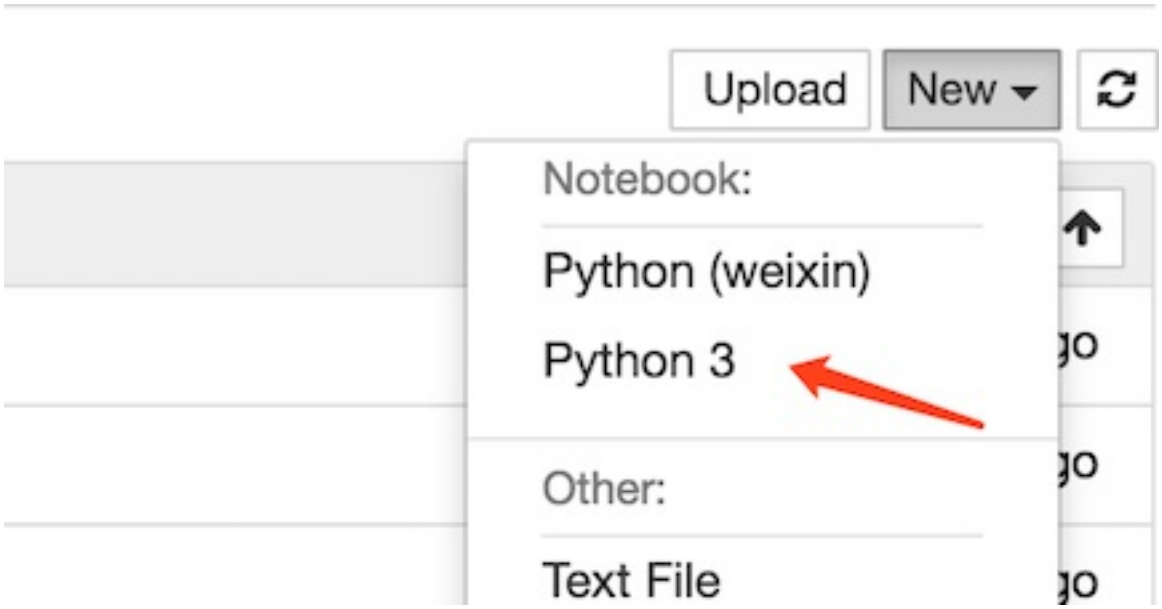
基于 Matplotlib 实现数据可视化

上节我们介绍了 Pandas 的基本操作，这节我们使用 Pandas 结合 Matplotlib 对数据进行可视化展示。首先我们把数据加载到 Pandas，现在假设你已经拥有了数据，如果没有数据可以下载我给你准备的[JSON文件](https://github.com/pythonzhichan/weixincrawler/blob/mast

加载数据

启动 juypter notebook 之后基于Python3 新建一个 notebook，之所以不选择叫 weixin 的 Python 解释器是因为默认的 Python3 已经包含了所有的数据分析相关包，无需另外下载。



在终端查看我的系统里有哪些虚拟环境

```
conda info -e
# conda environments:
#
crawler-toturial
/Users/lzjun/anaconda3/envs/crawler-toturial
crawler_test
/Users/lzjun/anaconda3/envs/crawler_test
weixin
/Users/lzjun/anaconda3/envs/weixin
root                  *  /Users/lzjun/anaconda3
```

以上是我系统里面用 conda 管理的虚拟环境，juypter notebook 中的 Python3 对应的就是 root 环境，我们现在切换到 root 环境来安装其它第三方包。

```
# windows 不需要加 source
source activate root
# 安装 pymongo
pip install pymongo
```

回到 jupyter notebook ，导入基础包（以下代码都是在 jupyter notebook 中完成）

```
# 加这行不需要再写plt.show()，直接显示图像出来
%matplotlib inline

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

display_columns =
["title","read_num","like_num","comment_num","rew
ard_num","p_date"]
```

从 MongoDB 导入数据

```
import pymongo
from pymongo import MongoClient
# 连接 mongodb
c = MongoClient()
cursor = c.weixin3['post'].find()
df = pd.DataFrame(list(cursor))

# 删除 "_id"列
df = df.drop("_id", axis=1)
# 重新设置列的顺序
df = df.reindex(columns=display_columns)
# 将p_date的数据类型从timestamp 转换成 datetime
df.p_date = pd.to_datetime(df['p_date'])
df.head()
```

前5条数据：

	title	p_date	read_num	like_num	comment_num	reward_num	author
0	为什么我推荐你学习数据分析	2018-01-15 19:59:14	3549	22	0	0	刘志军
1	推荐几个公众号（文末彩蛋）	2018-01-12 12:13:55	4866	37	0	0	刘志军
2	普通程序员的逆袭：如何学习数据挖掘与人工智能	2018-01-11 21:06:24	3582	14	0	0	万门大学
3	这10个题，有 68% 的人答不对	2018-01-10 07:31:18	3643	49	0	1	刘志军
4	Python知识冲顶大会	2018-01-08 08:00:00	5308	30	0	0	刘志军

从 CSV 文件中导入

如果你的 MongoDB 没有数据，可以用我给你准备的[JSON文件](https://github.com/pythonzhichan/weixincrawler/blob/mast)
下载到本地后用 Pandas 导入进来

```
# 从csv文件中加载
df = pd.read_csv("post.csv")
# 重新设置列的顺序
df = df.reindex(columns=display_columns)
# 将p_date的数据类型从timestamp 转换成 datetime
df.p_date = pd.to_datetime(df['p_date'])
```

文章与阅读数

数据加载到 Pandas 之后，先来看下数据的总体概览情况

```
In [18]: df.describe()
```

Out[18]:

	read_num	like_num	comment_num	reward_num
count	203.000000	203.000000	203.0	203.000000
mean	2404.694581	31.891626	0.0	2.551724
std	2005.320602	31.026467	0.0	7.322895
min	124.000000	0.000000	0.0	0.000000
25%	1032.500000	8.500000	0.0	0.000000
50%	1844.000000	25.000000	0.0	0.000000
75%	3498.000000	43.500000	0.0	3.000000
max	8628.000000	190.000000	0.0	83.000000

从上面看出公众号一共发了 203 篇文章，文章平均阅读量是 2404，标准差 2005 说明文章阅读量波动非常大，从最高阅读量 8628 到最低阅读量 124 可以证明其波动性。为什么标准差这么大呢？这个其实很容易说明，因为公众号初期订阅读者少，阅读量也不高，但是随着你读者越来越多，阅读量也会越来越高。

这里的文章赞赏数和点赞数有一定的误差，因为我在初始化数据的时候，给每篇文章赞赏数默认设置为了0，而正确的方式应该是设置为None，如果为None 数据就不会统计进来。

获取阅读量最高的10篇文章

```
# 根据阅读数排序，ascending 表示降序排列
top_read_num_10 = df.sort_values(by=['read_num'],
ascending=False)[:10]
top_read_num_10 =
top_read_num_10[display_columns]
# 重置行索引，drop 表示删除原来的行索引
top_read_num_10.reset_index(drop=True)
```

	title	read_num	like_num	comment_num	reward_num	p_date
0	微软考虑将 Python 作为 Excel 官方脚本语言	8628	66	0	0	2017-12-17 16:57:23
1	5个酷毙的Python工具	8443	39	0	0	2017-12-27 08:00:00
2	Python是怎么火起来的	8423	67	0	3	2017-06-30 07:54:56
3	Python爬虫知识点梳理	8370	128	0	5	2017-09-20 18:12:54
4	如何快速入门Python	8132	123	0	13	2017-06-05 08:47:48
5	最新技能 get：用 AI 玩微信跳一跳，自动刷分可破 10000 分	8078	75	0	0	2018-01-05 16:30:00
6	推荐几个公众号	7726	105	0	7	2017-11-12 14:55:30
7	Python语言的2017年终总结	7446	56	0	1	2017-12-25 08:24:12
8	10行代码爬取微信公众号文章评论	7125	62	0	3	2017-07-30 21:49:18
9	零基础如何自学Python	6811	124	0	6	2017-12-15 07:50:00

历史文章阅读量变化曲线

```
ax = df.plot(y='read_num', x='p_date', title="文章
阅读量趋势",figsize=(9,6))
# 设置y轴标签
ax.set_ylabel("阅读量")
# 设置x轴标签
ax.set_xlabel("")
# 隐藏图例
ax.legend().set_visible(False)
```

一眼就看出来，阅读量都集中在 2017 这一年，那么前几年究竟发生什么了？是没写文章还是写了文章没人看？我们来统计一下这几年的文章数。

In [ ]:
```
按年分组统计每年写文章的数量
```

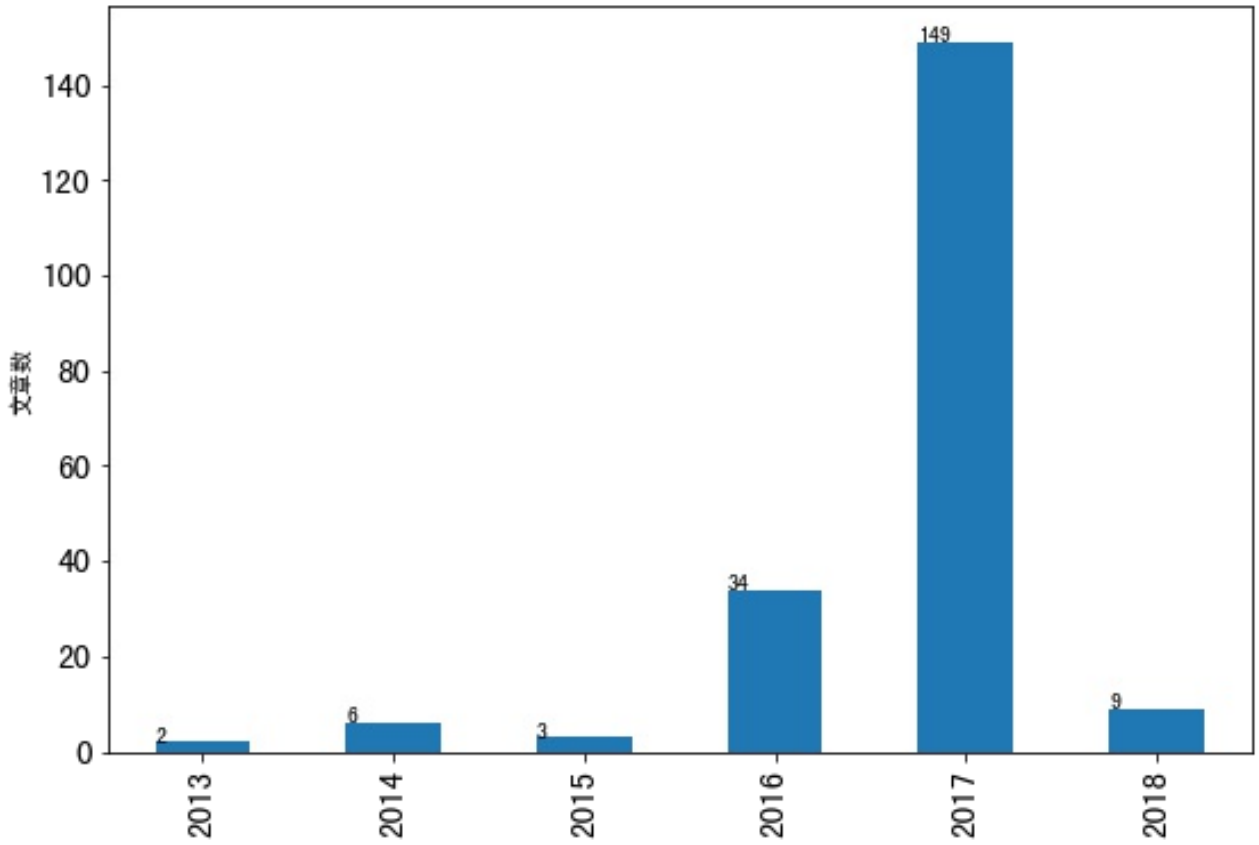
In [20]:
```
# 按年分组
year_df = df.groupby(df.p_date.dt.year).size().reset_index(name='total')
year_df
```

Out[20]:

	p_date	total
0	2013	2
1	2014	6
2	2015	3
3	2016	34
4	2017	149
5	2018	9

数据告诉我们，13年发了2篇文章（笑cry表情），而17年发了 149 篇文章（棒棒哒），平均每周大概有近 3 篇文章的更新频率，用柱状图展示就是这样：

```
ax = year_df.plot(x='p_date', y='total',
kind='bar', figsize=(9,6), fontsize=15)
ax.set_ylabel("文章数")
ax.set_xlabel("")
ax.legend().set_visible(False)
# 柱状图上显示数字
for p in ax.patches:
    ax.annotate(str(p.get_height()), xy=
(p.get_x(), p.get_height()))
```



文章与赞赏

再来分析我们的文章赞赏情况

```
In [26]: # 过滤赞赏数大于0的数据
         reward_count = len(df[df.reward_num>0])
         reward_count

Out[26]: 101

In [27]: reward_count/len(df)

Out[27]: 0.4975369458128079

In [28]: # 总共收到518次赞赏
         total = df['reward_num'].sum()
         total

Out[28]: 518
```

总共有101篇文章赞赏，平均两篇文章就有1次赞赏，读者一共贡献了 518 次赞赏，谢谢可爱读者们支持（微笑表情）

用同样的方式可以得到文章赞赏数前10的数据：

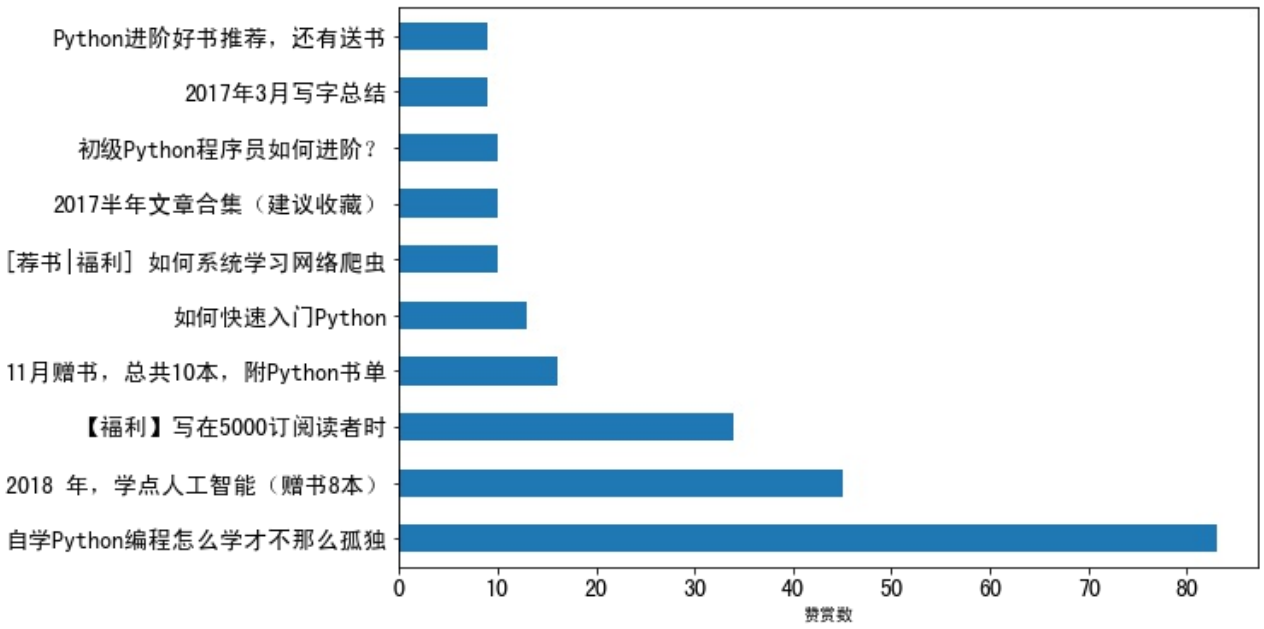
```
top_reward_num = df.sort_values(by=
['reward_num'], ascending=False)[:10]
top_reward_num = top_reward_num[display_columns]
top_reward_num
top_reward_num.reset_index(drop=True)
```

	title	read_num	like_num	comment_num	reward_num	p_date
0	自学Python编程怎么学才不那么孤独	5763	100	0	83	2017-10-24 08:36:40
1	2018 年，学点人工智能（赠书8本）	6353	137	0	45	2017-12-29 08:00:00
2	【福利】写在5000订阅读者时	1246	58	0	34	2017-04-14 18:24:18
3	11月赠书，总共10本，附Python书单	6485	190	0	16	2017-11-28 08:00:00
4	如何快速入门Python	8132	123	0	13	2017-06-05 08:47:48
5	[荐书|福利] 如何系统学习网络爬虫	4305	76	0	10	2017-09-28 17:16:31
6	2017半年文章合集（建议收藏）	3031	37	0	10	2017-07-09 09:25:45
7	初级Python程序员如何进阶?	4798	115	0	10	2017-05-18 17:09:30
8	2017年3月写字总结	1286	52	0	9	2017-04-02 01:35:01
9	Python进阶好书推荐，还有送书	5110	141	0	9	2017-07-26 07:30:00

最高的一篇文章有83个打赏，这究竟是一篇什么文章，戳-->[自学Python编程怎么学才不那么孤独 (http://mp.weixin.qq.com/s?__biz=MjM5MzgyODQxMQ==&mid=2650367720&idx=1&sn=](http://mp.weixin.qq.com/s?__biz=MjM5MzgyODQxMQ==&mid=2650367720&idx=1&sn=)

```
ax = top_reward_num.plot(x='title',
                         y='reward_num',
                         kind='barh',
                         figsize=(9,6),
                         fontsize=14)
ax.set_ylabel("")
ax.set_xlabel("赞赏数")
ax.legend().set_visible(False)
```

这里的 kind 用 "barh" 表示横向的条形图

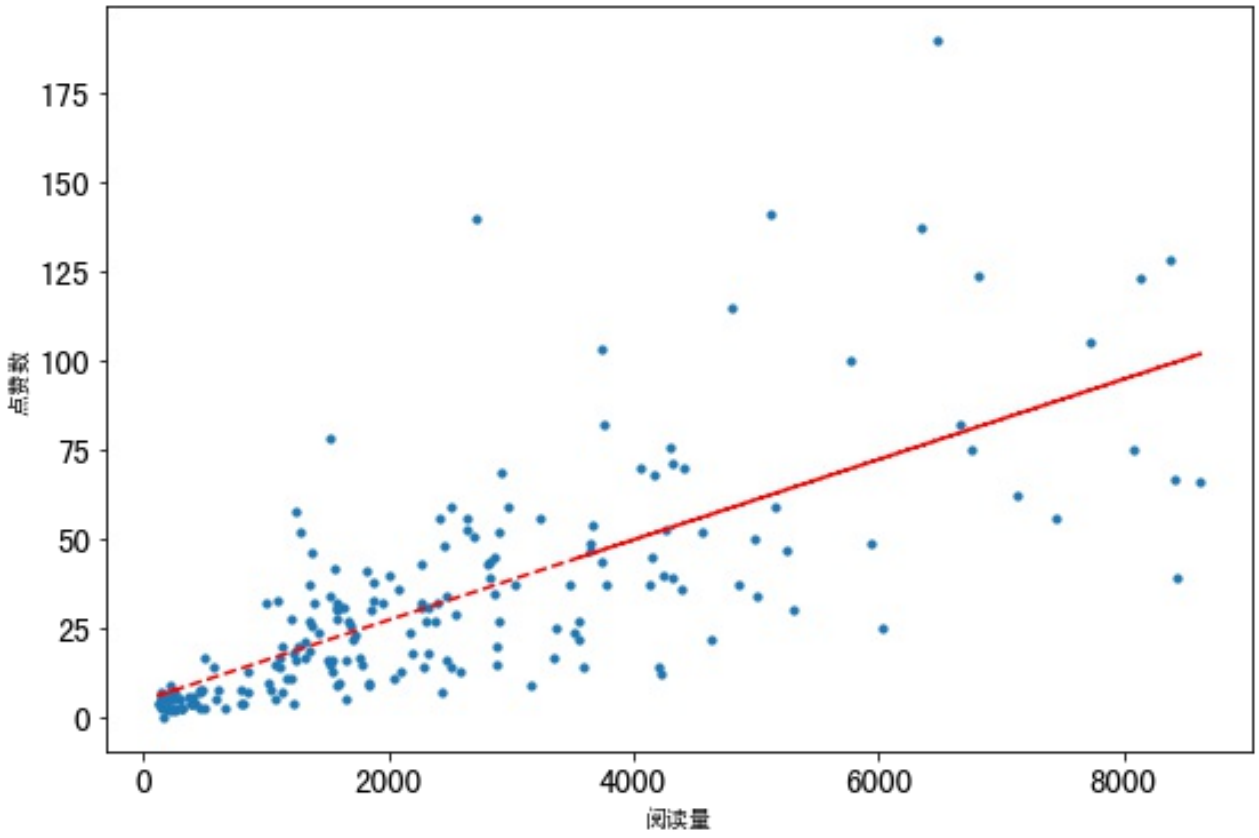


文章与点赞

说完赞赏的数据，再来看看点赞数与文章阅读数有什么关系，我们可以用散点图来表示二者之间关系，散点图用两组数据构成多个坐标点，表示因变量随自变量而变化的大致趋势。

```
# 散点图
ax = df.plot(kind="scatter", y='like_num',
x='read_num',s=10, figsize=(9,6), fontsize=15)
ax.set_xlabel("阅读量")
ax.set_ylabel("点赞数")

z = np.polyfit(df.read_num, df.like_num, 1)
p = np.poly1d(z)
plt.plot(df.read_num,p(df.read_num),"r--")
```



可以看出文章点赞数大部分集中在10~50之间，而且存在某种线性正相关性，也就是说，文章阅读数越高，点赞数也就越高，如果某篇文章阅读量很高，但是点赞数却很低，这样的文章是标题党或者是资讯类的文章的可能性比较大。

标题关键字

最后，我想基于文章标题做一个词云效果展示，看看这些文章标题都用了哪些关键字。这里需要用到另个包，一个是结巴分词，另一个词云包

```
conda install jieba
conda install wordcloud
```

```
from wordcloud import WordCloud
import jieba

words = []
for i in  df.title:
    seg_list = jieba.cut(i, cut_all=False)
    words.append(" ".join(seg_list))
wordcloud =
WordCloud(font_path='/Library/Fonts/Songti.ttc',
                        background_color="white",
                        max_words=80,).generate("
".join(words))
plt.figure(figsize=(9,6))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

把所有文章的标题同结巴库分词处理加入到 words 列表中，传递给 WordCloud 组件，另外还需要指定一个中文字体，因为 wordcloud 默认无法处理中文。max_words 用于指定最多显示多少词语

小结

到这里，我们就完成了一个公众号基本分析工作，得到一些结论，比如阅读量高的往往不是某个具体的知识点干货内容，而是一些更通俗的文章，要么是资讯，要么是一些工具介绍，或者是编程的方法论等文章。而赞赏文章基本集中在带有福利的文章里面，从文章标题得知公众号文章都是围绕Python写的文章。

本节ipynb源代码地址：[https://github.com/pythonzhichan/weixincrawler (https://github.com/pythonzhichan/weixincrawler)](https://github.com/pythonzhichan/weixincrawler)