customer_action_report

2020 年 6 月 11 日

用户行为分析消费报告

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pymysql
from sqlalchemy import create_engine
%matplotlib inline
```

[4]:
```
sql = "select customer_id,order_date,quantity,sales from spm_order where␣
 ↪order_date between '2018-01-01' and '2018-12-31'"
engine = create_engine("mysql+pymysql://root:root@192.168.1.113:3306/
 ↪data_analysis?charset=utf8")
df = pd.read_sql(sql, engine)
# 写入数据库
# df.to_slq(name="new_table", con="mysql+pymysql://root:root@192.168.8.107:3306/
 ↪data_analysis?charset=utf8", name=, if_exit=)
```

- customer_id：客户 ID
- order_date：购买日期
- quantity：购买产品数
- sales：购买金额

[5]:
```
df.head()
```

[5]:

	customer_id	order_date	quantity	sales
0	曾惠-14485	2018-04-27	2	129.696
1	许安-10165	2018-06-15	2	125.44
2	许安-10165	2018-06-15	2	31.92

```
3    宋良-17170  2018-12-09        4  321.216
4    康青-19585  2018-06-01        4   2368.8
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3377 entries, 0 to 3376
Data columns (total 4 columns):
customer_id    3377 non-null object
order_date     3377 non-null object
quantity       3377 non-null object
sales          3377 non-null object
dtypes: object(4)
memory usage: 105.7+ KB
```

[7]:
```
df[['quantity', 'sales']] = df[['quantity', 'sales']].astype(float)
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3377 entries, 0 to 3376
Data columns (total 4 columns):
customer_id    3377 non-null object
order_date     3377 non-null object
quantity       3377 non-null float64
sales          3377 non-null float64
dtypes: float64(2), object(2)
memory usage: 105.7+ KB
```

[9]:
```
df.describe()
```

[9]:

	quantity	sales
count	3377.000000	3377.000000
mean	3.823216	1617.541720
std	2.270866	2658.536943
min	1.000000	13.440000
25%	2.000000	250.600000
50%	3.000000	657.300000

```
75%        5.000000    1765.260000
max       14.000000   30306.640000
```

- 大部分订单消费了少量的商品，平均在 3.8，存在一定的极值干扰
- 购买金额的的均值大于中位数较多，受到严重的极值干扰
- 根据数据特征描述一些结论
- 分析是什么原因造成的，有什么解决方案

[10]:
```
df["order_date"] = pd.to_datetime(df.order_date,format="%Y-%m-%d")
```

[11]:
```
df.dtypes
```

[11]:
```
customer_id              object
order_date       datetime64[ns]
quantity                float64
sales                   float64
dtype: object
```

[12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3377 entries, 0 to 3376
Data columns (total 4 columns):
customer_id    3377 non-null object
order_date     3377 non-null datetime64[ns]
quantity       3377 non-null float64
sales          3377 non-null float64
dtypes: datetime64[ns](1), float64(2), object(1)
memory usage: 105.7+ KB
```

[13]:
```
df["month"] = df.order_date.values.astype('datetime64[M]')
```

[14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3377 entries, 0 to 3376
Data columns (total 5 columns):
customer_id    3377 non-null object
order_date     3377 non-null datetime64[ns]
```

```
quantity       3377 non-null float64
sales          3377 non-null float64
month          3377 non-null datetime64[ns]
dtypes: datetime64[ns](2), float64(2), object(1)
memory usage: 132.0+ KB
```

[15]:
```
df.head()
```

[15]:

	customer_id	order_date	quantity	sales	month
0	曾惠-14485	2018-04-27	2.0	129.696	2018-04-01
1	许安-10165	2018-06-15	2.0	125.440	2018-06-01
2	许安-10165	2018-06-15	2.0	31.920	2018-06-01
3	宋良-17170	2018-12-09	4.0	321.216	2018-12-01
4	康青-19585	2018-06-01	4.0	2368.800	2018-06-01

0.1 1. 进行用户消费趋势的分析

- 每月的消费总金额
- 每月的消费总次数
- 每月的产品购买量
- 每月的消费人数

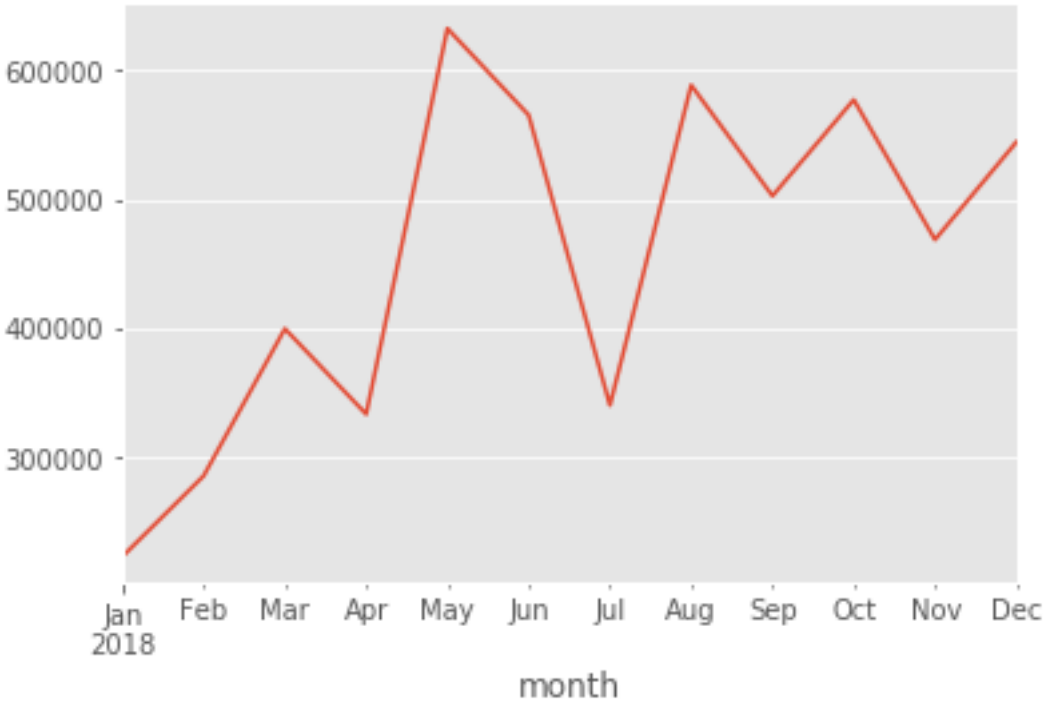
[16]:
```
grouped_month = df.groupby('month')
order_month_amount = grouped_month.sales.sum()
order_month_amount
```

[16]:
```
month
2018-01-01    222862.829
2018-02-01    285475.428
2018-03-01    399711.781
2018-04-01    333398.317
2018-05-01    632800.350
2018-06-01    565523.427
2018-07-01    340308.682
2018-08-01    588746.354
2018-09-01    502799.255
2018-10-01    577450.321
2018-11-01    468822.543
```

```
2018-12-01    544539.100
Name: sales, dtype: float64
```

[17]:
```
plt.style.use('ggplot')  # 更改设计风格
order_month_amount.plot()  # 调用的是 dataframe 下的绘图函数，并不是 matplotlib.
↪pyplot 下的
```

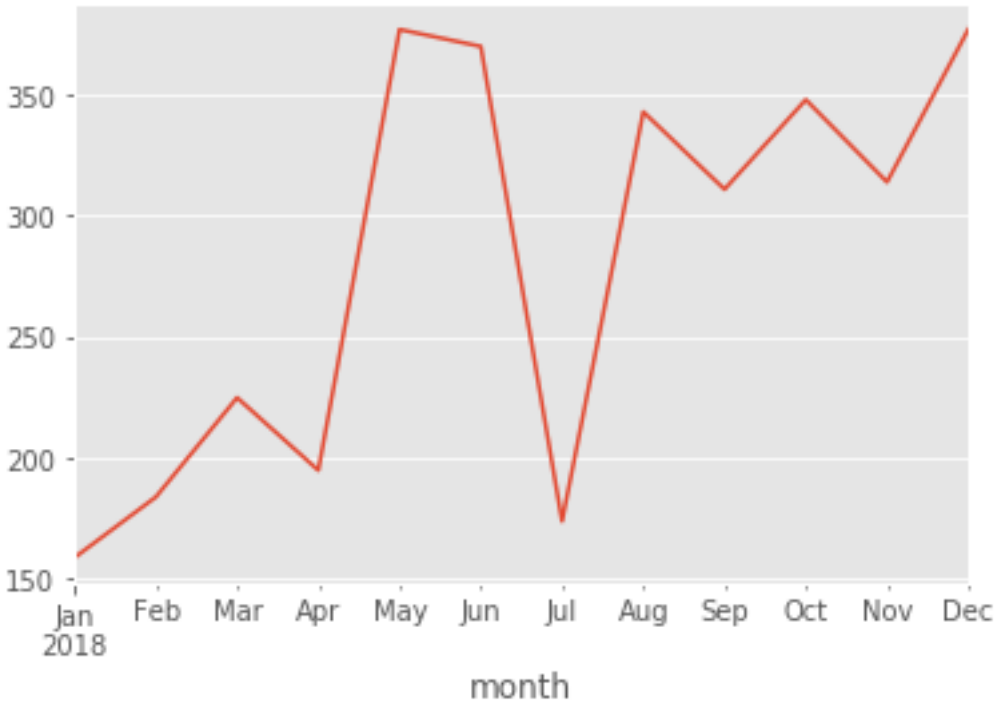
[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x95660c8>
```



0.1.1 由上图可知，消费金额在一月达到最低谷，五月达到最高峰，在八月后达到相对稳定（由于没有实际业务背景，无法分析出具体原因）

[18]:
```
grouped_month.customer_id.count().plot()
```

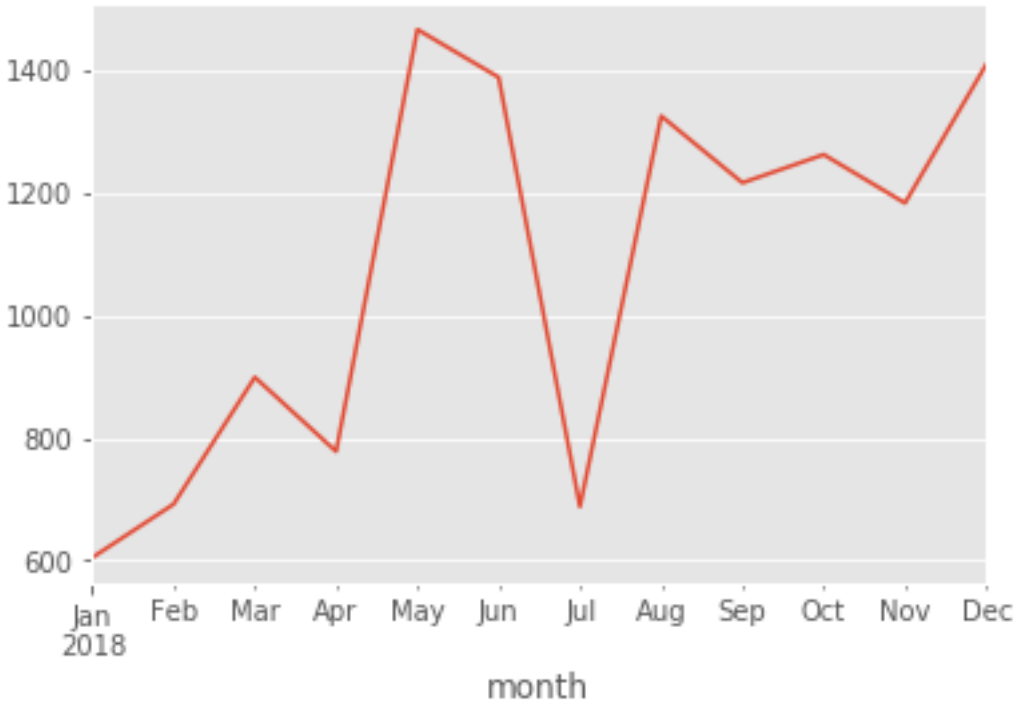
[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x96fffc8>
```



0.1.2 消费次数最多时达到 **390** 左右，最低时有 **155** 左右，在八月后稳定在 **325** 左右，有上升的趋势

[19]:
```
grouped_month.quantity.sum().plot()
```

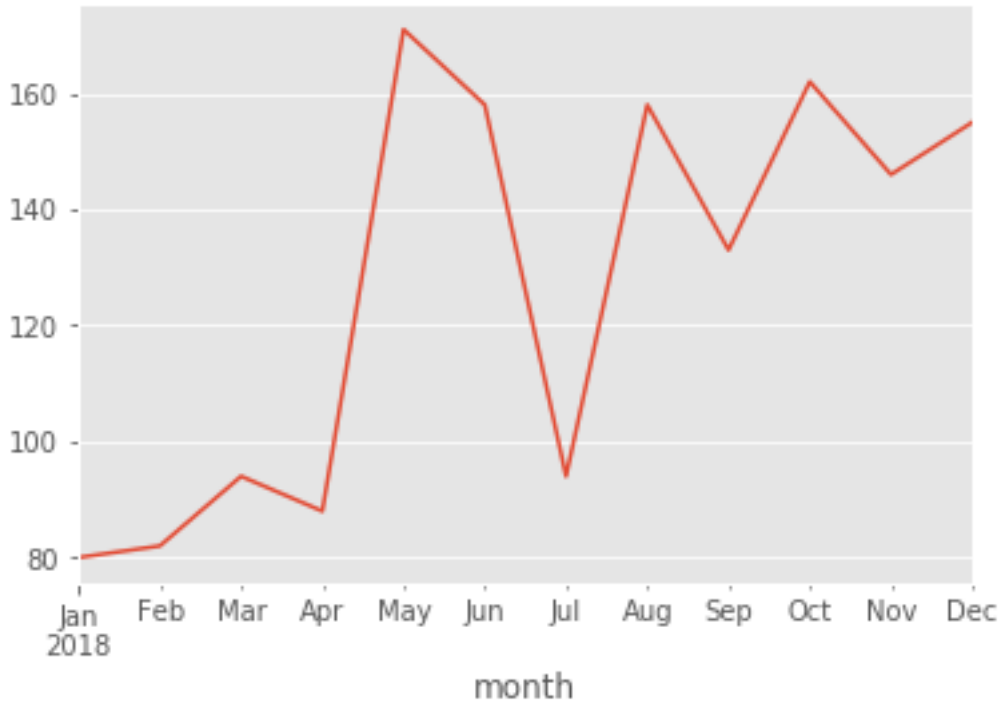
[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x9bf8048>
```



0.1.3 产品的购买量与消费人数呈现正相关关系，购买量在五月份达到最大值，在一月份时达到最小值

[20]:
```
# grouped_month.customer_id.apply(lambda x:len(x.drop_duplicates())).plot()
df.groupby('month').customer_id.apply(lambda x:len(x.drop_duplicates())).plot()
# df.groupby(['month','customer_id']).count().reset_index()  # 多重分组
```

[20]: <matplotlib.axes._subplots.AxesSubplot at 0x9c8d088>



0.1.4 前三月的消费人数在 80-100 之间，在五月份达到最大值，在八月份后稳定在 140-160 左右

[21]:
```
df.head()
```

[21]:

	customer_id	order_date	quantity	sales	month
0	曾惠-14485	2018-04-27	2.0	129.696	2018-04-01
1	许安-10165	2018-06-15	2.0	125.440	2018-06-01
2	许安-10165	2018-06-15	2.0	31.920	2018-06-01
3	宋良-17170	2018-12-09	4.0	321.216	2018-12-01
4	康青-19585	2018-06-01	4.0	2368.800	2018-06-01

[22]:
```
# 利用透视表
df.pivot_table(index='month',
               values=['sales','quantity','customer_id'],
               aggfunc={'quantity':'sum',
                        'sales':'sum',
                        'customer_id':'count'})
```

[22]:

month	customer_id	quantity	sales
2018-01-01	159	605.0	222862.829
2018-02-01	184	692.0	285475.428
2018-03-01	225	899.0	399711.781
2018-04-01	195	778.0	333398.317
2018-05-01	377	1466.0	632800.350
2018-06-01	370	1388.0	565523.427
2018-07-01	174	688.0	340308.682
2018-08-01	343	1325.0	588746.354
2018-09-01	311	1216.0	502799.255
2018-10-01	348	1262.0	577450.321
2018-11-01	314	1183.0	468822.543
2018-12-01	377	1409.0	544539.100

初次之外还可以计算: - 每月用户平均消费金额趋势 - 每月用户平均消费次数的趋势 - 等

0.2 2. 用户个体消费分析

- 用户消费金额、消费次数的描述统计
- 用户消费金额和消费的次数散点图
- 用户消费金额的分布图
- 用户消费次数的分布图
- 用户累计消费金额占比（百分之多少的用户占了百分之多少的消费额）

[23]:
```
grouped_user=df.groupby('customer_id')
```

[24]:
```
grouped_user.sum().describe()
```

[24]:

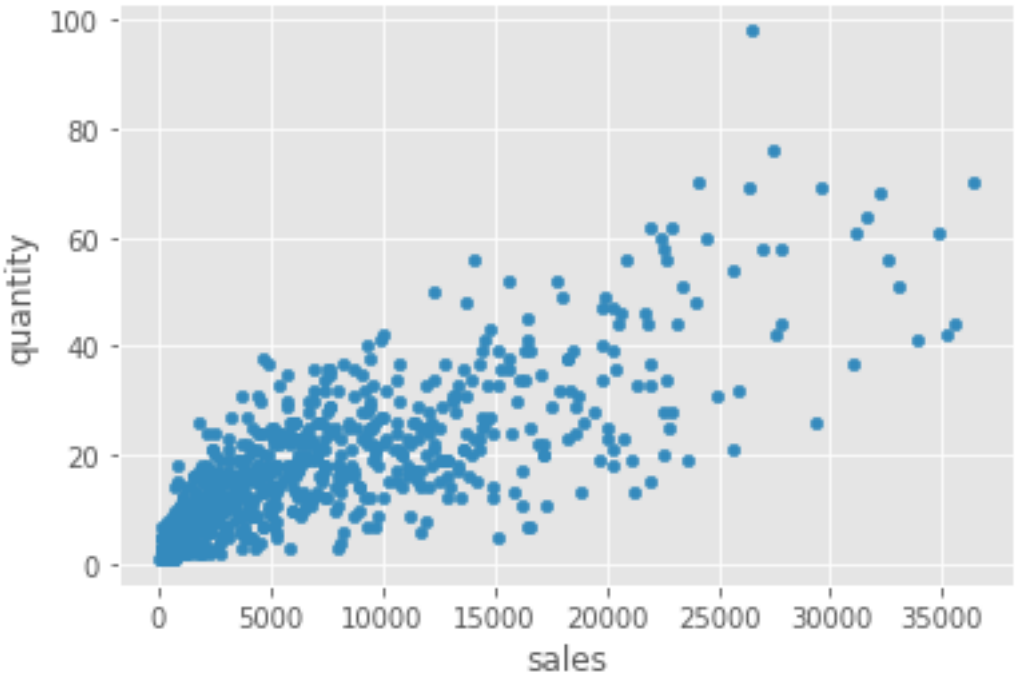
	quantity	sales
count	698.000000	698.000000
mean	18.497135	7825.842961
std	14.002997	7752.744518
min	1.000000	39.200000
25%	8.000000	2032.310000
50%	16.000000	5321.512000
75%	25.000000	11472.951000
max	98.000000	64066.380000

0.2.1 每个用户平均够购买数量 **18**，中位数是 **16**，受到一定极值的影响

0.2.2 每个用户平均购买 **7825** 元，中位数是 **5321**，也有极值干扰

[25]:
```
grouped_user.sum().query('sales<40000').plot.scatter(x='sales',y='quantity')
```

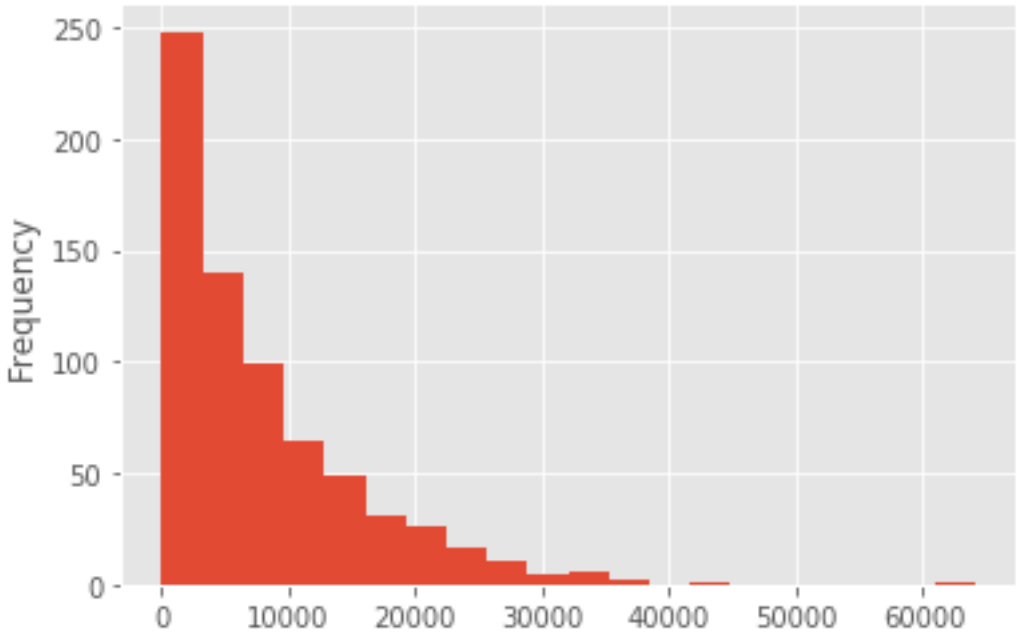
[25]: <matplotlib.axes._subplots.AxesSubplot at 0xad2d248>



0.2.3 购买的产品数量与购买金额相关性不强，可能与产品的种类有关

[26]:
```
grouped_user.sum().sales.plot.hist(bins=20)
```

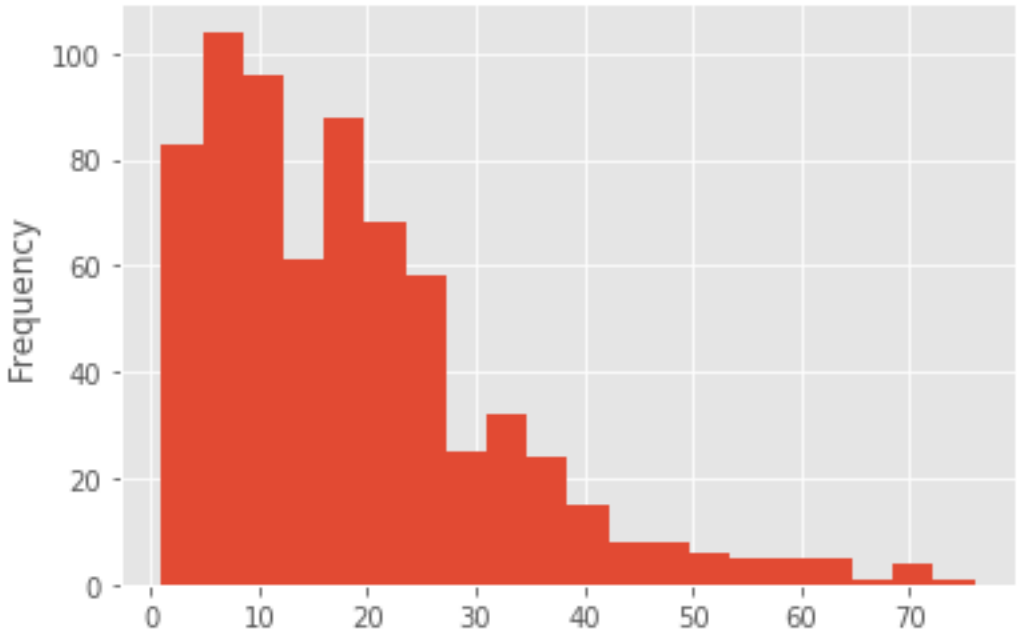
[26]: <matplotlib.axes._subplots.AxesSubplot at 0xad19cc8>



0.3 从上图可知，用户金额绝大多数呈现集中趋势, 用户大部分都处于低层消费

[27]:
```
grouped_user.sum().query('quantity<98').quantity.plot.hist(bins=20)
```

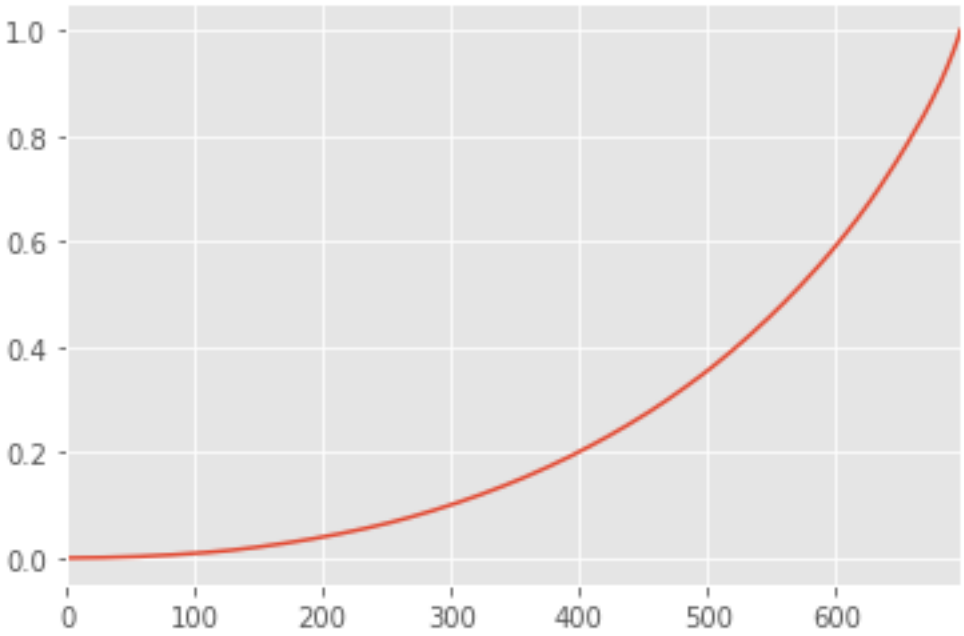
[27]: <matplotlib.axes._subplots.AxesSubplot at 0xadfe708>



0.3.1 使用切比雪夫定理过滤掉异常值，计算 95% 的数据分布情况，

[28]:
```
user_cumsum = grouped_user.sum().sort_values('sales').apply(lambda x:x.cumsum()/
↪x.sum())
user_cumsum.reset_index().sales.plot()
```

[28]: <matplotlib.axes._subplots.AxesSubplot at 0xb514988>

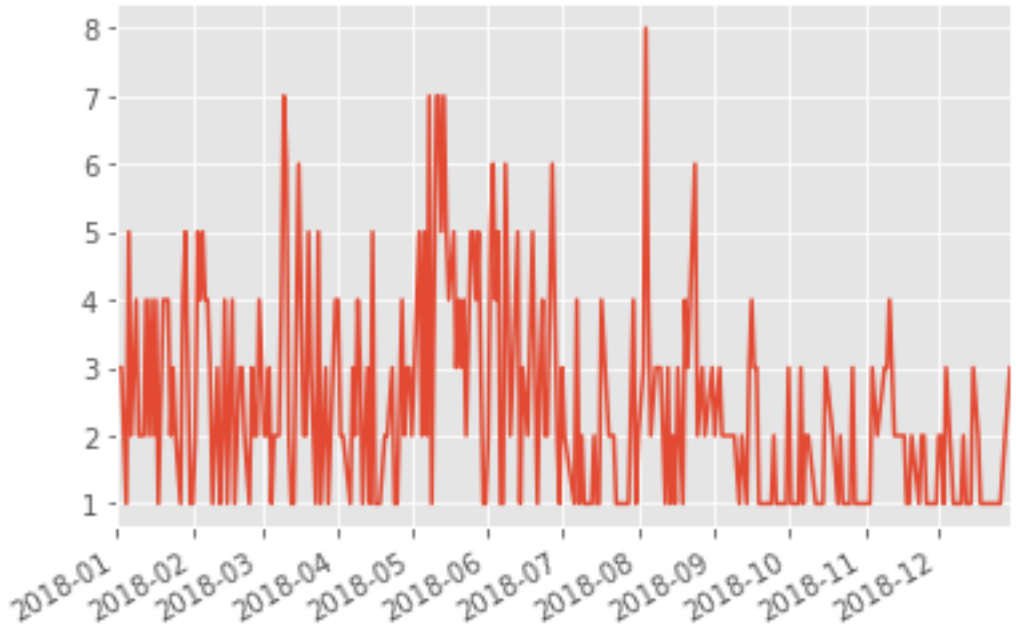


0.4 3. 用户消费行为

- 用户第一次消费（首购）
- 用户最后一次消费
- 新老客户消费比
  - 多少用户仅消费了一次
  - 每月新客占比
- 用户分层
  - RFM

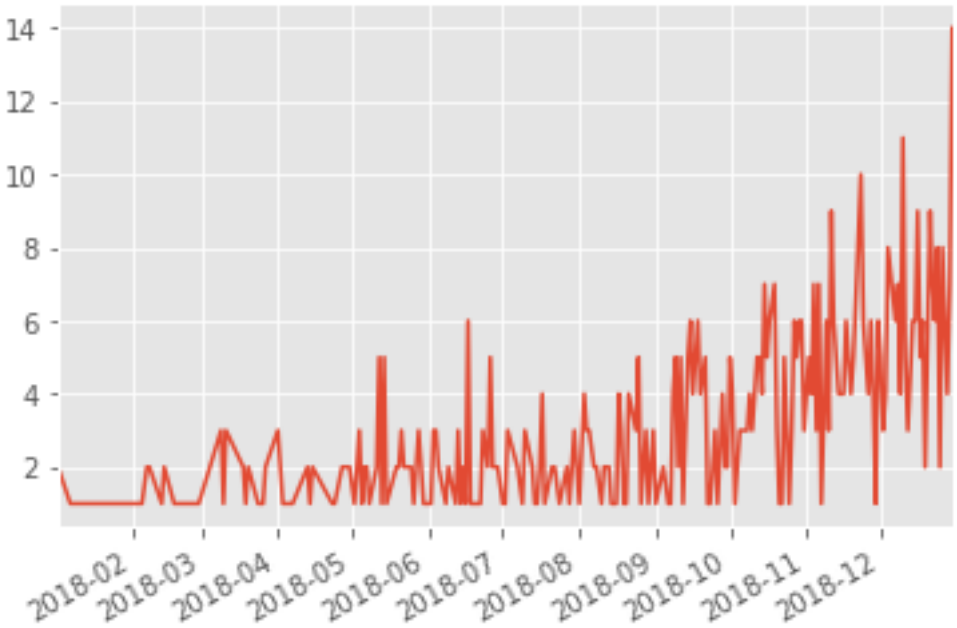
  - 新、老、活跃、回流、流失
- 用户购买周期（按订单）
  - 用户消费周期描述
  - 用户消费周期分布
- 用户生命周期（按第一次 & 最后一次消费）
  - 用户生命周期描述
  - 用户生命周期分布

[29]:
```
grouped_user.min().order_date.value_counts().plot()  # 用户第一次消费
plt.show()
```



[30]:
```
grouped_user.max().order_date.value_counts().plot()  # 用户最后一次消费
```

[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb566a88>
```



0.4.1 随着时间的递增，最后一次购买数也在递增，消费呈现上升的趋势，可能是因为运营恰当，用户忠诚度较高

[31]:
```
user_life = grouped_user.order_date.agg(['min','max'])
user_life.head()
```

[31]:

customer_id	min	max
丁君-15280	2018-06-23	2018-06-23
丁妮-18610	2018-07-20	2018-09-16
丁娇-14695	2018-06-08	2018-10-26
丁婵-10990	2018-07-01	2018-10-21
丁崆-15535	2018-03-16	2018-05-06

[32]:
```
(user_life['min'] == user_life['max']).value_counts()
```

[32]:
```
False    492
True     206
dtype: int64
```

0.4.2 有七分之二的用户就消费了一次

[33]:
```
rfm = df.pivot_table(index = 'customer_id',
                     values = ['quantity','sales','order_date'],
                      aggfunc = {
                          'order_date':'max',
                          'sales':'sum',
                           'quantity':'sum'
                      })
rfm.head()
```

[33]:

customer_id	order_date	quantity	sales
丁君-15280	2018-06-23	8.0	636.160
丁妮-18610	2018-09-16	19.0	6998.628
丁娇-14695	2018-10-26	26.0	7008.204
丁婵-10990	2018-10-21	9.0	2583.112
丁崆-15535	2018-05-06	19.0	12943.560

[34]:
```
rfm['R'] = -(rfm.order_date-rfm.order_date.max())/np.timedelta64(1,'D')
rfm.rename(columns = {'quantity':'F','sales':'M'},inplace=True)
```

[35]:
```
def rfm_func(x):
    level = x.apply(lambda x:'1' if x>=0 else '0')
    label = level.R + level.F + level.M
    d = {
        '111':'重要价值客户',
        '011':'重要保持客户',
        '101':'重要发展客户',
        '001':'重要挽留客户',
        '110':'一般价值客户',
        '010':'一般保持客户',
        '100':'一般发展客户',
        '000':'一般挽留客户'
    }
    result = d[label]
    return result
```

```
rfm['label'] = rfm[['R','F','M']].apply(lambda x:x-x.mean()).
 ↪apply(rfm_func,axis=1)
```

[36]:
```
rfm
```

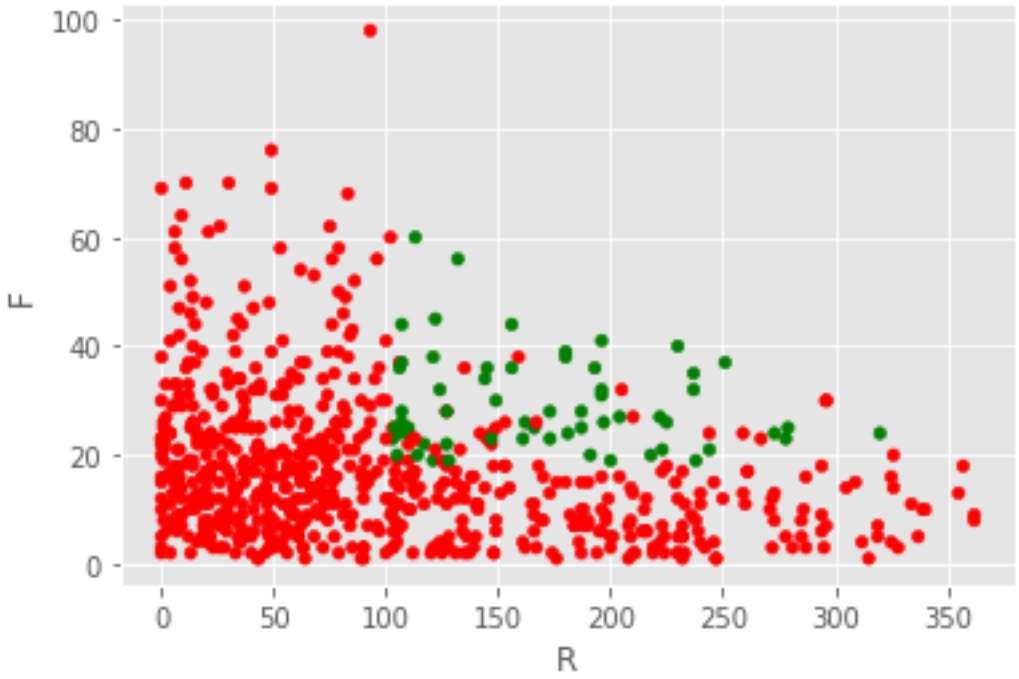
[36]:

customer_id	order_date	F	M	R	label
丁君-15280	2018-06-23	8.0	636.160	190.0	一般发展客户
丁妮-18610	2018-09-16	19.0	6998.628	105.0	一般价值客户
丁娇-14695	2018-10-26	26.0	7008.204	65.0	一般保持客户
丁婵-10990	2018-10-21	9.0	2583.112	70.0	一般挽留客户
丁崆-15535	2018-05-06	19.0	12943.560	238.0	重要价值客户
…	…	…	…	…	…
龙婷-21115	2018-12-21	12.0	3213.000	9.0	一般挽留客户
龙廷-21295	2018-04-02	12.0	1840.440	272.0	一般发展客户
龙谦-19300	2018-09-19	60.0	22491.014	102.0	重要保持客户
龙锦-14875	2018-12-15	37.0	10715.740	15.0	重要保持客户
龚松-20710	2018-11-23	25.0	16623.712	37.0	重要保持客户

[698 rows x 5 columns]

[37]:
```
rfm.loc[rfm.label == '重要价值客户','color'] = 'g'
rfm.loc[~(rfm.label == '重要价值客户'), 'color'] = 'r'
rfm.plot.scatter('R','F',c=rfm.color)
```

[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb5c7788>
```



[38]:
```
rfm.groupby('label').sum()
```

[38]:

label	F	M	R
一般价值客户	711.0	157065.804	4713.0
一般保持客户	1487.0	328521.340	2449.0
一般发展客户	1274.0	387476.068	31788.0
一般挽留客户	1808.0	480943.050	8099.0
重要价值客户	1761.0	993809.117	10213.0
重要保持客户	4980.0	2331268.373	6088.0
重要发展客户	433.0	396864.811	6485.0
重要挽留客户	457.0	386489.824	1579.0

[39]:
```
rfm.groupby('label').count()
```

[39]:

label	order_date	F	M	R	color
一般价值客户	28	28	28	28	28
一般保持客户	60	60	60	60	60

```
一般发展客户          160  160  160  160    160
一般挽留客户          188  188  188  188    188
重要价值客户           60   60   60   60     60
重要保持客户          135  135  135  135    135
重要发展客户           33   33   33   33     33
重要挽留客户           34   34   34   34     34
```

[40]:
```
pivoted_counts = df.pivot_table(index = 'customer_id',
#                                 columns = 'month',
                                values = 'order_date',
                                aggfunc = 'count').fillna(0)
pivoted_counts.head()
```

[40]:
```
             order_date
customer_id
丁君-15280             1
丁妮-18610             3
丁娇-14695             5
丁婵-10990             2
丁崆-15535             5
```

[41]:
```
df_purchase = pivoted_counts.applymap(lambda x:1 if x>0 else 0)
df_purchase.tail()
```

[41]:
```
             order_date
customer_id
龙婷-21115             1
龙廷-21295             1
龙谦-19300             1
龙锦-14875             1
龚松-20710             1
```

[42]:
```
def active_status(data):
    status = []
    for i in range(12):
        if data[i] == 0:  # 若本月没有消费
            if len(status) > 0:
```

```
                if status[i-1] == 'unreg':
                    status.append('unreg')
                else:
                    status.append('unactive')
            else:
                status.append('unreg')
        else:  # 若本月没有消费
            if len(status) == 0:
                status.append('new')
            else:
                if status[i-1] == 'unactive':
                    status.append('return')
                elif status[i-1] == 'unreg':
                    status.append('new')
                else:
                    status.append('active')
    return status
```

- 若本月没有消费
  - 若之前是未注册，则依旧为注册
  - 若之前有消费，则为流失/不活跃
  - 其他情况，为未注册
- 若本月有消费
  - 若是第一次消费，则为新用户
  - 如果之前有过消费，则上个月为不活跃，则为回流
  - 如果上个月未注册，则为新用户
  - 除此之外，为活跃

[53]:
```
# purchase_stats = df_purchase.apply(active_status,axis=1)
# purchase_stats.head()
```

[55]:
```
# purchase_stats_ct = purchase_stats.replace('unreg',np.NaN).apply(lambda x:pd.
↪value_counts(x))
# purchase_stats_ct
```

[56]:
```
# purchase_stats_ct.fillna(0).T.head()
```

[47]:
```
# purchase_stats_ct.fillna(0).T.plot.area()
```

[58]:
```
# order_diff = grouped_user.apply(lambda x:x.order_date - x.order_date.shift())
# order_diff.head()
```

0.5 4. 复购率与回购率分析

- 复购率
  - 自然月内，购买多次的用户占比
- 回购率
  - 曾经购买过的用户在某一时期内的再次购买的占比

[ ]:
```

```