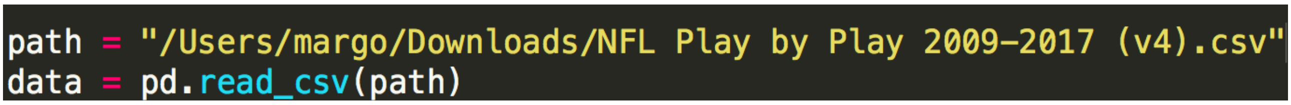
数据摘要

1.标称属性

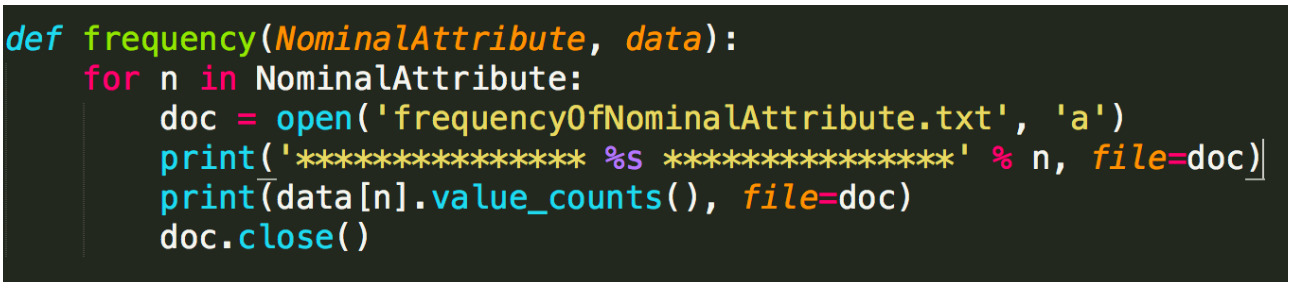
此数据集中的标称属性包括
['Date', 'GameID', 'Drive', 'qtr', 'down', 'time', 'TimeUnder', 'TimeSecs', 'SideofField', 'GoalToGo', 'FirstDown', 'posteam', 'DefensiveTeam', 'desc', 'PlayAttempted', 'sp', 'Touchdown', 'ExPointResult', 'TwoPointConv', 'DefTwoPoint', 'Safety', 'Onsidekick', 'PuntResult', 'PlayType', 'Passer', 'Passer_ID', 'PassAttempt', 'PassOutcome', 'PassLength', 'QBHit', 'PassLocation', 'InterceptionThrown', 'Interceptor','Rusher', 'Rusher_ID', 'RushAttempt', 'RunLocation', 'RunGap', 'Receiver', 'Receiver_ID', 'Reception', 'ReturnResult', 'Returner', 'BlockingPlayer', 'Tackler1', 'Tackler2', 'FieldGoalResult', 'Fumble', 'RecFumbTeam', 'RecFumbPlayer', 'Sack', 'Challenge.Replay', 'ChalReplayResult', 'Accepted.Penalty', 'PenalizedTeam', 'PenaltyType', 'PenalizedPlayer', 'HomeTeam', 'AwayTeam', 'Timeout_Indicator', 'Timeout_Team', ‘Season']

数据读取：



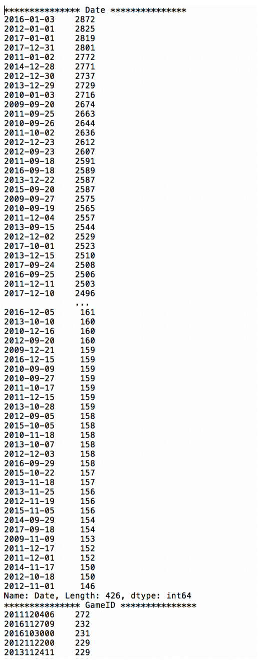
```
path = "/Users/margo/Downloads/NFL Play by Play 2009-2017 (v4).csv"
data = pd.read_csv(path)
```

对于标称属性，求出每个可能值的频数并保存在txt文件中，代码如下：

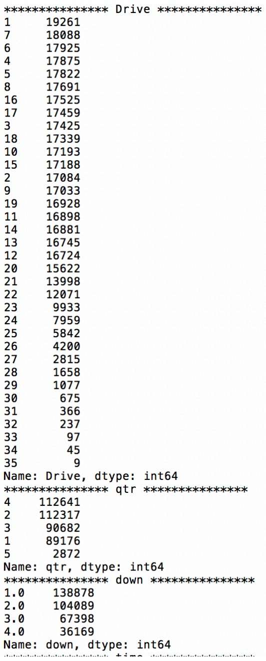


```
def frequency(NominalAttribute, data):
    for n in NominalAttribute:
        doc = open('frequencyOfNominalAttribute.txt', 'a')
        print('*************** %s ***************' % n, file=doc)
        print(data[n].value_counts(), file=doc)
        doc.close()
```

求出的部分结果如下：



```
*************** Date ***************
2016-01-03    2872
2012-01-01    2825
2017-01-01    2819
2017-12-31    2801
2011-01-02    2772
2014-12-28    2771
2012-12-30    2737
2013-12-29    2729
2010-01-03    2716
2009-09-20    2674
2011-09-25    2663
2010-09-26    2644
2011-10-02    2636
2012-12-23    2612
2012-09-23    2607
2011-09-18    2591
2016-09-18    2589
2013-12-22    2587
2015-09-20    2587
2009-09-27    2575
2010-09-19    2565
2011-12-04    2557
2013-09-15    2544
2012-12-02    2529
2017-10-01    2523
2013-12-15    2510
2017-09-24    2508
2016-09-25    2506
2011-12-11    2503
2017-12-10    2496
              ...
2016-12-05     161
2013-10-10     160
2010-12-16     160
2012-09-20     160
2009-12-21     159
2016-12-15     159
2010-09-09     159
2010-09-27     159
2011-10-17     159
2011-12-15     159
2013-10-28     159
2012-09-05     158
2015-10-05     158
2010-11-18     158
2013-10-07     158
2012-12-03     158
2016-09-29     158
2015-10-22     157
2013-11-18     157
2013-11-25     156
2012-11-19     156
2015-11-05     156
2014-09-29     154
2017-09-18     154
2009-11-09     153
2011-12-17     152
2011-12-01     152
2014-11-17     150
2012-10-18     150
2012-11-01     146
Name: Date, Length: 426, dtype: int64
*************** GameID ***************
2011120406    272
2016112709    232
2016103000    231
2012112200    229
2013112411    229
```

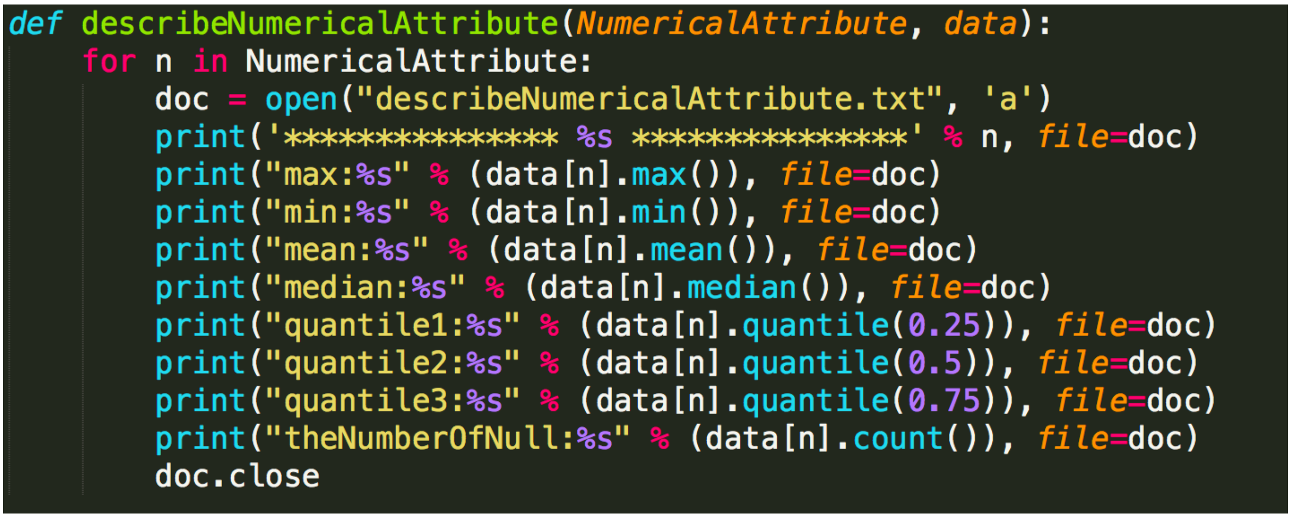


```
*************** Drive ***************
1     19261
7     18088
6     17925
4     17875
5     17822
8     17691
16    17525
17    17459
3     17425
18    17339
10    17193
15    17188
2     17084
9     17033
19    16928
11    16898
14    16881
13    16745
12    16724
20    15622
21    13998
22    12071
23     9933
24     7959
25     5842
26     4200
27     2815
28     1658
29     1077
30      675
31      366
32      237
33       97
34       45
35        9
Name: Drive, dtype: int64
*************** qtr ***************
4    112641
2    112317
3     90682
1     89176
5      2872
Name: qtr, dtype: int64
*************** down ***************
1.0    138878
2.0    104089
3.0     67398
4.0     36169
Name: down, dtype: int64
```

2.数值属性

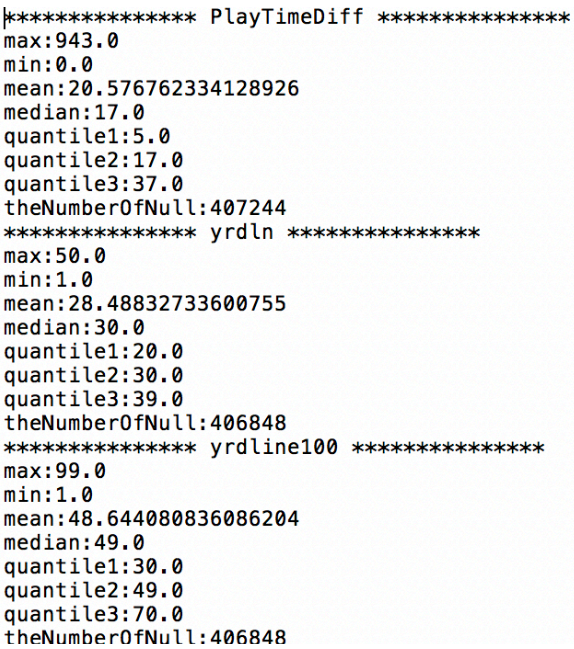
此数据集中的数值属性包括：
['PlayTimeDiff', 'yrdln', 'yrdline100', 'ydstogo', 'ydsnet', 'Yards.Gained', 'AirYards', 'YardsAfterCatch', 'FieldGoalDistance', 'Penalty.Yards', 'PosTeamScore', 'DefTeamScore', 'ScoreDiff', 'AbsScoreDiff', 'posteam_timeouts_pre', 'HomeTimeouts_Remaining_Pre', 'AwayTimeouts_Remaining_Pre', 'HomeTimeouts_Remaining_Post','AwayTimeouts_Remaining_Post', 'No_Score_Prob', 'Opp_Field_Goal_Prob', 'Opp_Safety_Prob', 'Opp_Touchdown_Prob', 'Field_Goal_Prob', 'Safety_Prob', 'Touchdown_Prob', 'ExPoint_Prob', 'TwoPoint_Prob', 'ExpPts', 'EPA', 'airEPA', 'yacEPA', 'Home_WP_pre', 'Away_WP_pre', 'Home_WP_post', 'Away_WP_post', 'Win_Prob', 'WPA', 'airWPA', 'yacWPA']

对于数值属性，求出了每个数值属性的最大值、最小值、均值、中位数、四分位数及缺失值的个数，代码如下：



```
def describeNumericalAttribute(NumericalAttribute, data):
    for n in NumericalAttribute:
        doc = open("describeNumericalAttribute.txt", 'a')
        print('*************** %s ***************' % n, file=doc)
        print("max:%s" % (data[n].max()), file=doc)
        print("min:%s" % (data[n].min()), file=doc)
        print("mean:%s" % (data[n].mean()), file=doc)
        print("median:%s" % (data[n].median()), file=doc)
        print("quantile1:%s" % (data[n].quantile(0.25)), file=doc)
        print("quantile2:%s" % (data[n].quantile(0.5)), file=doc)
        print("quantile3:%s" % (data[n].quantile(0.75)), file=doc)
        print("theNumberOfNull:%s" % (data[n].count()), file=doc)
        doc.close
```

求出的部分结果如下所示：

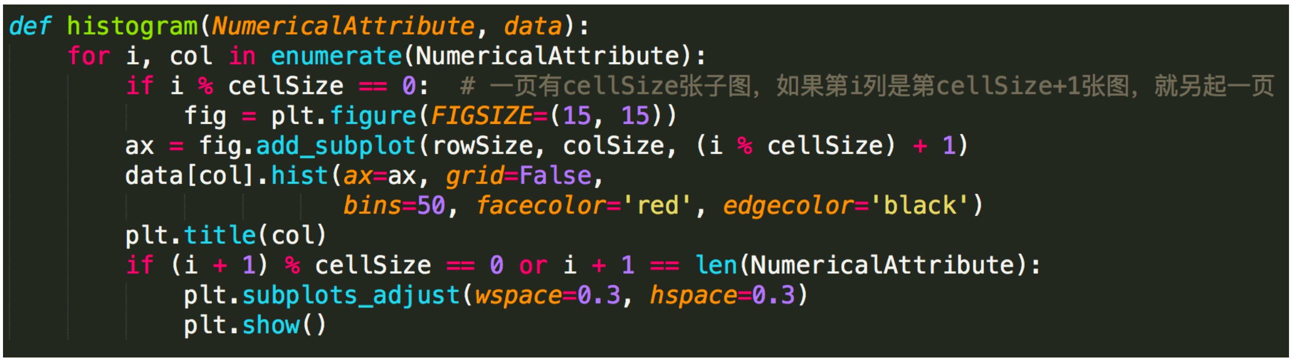


```
*************** PlayTimeDiff ***************
max:943.0
min:0.0
mean:20.576762334128926
median:17.0
quantile1:5.0
quantile2:17.0
quantile3:37.0
theNumberOfNull:407244
*************** yrdln ***************
max:50.0
min:1.0
mean:28.48832733600755
median:30.0
quantile1:20.0
quantile2:30.0
quantile3:39.0
theNumberOfNull:406848
*************** yrdline100 ***************
max:99.0
min:1.0
mean:48.644080836086204
median:49.0
quantile1:30.0
quantile2:49.0
quantile3:70.0
theNumberOfNull:406848
```

数据的可视化

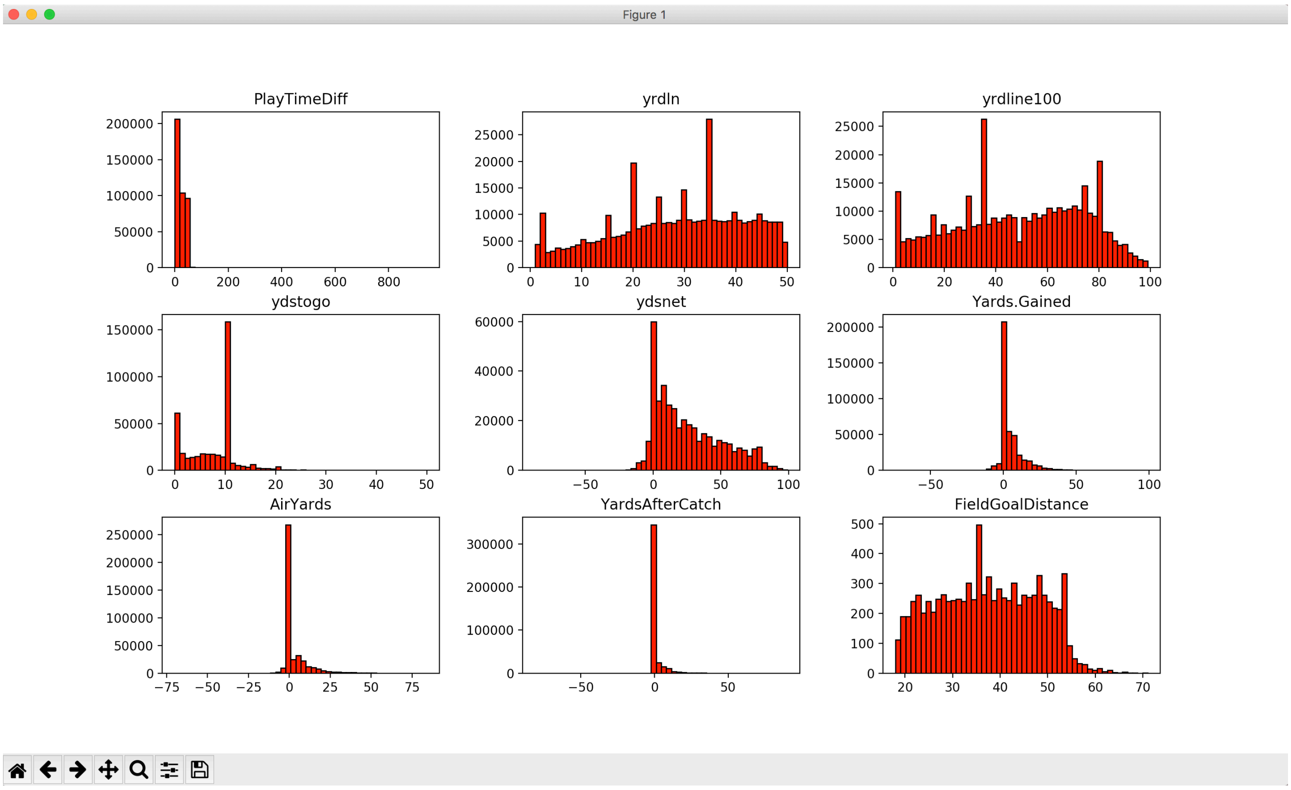
1.直方图

对每种数值属性绘制直方图，代码如下



```
def histogram(NumericalAttribute, data):
    for i, col in enumerate(NumericalAttribute):
        if i % cellSize == 0:  # 一页有cellSize张子图，如果第i列是第cellSize+1张图，就另起一页
            fig = plt.figure(FIGSIZE=(15, 15))
        ax = fig.add_subplot(rowSize, colSize, (i % cellSize) + 1)
        data[col].hist(ax=ax, grid=False,
                       bins=50, facecolor='red', edgecolor='black')
        plt.title(col)
        if (i + 1) % cellSize == 0 or i + 1 == len(NumericalAttribute):
            plt.subplots_adjust(wspace=0.3, hspace=0.3)
            plt.show()
```

绘制出的部分直方图如下所示：



PlayTimeDiff

yrdln

yrdline100

ydstogo

ydsnet

Yards.Gained

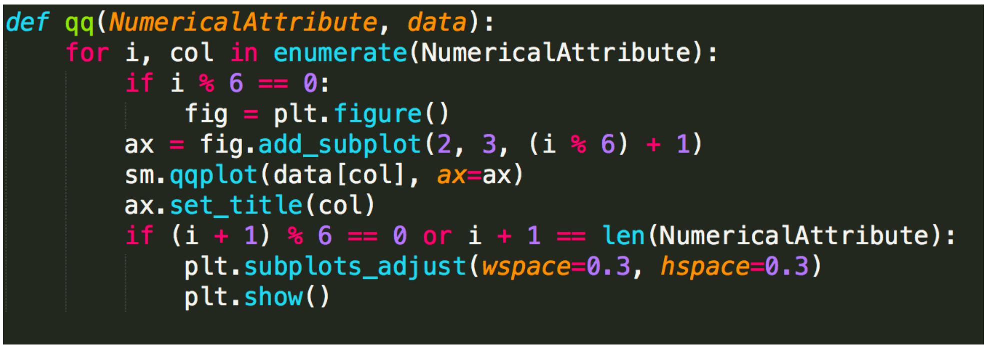
AirYards

YardsAfterCatch

FieldGoalDistance

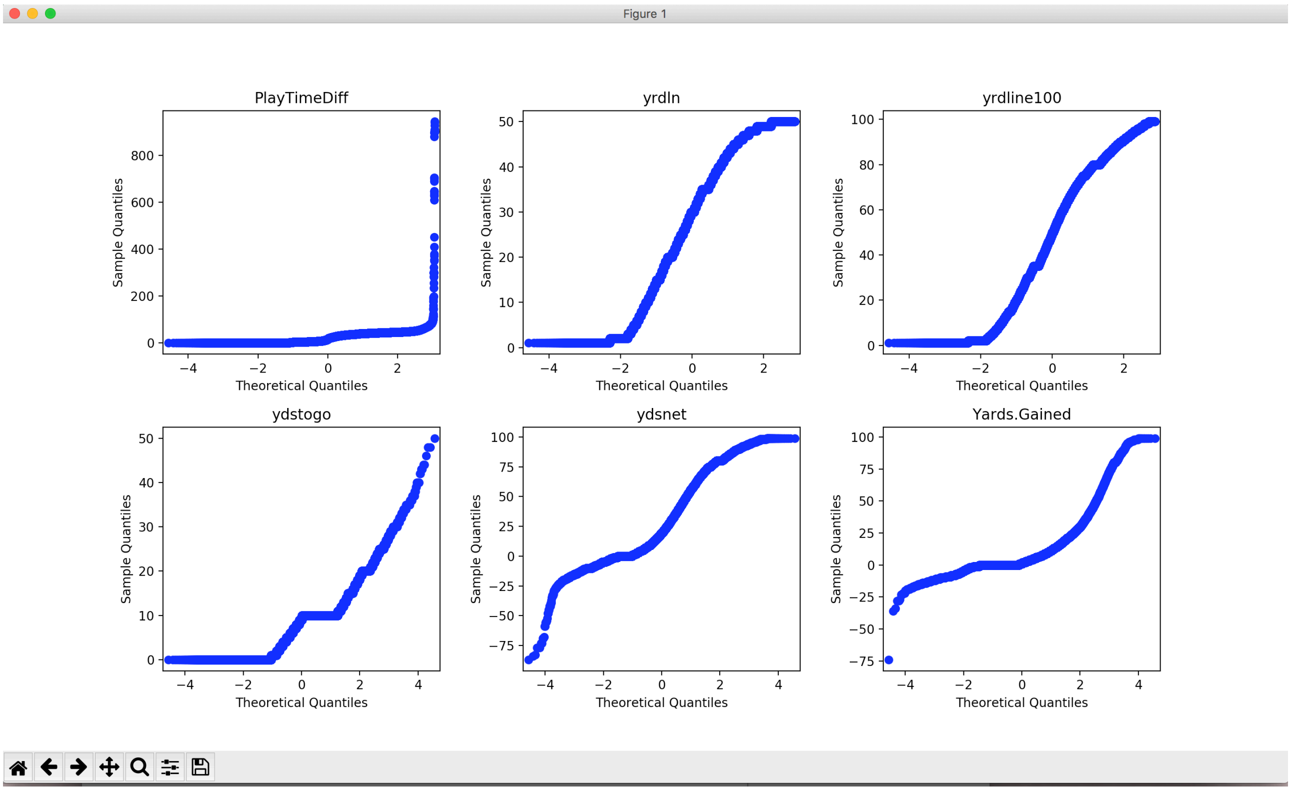
2.qq图

绘制qq图的代码如下所示：



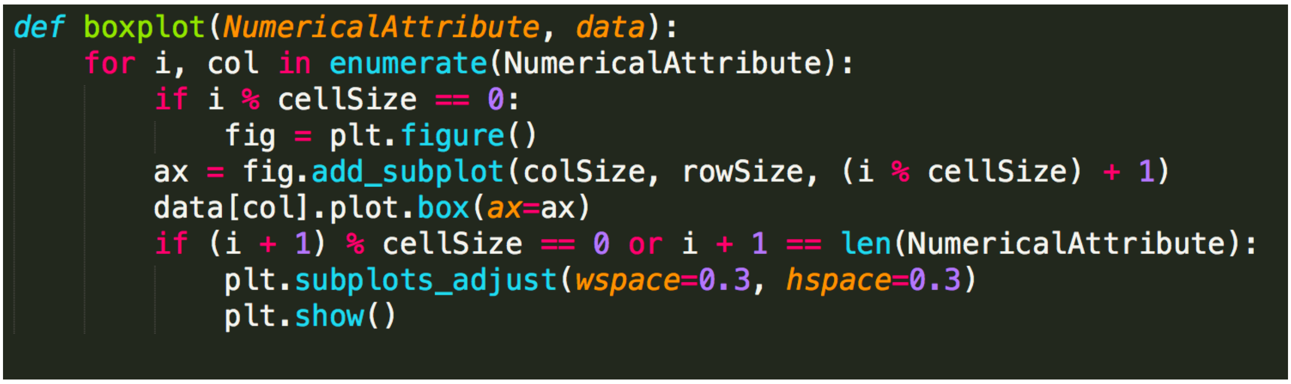
```
def qq(NumericalAttribute, data):
    for i, col in enumerate(NumericalAttribute):
        if i % 6 == 0:
            fig = plt.figure()
        ax = fig.add_subplot(2, 3, (i % 6) + 1)
        sm.qqplot(data[col], ax=ax)
        ax.set_title(col)
        if (i + 1) % 6 == 0 or i + 1 == len(NumericalAttribute):
            plt.subplots_adjust(wspace=0.3, hspace=0.3)
            plt.show()
```

绘制出的部分qq图如下，可以直观地利用qq图检验分布是否为正态分布



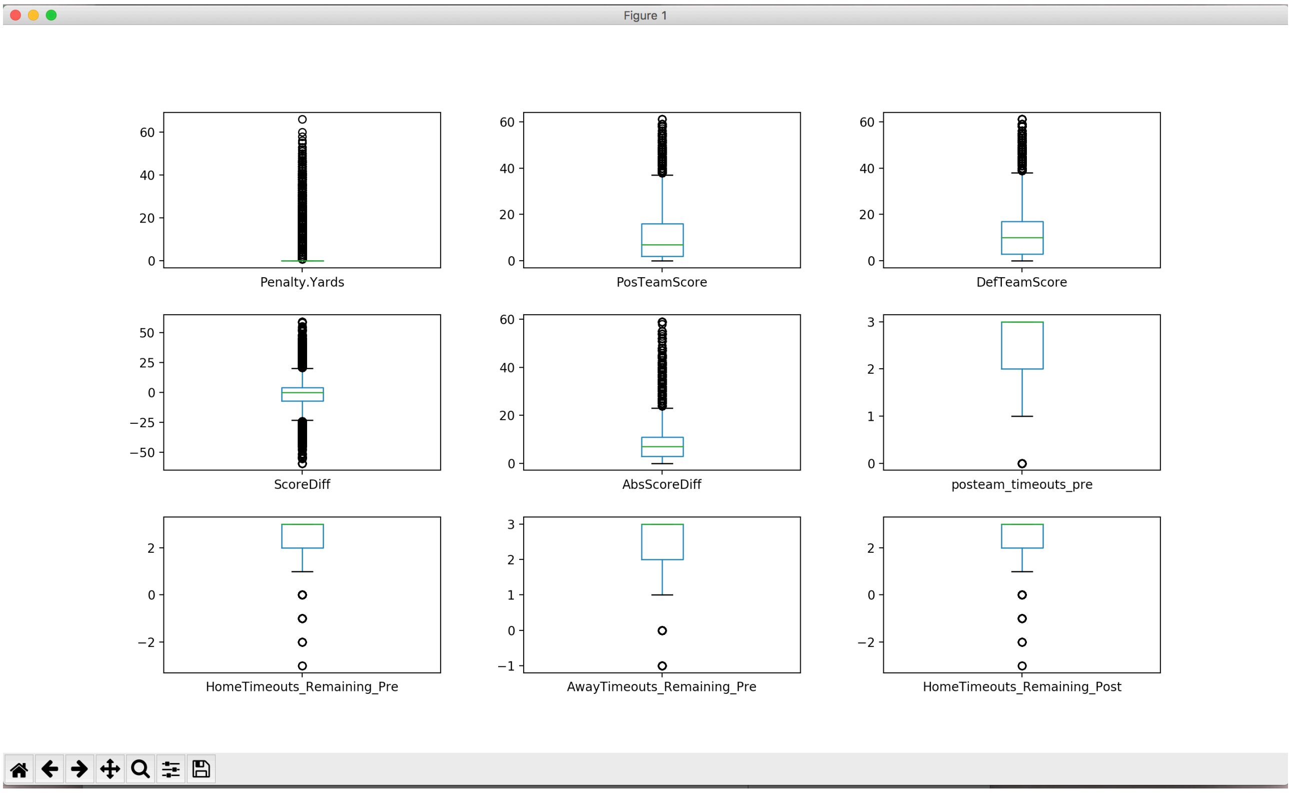
3.盒图

绘制盒图的代码如下：



```
def boxplot(NumericalAttribute, data):
    for i, col in enumerate(NumericalAttribute):
        if i % cellSize == 0:
            fig = plt.figure()
        ax = fig.add_subplot(colSize, rowSize, (i % cellSize) + 1)
        data[col].plot.box(ax=ax)
        if (i + 1) % cellSize == 0 or i + 1 == len(NumericalAttribute):
            plt.subplots_adjust(wspace=0.3, hspace=0.3)
            plt.show()
```

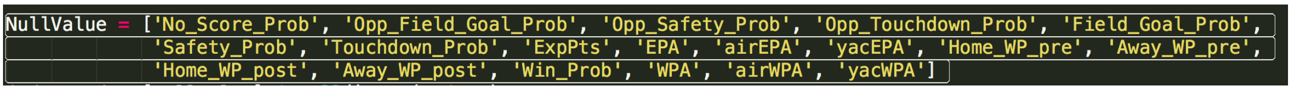
绘制出的部分盒图如下，可以直观的观察到离群值的分布



缺失数据的处理

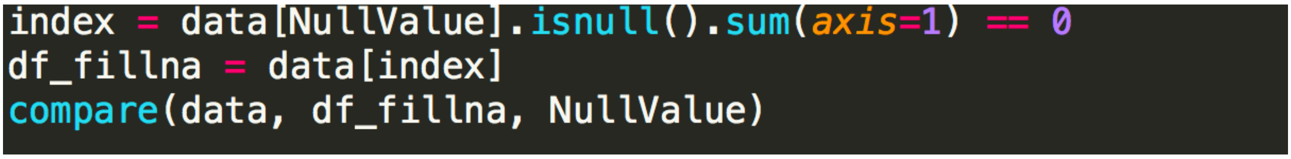
1.将缺失部分剔除

可以进行填充的字段有如下：



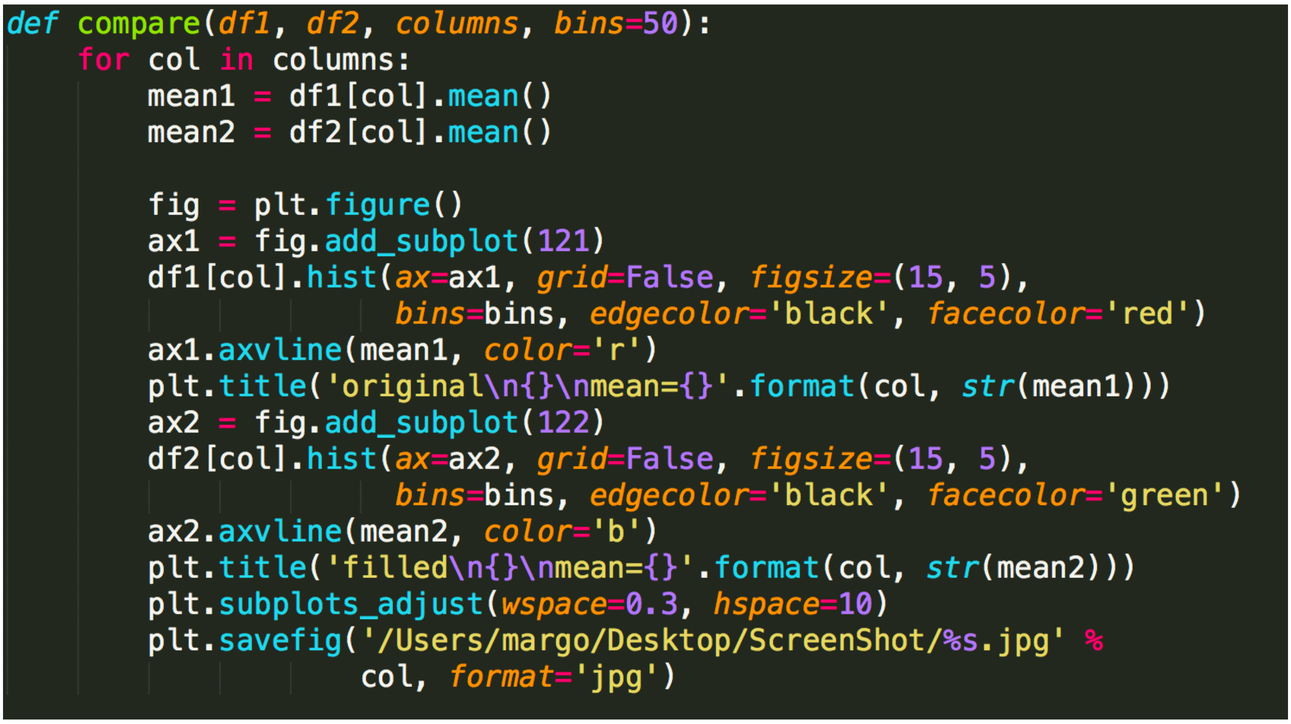
```
NullValue = ['No_Score_Prob', 'Opp_Field_Goal_Prob', 'Opp_Safety_Prob', 'Opp_Touchdown_Prob', 'Field_Goal_Prob',
             'Safety_Prob', 'Touchdown_Prob', 'ExpPts', 'EPA', 'airEPA', 'yacEPA', 'Home_WP_pre', 'Away_WP_pre',
             'Home_WP_post', 'Away_WP_post', 'Win_Prob', 'WPA', 'airWPA', 'yacWPA']
```

剔除缺失数据的代码如下：



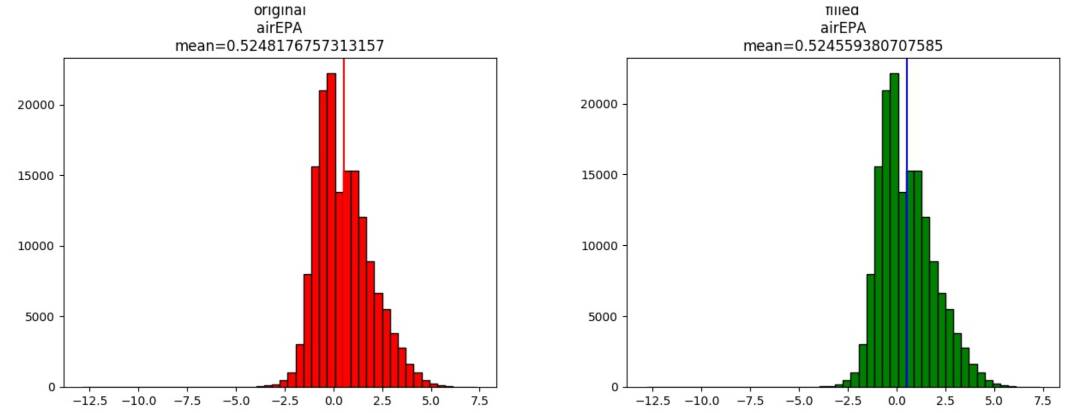
```
index = data[NullValue].isnull().sum(axis=1) == 0
df_fillna = data[index]
compare(data, df_fillna, NullValue)
```

将剔除之后的数据与剔除前的数据进行直方图的比较，代码如下：



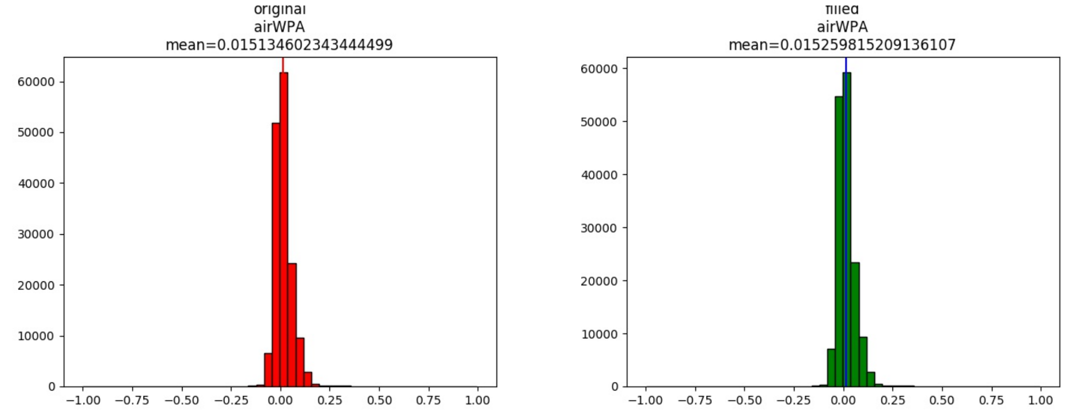
```python
def compare(df1, df2, columns, bins=50):
    for col in columns:
        mean1 = df1[col].mean()
        mean2 = df2[col].mean()

        fig = plt.figure()
        ax1 = fig.add_subplot(121)
        df1[col].hist(ax=ax1, grid=False, figsize=(15, 5),
                      bins=bins, edgecolor='black', facecolor='red')
        ax1.axvline(mean1, color='r')
        plt.title('original\n{}\nmean={}'.format(col, str(mean1)))
        ax2 = fig.add_subplot(122)
        df2[col].hist(ax=ax2, grid=False, figsize=(15, 5),
                      bins=bins, edgecolor='black', facecolor='green')
        ax2.axvline(mean2, color='b')
        plt.title('filled\n{}\nmean={}'.format(col, str(mean2)))
        plt.subplots_adjust(wspace=0.3, hspace=10)
        plt.savefig('/Users/margo/Desktop/ScreenShot/%s.jpg' %
                       col, format='jpg')
```

比较的部分直方图如下，其中左边的红线为原有数据集的均值，右边的蓝线为剔除后的均值



original
airEPA
mean=0.5248176757313157

filled
airEPA
mean=0.524559380707585

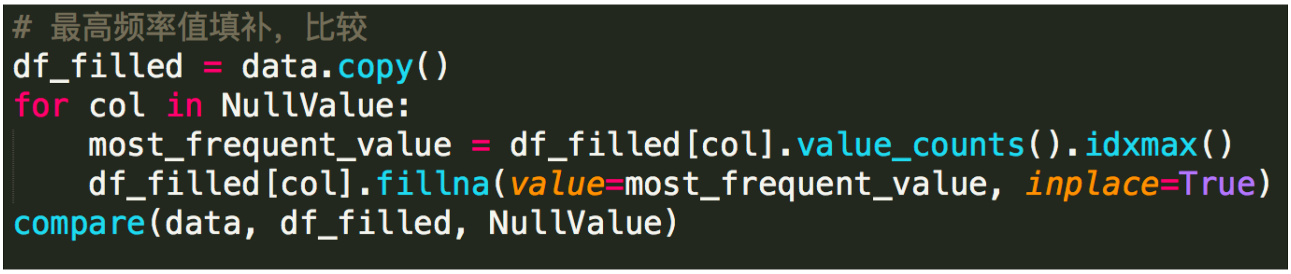


original
airWPA
mean=0.015134602343444499

filled
airWPA
mean=0.015259815209136107

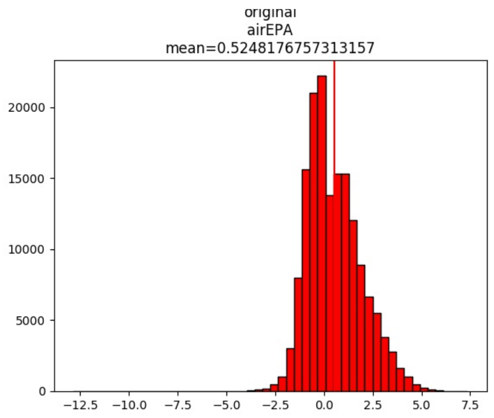
2.用最高频率值填补缺失数据

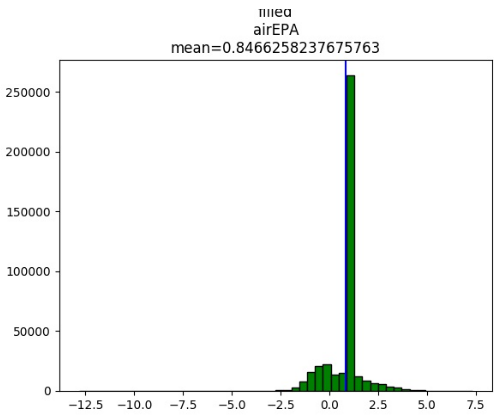
用最高频率值填补缺失数据的代码如下：

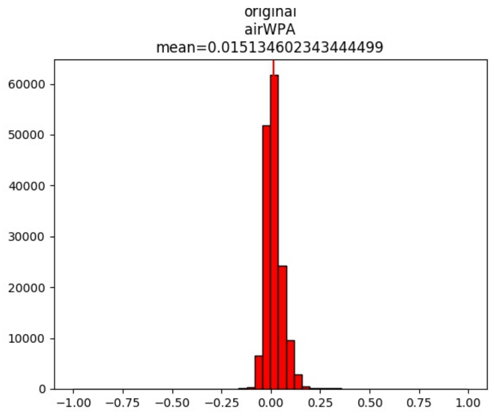


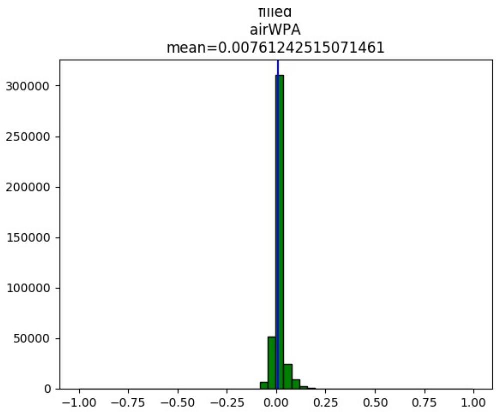
```
# 最高频率值填补，比较
df_filled = data.copy()
for col in NullValue:
    most_frequent_value = df_filled[col].value_counts().idxmax()
    df_filled[col].fillna(value=most_frequent_value, inplace=True)
compare(data, df_filled, NullValue)
```

比较的部分直方图如下，其中左边的红线为原有数据集的均值，右边的蓝线为新数据的均值



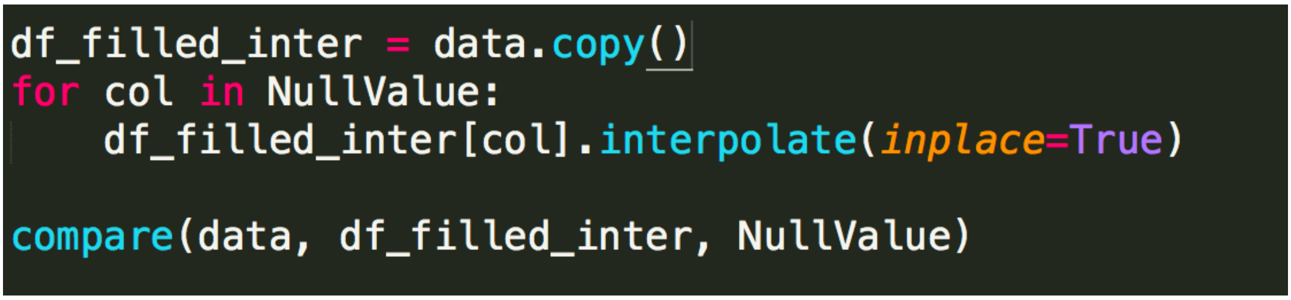






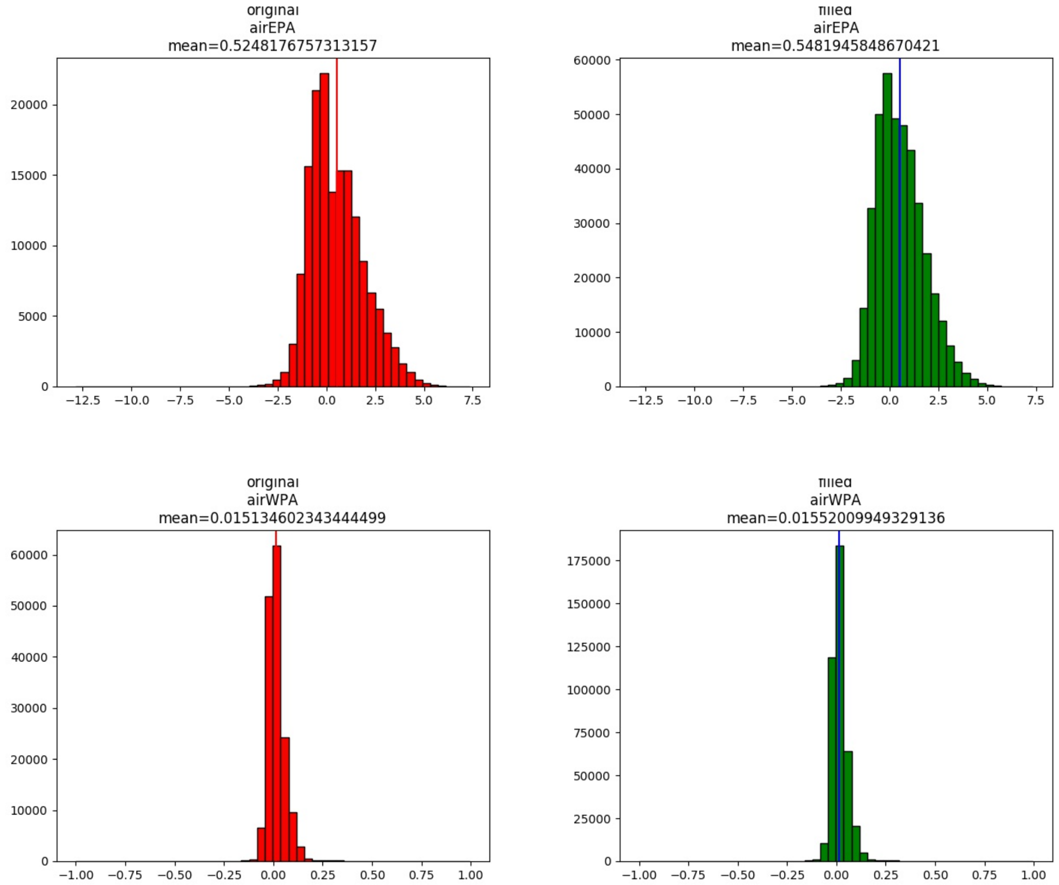
3.通过属性的相关关系来填补缺失值

通过属性的相关关系来填补缺失值的代码如下：



```
df_filled_inter = data.copy()
for col in NullValue:
    df_filled_inter[col].interpolate(inplace=True)

compare(data, df_filled_inter, NullValue)
```

得到的对比直方图如下所示，其中左边的红线为原有数据集的均值，右边的蓝线为新数据的均值


original
airEPA
mean=0.5248176757313157
filled
airEPA
mean=0.5481945848670421
original
airWPA
mean=0.015134602343444499
filled
airWPA
mean=0.01552009949329136