Python 图表利器 pyecharts

原创 派森酱 Python技术 昨天

文 | 豆豆

来源：Python 技术「ID: pythonall」

随着互联网的高速发展，数据量也在疯狂增长，近几年数据分析，数据挖掘的岗位越来越吃香。说到数据分析，就离不开数据的可视化，毕竟图表比冷冰冰的数字直观，一眼就可以看出趋势和结论，毕竟一图胜千言。

而 Python 作为数据分析的主力语言，自然也有不少可视化的类库，比如 matplotlib，常用的柱状图、散点图、折线图都可以生成。但如果想在网页端展示的话就显得有些捉襟见肘了。

做过 web 端数据可视化的基本都知道 Echarts 这个库，这是由百度开源的数据可视化类库。讲真，虽然我对百度这个企业没有一点好感，但这款工具确实好用，咱一码归一码，不能因为不喜欢百度就全盘否定百度的一切产品。其凭借着良好的交互性，精美的图表设计，以及开发者容易接入等优点，在数据可视化这块占据着举足轻重的位置。

而 Python 是一门富有表达力的语言，非常适合用于数据处理。当数据分析遇上数据可视化时，pyecharts 诞生了。其不仅可以生成独立的网页，还可以在 flask , Django 等框架中集成使用。

今天我们就聊一聊 pyecharts 中几种我们常用的图表。

安装

直接通过 pip 安装即可。

```
pip install pyecharts
```

老规矩，为了故事的顺利发展，我们先导入本文所需的模块。

```
from pyecharts.charts import Bar
from pyecharts.charts import Line
from pyecharts import options as opts
from pyecharts.charts import EffectScatter
from pyecharts.globals import SymbolType
from pyecharts.charts import Grid
from pyecharts.charts import WordCloud
from pyecharts.charts import Map
import random
```

柱状图

平时使用最多的图就是柱状图了，pyecharts 生成柱状图非常简单。直接填入 x 轴和 y 轴的数据即可。

```
x = ['1月', '2月', '3月', '4月', '5月', '6月', '7月', '8月', '9月', '10月', '11月', '12月']
data_china = [2.6, 5.9, 9.0, 26.4, 28.7, 70.7, 175.6, 182.2, 48.7, 18.8, 6.0, 2.3]
data_russia = [1.6, 5.4, 9.3, 28.4, 22.7, 60.7, 162.6, 199.2, 56.7, 43.8, 3.0, 4.9]

bar = Bar()
bar.add_xaxis(x)
bar.add_yaxis("降水量", data_china)
bar.set_global_opts(title_opts=opts.TitleOpts(title="Bar - 基本示例"))
bar.render()
```

在 PyCharm 中运行以上代码之后你会发现，控制台什么也没有，也不会像 matplotlib 一样生成一张图片，是不是我们姿势不对，但细心的你会在 Python 文件的同级目录下发现一个 html 文件，打开它，咦，原来在这里。

事实上 render 会生成本地 HTML 文件，默认会在当前目录生成 render.html 文件，当然我们也可以传入路径参数，如 bar.render("mycharts.html")。不过这样子来测试的话实在是太麻烦了，好在 pyechars 提供了贴心的 Notebook 模式，使得我们可以在 Jupyter Notebook / Jupyter Lab / Nteract / Zeppelin 四种环境中渲染。

本文均是在 Jupyter Notebook 下做的测试，只需将 bar.render() 改为 bar.rerender_notebook() 即可。改完之后再次 run 会得到下图：

同时，pyechars 还支持链式调用。

```
bar = (
    Bar()
    .add_xaxis(x)
    .add_yaxis('china', data_china)
    .set_global_opts(title_opts=opts.TitleOpts(title="Bar - 基本示例"))
)
bar.render_notebook()
```

另外，pyechars 还支持在一个柱状图中添加多个 y 轴记录，只需调用多一次 add_yaxis 即可。

```
bar = (
    Bar()
    .add_xaxis(x)
    .add_yaxis('china', data_china)
    .add_yaxis("sussia", data_russia)
    .set_global_opts(title_opts=opts.TitleOpts(title="Bar - 多柱状图"))
)
bar.render_notebook()
```

有时觉得柱状图太高不方便看，我们还可以将 x 轴和 y 轴互换，生成横向的柱状图。多柱状图和 xy 轴互换不冲突，可叠加使用。

```
bar = (
    Bar()
    .add_xaxis(x)
    .add_yaxis('china', data_china)
    .add_yaxis('russia', data_russia)
    .reversal_axis()
    .set_series_opts(label_opts=opts.LabelOpts(position="right"))
    .set_global_opts(title_opts=opts.TitleOpts(title="Bar - 翻转 XY 轴"))
)
bar.render_notebook()
```

饼状图

饼状图也是使用频率极高的图表之一，尤其是适用于占据百分比类的图，可以很直观的看出来各个类别所占据总体份额的比例。

```
pie = (
    Pie()
    .add("", [list(z) for z in zip(x, data_china)])
    .set_global_opts(title_opts=opts.TitleOpts(title="Pie - 基本示例"))
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}: {c}"))
)
pie.render_notebook()
```

圆环状的饼状图。

```
pie = (
    Pie(init_opts=opts.InitOpts(width="600px", height="400px"))
    .add(
        series_name="降雨量",
        data_pair=[list(z) for z in zip(x, data_china)],
        radius=["50%", "70%"],
        label_opts=opts.LabelOpts(is_show=False, position="center"),
    )
    .set_global_opts(legend_opts=opts.LegendOpts(pos_left="legft", orient="vertical"))
    .set_series_opts(
        tooltip_opts=opts.TooltipOpts(
            trigger="item", formatter="{a} <br/>{b}: {c} ({d}%)"
        ),
    label_opts=opts.LabelOpts(formatter="{b}: {c}")
    )
```

```
)
pie.render_notebook()
```

折线图

折线图通常用于展示数据在不同时间段的走势，例如股市的 K 线图就是折线图的一种。

```
line = (
    Line()
    .add_xaxis(x)
    .add_yaxis('china', data_china)
    .set_global_opts(title_opts=opts.TitleOpts(title="Line - 基本示例"))
)
line.render_notebook()
```

同样，和柱状图类似，折线图也可以在一个图中添加多个 y 轴记录。

```
line = (
    Line()
    .add_xaxis(x)
    .add_yaxis('china', data_china)
    .add_yaxis('russis', data_russia)
    .set_global_opts(title_opts=opts.TitleOpts(title="Line - 双折线图"))
)
line.render_notebook()
```

同样支持阶梯折线图。

```
line = (
    Line()
    .add_xaxis(x)
    .add_yaxis('china', data_china, is_step=True)
    .set_global_opts(title_opts=opts.TitleOpts(title="Line - 阶梯折线图"))
)
line.render_notebook()
```

散点图

```
scatter = (
    EffectScatter()
```

```
    EffectScatter()
    .add_xaxis(x)
    .add_yaxis("", data_china)
    .set_global_opts(title_opts=opts.TitleOpts(title="EffectScatter - 基本示例"))
)
scatter.render_notebook()
```

数据对比不是很清晰，我们可以给散点图加上网格，使各个点对应的 y 轴数据更清晰可见。

```
scatter = (
    EffectScatter()
    .add_xaxis(x)
    .add_yaxis("china", data_china, symbol=SymbolType.ARROW)
    .set_global_opts(
        title_opts=opts.TitleOpts(title="EffectScatter - 显示分割线"),
        xaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
        yaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
    )
)
scatter.render_notebook()
```

同时，我们可以指定点的形状，还可以在一个散点图上加多个 y 轴记录。这些配置就像积木一样，随意堆叠。

```
scatter = (
    EffectScatter()
    .add_xaxis(x)
    .add_yaxis("china", [x + 30 for x in data_russia],symbol=SymbolType.ARROW)
    .add_yaxis("russia", data_russia, symbol=SymbolType.TRIANGLE)
    .set_global_opts(
        title_opts=opts.TitleOpts(title="EffectScatter - 显示分割线"),
        xaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
        yaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
    )
)
scatter.render_notebook()
```

图表合并

有时候，我们需要将多种图放在一张图上来集中显示，pyechars 也考虑到了。基本步骤就是先单独生成各自类别的图，然后用 Grid 将二者合并起来即可。

比如我们想将柱状图和折线图放在一起，那就先分别生成 Bar 和 Line，然后将二者合并即可。

```
from pyecharts.charts import Grid

bar = (
    Bar()
    .add_xaxis(x)
    .add_yaxis('china', data_china)
    .add_yaxis("sussia", data_russia)
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Grid - 多图合并"),
    )
)

line = (
    Line()
    .add_xaxis(x_data)
    .add_yaxis("蒸发量", [x + 50 for x in data_china]
    )
)

bar.overlap(line)
grid = Grid()
grid.add(bar, opts.GridOpts(pos_left="5%", pos_right="5%"), is_control_axis_index=True)
grid.render_notebook()
```

词云

同样，功能强大的 pyechars 对词云也是支持的，更贴心的是中文也完全没问题，不会出现乱码。

```
import pyecharts.options as opts
from pyecharts.charts import WordCloud

data = [("生活资源", "999"),("供热管理", "888"),("供气质量", "777"),("生活用水管理", "688"),("一次供水问题

wordCloud = (
    WordCloud()
    .add(series_name="热点分析", data_pair=data, word_size_range=[6, 66])
    .set_global_opts(
        title_opts=opts.TitleOpts(
            title="热点分析", title_textstyle_opts=opts.TextStyleOpts(font_size=23)
        ),
        tooltip_opts=opts.TooltipOpts(is_show=True),
    )
)

wordCloud.render_notebook()
```

地图

最后，来看看 pyecharts 对地图图表的支持。

有时我们会很希望将数据展示在地图上，比如全国各省份人口数据，微信好友各省份分布等。

```
provinces = ['广东', '北京', '上海', '湖南', '重庆', '新疆', '河南', '黑龙江', '浙江', '台湾']
values = [random.randint(1, 1024) for x in range(len(provinces))]

map = (
    Map()
    .add("", [list(z) for z in zip(provinces, values)], "china")
    .set_global_opts(
        title_opts=opts.TitleOpts(title="map - 基本示例"),
        visualmap_opts=opts.VisualMapOpts(max_=1024, is_piecewise=True),
    )

)
map.render_notebook()
```

总结

今天我们分析了 pyecharts 常用的几种图表，俗话说一图胜千言，数据分析离不开数据的可视化，尤其是向领导做汇报工作时，图表更能清晰明了的表达成果。

生成图表的基本步骤大致可分为三个步骤，准备相关数据、利用链式调用法设置数据和相关配置、调用 render_notebook() 或者 render() 函数生成图表。

另外，pyecharts 还支持好多好玩有趣的 3D 图表，大家可自行查阅官方文档