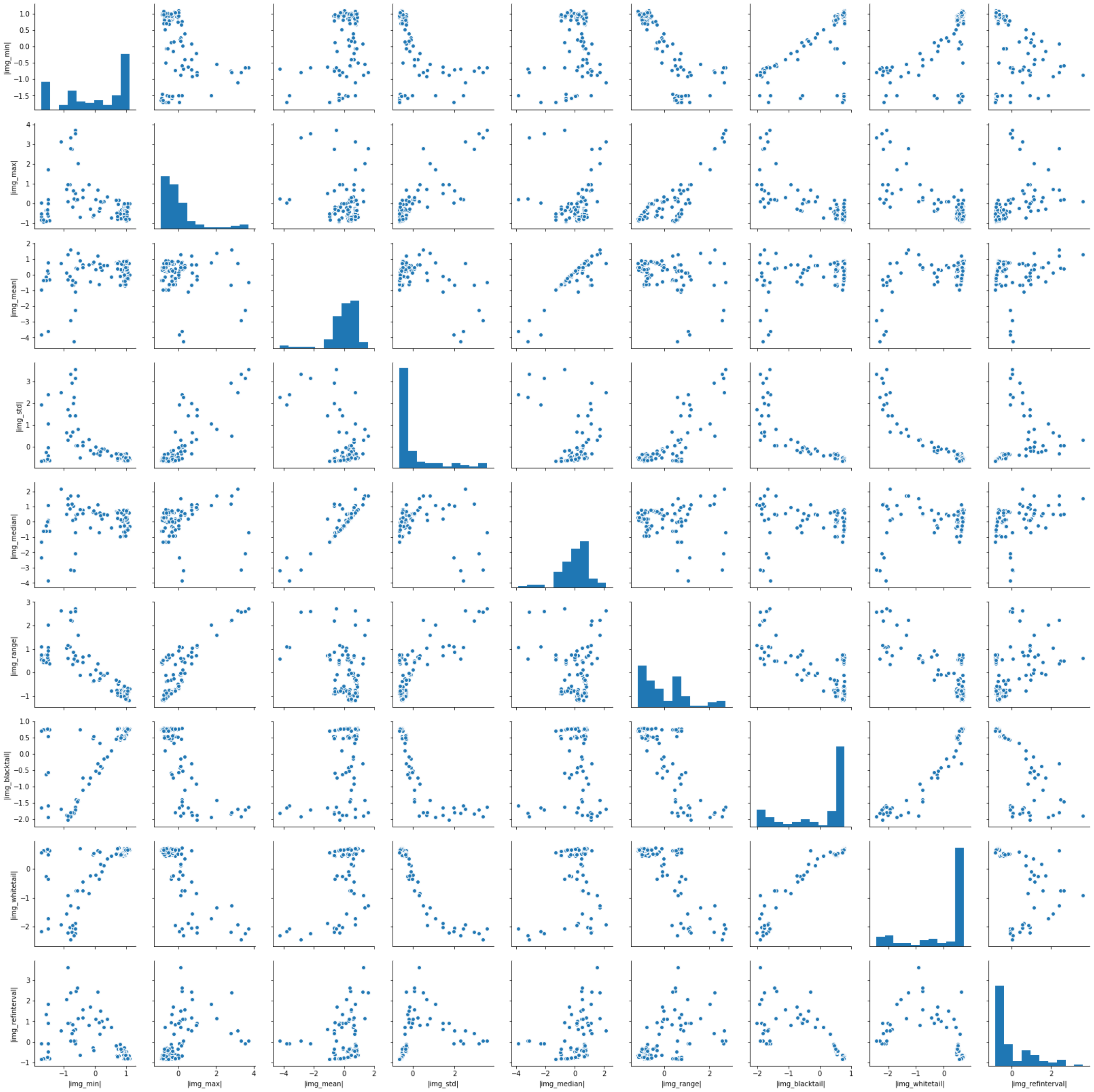
3. Check some combinations for patterns

(using the seaborn pairplot)

In [66]:

```
import seaborn as sb
```

In [67]:
```
%matplotlib inline
sb.pairplot(df, vars=stat_normnames)
plt.show()
```

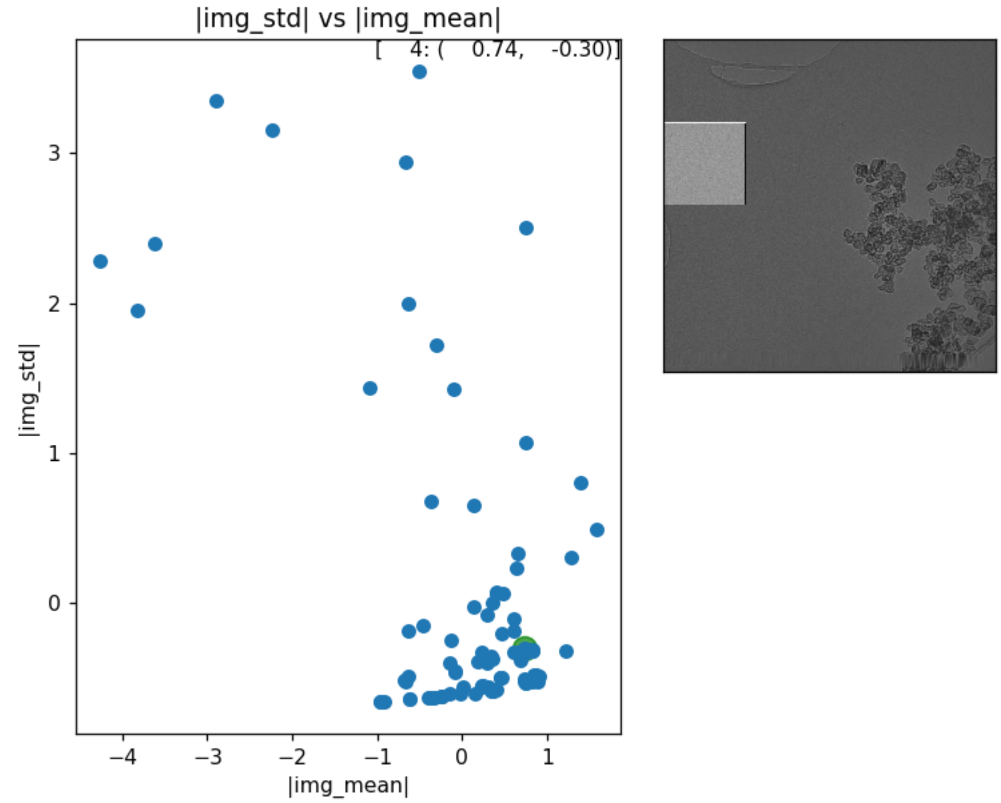


4. Inspect interactively

Let's inspect some combinations that have 'signs of clustering' in the interactive graph

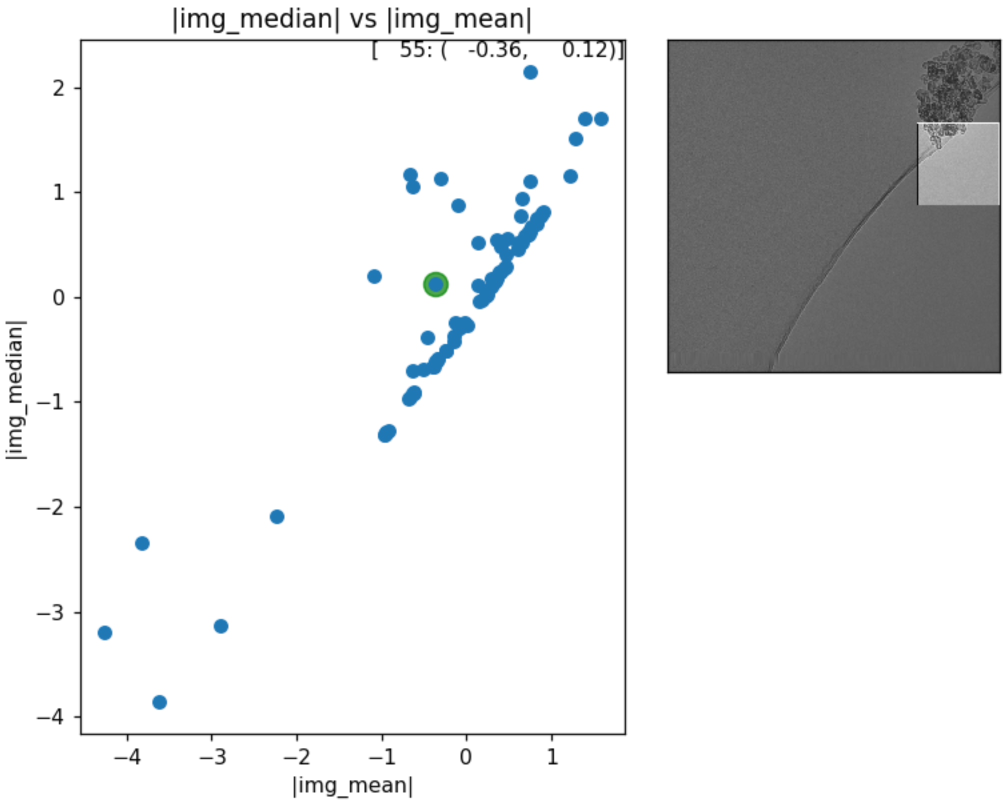
In [68]:
```
%matplotlib notebook
```

```
In [69]: imgutils.plotwithimg(df, '|img_mean|', '|img_std|', imgutils.highlightimgslice, True)
```



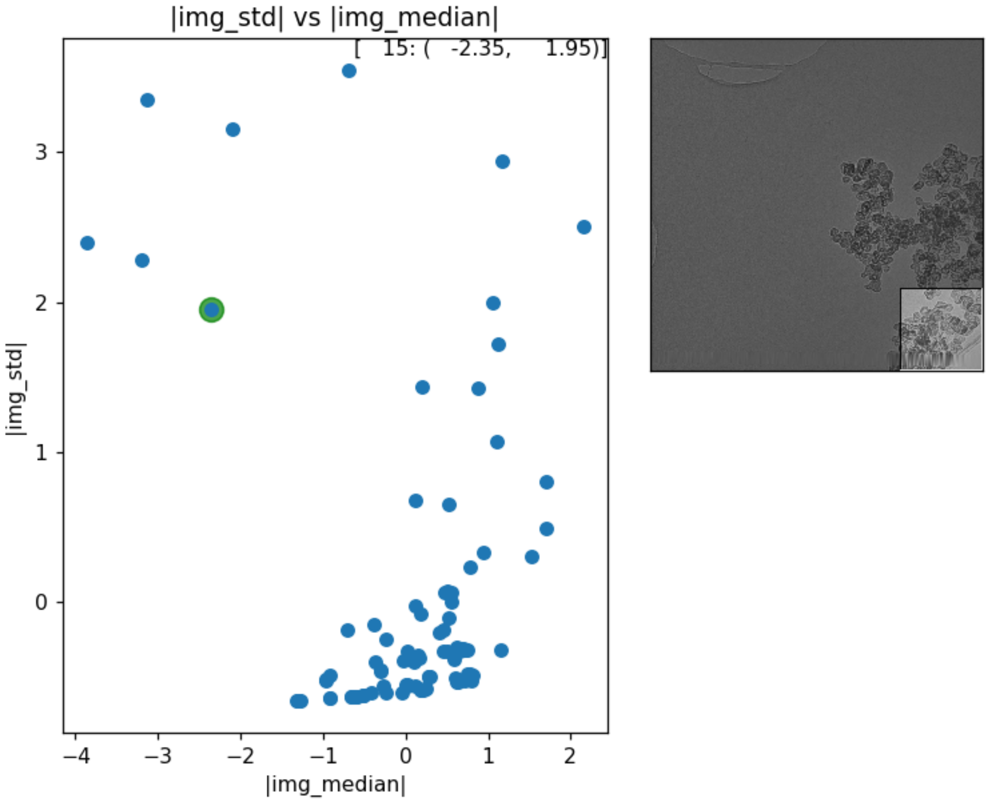
Looks likt the sort-of cluster in lower right are points without a crystal

```
In [70]: imgutils.plotwithimg(df, '|img_mean|', '|img_median|', imgutils.highlightimgslice, True)
```



The separation is not representative, the group at top-left contains both with and without micro crystals

```
In [73]: imgutils.plotwithimg(df, '|img_median|', '|img_std|', imgutils.highlightimgslice, True)
```



This looks better, bottom right are empty regions, top-left have crystals.

5. Heatmaps

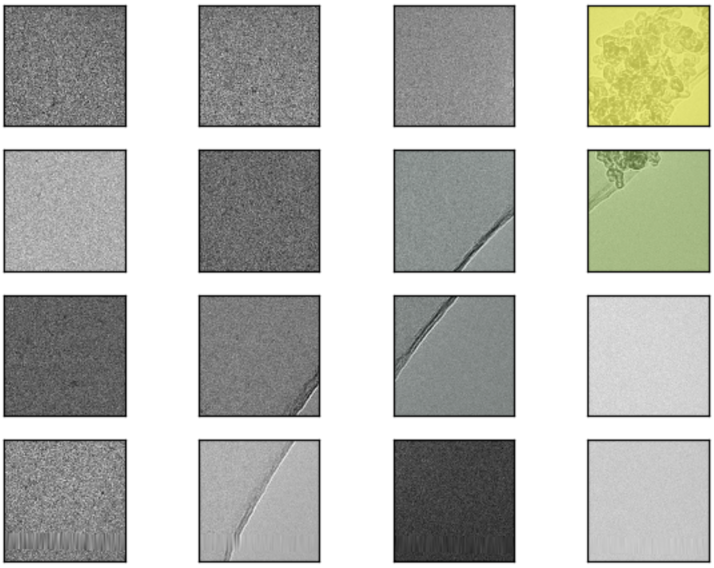
Let's do an attempt to create a score for a heatmap. Looks like |img_std| is most infromative

```
In [74]: imgname = df_imgfiles.iloc[3]['filename']
         print(imgname)

         ..\data\Crystals_Apr_12\Tileset7\Tile_002-001-000_0-000.tif
```

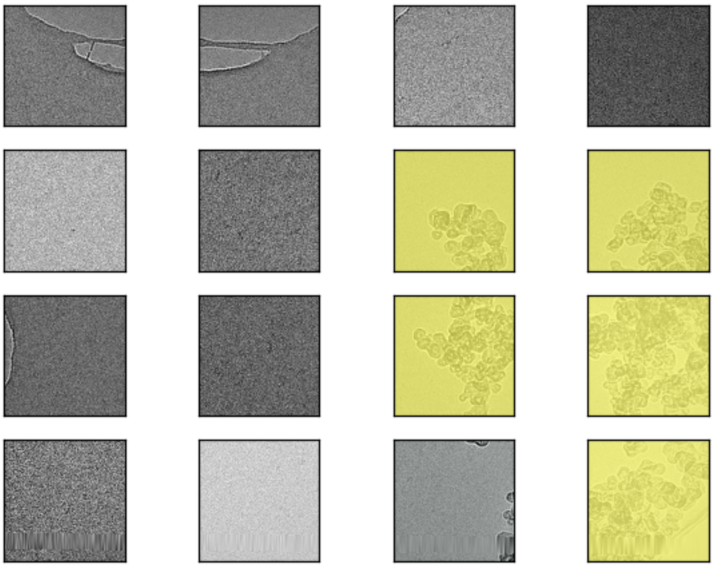
```
In [75]: imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
```

In [76]:
```
imgutils.showheatmap(imgs, heats)
```

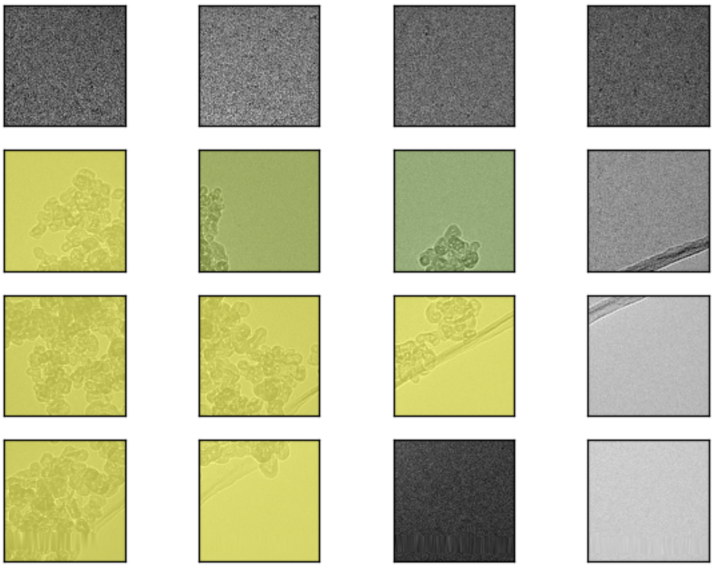


Yes, looks great!. Let's check for some other images as well

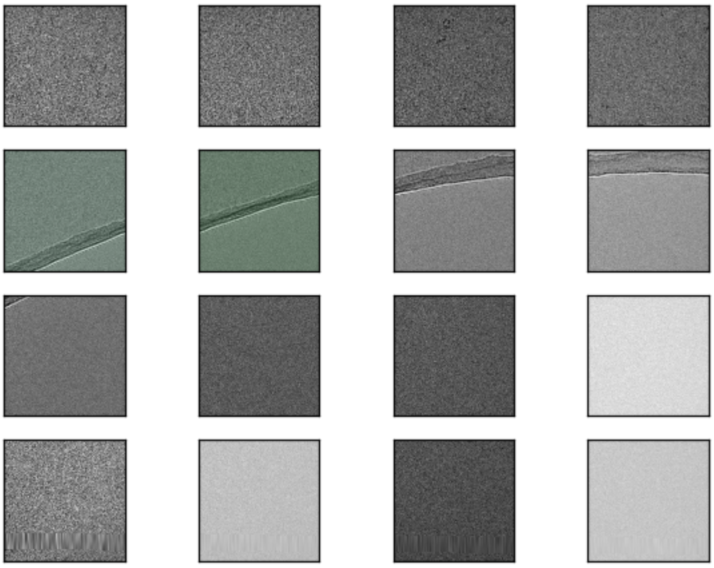
In [78]:
```
imgname = df_imgfiles.iloc[0]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



In [79]:
```
imgname = df_imgfiles.iloc[1]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```

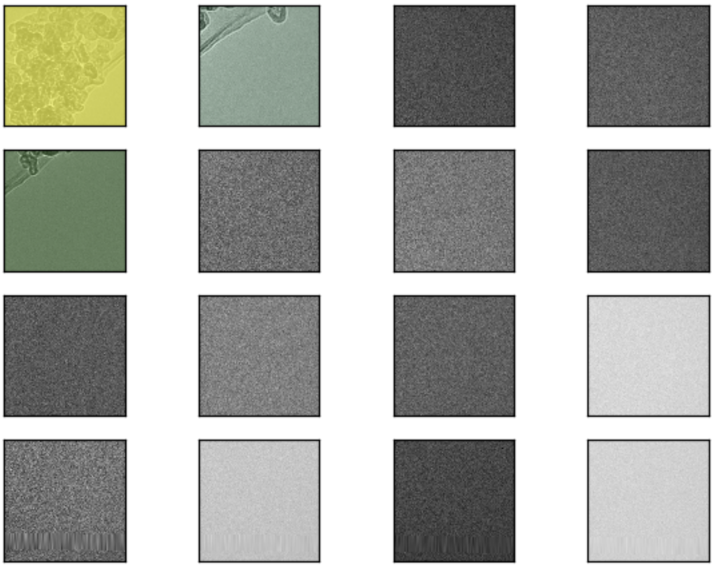


In [80]:
```
imgname = df_imgfiles.iloc[2]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



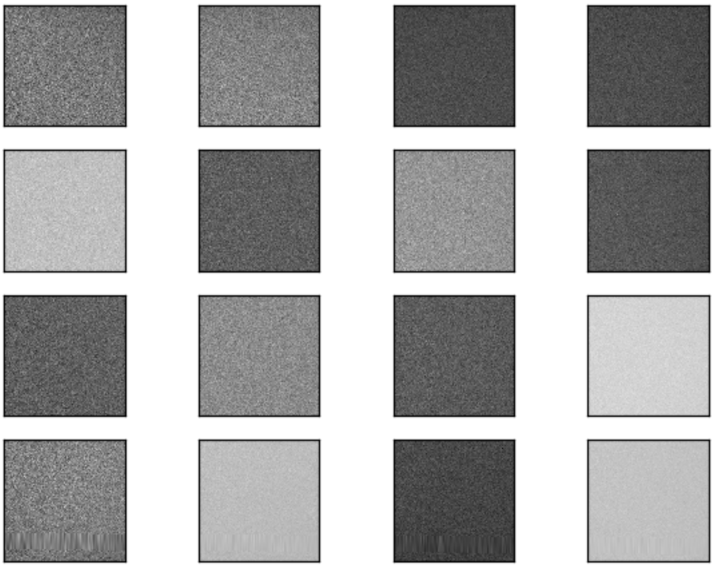
In [81]:

```
imgname = df_imgfiles.iloc[4]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



In [82]:

```
imgname = df_imgfiles.iloc[5]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



[So far, this was a repeat of previous session of June 15]

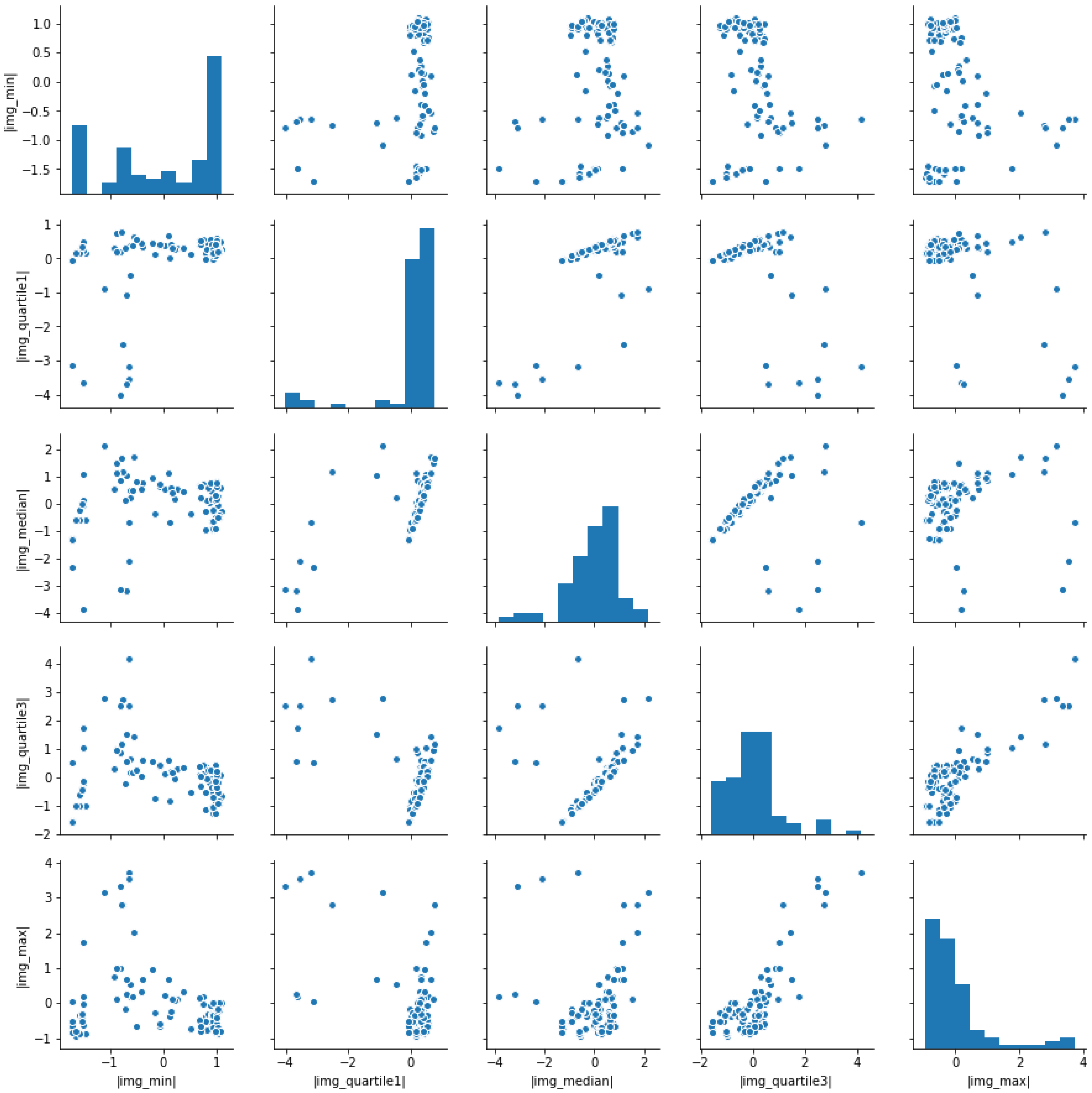
6. Try some more stats (June 19)

The '5 number statistics'

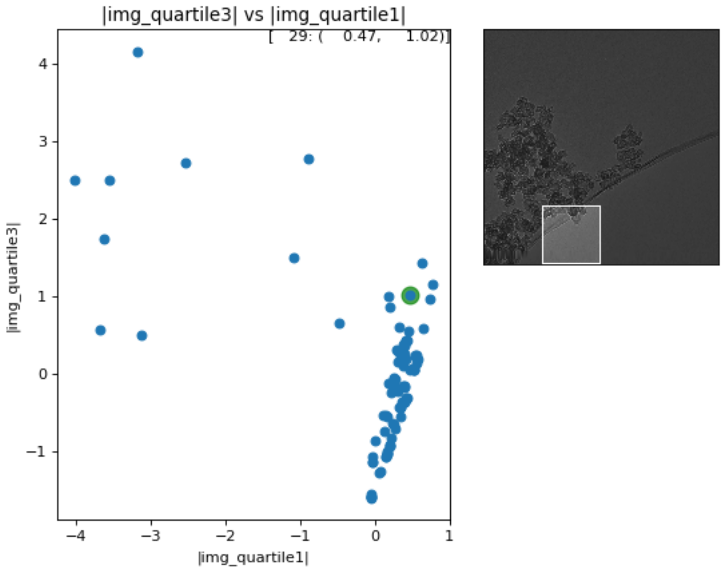
```
In [85]: %matplotlib inline
         statfuncs = imgutils.statfuncs_5numsummary()
         stat_names = imgutils.stat_names(statfuncs)
         stat_normnames = imgutils.normalized_names(stat_names)
         df = imgutils.slicestats(list(df_imgfiles['filename']), 4, 4, statfuncs)
         imgutils.normalize(df, stat_names)

         sb.pairplot(df, vars=stat_normnames)
```

```
Out[85]: <seaborn.axisgrid.PairGrid at 0x15464278>
```

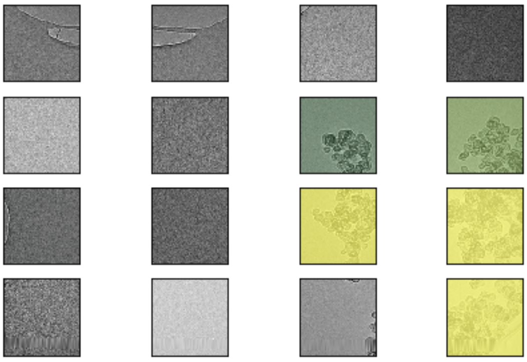


In [87]:
```
%matplotlib notebook
imgutils.plotwithimg(df, '|img_quartile1|', '|img_quartile3|', imgutils.highlightimgslice, True)
```



In [88]:
```
# Looks like here that if quartile 2 needs to be > 0 and quartile 1 < -1
df['score'] = df['|img_quartile3|'] - df['|img_quartile1|']
df['|score|'] = imgutils.norm_standardize(df, 'score')
```

In [89]:
```
imgname = df_imgfiles.iloc[0]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|score|')
imgutils.showheatmap(imgs, heats)
```

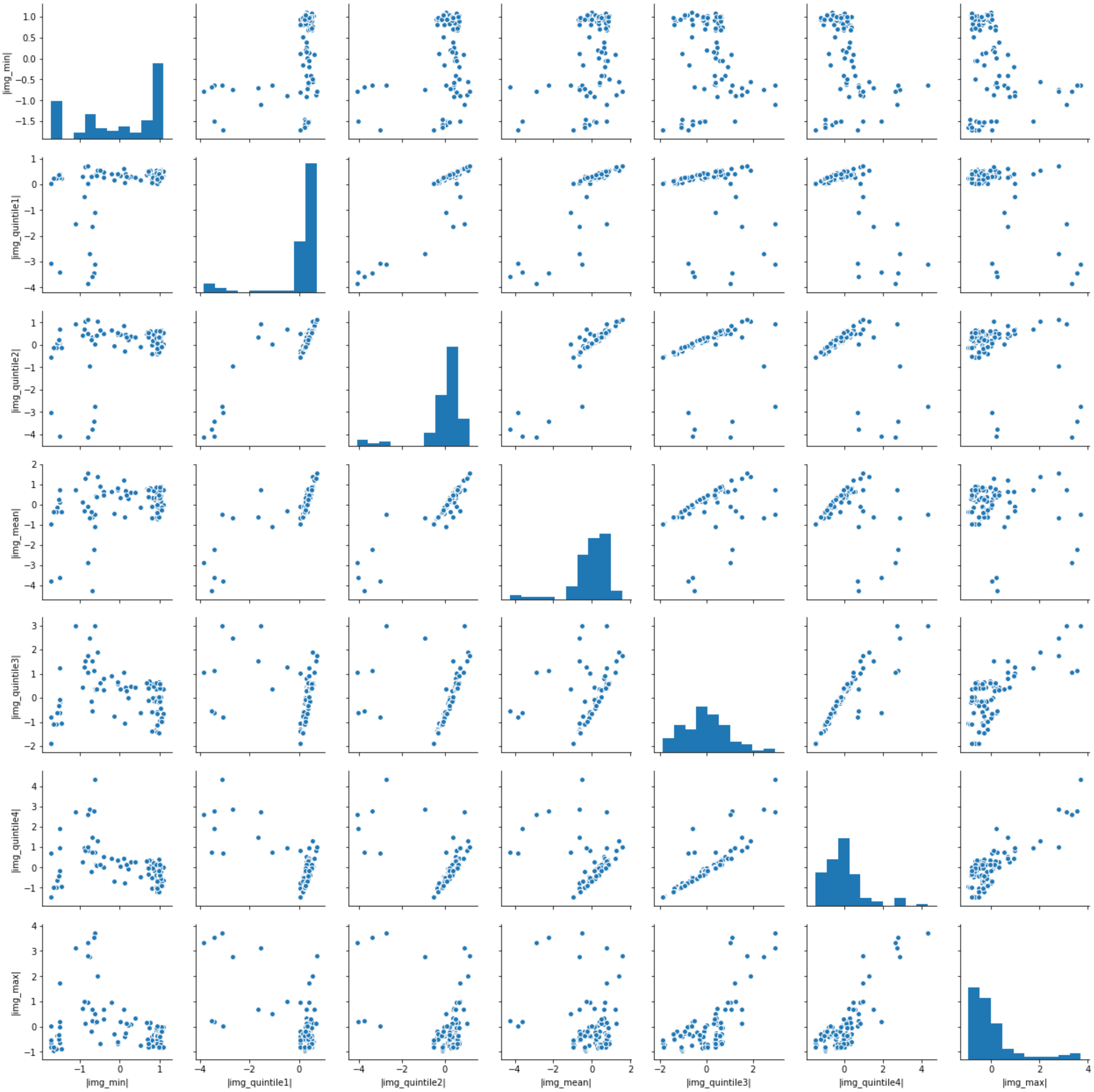


7 number stats *

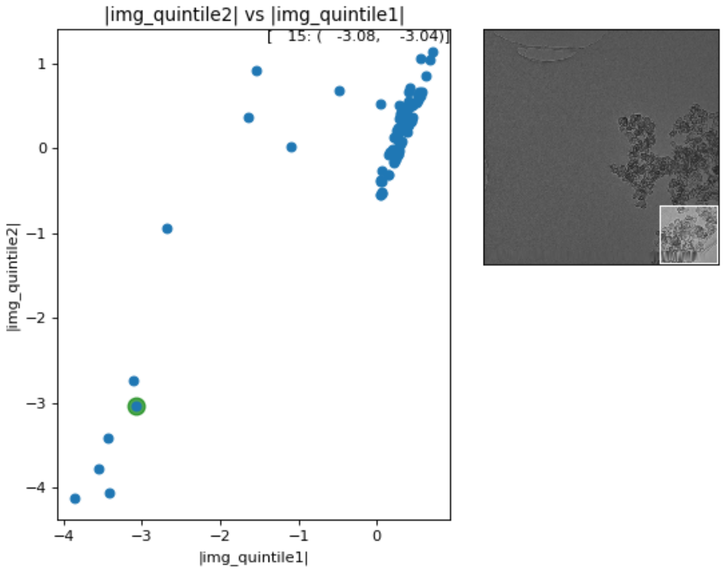
In [90]:
```
%matplotlib inline
statfuncs = imgutils.statfuncs_7numsummary()
stat_names = imgutils.stat_names(statfuncs)
stat_normnames = imgutils.normalized_names(stat_names)
df = imgutils.slicestats(list(df_imgfiles['filename']), 4, 4, statfuncs)
imgutils.normalize(df, stat_names)

sb.pairplot(df, vars=stat_normnames)
```

Out[90]:
```
<seaborn.axisgrid.PairGrid at 0x1518ad30>
```

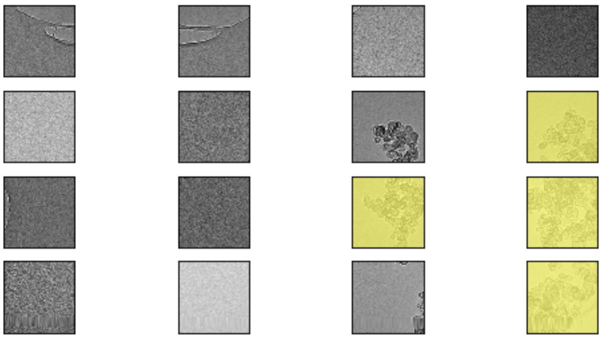


```
In [91]: %matplotlib notebook
         imgutils.plotwithimg(df, '|img_quintile1|', '|img_quintile2|', imgutils.highlightimgslice, True)
```



here, quintile 1 looks like pretty good separting statistics

```
In [92]: # looks like here that if quartile 2 needs to be > 0 and quartile 1 < -1
         df['score'] = -df['|img_quintile1|']
         df['|score|'] = imgutils.norm_minmax(df, 'score')
         imgname = df_imgfiles.iloc[0]['filename']
         imgs, heats = imgutils.getimgslices_fromdf(df, imgname, 'score')
         imgutils.showheatmap(imgs, heats)
```



hmm, one obvious one was missed, so need clustering and not one statistics!

box-and-whisker stats

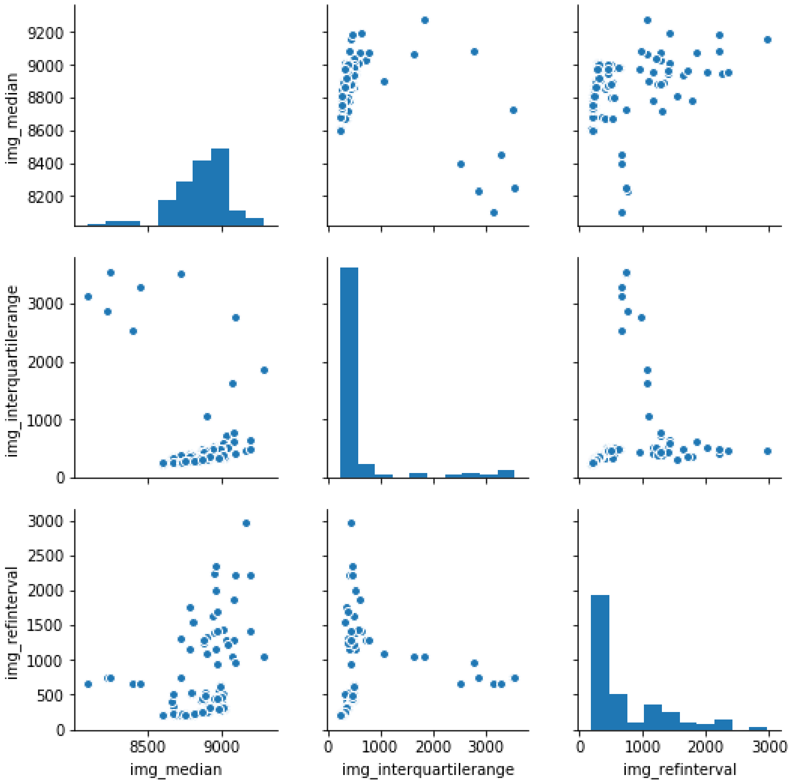
In [94]:

```
statfuncs = imgutils.statfuncs_boxandwhisker()
stat_names = imgutils.stat_names(statfuncs)
stat_normnames = imgutils.normalized_names(stat_names)
df = imgutils.slicestats(list(df_imgfiles['filename']), 4, 4, statfuncs)
imgutils.normalize(df, stat_names)


%matplotlib inline
sb.pairplot(df, vars=stat_names)
```

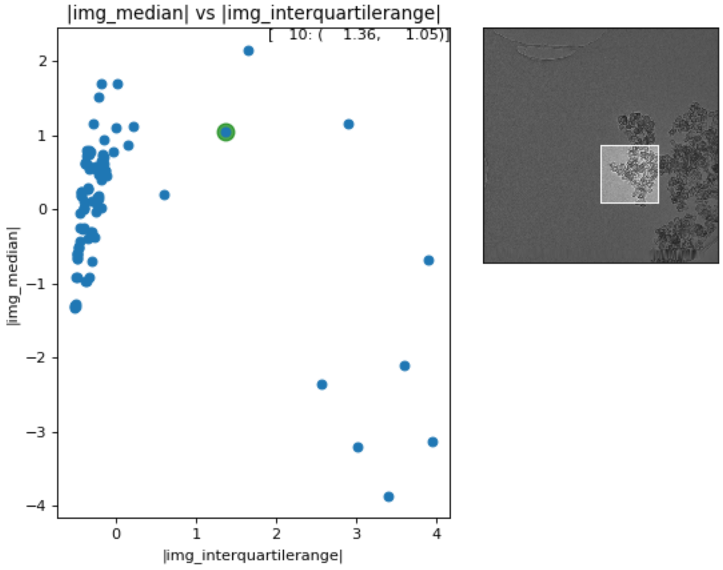
Out[94]:

```
<seaborn.axisgrid.PairGrid at 0x233aaf60>
```



In [95]:

```
# check one interactively
%matplotlib notebook
imgutils.plotwithimg(df, '|img_interquartilerange|', '|img_median|', imgutils.highlightimgslice, True)
```

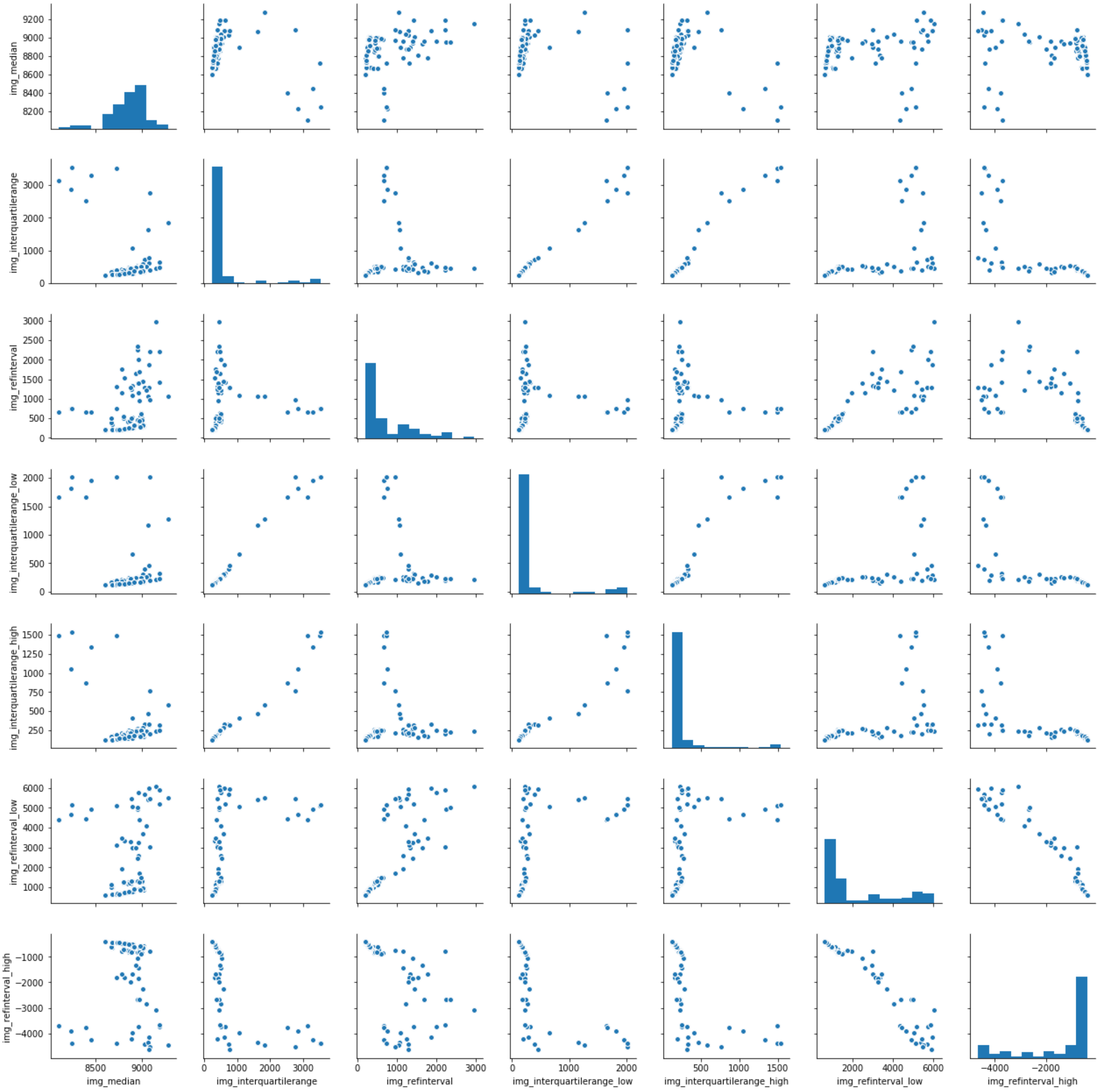


separtion of clusters is just at top of dense area of top left (where it becomes more sparse)

```
In [96]: statfuncs = imgutils.statfuncs_boxandwhisker_ext()
         stat_names = imgutils.stat_names(statfuncs)
         stat_normnames = imgutils.normalized_names(stat_names)
         df = imgutils.slicestats(list(df_imgfiles['filename']), 4, 4, statfuncs)
         imgutils.normalize(df, stat_names)


         %matplotlib inline
         sb.pairplot(df, vars=stat_names)

Out[96]: <seaborn.axisgrid.PairGrid at 0x2acfd7f0>
```

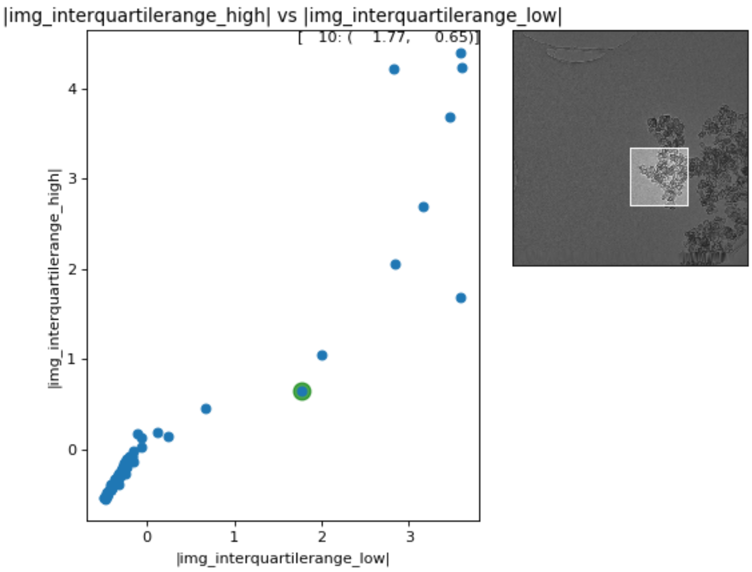


```
In [97]: print(stat_normnames)

         ['|img_median|', '|img_interquartilerange|', '|img_refinterval|', '|img_interquartilerange_low|', '|img_interquartilera
         nge_high|', '|img_refinterval_low|', '|img_refinterval_high|']
```

In [98]:

```
%matplotlib notebook
imgutils.plotwithimg(df, '|img_interquartilerange_low|', '|img_interquartilerange_high|', imgutils.highlightimgslice, T
rue)
```

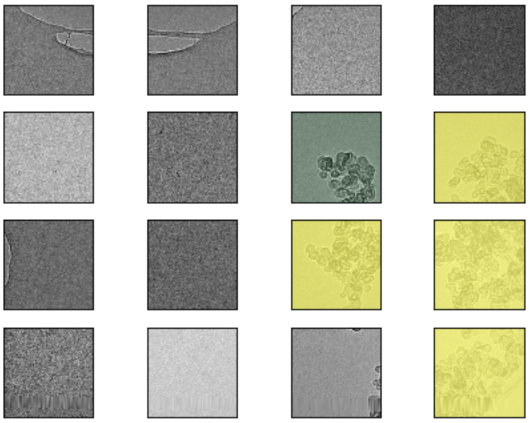


In [99]:

```
# lower left cluster is non-particles; lets try to separate them:
df['score'] = df['|img_interquartilerange_low|'] + df['|img_interquartilerange_high|']
#df['|score|'] = imgutils.norm_standardize(df, 'score')
#df['|score|'] = imgutils.norm_minmax(df, 'score')
```

In [100]:

```
imgname = df_imgfiles.iloc[0]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, 'score')
imgutils.showheatmap(imgs, heats)
```



In [101]:

```
# mwa
```