3.3

实验分析

我利用小组实验产生的输出做出了聚类分析.

我阅读了课程链接中的表格部分——"聚类算法比较". 我发现 `Affinity propagation` 方法的用例是"Many clusters, uneven cluster size, non-flat geometry", 即"多簇, 簇的大小无需均匀, 几何分布无需平坦". 这契合对高度分布的分析, 我希望了解高楼的分布, 是否存在高楼群, 即高楼附近环绕小楼. 在区分高楼群的过程, 也无需将群的大小固定, 而且高楼群的楼层高度也无需均匀分布. 因此我想尝试这个方法.

我查阅了这个方法的文档. 它的参数如下:

- `damping` , default=0.5. 衰减因子. 它决定了集簇后期进入的点所占的权重. 衰减因子越大, 后期进入的点所占的比重越低.
- `max_iter` , default=200. 最大迭代次数.
- `convergence_iter` , default=15. 收敛迭代次数, 如果到了集簇后期, 簇的数目在数次(`convergence_iter`)迭代之后不再改变, 即代表收敛已经达到, 无须继续迭代.
- `copy` , default=True. 是否复制输入数据.
- `preference` , array-like of shape (n_samples,) or float, default=None. 偏好, 数字越大的越容易成为簇的典型代表.
- `affinity` , {'euclidean', 'precomputed'}, default='euclidean'. 联络, 默认选择欧几里得联络计算点之间的距离.
- `verbose` , default=False. 是否输出更多的信息, 运行记录.
- `random_state` , RandomState instance or None, default=None. 是否选择使用固定的随机数序列, 选择固定的即可重复分簇结果.

我将研究 `damping` 的影响, 用楼的高度或者 `None` 固定 `preference` , 设置 `random_state=5` 来重复结果. 其他均设置为默认参数.

In [1]:
```
import geopandas as gpd
import numpy as np
import pandas as pd
```

In [2]:
```
selected_heights = gpd.read_file("outputs/selected_heights.geojson")
```

In [3]:
```
from sklearn.cluster import AffinityPropagation
```

In [4]:
```
#display(selected_heights)
```

In [5]:
```
selected_heights_centroid = selected_heights.copy()

# convert Polygon to centroids
selected_heights_centroid["geometry"] = selected_heights_centroid["geometry"].apply(lambda p : p.centroid)

# skim off entries without floor information
selected_heights_centroid = selected_heights_centroid[~np.isnan(selected_heights_centroid["Floor"])]
```

In [6]:
```
# create input format for clustering
selected_heights_centroid_x = pd.Series(selected_heights_centroid['geometry'][:].x)
selected_heights_centroid_y = pd.Series(selected_heights_centroid['geometry'][:].y)
selected_heights_centroid_P = np.column_stack((selected_heights_centroid_x, selected_heights_centroid_y))
```

In [7]:
```
def APClusters(prep):
    """
    return the AffinityPropagation, dampings, and dataframes predicted by AffinityPropagation
    """
    clustering = []
    damps = []
    for damp in np.arange(5, 10)/10:
        #print (damp)
        damps.append(damp)
        clustering.append(AffinityPropagation(damping=damp, preference=prep, random_state=5).fit(selected_heights_centroid_P))
    predicted = []
    for c in clustering:
        predicted.append(c.predict(selected_heights_centroid_P))
        #print(c)
    return clustering, damps, predicted
```

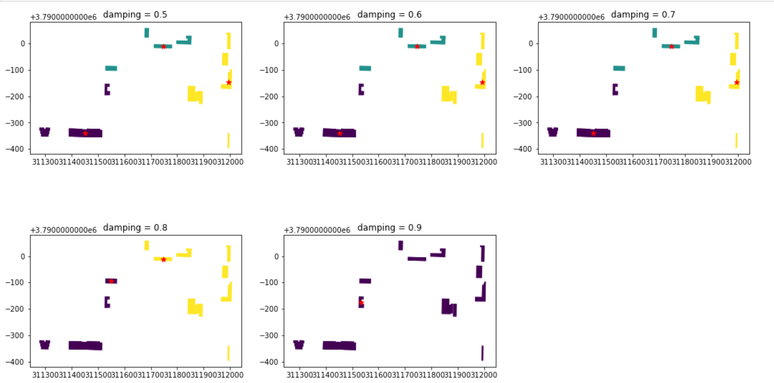
In [8]:
```
noprep = APClusters(None)
```

In [9]:
```
#selected_heights.head()
```

In [10]:
```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2, 3, figsize=(18, 10))
fig.delaxes(axs[1][2])

clustering, damps, predicted = noprep

i = 0
for damp, c, p in zip(damps, clustering, predicted):
    row = int(i/3)
    col = i % 3
    ax = axs[row, col]
    selected_heights_copy = selected_heights.copy()
    selected_heights_copy["damping_%.1f" % damp] = p
    ax.set_title("damping = %.1f" % damp)
    selected_heights_copy.plot(column="damping_%.1f" % damp, ax=ax)
    centroids = c.cluster_centers_
    ax.scatter(centroids[:, 0], centroids[:, 1], label="centroids from fit", marker='*', s=50, c='r')
    i += 1
```



小结 —— preference = None

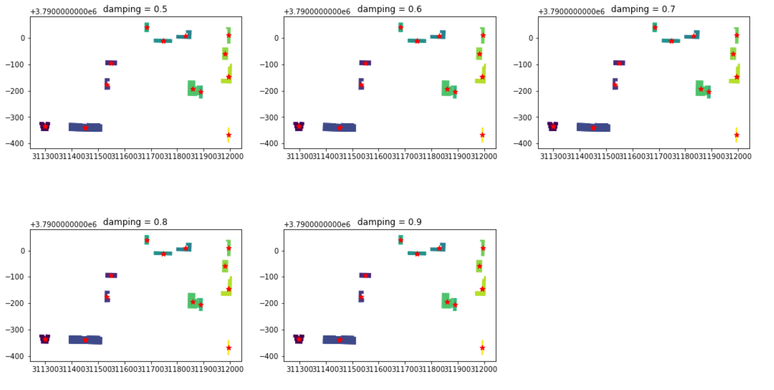
使用 `preference = None` 的时候, 默认会取输入数据的平均值作为典型数据. 我将结果绘制如上图.

In [11]:
```
prep_floor = APClusters(selected_heights["Floor"])
```

In [12]:
```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2, 3, figsize=(18, 10))
fig.delaxes(axs[1][2])

clustering, damps, predicted = prep_floor

i = 0
for damp, c, p in zip(damps, clustering, predicted):
    row = int(i/3)
    col = i % 3
    ax = axs[row, col]
    selected_heights_copy = selected_heights.copy()
    selected_heights_copy["damping_%.1f" % damp] = p
    ax.set_title("damping = %.1f" % damp)
    selected_heights_copy.plot(column="damping_%.1f" % damp, ax=ax)
    centroids = c.cluster_centers_
    ax.scatter(centroids[:, 0], centroids[:, 1], label="centroids from fit", marker='*', s=50, c='r')
    i += 1
```



小结 —— preference = Floor

使用 `preference = dataframe['Floor']` 的时候, 会取输入数据的各个楼的高度作为典型数据. 我将结果绘制如上图.

实验结果

来自preference的影响

我尝试了两种 `preference` 的设置

- 设置其为 `None` : 这时默认的典型样本是均值, 因此我们仍然可以集簇.
- 设置其为 `selected_heights["Floor"]` : 这时的典型样本数据是各个楼的高度, 因此很难将他们集簇起来. 从图中可以看出, 各个建筑各自为簇. #### 来自damping的影响 我只分析 `preference=None` 的影响, 另一种设置已经被证明失败. 选取不同的 `damping` 会出现不同的集簇数据, 当 `damping` 变大时, 集簇数目变少. 这是符合预期的, 因为 `damping` 很大时, 后面读取的数据所占的权重已经很小了, 不影响整个集簇的过程, 因此它们只能被放入最先生成的簇中.

In [ ]:
```
```