Introduction to pyecharts

Author: Shihang Wang, Jingyuan Wang

Pyecharts is a powerful data visualization tool which combines 'python' and 'echarts'. In this article, we will explain detailed usage rules for pyecharts.

preface

We all know Matplotlib, a visualization tool on python. A few days ago, when we did a spark project, we used ecarts, a visualization JS tool open source by Baidu. There are many types of visualization, but we have to import JS library to run on Java Web project. We tend to use Python more, so we wonder if there is a combination of Python and ecarts.

Installation

Firstly open the terminal and input:

```
pip install pyecharts
```

However, we found that due to the firewall, the download will be broken and slow, leading to download failure. So it is recommended to download through Tsinghua mirror. After we download successfully, we can go to next step.

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple pyecharts
```

Usage Examples

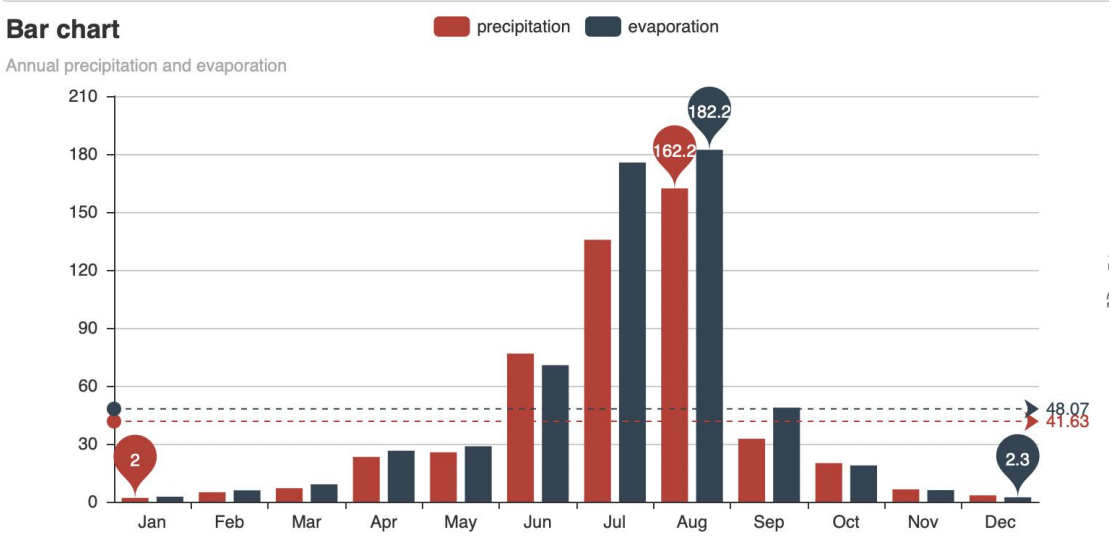
Before using, we want to emphasize one point: the encoding problem of python2.X and python3.X. You can regard it as the default Unicode encoding in python 3.X, but it is not the default in python2.X. The reason is that its bytes object definition is confusing, and pycharts uses unicode encoding to handle strings and files. So when you use python2.X, be sure to insert this code:

```
from __future__ import unicode_literals
```

Now let's officially use pycharts. Here we directly use the official data:

Bar chart

```
from pyecharts import Bar
## column
columns = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct",
           "Nov", "Dec"]
## data
data1 = [2.0, 4.9, 7.0, 23.2, 25.6, 76.7, 135.6, 162.2, 32.6, 20.0, 6.4, 3.3]
data2 = [2.6, 5.9, 9.0, 26.4, 28.7, 70.7, 175.6, 182.2, 48.7, 18.8, 6.0, 2.3]
## title
bar = Bar("Bar chart", "Annual precipitation and evaporation")
## Add data and configuration items of histogram
bar.add("precipitation", columns, data1, mark_line=["average"], mark_point=["max", "min"])
bar.add("evaporation", columns, data2, mark_line=["average"], mark_point=["max", "min"])
#Render will generate a local HTML file, which is generated in the
#current directory by default render.html
# bar.render()
#You can pass in path parameters such as bar.render (" mycharts.html ")
#You can output graphics in jupyter, such as bar.render_ notebook()
bar
```

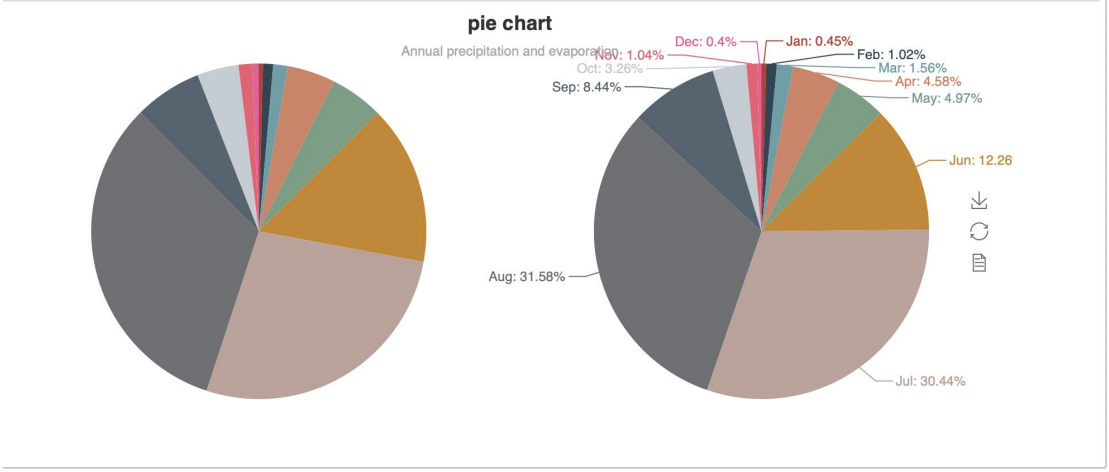


A few simple lines of code can visualize the data very well, and it's still dynamic. Here we recommend jupyter. Pyecharts starts with version v0.1.9.2. You can directly represent the chart by calling the instance on jupyter (for example, calling bar directly above), which is very convenient.

There are more than 20 kinds of charts supported by pyecharts. Next, we use the data above to generate several common charts for data mining.

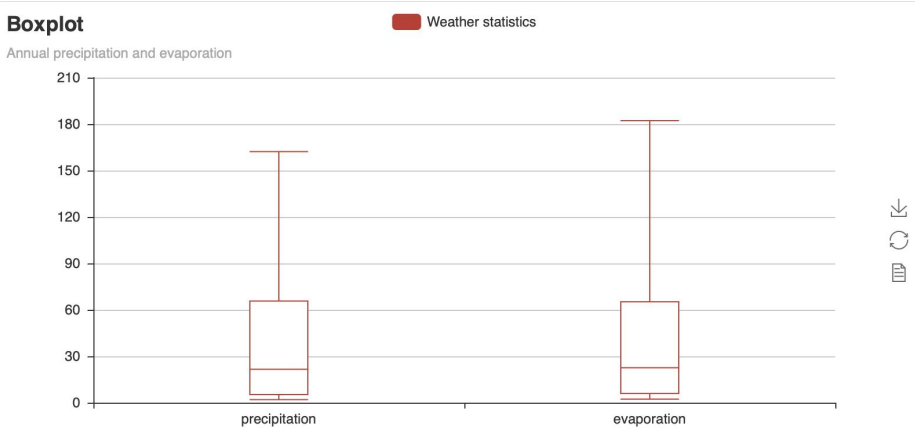
Pie chart

```
from pyecharts import Pie
## Set the main title and subtitle;set the center of the title and set the width to 900
pie = Pie("pie chart", "Annual precipitation and evaporation",title_pos='center',width=900)
## Add the data, set the coordinate position to [25, 50]
# and the colors option above will not be displayed.
pie.add("precipitation", columns, data1 ,center=[25,50],is_legend_show=False)
## Add the data, set the coordinate position to [75, 50]
# and the colors option above will not be displayed.
pie.add("evaoration", columns, data2 ,center=[75,50],is_legend_show=False,is_label_show=True
pie.render_notebook()
```



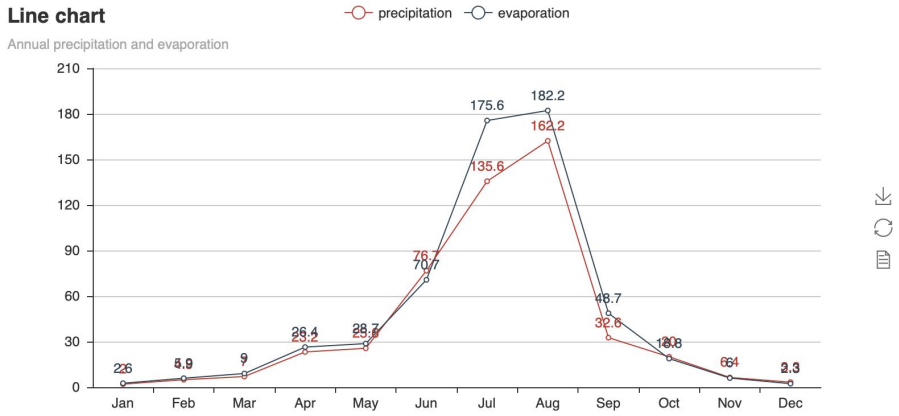
Boxplot

```
from pyecharts import Boxplot
boxplot = Boxplot("Boxplot", "Annual precipitation and evaporation")
x_axis = ['precipitation','evaporation']
y_axis = [data1,data2]
## prepare_data method can convert data into nested interval [min, Q1, median (or Q2), Q3, Max].
yaxis = boxplot.prepare_data(y_axis)
boxplot.add("Weather statistics", x_axis, yaxis)
boxplot
```



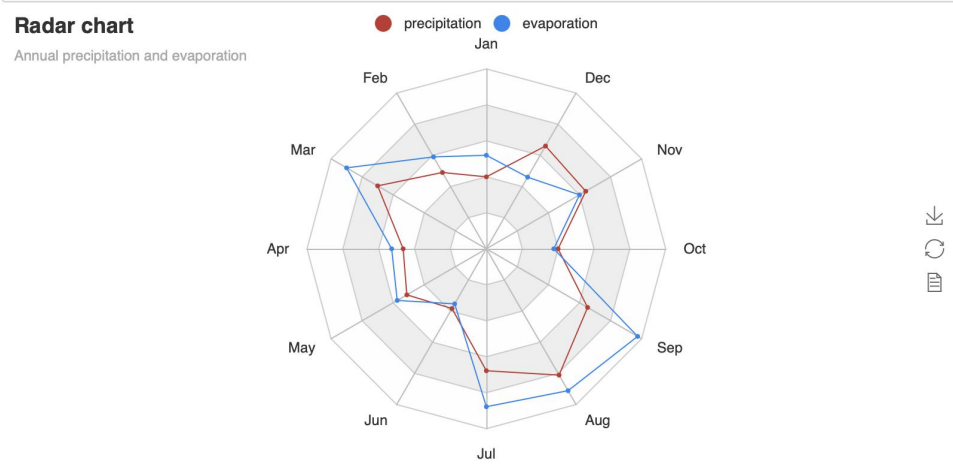
Line chart

```
from pyecharts import Line
line = Line("Line chart","Annual precipitation and evaporation")
## is_ label_ Show is to set whether the data above is displayed
line.add("precipitation", columns, data1, is_label_show=True)
line.add("evaporation", columns, data2, is_label_show=True)
line
```



Radar chart

```
from pyecharts import Radar
radar = Radar("Radar chart", "Annual precipitation and evaporation")
## Since the data of radar chart must be multidimensional data, we need to do some pre-processing here
radar_data1 = [[2.0, 4.9, 7.0, 23.2, 25.6, 76.7, 135.6, 162.2, 32.6, 20.0, 6.4, 3.3]]
radar_data2 = [[2.6, 5.9, 9.0, 26.4, 28.7, 70.7, 175.6, 182.2, 48.7, 18.8, 6.0, 2.3]]
## Set the maximum value of column. In order to make the radar chart more intuitive,
#the maximum value of month here is set differently.
schema = [
    ("Jan", 5), ("Feb",10), ("Mar", 10),
    ("Apr", 50), ("May", 50), ("Jun", 200),
    ("Jul", 200), ("Aug", 200), ("Sep", 50),
    ("Oct", 50), ("Nov", 10), ("Dec", 5)
]
radar.config(schema)
radar.add("precipitation",radar_data1)
## Generally, it is the same color by default.
# Here, in order to distinguish easily, we need to set the color of different item
radar.add("evaporation",radar_data2,item_color="#1C86EE")
radar
```



Scatter chart

```
from pyecharts import Scatter
scatter = Scatter("Scatter chart", "Annual precipitation and evaporation")
## xais_name is to set the horizontal ordinate name. Here due to the display issue,
#we need to set the distance for the y-axis.
scatter.add("Scatter distribution of precipitation and evaporation",
            data1,data2,xaxis_name="precipitation",yaxis_name="evaporation",
            yaxis_name_gap=40)
scatter
```

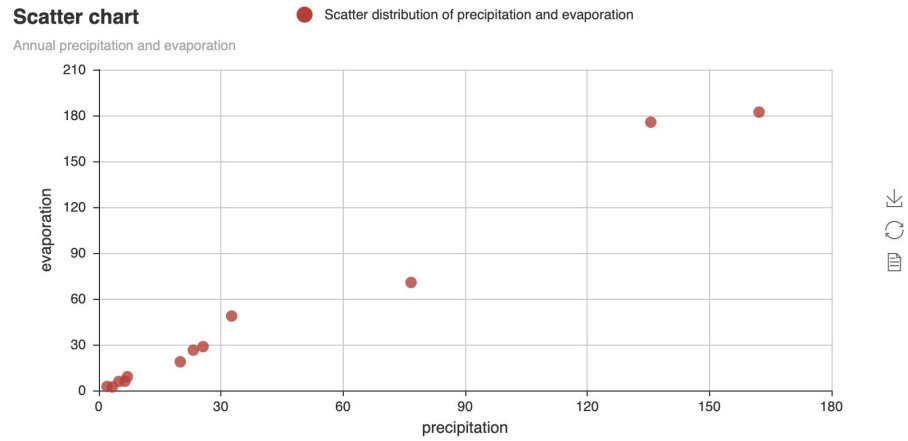
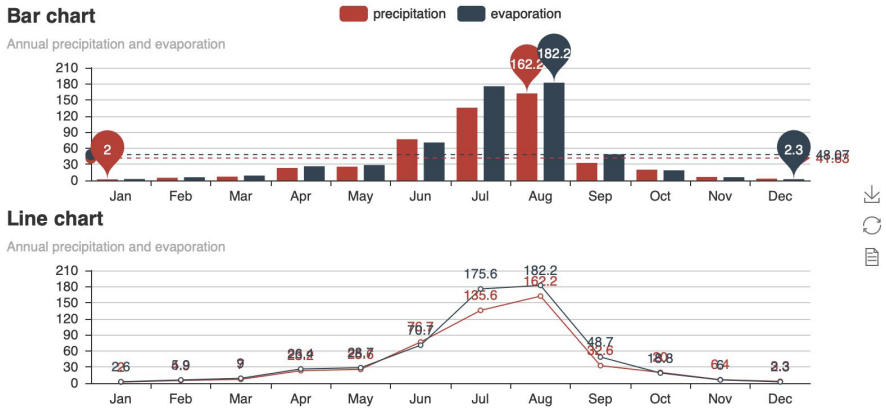


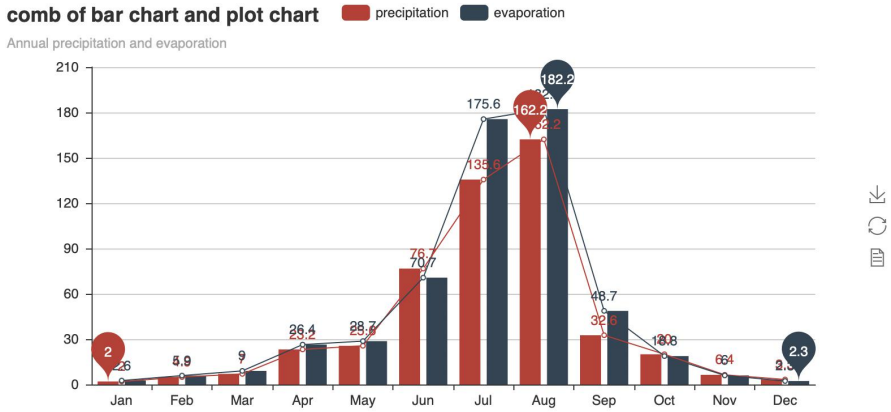
Chart layout: combination of graphs

As the title and chart belong to different controls, the title position of the chart line must be set, otherwise the titles may be overlapped.

```
from pyecharts import Grid
## Set line chart title position
line = Line("Line chart","Annual precipitation and evaporation",title_top="45%")
line.add("precipitation", columns, data1, is_label_show=True)
line.add("evaporation", columns, data2, is_label_show=True)
grid = Grid()
## Set the relative position of the two charts
grid.add(bar, grid_bottom="60%")
grid.add(line, grid_top="60%")
grid
```




```
from pyecharts import Overlap
overlap = Overlap()
bar = Bar("comb of bar chart and plot chart", "Annual precipitation and evaporation")
bar.add("precipitation", columns, data1, mark_point=["max", "min"])
bar.add("evaporation", columns, data2, mark_point=["max", "min"])
overlap.add(bar)
overlap.add(line)
overlap
```



Conclusion

1. Import related chart package.
2. Set basic information of the chart and create chart objects.
3. Use the add ( ) method for data input and chart setting (You can use print_echarts_ options( ) to output all configurable items).
4. Use render ( ) method to save chart.

A Real-World Case:

In this case, we will implement Pyecharts along with some auxiliary tools to have an analysis of the author's WeChat Friends (Under the Web Crawler license with all sensitive information eliminated).

Firstly, install the packages we need to use:

```
! pip install pyecharts==0.5.11
! pip install pyecharts-snapshot
! pip install itchat
! pip install jieba
! pip install wordcloud
```

Then, import the required packages:

```
import itchat
import re
import jieba
import matplotlib.pyplot as plt
from wordcloud import WordCloud,ImageColorGenerator
import numpy as np
import PIL.Image as Image
from os import path
from imageio import imread
```

Login the author's WeChat account via "itchat" method:

```
itchat.auto_login()#Scan the QR code and log in
friends = itchat.get_friends(update = True)#get WeChat friends' available information
```

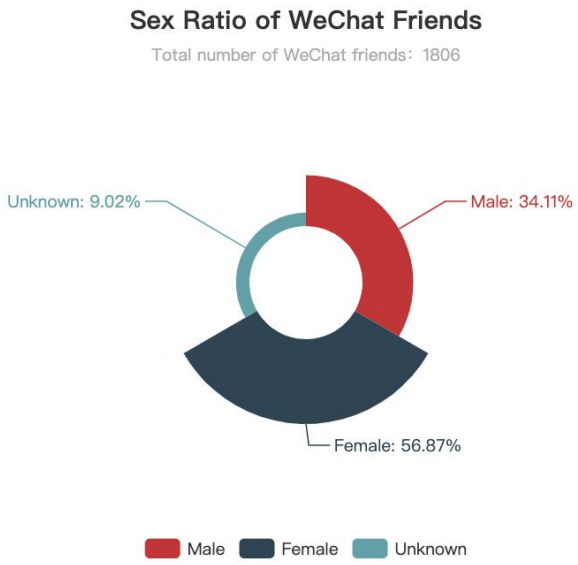
```
Getting uuid of QR code.
Downloading QR code.
Please scan the QR code to log in.
Please press confirm on your phone.
Loading the contact, this may take a little while.
Login successfully as Nucky
```

Apply "pandas" to integrate all the available information in a DataFrame.

```
import pandas as pd
sex=list(map(lambda x:x['Sex'],friends[1:]))
nickname=list(map(lambda x:x['NickName'],friends[1:]))
signature=list(map(lambda x:x['Signature'],friends[1:]))
province=list(map(lambda x:x['Province'],friends[1:]))
city=list(map(lambda x:x['City'],friends[1:]))
headimg=list(map(lambda x:x['HeadImgUrl'],friends[1:]))
info={"nickname":nickname,"sex":sex,"province":province,"city":city,"signature":signature,'headimgurl':headimg}
data=pd.DataFrame(info)
data.head(10)
```

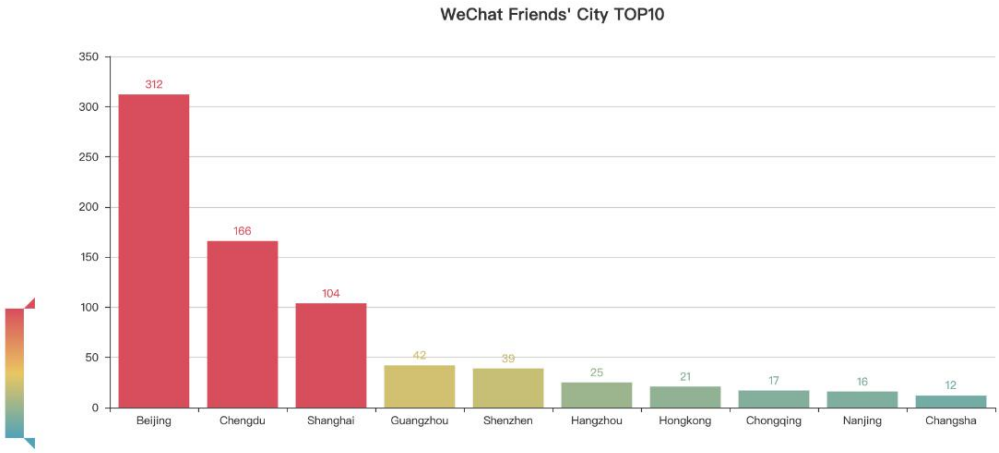
Now, we can draw the plots with the DataFrame. First, let's analysis the distribution of the author's WeChat friends' gender. Here we use the "Nightingale Rose Diagram", where parts are separated with equal angle and area is the indication of proportion. We can observe that the author has more female friends than male friends, and there are also 9.02% of friends who don't fill in their gender in WeChat (In WeChat, there is only 2 gender option, Male and Female):

```
from pyecharts import Pie
sexs=list(map(lambda x:x['Sex'],friends[1:])) #Get the gender of Wechat friends
value = [sexs.count(1), sexs.count(2), sexs.count(0)]#Count the amount of each gender
sex_attr=['Male','Female','Unknown']
pie = Pie('Sex Ratio of WeChat Friends', 'Total number of WeChat friends: %d' % len(sex), title_pos='center')
pie.add('', sex_attr, value, radius=[20, 50], rosetype='area', is_label_show=True,
        is_legend_show=True, legend_top='bottom')#Here we use "rosetype='area'" to get Nightingale rose diagram
pie
```



Then, we analysis the Top10 cities that the author's friends located in with a bar plot. We observe that friends in the author's two working place (Beijing & Shanghai) and hometown (Chengdu) make up a large proportion of the author's WeChat friends. Besides, the rest 7 cities of Top10 are all the so called "first-tier" cities (developed cities) in China.

```
from pyecharts import Bar
from collections import Counter
city_10=Counter(city).most_common(10)
# Return the top 10 cities with highest frequency
bar = Bar("WeChat Friends' City Top10", '', title_pos='center', width=1000, height=500)
attr, value = bar.cast(city_10)
bar.add('', attr, value, is_visualmap=True, visual_text_color='#fff', is_label_show=True)
bar
```



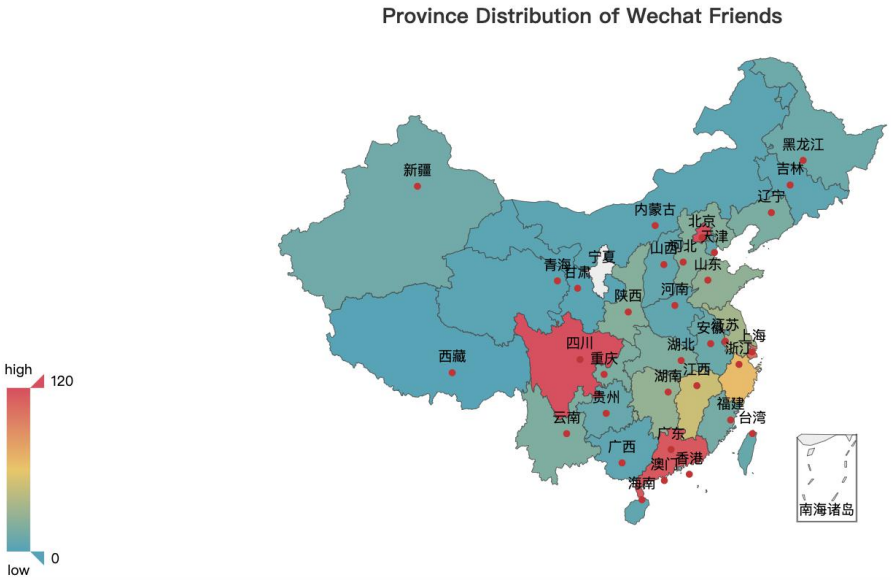
To make the location distribution more intuitive to read, we draw them on Chinese Maps. Note that we need to download the maps we would use before call the "Map" method. Here we download the maps:

```
! pip install echarts-countries-pypkg
! pip install echarts-china-provinces-pypkg
! pip install echarts-china-cities-pypkg
! pip install echarts-china-counties-pypkg
```

Then we can be obtained the distribution on Chinese Map with the following code:

```
from pyecharts import Map
#Get the province frequency via "groupby" and "count" method for DataFrame
provinces_count = data.groupby('province', as_index=True)['province'].count().sort_values()
#If the location is not filled in, it will be changed to unknown
attr = list(map(lambda x: x if x != '' else 'unknown', list(provinces_count.index)))
value = list(provinces_count)
#Draw the Map for WeChat friends located in China
map_1 = Map("Province Distribution of Wechat Friends",title_pos="center",width=1000, height=500)
#Set "visual_range" to [0,120], making the provinces more differential to read
map_1.add('', attr, value, is_label_show=True, is_visualmap=True, visual_text_color='#000',visual_range=[0,120])
map_1
```

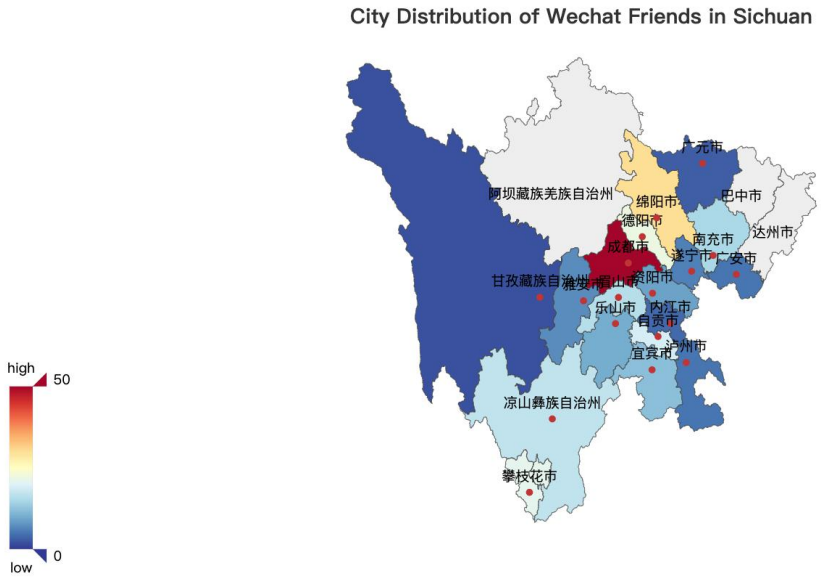
The Map is almost aligned with the bar plot we obtain above, we see that many friends located in Sichuan, Beijing, Shanghai and Guangdong. And the author's friends are also located in the provinces adjacent to the three developed Cities/Province (Beijing, Shanghai and Guangdong). These provinces are shown in "yellow" color on the map, and the author has never been to some of them before, indicating the tendency that people in the nearby provinces are incline to work or study in the more developed places (The author also worked as an intern in those 3 cities/ provinces, thus the majority of friends were added there).



To give Sichuan Province as an example, we can also show how the WeChat friends are located at the City-Level. Instead of the default color range, we can also set our own color range, e.g. from cold color to warm color by setting "range_color".

```
data_sc=data[data['province']=='四川']#Select friends whose province is Sichuan
cities_count = data_sc.groupby('city', as_index=True)['city'].count().sort_values()#Same method as above
attr = list(map(lambda x: x if x != '' else 'unknown', list(cities_count.index)))
value = list(cities_count)
map_2 = Map("City Distribution of Wechat Friends in Sichuan",title_pos="center",width=1000, height=500)
#instead of the default color range, we can also set our own color range, e.g. from cold color to warm color
range_color = ['#313695', '#4575b4', '#74add1', '#abd9e9', '#e0f3f8', '#ffffbf',
               '#fee090', '#fdae61', '#f46d43', '#d73027', '#a50026']
#For Provincial level map, we need to assign the province name via "maptype='四川'(SiChuan)"
map_2.add('', attr, value, is_label_show=True, is_visualmap=True, maptype='四川',
          visual_text_color='#000',visual_range=[0,50], visual_range_color=range_color)
map_2
```

The graph shows that the majority of the author's WeChat friends in Sichuan located in Chengdu (the author's hometown) and Chengdu's adjacent cities. Since the majority of the friends in Sichuan are added during the author's study in university (top 1 university in Sichuan Province), so this graph may also indicate the education qualities of different cities in Sichuan to some certain extent.



Now, we implement the “jieba”, “wordcloud” and “snownlp” tools to analysis the signatures of the author’s friends. Firstly, we draw a word cloud of signatures with the following code:

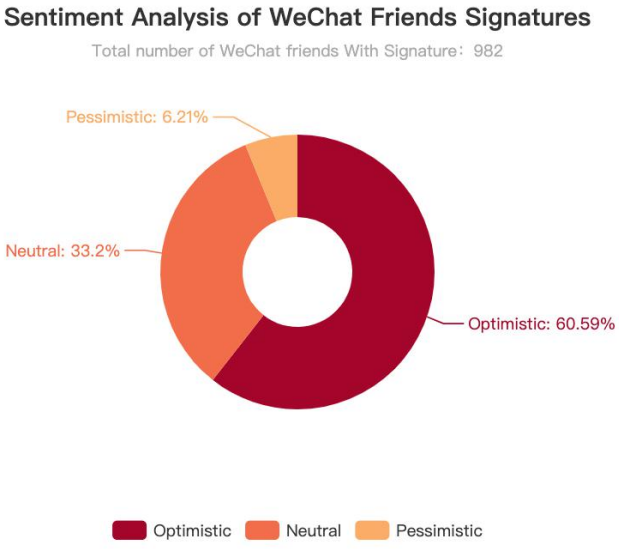
```
import wordcloud# Use wordcloud package to draw the word cloud plot of WeChat friends' signatures
import jieba.analyse# "jieba" is a very popular package for word cutting
# "snownlp" is also a very popular package for text analysis, here we use it to get the emotion of signatures
import snownlp
plt.rcParams['font.sans-serif']=['SimHei']#Chinese can be displayed when drawing word cloud plot
plt.rcParams['axes.unicode_minus']=False#Chinese can be displayed when drawing word cloud plot

signatures = ''
emotions = []
for friend in friends:
    signature = friend['Signature']
    if signature != None:
        # Remove irrelevant content
        signature = signature.strip().replace("span","").replace("class","").replace("emoji","")
        signature = re.sub(r'1f(\d.+)',"",signature)

    # Only signature with length larger than 0 would be counted
    if len(signature) > 0:
        nlp = snownlp.SnowNLP(signature)
        emotions.append(nlp.sentiments)#nlp.sentiments: the probability that a signature tend to be optimistic
        signatures += " ".join(jieba.analyse.extract_tags(signature,5)) #get words from signatures via "jieba"
#We can use any plot as base plot to define the shape of word cloud plot
back_coloring = np.array(Image.open("/Users/wangshihang01/Downloads/Study/Wechat_logo.jpg"))
word_cloud2 = WordCloud(font_path = 'zt.ttf',
                        background_color = 'white',
                        max_words = 1200,
                        mask = back_coloring,
                        margin = 15)
word_cloud2.generate(signatures)
image_colors = ImageColorGenerator(back_coloring)
plt.figure(figsize=(6,5),dpi=160)
plt.imshow(word_cloud2.recolor(color_func=image_colors))
plt.axis("off")
plt.show()
word_cloud2.to_file("signatures.jpg")
```

The word cloud is displayed with the WeChat logo’s shape. It shows that the majorities of words are those with positive meaning, for instance, “Freedom”, “Tender”, “Like”, “Friends” etc. However, more works should be conduct to make more valid analysis.

Then, with the sentiment analysis results of "snownlp", the author can also display the distribution of the friends' sentiment (optimistic, pessimistic, neutral). (We can also customize our graph's color via "label_color" here)

```
#Signature Sentiment
count_optimistic = len(list(filter(lambda x:x>0.66,emotions)))#Optimistic if the emo prob larger than 0.66
count_pessimistic = len(list(filter(lambda x:x<0.33,emotions)))#Pessimistic if the emo prob less than 0.33
count_neutral = len(list(filter(lambda x:x>=0.33 and x <= 0.66,emotions)))# else, neutral
value=[count_optimistic,count_neutral,count_pessimistic]
attr=['Optimistic','Neutral','Pessimistic']
pie = Pie('Sentiment Analysis of WeChat Friends Signatures',
          'Total number of WeChat friends With Signature: %d' % sum(value),
          title_pos='center')
# with "rosetype=None", a simple pie chart instead of Nightingale rose diagram is obtained
# here instead of using the default colors, we can also set our own colors via "label_color="
pie.add('', attr, value, radius=[20, 50], rosetype=None, is_label_show=True,
        is_legend_show=True, legend_top='bottom',label_color=['#a50026', '#f46d43', '#fdae61'])
pie
```

This result is aligned with the word cloud we obtained above, the majority of WeChat friends select an optimistic text as their signature, and only a little friends use pessimistic content as their own signature.



With this case study above, the readers can have a clear view on how Pyecharts can be applied along with other popular Python tools to conduct interesting and valuable analysis in our academic and daily life.

Reference:

https://www.jianshu.com/p/554d64470ec9
https://github.com/pyecharts/pyecharts
https://05x-docs.pyecharts.org/#/zh-cn/prepare