MATLAB 微博数据分析与挖掘

简介

```
MyProject = BasicClass
MyProject.Name = "MATLAB 微博数据分析与挖掘";
MyProject.Author = "刘有哲";
MyProject.StudentID = "1830710052";
MyProject.CodeLines = 1071;   % Containing all the comments
MyProject.GitHubRepository = https://github.com/pickerxxr/MATLAB-weibo-pj
MyProject.Time = 2020.05.27
```

摘要

给定的微博数据有二十个维度，其中有一些我认为从实际生活的角度来说，对于用户的分析和使用体验的提升并没有太大作用，暂时没有考虑,而是将重心放在和现实生活（如**新冠肺炎疫情**相关）分析的结合上，利用社交网络和统计学的相关知识来解释数据的分布规律。除了一些基本的**数据分析**以及**可视化**部分，另外尝试学习并实现了**用CVX解决K-means**分类算法的实际实现，**GUI界面**的制作以及**虚假用户的筛选**的挖掘。代码的实现全部为原创，如算法中有具体理论知识的借鉴已经在下面的文章中标明。

主要实现的有（目录大纲）

|— 完美读取所有数据至一张表中（主要解决了**分数据类型精确读取**）

|— 基本统计量的统计及可视化

|　　|— 每个用户发送的微博数量（主要解决快速技术的实现）

|　　|— 微博的发布平台分布（主要解决小项的合并）

|　　|— 一些特征量的饼图分布（位置、是否原创）

|　　|— 情感指数的分布特征

|—点赞、评论、转发三大要素一系列分析（相关性、离散性）

|—时间序列分析

|　　|—情感随肺炎疫情发展的波动（观察时间节点的情感波动）

|　　|—以标签数量为标志的事件发生（观察平均标签数量的变化）

|—几组数据之间的相关性分析

|　　|—探寻微博影响力的因素（数据有限，暂时以点赞评论转发为标准）

|　　|—一些变量之间的相关系数大小比较

|　　|— CVX 工具包的使用（解决一个简单的最小二乘的最优化问题）

|—K-means分类算法（利用点赞、评论数量，采取一部分数据来进行K-means算法实现分类）

|—GUI图形化界面（实现了输入用户ID，输出此用户的所有微博的点赞评论数的图像）

|—虚假用户的检测（对数据做了稍微深入的挖掘，利用社交网络的尝试找出可能存在的虚假用户）

读取数据文件

> 以导入广州的数据文件为例

```
%% Guangzhou
filename = '.\Guangzhou.csv';
delimiter = ',';
startRow = 2;
formatSpec = '%s%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%[^\n\r]';
fileID = fopen(filename,'r');
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter, 'TextType',
'string', 'EmptyValue', NaN, 'HeaderLines' ,startRow-1, 'ReturnOnError', false,
'EndOfLine', '\r\n');
fclose(fileID);
Guangzhou = table(dataArray{1:end-1}, 'VariableNames',
{'author','source','forward_num','comment_num','likes_num','month','day','hour',
'is_original','forward_link_length','original_word_num','content_word_num','forw
ard_word_num','sentiment','emoji_num','tag_num','mention_num','is_located','link
_num','topic_num'});
clearvars filename delimiter startRow formatSpec fileID dataArray ans;
```

> 运行结果如下

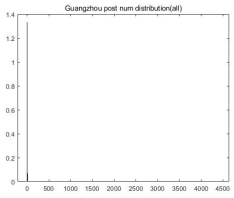
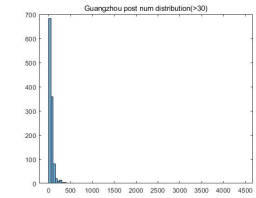
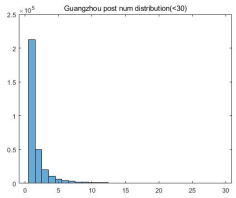
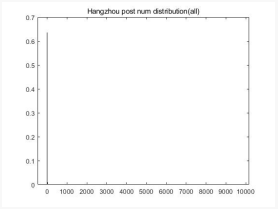
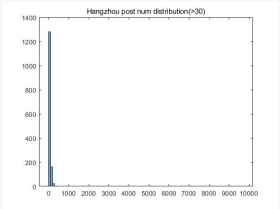
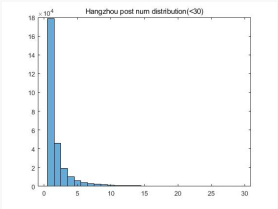
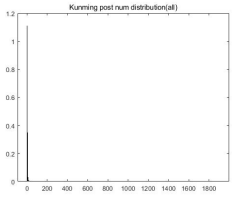
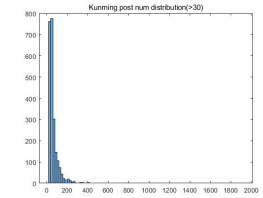
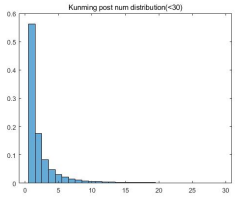
```
%%
>> who

您的变量为:

Guangzhou  Hangzhou   Kunming

%%
>> class(Guangzhou)

ans =

    'table'
```

- 这里一个实现过程中发现的问题是用户的id是一个六位的序列号，而不是数值型的元素，所以不能直接用`csvread(filename);`命令（只能读入数据类型一致的csv文件），如果序列号出现开头为0的项，会出现格式改变的问题，所以尝试了采用`textscan(fileID);`的命令，而把第一项当成`%s`来看。

一些基本统计量的分析

用户发送微博条数

1、首先是对于第一个用户名ID `author` 变量的统计，通过对于三个城市发送微博数量的统计，找出post的数目分布情况。其中以30条微博为分界线，通过简单的 `histogram()` 命令画出两个统计量的分布。这里取到30为分解的首要原因是，数据采样采取的是一个月($\approx$ 30)内的用户数据，而可以依据经验认为30以上的可以看作非常活跃的用户(平均每天发一条以上)，而30以下的相对没那么活跃；另外，通过所有数据的分布图可以看出，数据太过于集中于0~2附近，使得在高数量区域的分布无法显示，所以对于30以上的数据另作处理。

	Distribution of all post data ($\times 10^3$)	Distribution of more than 30	Distribution of less than 30
Guangzhou			
Hangzhou			
Kunming			

- 从图表中可以很明显看到大多数用户这一个月内大多数集中在1, 2条微博的区间，而在200条以上的就接近于0了。
- **这是符合二八定律的分布特点的，在社交网络上也就是著名的幂律分布特点，高频词访问、发帖的用户随着计量次数的升高而急剧减少。**

2. 另外，对每个城市发表次数最多的用户ID挑出，因为观察到发表最大的次数高达四位数，且收集数据都是整点附近收集，平均一天（整点）的发帖数甚至上百，让我对这种用户的行为产生了好奇，初步**猜测可能是新闻的官方号或者是僵尸流量粉**，所以将其用户ID挑出，在后续进行进一步账号分析，如果必要可能再尝试挑选发表的前几位进行分析。

```
%% Guangzhou
>> {gz_post_max_id; gz_post_max}

ans =

  2×1 cell 数组

    {'523187'}
    {[  4423]}

%% Hangzhou
>> {hz_post_max_id; hz_post_max}

ans =

  2×1 cell 数组

    {'428875'}
    {[  9685]}
%% Kunming
>> {km_post_max_id; km_post_max}
```

```
ans =

  2×1 cell 数组

    {'024326'}
    {[  1904]}
```

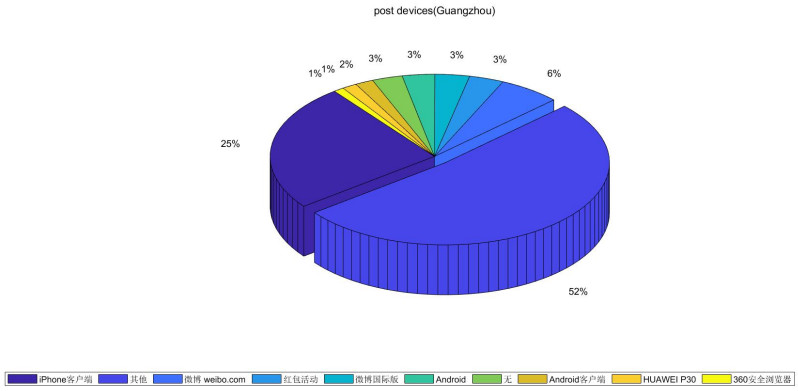
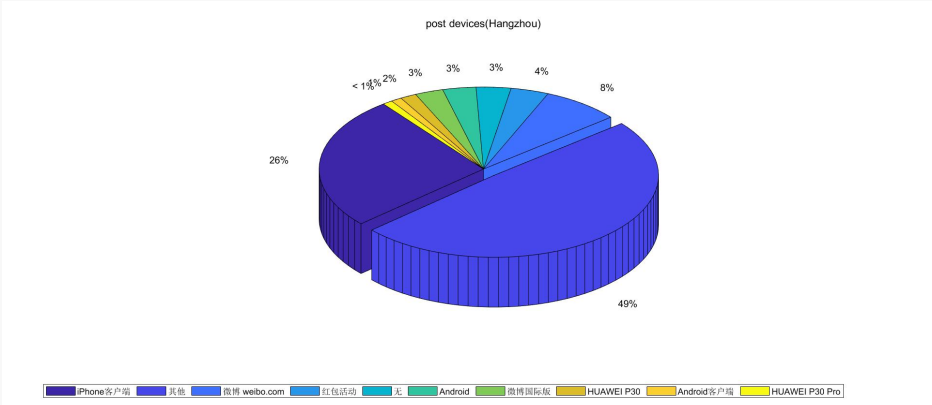
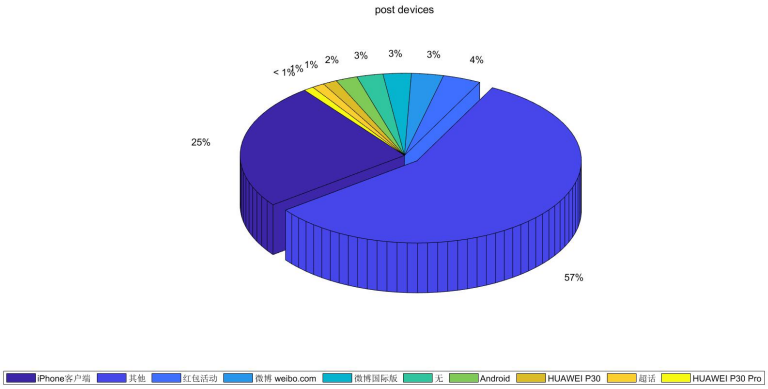
用户发表微博的平台分布

- 针对平台的第二项统计量，进行对于发布来源的统计分析。
- 首先统计0-200出现的次数，再进行饼图的绘制。这里需要处理的问题是，200个平台来绘制份额的饼图对于数据的查看便捷性和对比度都不会太好，因为200个切片对于数据标注来说太密集了，而且很大一部分都是很小的值，观测十分不方便，所以我将所有来源分为十个，其中九个为来源最多的平台，外加一个"其他"变量。而我采取将代码耦合性尽量降低的方式，使得三个地区的实现代码几乎相同。

> 广州地区的一个例子

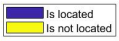
```
%% Import the data and select the device column

% Select the rows whose id is NOT NULL
rows_gz = Guangzhou.author ~= "NULL";
rows_hz = Hangzhou.author ~= "NULL";
rows_km = Kunming.author ~= "NULL";

% Put the device(source) column into a string vec
user_source_guangzhou = Guangzhou{rows_gz, "source"};
user_source_hangzhou= Hangzhou{rows_hz, "source"};
user_source_kunming = Kunming{rows_km, "source"};

device_times = zeros(201, 1);
for i = 1: length(user_source_guangzhou)
 tmp = user_source_guangzhou(i);
device_times(tmp + 1) = device_times(tmp + 1) + 1;
end

[B_gz, I_gz] = maxk(device_times, 10);

other_device_gz = sum(device_times) - sum(B_gz);
B_gz(2) = B_gz(2) + other_device_gz;

% Impoert source.csv by the function in the path.
% The imported array name is device_name
device_name = import_device_name();

% give the diffenrent part a legend
device_name_array = table2array(device_name);
legend_name_tmp = strings(1, 10);
for i = 1: 10
 name_tmp = I_gz(i);
 legend_name_tmp(i) = device_name_array(name_tmp);
end

% draw the graph to show devices parts
figure
```

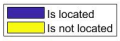
```
explode = [0 1 0 0 0 0 0 0 0 0];
pie3(B_gz, explode);
p = pie3(B_gz, explode);
legend(legend_name_tmp,
'Location','southoutside','Orientation','horizontal')
title("post devices")
% Set the property of the pie graph
for i = 1: 10
 t = p(i);
 t.FontSize = 8;
end
```

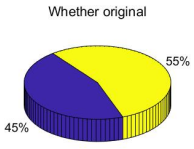
City	Source Pie Graph
Guangzhou	
Hangzhou	
Kunming	

是否原创和是否有位置的比例分布

> 实现方法：分别统计两个数组中0和1的个数，加和之后直接用饼图绘制

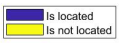


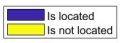


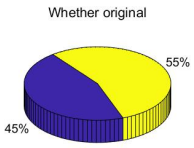


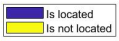
- 画法很简单，是否原创基本是持平的，非原创的更多一些。值得注意的是广州的标注位置的微博很少，考虑到数据抽取的时间大概在春节放假期间，而且正是疫情爆发的关键时期，广州基本只有本地人，而这一部分群体相对不会倾向于标注自己的位置。

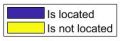
> **对比于杭州/ 昆明**

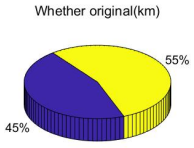










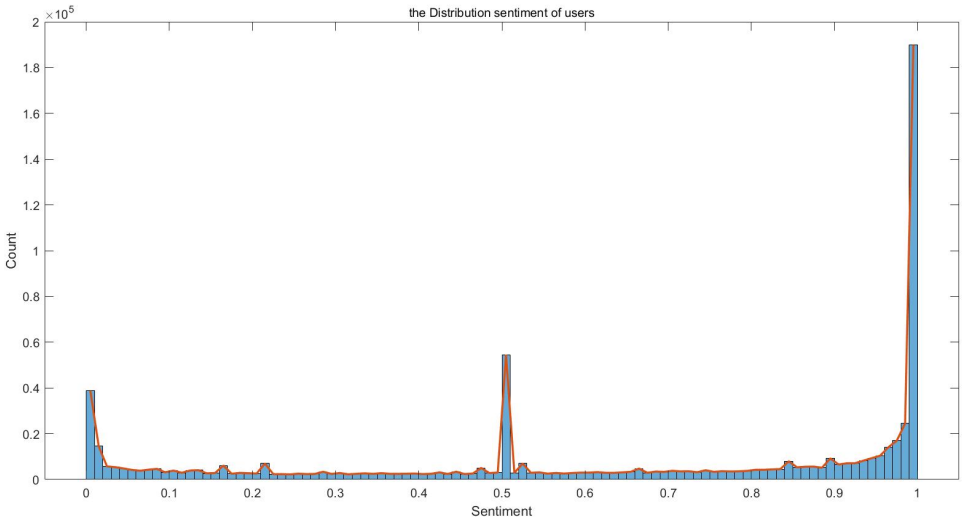


- 其中广州是外来人口的大市，务工人员回乡的较多，杭州次之，昆明最次。可以看出是否标注地点于此可能且很可能有相关的关系。

情感指数的分布

> 实现方法：将情感指数用histogram直接绘制，再将每一个柱体的中心点连接成为曲线图

```
%% sentiment

% Put the device(source) column into a string vec
user_source_guangzhou = Guangzhou{rows_gz, "sentiment"};
figure;
h = histogram(user_source_guangzhou);
hold on;
h_x = h.BinEdges(1 : end - 1) + 0.5 * h.BinWidth;
h_y = h.Values;
plot(h_x, h_y, 'LineWidth', 1.7);
xlabel('Sentiment');
ylabel('Count');
title('the Distribution sentiment of users', 'Fontsize', 10);

hold off;
```



总结有以下规律：

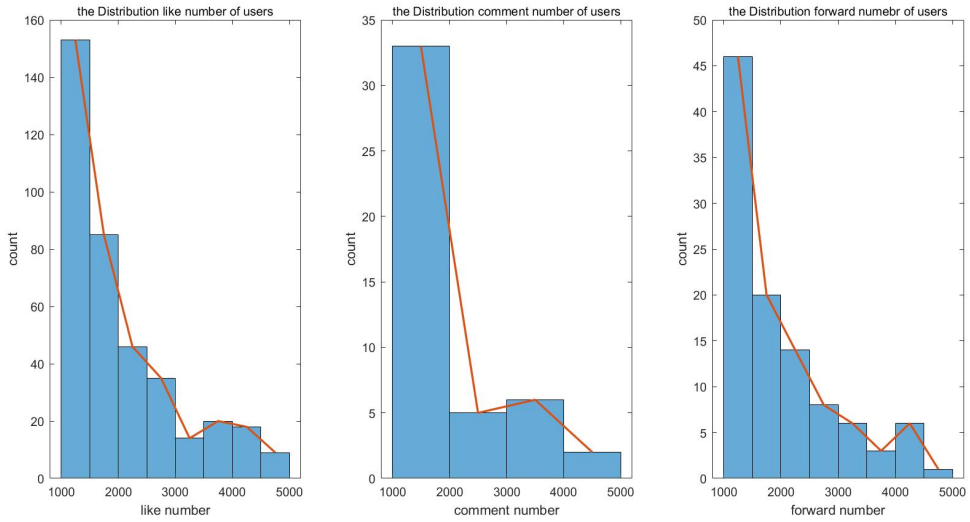
1. 接近零和1以及中位数0.5的分布最多。
2. 绝大多数趋近于1
3. 其他分布较为均匀

- 个人推测以上原因大概率是情感分析的ML模型不够完善（不然情感曲线不应该分布太过于陡峭）。在后面我将会使用sentiment这个指标来进行时间序列的分析，由分布图考虑应该去掉0，1， 0.5三个值的变量。

点赞数、评论数和转发数三大要素

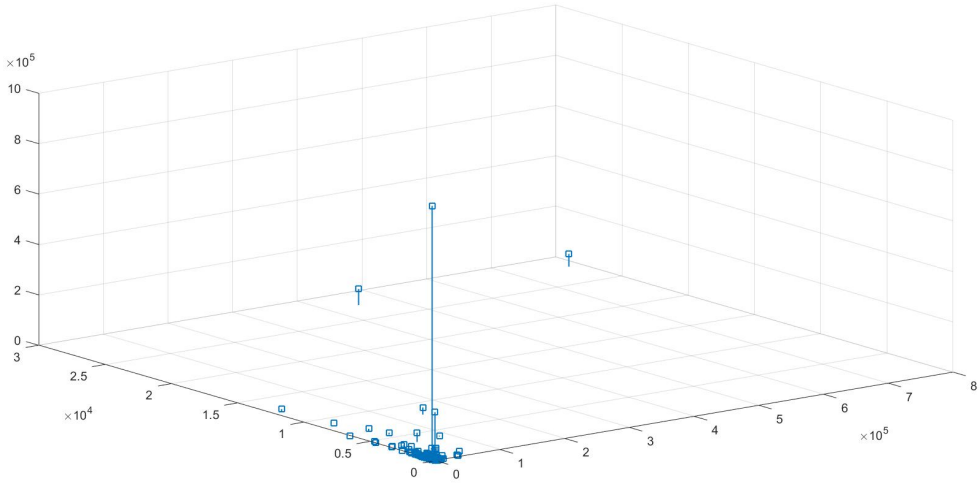
- 首先是三个量的分布图

> 实现方法：这里首先尝试了将所有的变量和对应的值画成分布图，但是明显太过于集中，收集到的数据大多数都集中在0，而且还有一部分很大的数，考虑到分布图的直观性，采用`rows_gz_l = Guangzhou.likes_num > 1000 & Guangzhou.likes_num < 5000;`的代码来选取中间区域的数值对。



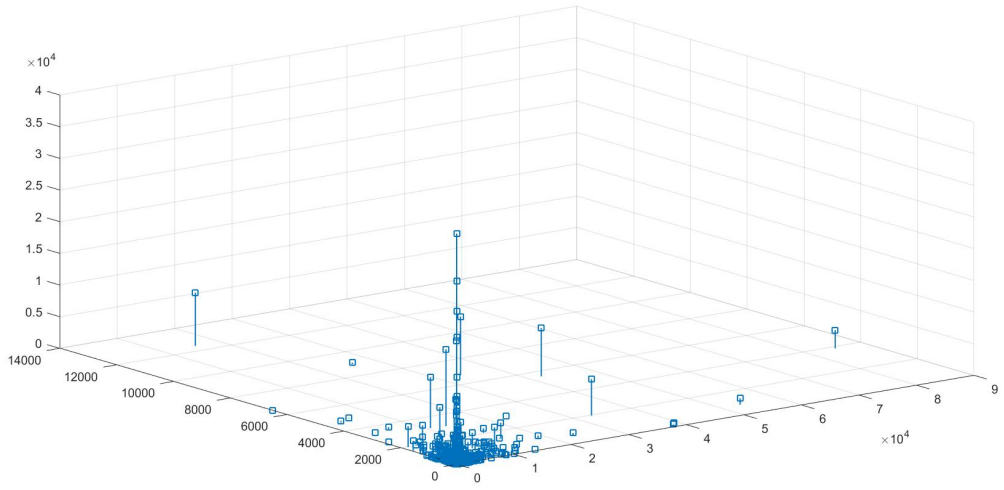
- 另外，对三个量绘制三维图（火柴图），从而更直观的观察转发量和点赞数、评论数是否呈现正相关关系。

> 实现方法：对三个变量进行统计，直接用`stem3();`命令绘制。



- 由于包含所有数据的图像存在一部分和其他数据差别较大的，从图中可以看到以一个转发数量过大的，有两个点赞和评论数量很大的，与大部分差距很大，猜测可能是点赞评论/转发抽奖互动或者是明星又出现了绯闻...于是对这几个变量进行处理，暂时删除几个偏差较大的值，代码如下：

```
[like_most, like_most_pos] = maxk(user_guangzhou_like, 2);
[forw_most, for_most_pos] = maxk(user_guangzhou_forward, 1);
user_guangzhou_comment(for_most_pos) = [];
user_guangzhou_like(for_most_pos) = [];
user_guangzhou_forward(for_most_pos) = [];
for i = 1: 2
 user_guangzhou_comment(i) = [];
 user_guangzhou_like(i) = [];
 user_guangzhou_forward(i) = [];
end

stem3([user_guangzhou_like, user_guangzhou_comment,
user_guangzhou_forward]);
```



- 从上面的三维图中可以看出，在大区域内是明显存在相关关系的，但是在小区域内，却没有明显的相关关系，这样比较容易理解，比如在点赞数和评论数很小的微博发表，大部分是一些小博主或者普通用户，点赞和评论也有很大的随机性，转发一般都是朋友之间或者多层转发的情况，所以与内容质量无关，也就不会和点赞数、评论数呈现同一规律；而点赞数和评论数较大的情况一般是官方账号的发布或者明星的新闻，也就会呈现点赞、评论以及转发数很高的情况。
- 后续会根据点赞数、评论数以及转发数的情况进行一系列相关性分析，对于上述的分布图可以用作参考进行数据的清洗和筛选。

时间序列分析

情感的时间序列分析情况

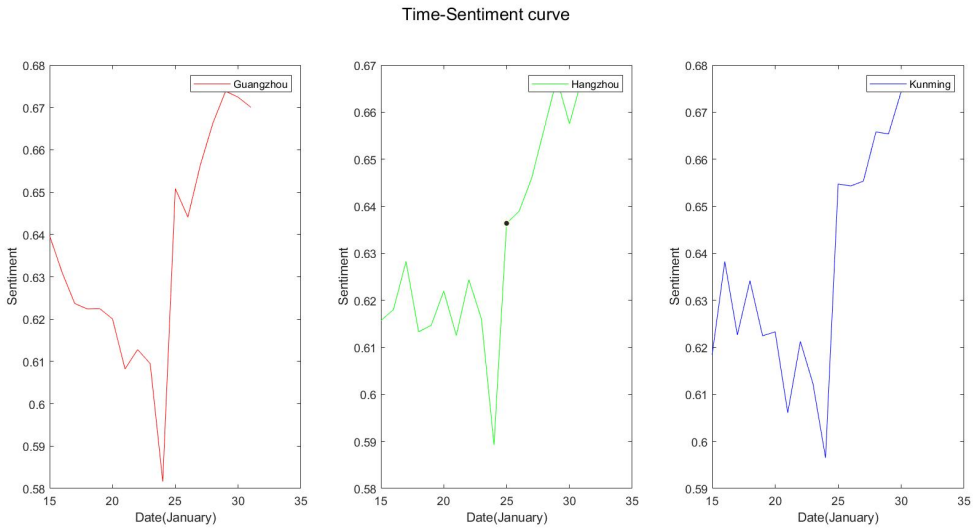
根据1月的数据，分别统计了三个城市的微博平均情感指数

> 实现方法：通过对月份为1的数据进行抽取，对于情感指数加和求平均值，再画出”时间-情感指数“曲线。

```
%% Guangzhou
gz_time_senti_count = zeros(17, 1);
gz_time_senti = zeros(17, 1);
for i = 1: length(user_sentiment_guangzhou)
tmp = user_time_guangzhou(i) - 14;
gz_time_senti(tmp) = gz_time_senti(tmp) + user_sentiment_guangzhou(i);
gz_time_senti_count(tmp) = gz_time_senti_count(tmp) + 1;
```

```
end
ave_gz_time_senti = zeros(17, 1);
for j = 1: 17
 ave_gz_time_senti(j) = gz_time_senti(j) / gz_time_senti_count(j);
end

January_days = 15: 31;
subplot(1,3,1);
plot(January_days, ave_gz_time_senti,'-r');
xlabel("Date(January)"); ylabel("Sentiment");
legend("Guangzhou");
```

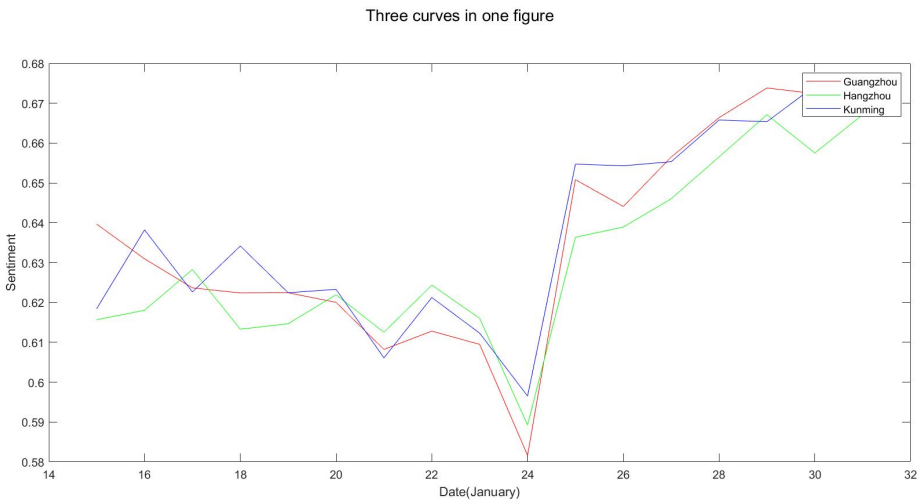
- 用`subplot()`画出的图像如下：



- 我们发现基本的趋势很相近，一方面体现了其实现在信息化越来越普遍，各地区的舆论趋势基本相同。另一方面，这时候正值疫情伊始，全国信息网络信息基本相同。

我们将三条曲线绘制到一张图中，发现有高度的重合性，如下所示：

```
figure;
plot(January_days, ave_gz_time_senti,'-r');
xlabel("Date(January)"); ylabel("Sentiment");
hold on;
plot(January_days, ave_hz_time_senti,'-g');
hold on;
plot(January_days, ave_km_time_senti,'-b');
hold off;
legend("Guangzhou", "Hangzhou", "Kunming");
suptitle("Three curves in one figure");
```



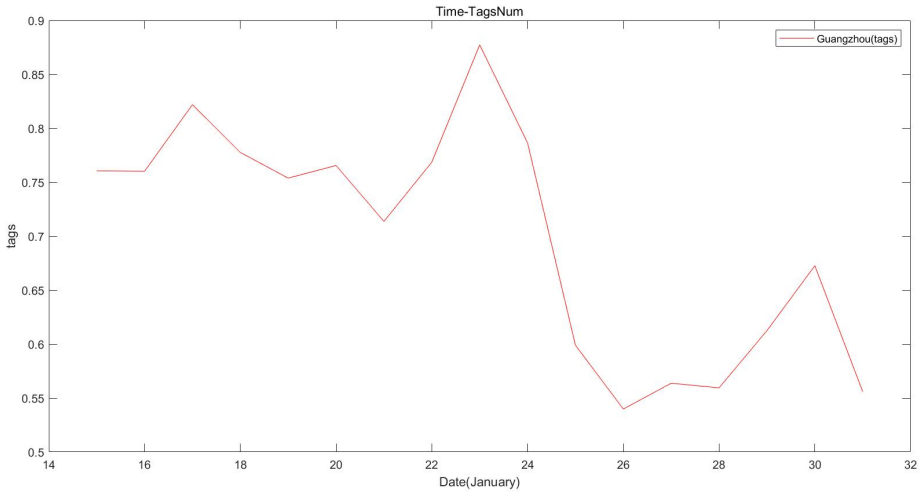
- 21号明确新冠肺炎病毒有人传人的存在，这时候出现了一个较为陡峭的下坡。
- 24号武汉封禁所有交通通道，各地进入紧急状态。值得注意的是，这一天是除夕，本该是情绪较为高的时间点，实际情况是情绪指数跌至最低，我推测是春晚涉及到的节目的某些词汇被算作情绪较低的词汇被大量使用的原因。
- 25号零点（大年初一）的数据情绪值突然反弹。

以标签数量为标志的突发话题

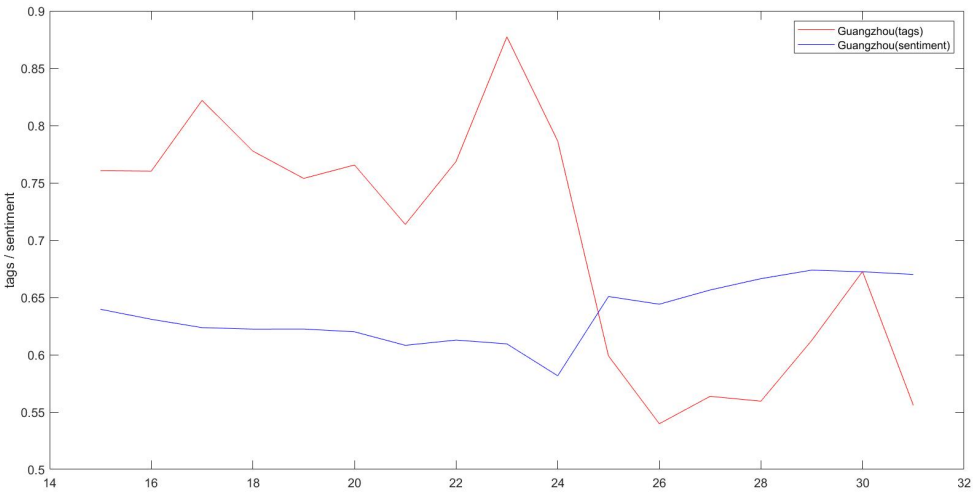
标签往往可以看作是微博上热点话题的一个缩影，一个热点话题的出现往往表现为标签数量的急剧增加。

实现方法：根据上述思路，统计时间和表情数量（平均值）之间的关系曲线。

```
%% Guangzhou_tags
gz_time_tags_count = zeros(17, 1);
gz_time_tags = zeros(17, 1);
for i = 1: length(user_tags_guangzhou)
tmp = user_time_guangzhou(i) - 14;
gz_time_tags(tmp) = gz_time_tags(tmp) + user_tags_guangzhou(i);
gz_time_tags_count(tmp) = gz_time_tags_count(tmp) + 1;
end
ave_gz_time_tags = zeros(17, 1);
for j = 1: 17
 ave_gz_time_tags(j) = gz_time_tags(j) / gz_time_tags_count(j);
end

January_days = 15: 31;
plot(January_days, ave_gz_time_tags,'-r');
xlabel("Date(January)"); ylabel("tags");
legend("Guangzhou(tags)");
title("Time-TagsNum");
hold on;
```



- 发现在23，24号数量达到了最大值，而这两天恰好也是新冠肺炎刚开始爆发的时间节点，在25（正月初一）大幅度下降。我们将其与情感指数绘制到一张图表中，结果如下：



- 可以明显看到22，24左右的两者呈现负相关，印证了和新冠肺炎的微博消息关系应该很大。

多组数据相关性分析

给定统计的数据中包含一条微博多种多样的信息，而这些数据变量之间都或多或少存在着一定的关系属性，或是相互促进，或是相互排斥，或者关系基本趋近于无，通过对这些变量之间的相关性分析，可以得到一些变量受影响的因素和趋势。

微博影响力的相关因素

微博影响力可以由以下指标刻画（理论来源于《Social Media Mining An Introduction》）

$$C_d(v_i) = d_i^{in} \qquad C_d(v_i) = d_i^{out} \qquad C_d(v_i) = d_i^{in} + d_i^{out}$$

也就是说，一个用户的度中心性包括了入度和出度，分别刻画声望（prominence）以及合群性（gregarousnesee）。但是由于微博中

数据中并没有包含也无法挖掘出度的具体信息，这里就只分析入度代表的度中心性。而根据经验，微博的点赞所代表的粘性用户可能性最低，评论次之，转发代表的黏性粉丝最为可靠，所以将三个变量分别分配0.1，0.2，0.7的加权值。计算出中心性的一个指标，并利用最大度数为基准进行归一化：

$$C_d^{max}(v_i) = \frac{d_i}{max_j d_j}$$

```
%% Calculate the centrality of users
rows_gz = Guangzhou.author ~= "NULL";
% Put the three column into a string vec
user_like_guangzhou = Guangzhou{rows_gz, "likes_num"};
user_comment_guangzhou= Guangzhou{rows_gz, "comment_num"};
user_forward_guangzhou = Guangzhou{rows_gz, "forward_num"};

centrality_gz = 0.1 * user_like_guangzhou + 0.2 * user_comment_guangzhou +
...
 0.7 * user_forward_guangzhou;

centrality_gz = centrality_gz / max(abs(centrality_gz));

% 在处理过程中发现最大数据明显偏差过大，暂时删除，使得分析更加有普适性。
[lar_cen, lar_cen_pos] = maxk(centrality_gz, 1);
centrality_gz(lar_cen_pos) = [];
% 再进行一次归一化操作
centrality_gz = centrality_gz / max(abs(centrality_gz));
```

通过`corrcoef();`函数的使用来计算个变量之间的相关系数

1. 与情绪值的相关系数：

```
senti_co =

    1.0000    0.0002
    0.0002    1.0000
```

2. 与原创字数之间的关系

```
orig_word_num_co =

    1.0000    0.0013
    0.0013    1.0000
```

3. 与提到的人数的相关系数

```
mention_num_co =

    1.0000    0.0016
    0.0016    1.0000
```

4. 与话题数之间的相关关系

```
topic_num_co =

    1.0000    0.0020
    0.0020    1.0000
```

5. 与转发博文之间的相关系数

```
forward_num_co =

    1.0000    0.0005
    0.0005    1.0000
```

- 根据上述相关系数的计算，可以看到大部分数据之间都是没有太大的相关关系的，相比之下，影响力和话题数量之间的相关性比较明显。

其他的相关性简述

1. 点赞数、评论数 / 点赞数、转发数

```
% 点赞数和评论数
like_comment = corrcoef(user_like_guangzhou, user_comment_guangzhou);

like_comment =

    1.0000    0.8552
    0.8552    1.0000

% 点赞数和转发数
like_forward =

    1.0000    0.0860
    0.0860    1.0000
```

- 这一组对比非常直观的说明了点赞和评论之家的相关性非常强，相比之下转发数和评论数之间就不存在这么强的相关关系，而这也是点赞和评论在度中心性计算相对比重较小的原因。

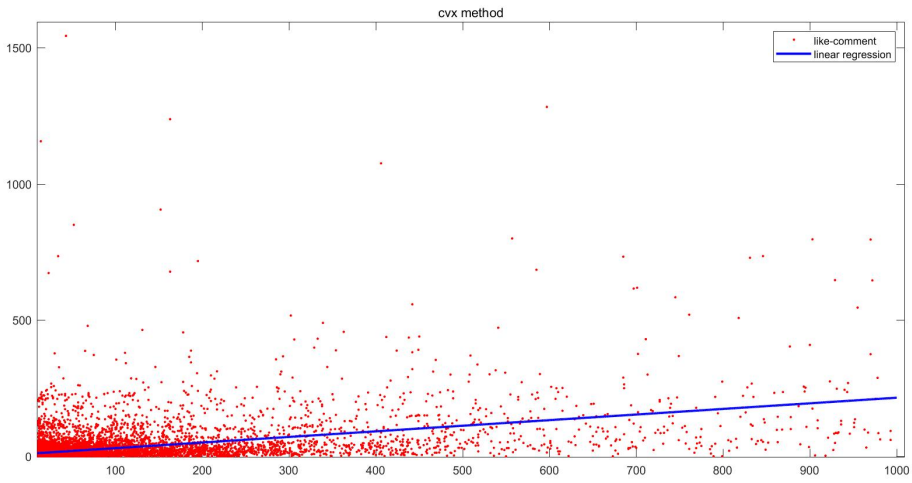
CVX的优化工具

- 尝试应用Boyd编写的cvx包工具来进行对于优化问题的解决。
- 首先下载安装了cvx工具（一个利用多种方法快速解决优化问题的工具包）
- 利用`cvx_begin`
  `variables x(2)`
  `minimize  norm(A*x - b)  % minimize the sum of distance`
  `cvx_end`来优化一个最小二乘问题，从而得到了用户点赞数和评论数的最佳拟合结果。
- 得到的优化结果：

```
Calling SDPT3 4.0: 5237 variables, 3 equality constraints
   For improved efficiency, SDPT3 is solving the dual problem.
------------------------------------------------------------

 num. of constraints =  3
 dim. of socp   var  = 5237,   num. of socp blk  =  1
*****************************************************************
   SDPT3: Infeasible path-following algorithms
*****************************************************************
 version  predcorr  gam  expon  scale_data
    NT      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
```

```
 0|0.000|0.000|3.6e+01|2.6e+00|1.3e+06|  0.000000e+00   0.000000e+00| 0:0:00| chol
 1  1
 1|0.971|1.000|1.0e+00|1.4e-05|5.4e+04|-4.055726e+03 -1.894285e+04| 0:0:00| chol
 1  1
 2|1.000|1.000|9.8e-08|1.4e-06|7.0e+03|-3.454003e+03 -1.041139e+04| 0:0:00| chol
 1  1
 3|1.000|0.942|1.7e-08|2.3e-07|2.6e+02|-5.861186e+03 -6.125128e+03| 0:0:00| chol
 1  1
 4|0.987|0.988|6.2e-09|2.0e-08|3.3e+00|-6.014205e+03 -6.017504e+03| 0:0:00| chol
 1  1
 5|0.989|0.989|6.9e-11|2.8e-09|3.6e-02|-6.016121e+03 -6.016157e+03| 0:0:00| chol
 1  1
 6|0.989|0.989|1.5e-12|4.5e-11|4.0e-04|-6.016142e+03 -6.016142e+03| 0:0:00| chol
 1  1
 7|0.989|0.989|4.6e-13|1.5e-12|4.4e-06|-6.016142e+03 -6.016142e+03| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
------------------------------------------------------------------------
 number of iterations   =  7
 primal objective value = -6.01614214e+03
 dual   objective value = -6.01614215e+03
 gap := trace(XZ)       = 4.40e-06
 relative gap           = 3.66e-10
 actual relative gap    = 3.65e-10
 rel. primal infeas (scaled problem)   = 4.56e-13
 rel. dual     "        "       "      = 1.49e-12
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 1.4e+00, 6.0e+03, 8.5e+03
 norm(A), norm(b), norm(C) = 1.7e+04, 2.0e+00, 7.3e+03
 Total CPU time (secs)  = 0.38
 CPU time per iteration = 0.05
 termination code       =  0
 DIMACS: 4.6e-13  0.0e+00  7.1e-12  0.0e+00  3.6e-10  3.7e-10
------------------------------------------------------------------------

-------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +6016.14
```

- 得到的图像绘制如下：



K-means 用户分类

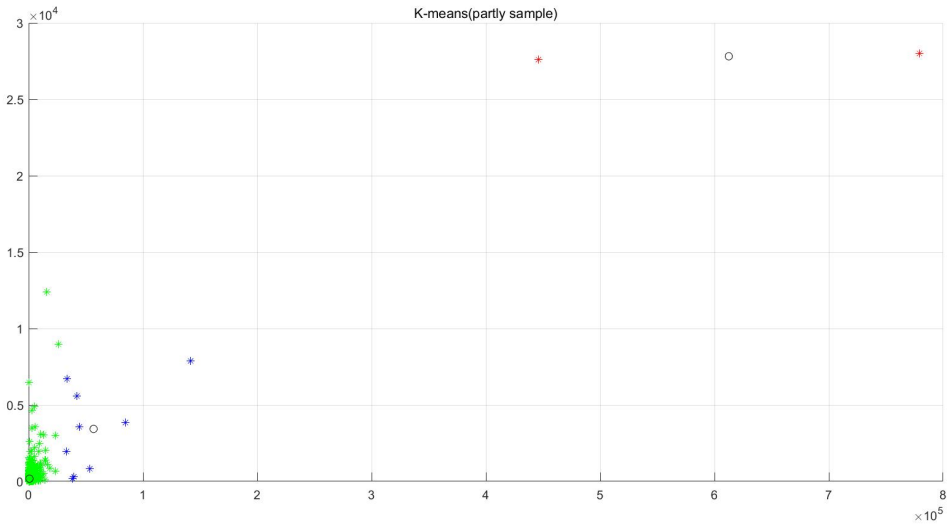
- 我们都知道微博其实是一个各种各样人员组成的平台，这其中有普通的用户来记录日常生活或者参与话题讨论，有明星来进行娱乐八卦等热点，也有官方部门和公司借微博这个平台来进行官方信息的发布和宣传，当然也有一些僵尸粉来刷流量发送广告的，这些用户的行为特征各不相同，通过数据体现出来的差别也可以观察找出来的；同理，发送的每一条微博其实都有一定的分类特征。当然，需要各种参数的调试和辅助，工作比较困难。但是通过给定的二十组数据指标，特征提取进行一定程度的分类还是可行的。
- 这里利用K-means方法来进行基本的用户特征类型分类。（根据上面的分布情况，其实观察到其实用谱聚类的方法更为精确可靠一些，但是由于数据量过大，迭代过程效率很低而且数据的预处理较为困难，暂时用K-means实现相应的功能）。

> 实现方法：首先将数据进行一定程度的清洗，将数据取出后，由于原点周围的数据占了99.8%左右，所以对这些数据进行抽取，使得分布较为均衡，同时也加速算法的运行（否则程序的运行计算次数太多，CPU速度达不到），然后将数据大体分为三块区域，分别选出一个中心点，进行K-means的详细算法。算法的具体过程见`k_means_static.m`文件，算法核心如下所示：

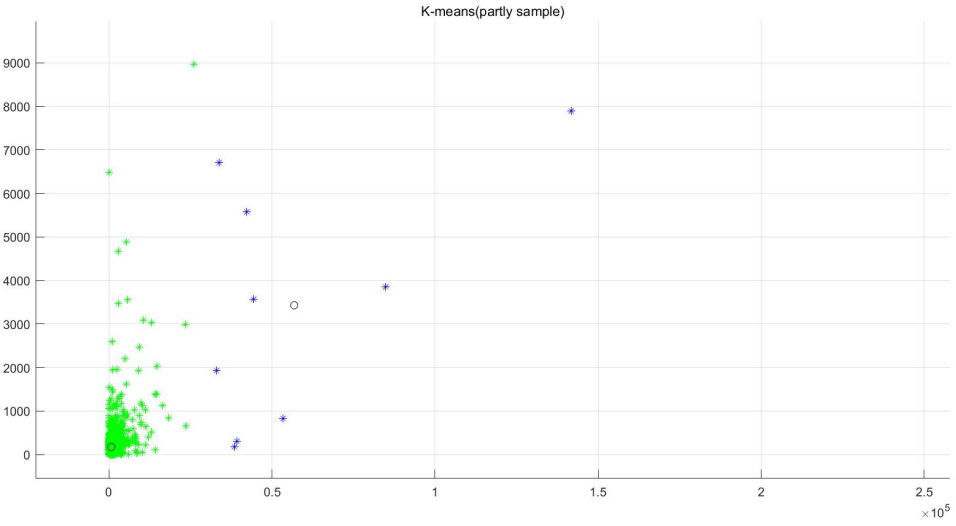
```
data = [data1; data2; data3];
N = 3;
[m, n]=size(data);
pattern = zeros(m,n+1);
center = zeros(N,n);
pattern(:, 1:n) = data(:,:);
for x = 1: N
 center(x, :) = data(randi(2244, 1),:);
end
while 1
 distence = zeros(1, N);
 num = zeros(1, N);
 new_center = zeros(N, n);

 for x = 1: m
     for y = 1: N
         distence(y) = norm(data(x,:) - center(y,:));
     end
     [~, temp] = min(distence);
     pattern(x, n + 1) = temp;
 end
 k = 0;
 for y = 1: N
     for x = 1: m
         if pattern(x, n + 1) == y
             new_center(y,:) = new_center(y, :) + pattern(x, 1: n);
             num(y) = num(y)+1;
         end
     end
     new_center(y,:) = new_center(y,:)/num(y);
     if norm(new_center(y,:) - center(y,:)) < 0.1
         k = k + 1;
     end
 end
 if k == N
     break;
 else
```

```
        center = new_center;
    end
end
```

- 实现的结果如图所示：



- 放大一定程度的图像如下：

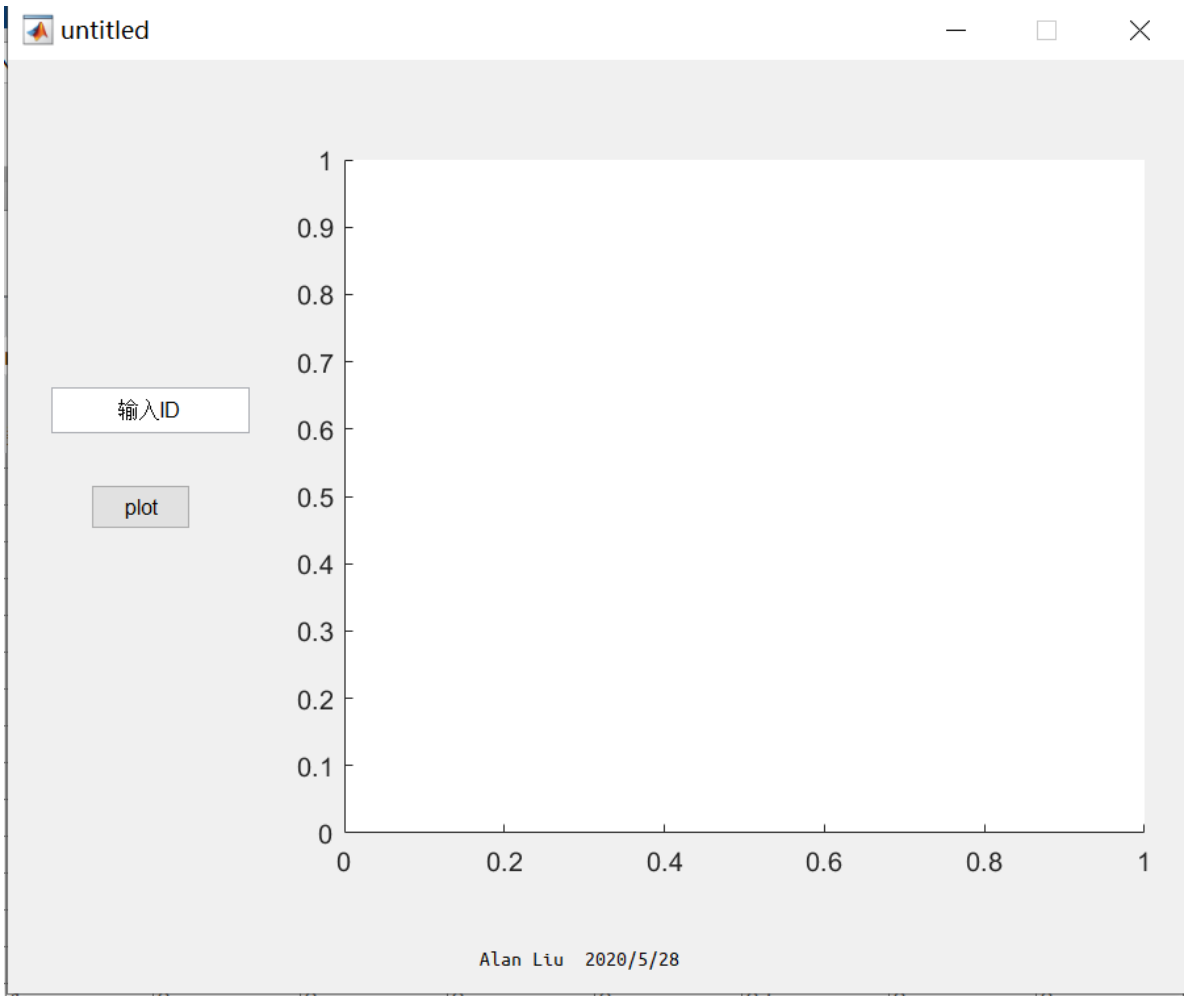


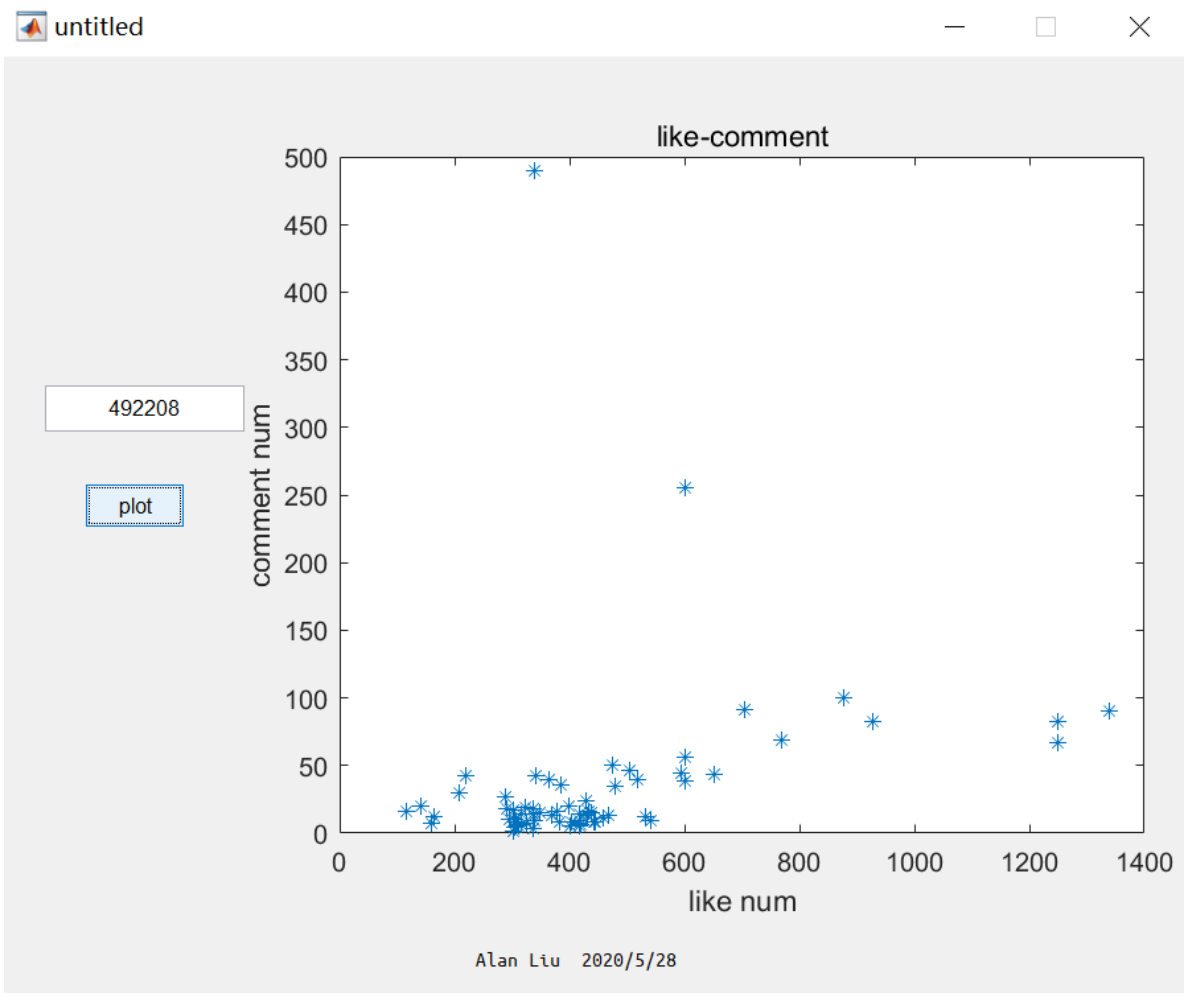
GUI 图形化用户界面

- 利用简单的GUI图形化界面操作，完成了数据的自动读取（以广州的用户数据为例），输入一个用户的ID，能够自动绘制此用户所有微博的点赞数和评论数的二维图像。

> 实现方式：利用MATLAB GUIDE工具来进行制作，其中集成了所需要的功能的代码

- 具体实现结果如下图所示：

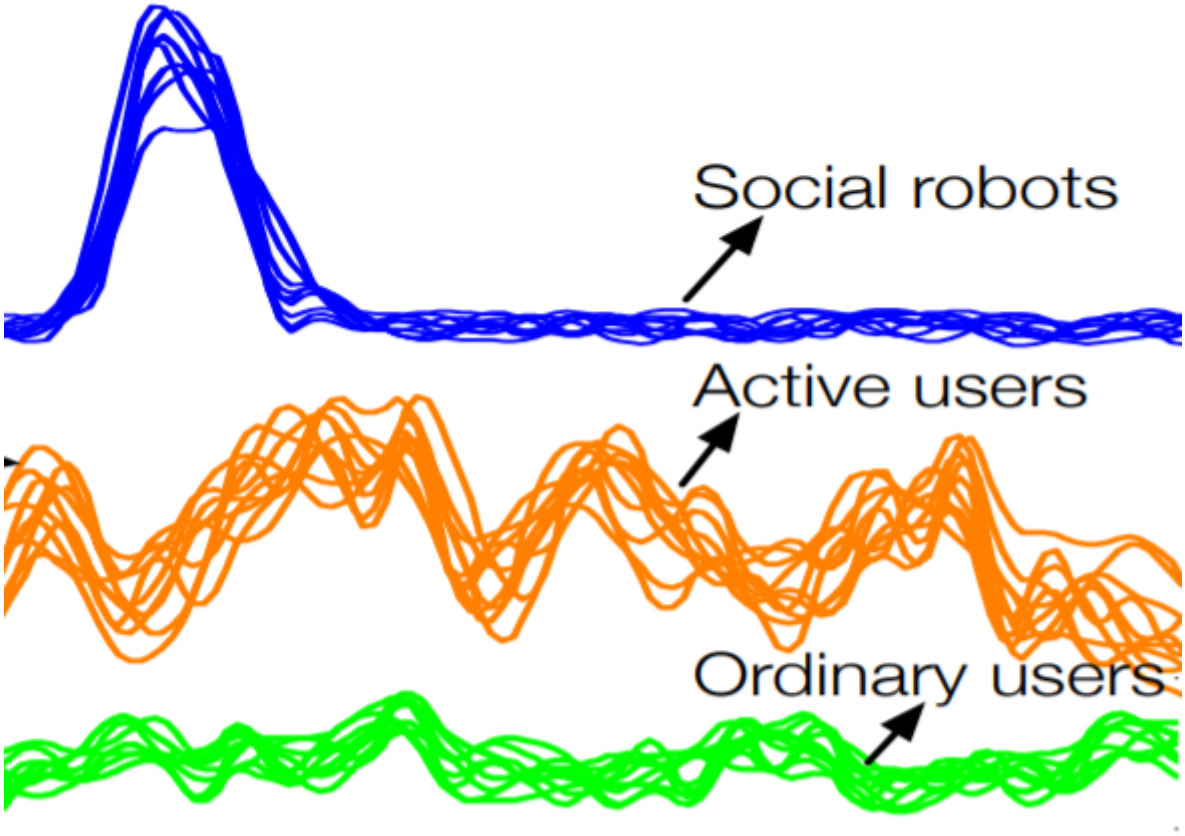

untitled
输入ID
plot
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
Alan Liu 2020/5/28


untitled
492208
plot
like-comment
comment num
like num
500
450
400
350
300
250
200
150
100
50
0
0
200
400
600
800
1000
1200
1400
Alan Liu 2020/5/28

- 此处选择了一个发表微博数量较多的用户492208为例，画出点赞数和评论数的图像如上图所示。

虚假用户的简单判断（流量水军或者恶意用户）

- 虚假用户（也称为僵尸粉、流量机器人、恶意用户）是一类存在于各大论坛、社交软件、邮箱中的，一般是由程序实现的虚假用户，以达到赚取流量、发送恶意信息、散布广告的目的。在这其中，现存有很多方法来捕捉这种虚假用户，比如常见的有IP检测、用户行为单独分析、利用机器学习的方法自动检测等手段。
- 在给定的微博数据集中，并没有具体的**评论内容文本**，也没有**评论重复性**，所以很难判断僵尸粉的存在，但是根据所得到的数据：在前面也有所体现，有一些用户的一个月内发帖数相当之高，有的帖子也存在评论很多却没有点赞的情况，反之亦然，这些情况下都有可能是虚假机器人的操作行为。
- 下图来自于Haishuai Wang, Jia Wu的研究“Learning Shapelet Patterns from Network-Based Time Series“



> 实现方法：大体参考上图的思路，对几个抽取出来的特定用户进行时间上的绘图，如符合social robots的特征，大体可以判断为虚假机器人
>
> 注意事项：需要以其他用户作为参考才有一定价值

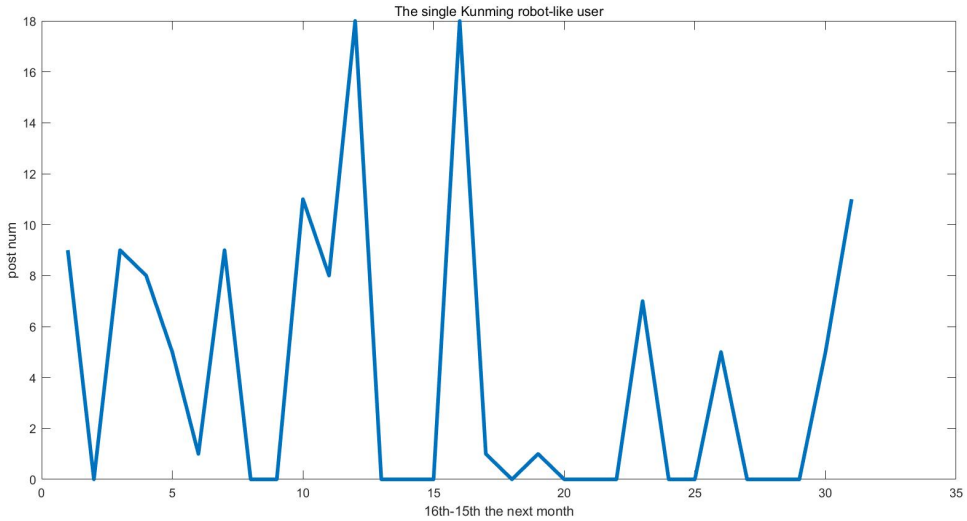
1. 评论数量很多但是没有点赞的

- 首先我们暂时不不考虑在一个时间线上进行分析，而是考虑点赞数很少（考虑0或1）但是评论数很多（大于500）的数据，然后重点分析这些用户的行为.

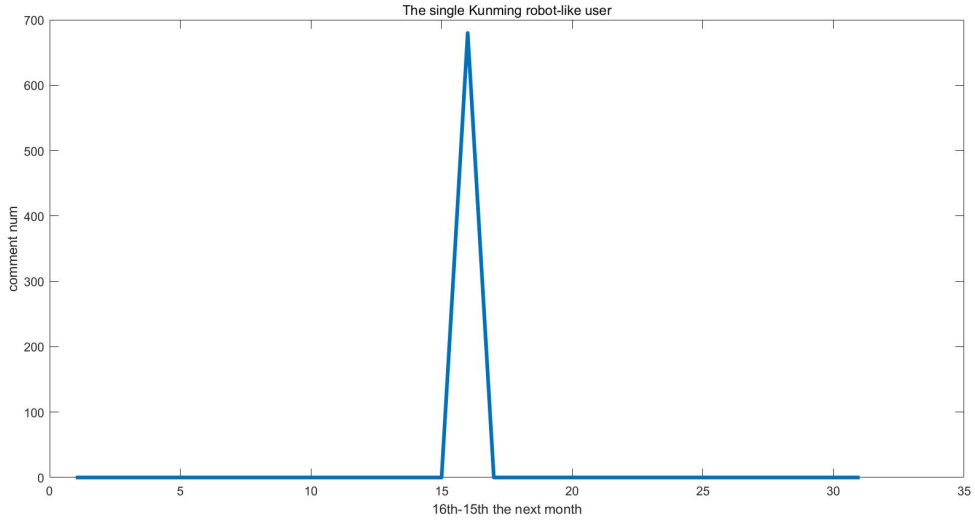
```
rows_gz = Guangzhou.likes_num <= 1 & Guangzhou.comment_num >= 5000;
rows_hz = Hangzhou.likes_num <= 1 & Hangzhou.comment_num >= 500;
rows_km = Kunming.likes_num <= 1 & Kunming.comment_num >= 500;
```

> 最终选出的数据中，昆明有一个这样的案例，杭州有四个，广州不存在。所以单独抽取出这些用户的所有数据进行时间线的分析

- 首先是昆明的一个案例
  -

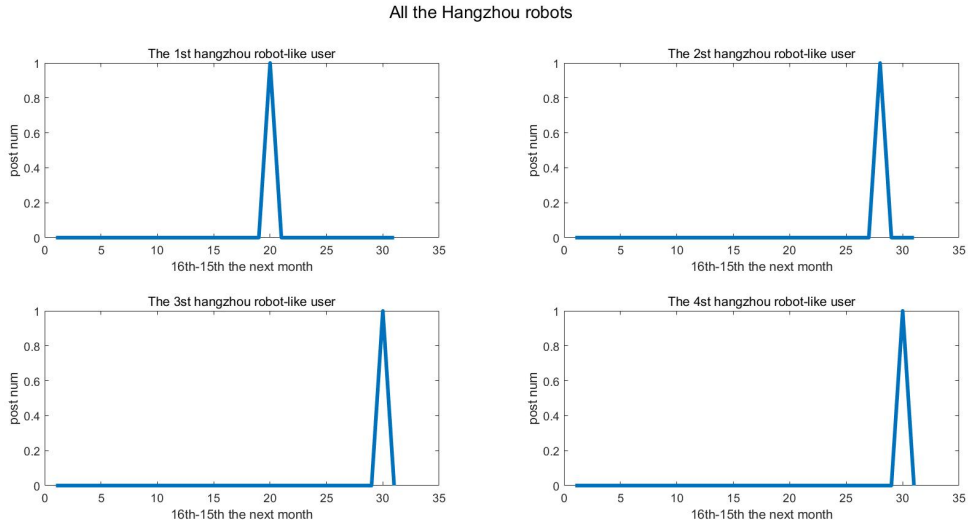


- 单纯对于每天的发布数量来看，波动很大而且似乎有些均匀，像是追星的用户或是官方发布的用户，依据此无法判断，再来看评论数的分布。
  -



- 结果显而易见，这个用户大概率就是使用了机器人来刷热度。

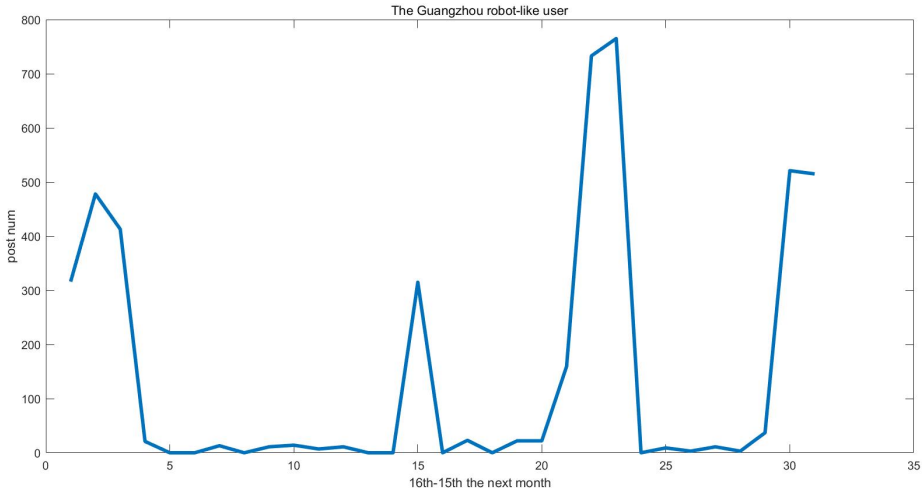
- 我们再将杭州的用户使用第一步同样的方法：



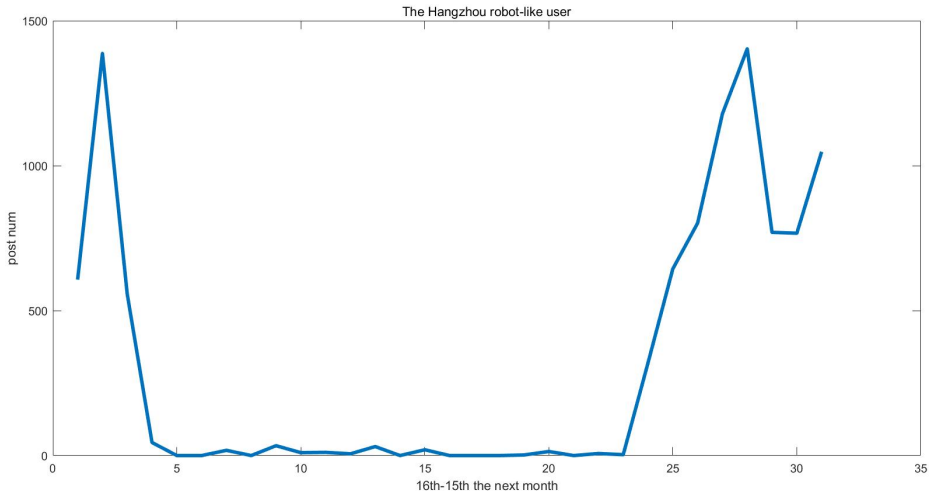
- 结果是显而易见的，如果说昆明的那几名用户是其他类型用户，杭州的这四个用户显然是机器型的用户，在长达一个月内只在某一天发帖一次，**重点在于没有点赞而评论超过500条**，这足以很大程度证明是虚假用户。

2. 上面的分析建立在有了点赞数很少但评论数很多的基础之上，下面将三个城市发帖数最多的几位用户单独拿出来进行分析。在基本数据量的统计中，已经得到了用户的基本信息，也定位了三个城市中发帖最多的几位用户。

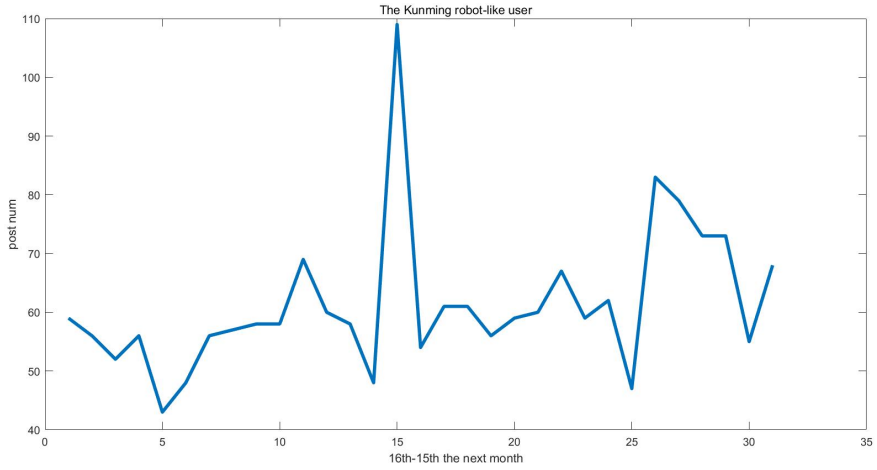
    1. 广州（ID: 523187）



    2. 杭州（ID: 428875）



3. 昆明（ID: 024326）



- 在这三者之中，杭州的用户为虚假用户的可能性最大（分布不均匀），广州次之，昆明居尾。

缺点与反思

1. 代码鲁棒性比较低，有一部分程序的运行需要手动来执行；同时由于代码几乎是叙述性的运行，导致耦合性很高，很难进行其他方面的的更改。
2. 数据挖掘的深度不够，一方面是计算机计算能力不够（曾尝试过用高迭代的方法但速度很慢），另一方面对于数据的属性没有深入探究。
3. 可视化做的确实不够漂亮。

引用

所有相关引用在原文中已经标注

致谢

由衷感谢老师助教的耐心讲解和助教们的作业答疑，这是一次十分愉快的学习之旅。