第 5 章

体验 Web 数据分析

第 4 章简要介绍了如何使用 Pandas 进行数据分析，并将数据分析的结果通过 Matplotlib 等第三方图库可视化展现出来，但是相对而言，这种形式展示的图形偏向于静态，数据互动性小，界面过于简陋，美观性和优雅性不足，因此如果想让自己的数据分析的结果优雅地展示出来，很多时候需要借助于网页，在网页上进行样式设置和界面美化，使展示出来的数据图表更加美观，这就是 Web 端数据分析可视化，但实际上，Web 界面代码编写也十分复杂，色彩、界面、功能搭配也是一门大学问，很多专业的数据分析人员及后端人员并不擅长此道，因此为帮助读者简单领略 Web 端数据分析可视化的功能及效果，本章特意介绍一个第三方库 Streamlit，借助 Streamlit 即便是纯后端数据分析人员也能构建简单的 Web 程序，将数据分析结果展示在网页端。

5.1 Streamlit 简介

Streamlit 是一个第三方库，借助 Streamlit 不懂前端的科研人员和数据分析人员也可以将自己的数据分析脚本快速地转换为网页应用，而不需要去考虑路由、协议、HTML、CSS 这些网站专业技术。Streamlit 开包即用，如果脚本更改或者数据发生改变，前端则会相应地做出变化，十分便捷。Streamlit 的官网为 https://streamlit.io/，在浏览器网址栏输入这个网址即可访问 Streamlit 的官网。如图 5-1 所示，在官网可以查看相应的代码示例和说明文档，本章仅抛砖引玉，进行简要介绍，并使用 Streamlit 构建一个简单的 Web 可视化界面。

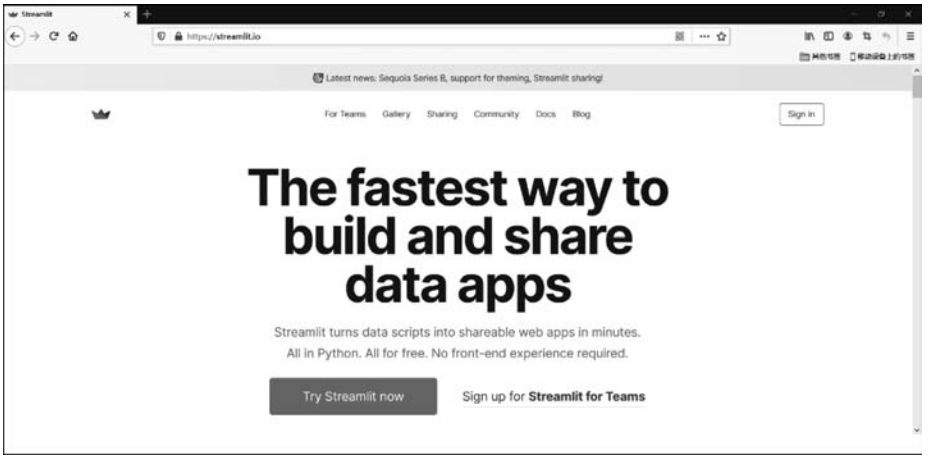

图 5-1　Streamlit 官网

5.2　安装 Streamlit

与安装其他第三方库一样，在 PyCharm 左下角单击 Terminal 按钮，会弹出当前虚拟环境（venv 开头）的路径，在后面输入 pip install streamlit 命令，输入后按 Enter 键，程序会自动下载并安装。由于版本兼容问题，此次并没有指定 pip 的镜像源参数（如-i https://pypi.tuna.tsinghua.cn/simple），而是直接下载并安装，安装完成后的提示如图 5-2 所示。

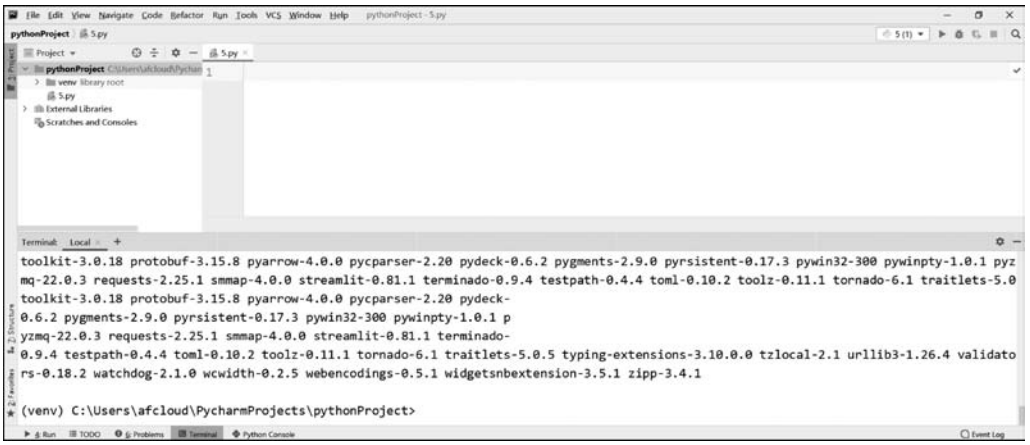

图 5-2　Streamlit 安装完成示例图

5.3　Streamlit 开发

本次使用 Streamlit 开发一个简单的 Web 界面，展示不同可视化库的绘图效果，这里选用 Matplotlib、Plotly、Altair 及 Streamlit 自带的图表。当然在正式开发之前，读者需先参照上述 Streamlit 的安装方法，自行安装 Plotly、Altair 两个库。最后的界面效果如图 5-3～图 5-6 所示。

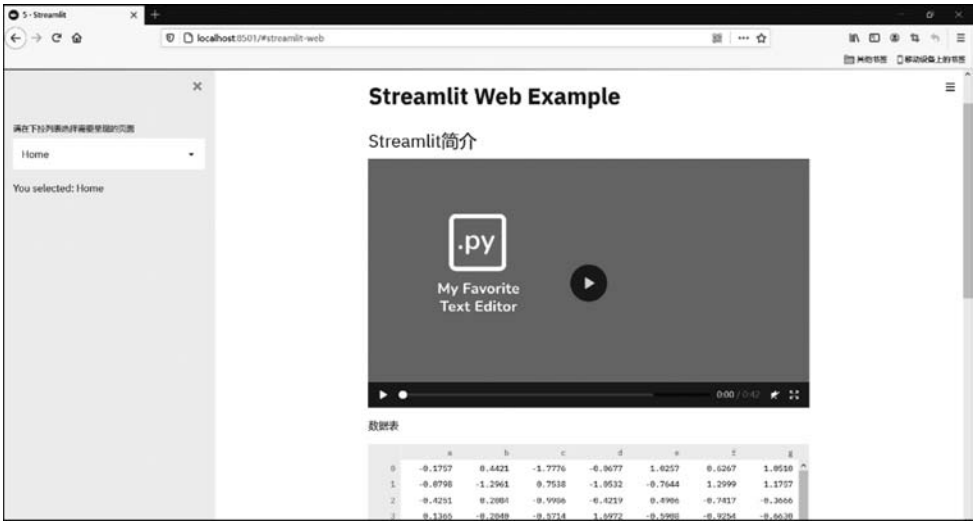

图 5-3　Web 效果图（1）

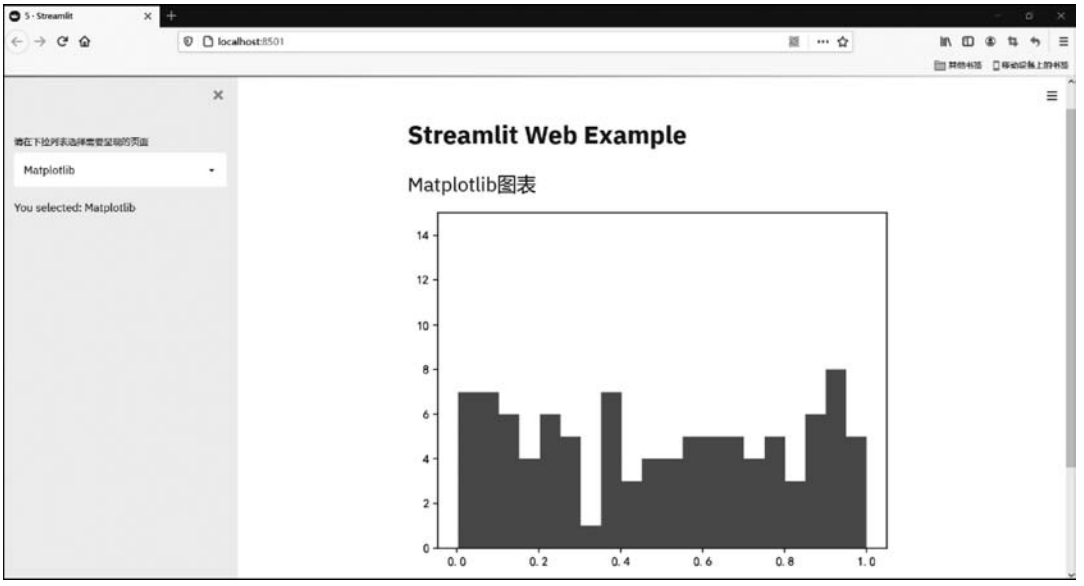

图 5-4　Web 效果图(2)

图 5-5　Web 效果图(3)

此次开发的 Streamlit Web 程序较为简单,主要由左侧侧边栏和右侧图表展示区两部分组成。左侧侧边栏采取下拉菜单方式完成,可以查看几种可视化图表库的不同图表效果,分别是 Streamlit 自带图表、Matplotlib 图表、Plotly 图表和 Altair 图表。单击后下拉列表下方的文字会相应地显示选项,右侧则是简单的展示每种图表库的一张图表示例。下面分步讲述如何开发这样一个简单的 Web 脚本。

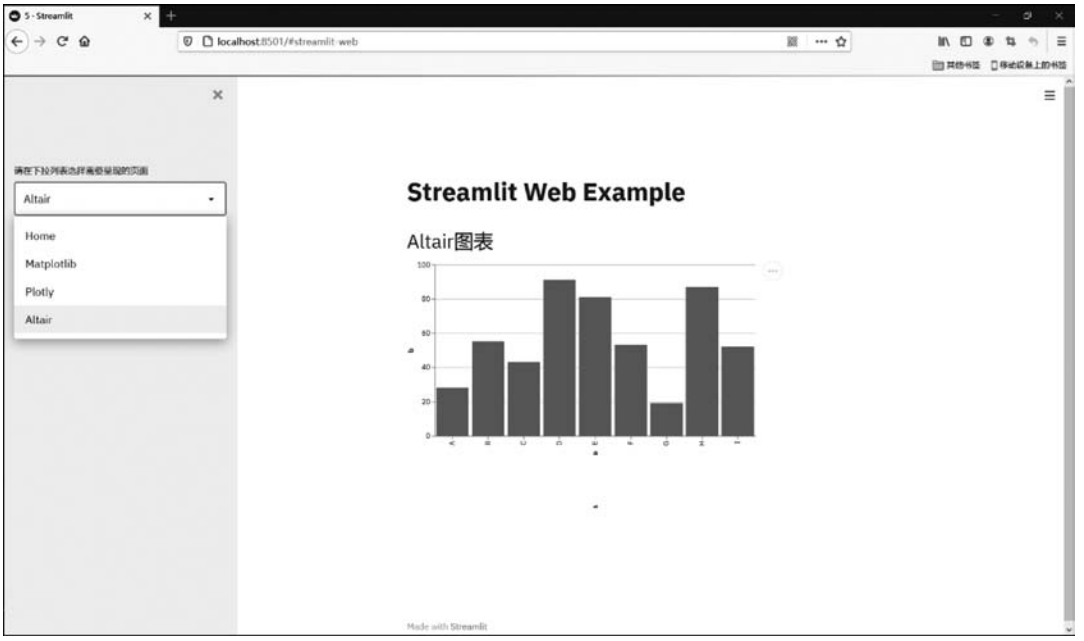

图 5-6　Web 效果图(4)

5.3.1　导入第三方库

5.2 节讲解了如何安装 Streamlit，并建议将其他第三方库(如 Matplotlib、Plotly、Altair)一并安装，为方便后续使用，下面将这几个库一并导入，代码如下：

```
//第 5 章/5.3.1/引入第三方库
import streamlit as st
import pandas as pd
import numpy as np
import altair as alt
import plotly.figure_factory as ff
import matplotlib
matplotlib.use('TkAgg')
matplotlib.get_backend()
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

细心的读者可能会发现这段代码在导入 Matplotlib 之后，又添加了其他内容，这是为了避免 Matplotlib 图标不显示及汉字字体不显示等问题。如果一切正常，则导入后的效果如图 5-7 所示，如果提示缺少一些库，则按提示安装即可。

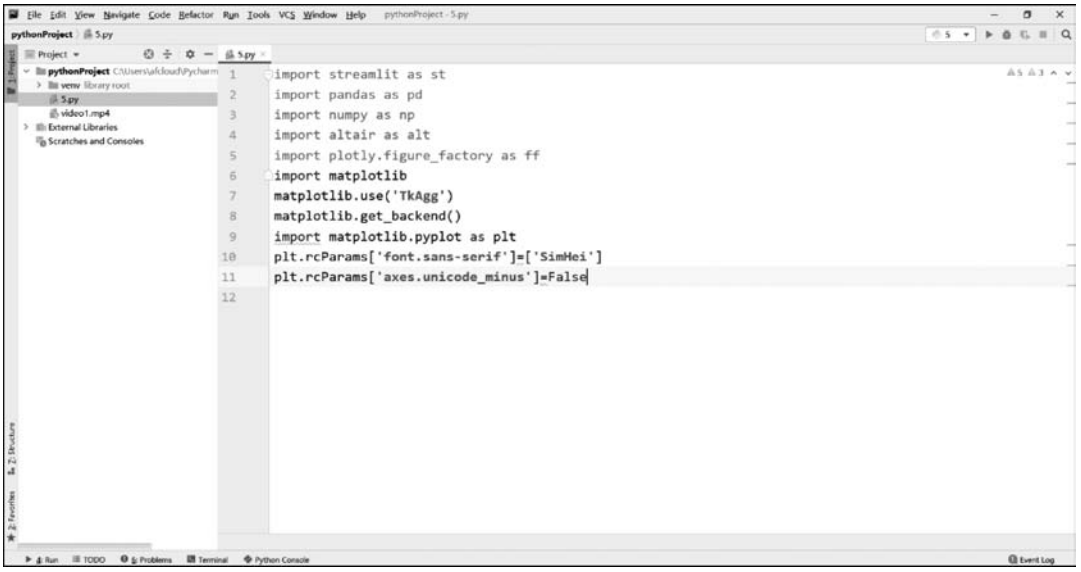

图 5-7　导入第三方库示意图

5.3.2　添加标题和侧边栏

下面先为程序添加一个标题，相当于网站的名称，这里的标题为 Streamlit Web Example，代码非常简单，代码如下：

```
#标题
st.title("Streamlit Web Example")
```

然后增加一个侧边栏的下拉选择框，代码如下：

```
#侧边栏选择框
option = st.sidebar.selectbox(
    '请在下拉列表选择需要呈现的页面',
     ['Home','Matplotlib','Plotly','Altair'])
st.sidebar.write('You selected:', option)
```

这段代码将下拉菜单的按钮设置为 4 个，名称分别为 Home、Matplotlib、Plotly、Altair，并将用户选取的按钮值赋值给变量 option。代码完成后，如果要查看代码的运行效果，在 PyCharm 左下角单击 Terminal 按钮，则会弹出当前虚拟环境(venv 开头)的路径，在后面输入 streamlit run 5.py 命令并按 Enter 键即可。5.py 是当前脚本的文件名，读者可以根据自己的实际情况命名。streamlit run 5.py 命令执行后 PyCharm 左下角会弹出页面的 URL 网址，同时弹出默认浏览器打开页面。如果用户没有设置默认浏览器或者浏览器没有弹出，则可以自行复制 URL 并在任意浏览器打开。程序执行的界面如图 5-8 所示，页面显示效果如图 5-9～图 5-10 所示。

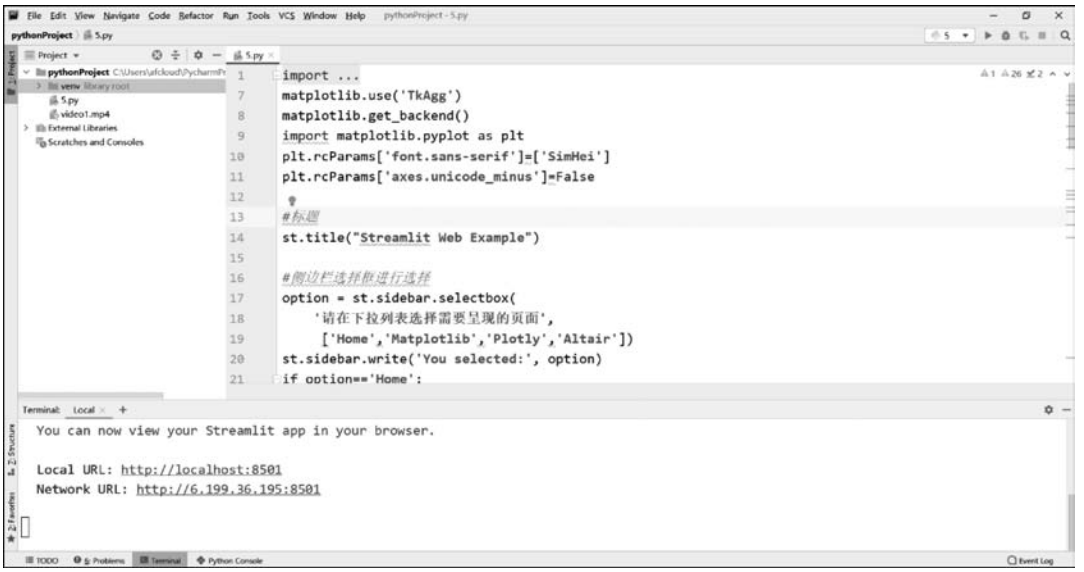

图 5-8　Streamlit 执行示意图

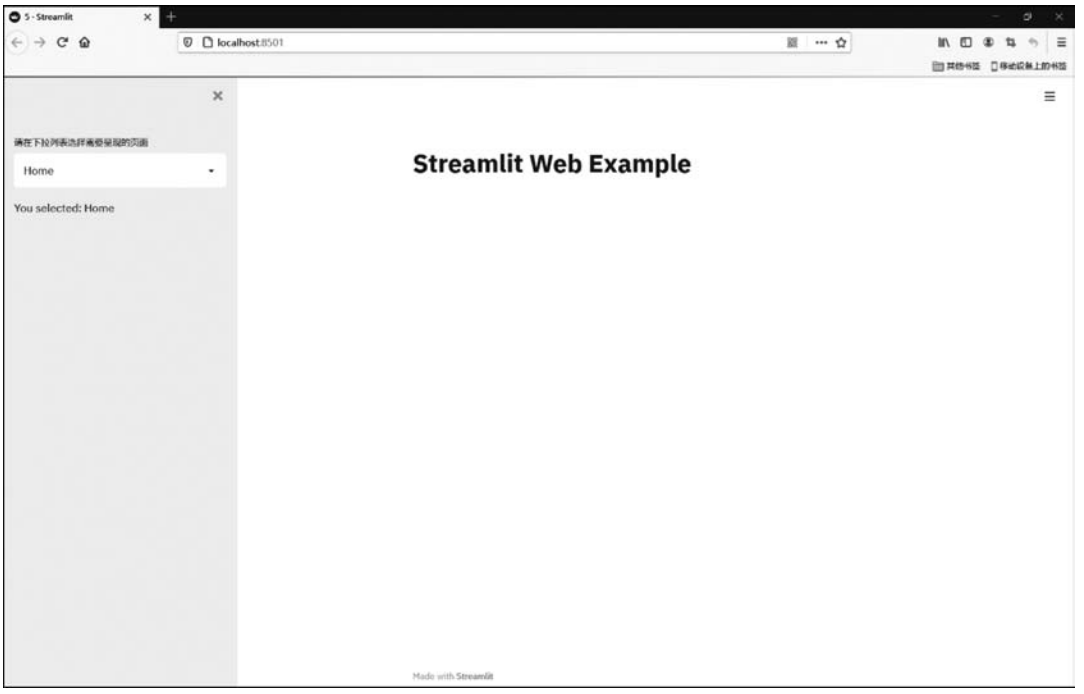

图 5-9　Streamlit 标题界面

图 5-10　Streamlit 侧边栏界面

5.3.3　为 Home 选项制作界面

Home 为 option 的默认值，本节为 Home 页设计了 4 个元素，分别是一个小标题、一个视频、一个数据表和一个折线图，这 4 个元素都是通过运用 Streamlit 自带函数实现的。视频预先放到与 5.py 同级的文件夹中，名为 Streamlit 简介，读者可以自行寻找一个视频替代。图表和折线图则是随机生成的一些数据，读者可以参考示例代码也可以自己制订。这部分的代码如下：

```
//第 5 章/Home 设计:
if option == 'Home':
    st.header('Streamlit 简介')
    video_file = open('video1.mp4', 'rb')
    video_Bytes = video_file.read()
    st.video(video_Bytes)
    #数据表
    st.write('数据表')
    chart_data = pd.DataFrame(
        np.random.randn(20, 7),
        columns = ['a', 'b', 'c','d','e','f','g'])
    chart_data
    #条形图
    st.write('条形图')
    st.bar_chart([1,2,3,4,5,6,7])
```

这段代码的含义是当 option 选项值为 Home 时，执行代码后创建页面并显示 4 个元素。代码执行完成后在浏览器中刷新，执行的效果如图 5-11 所示，因内容过多，纵向无法完全显示，所以图 5-11 中的值只保留了一部分供读者对比。

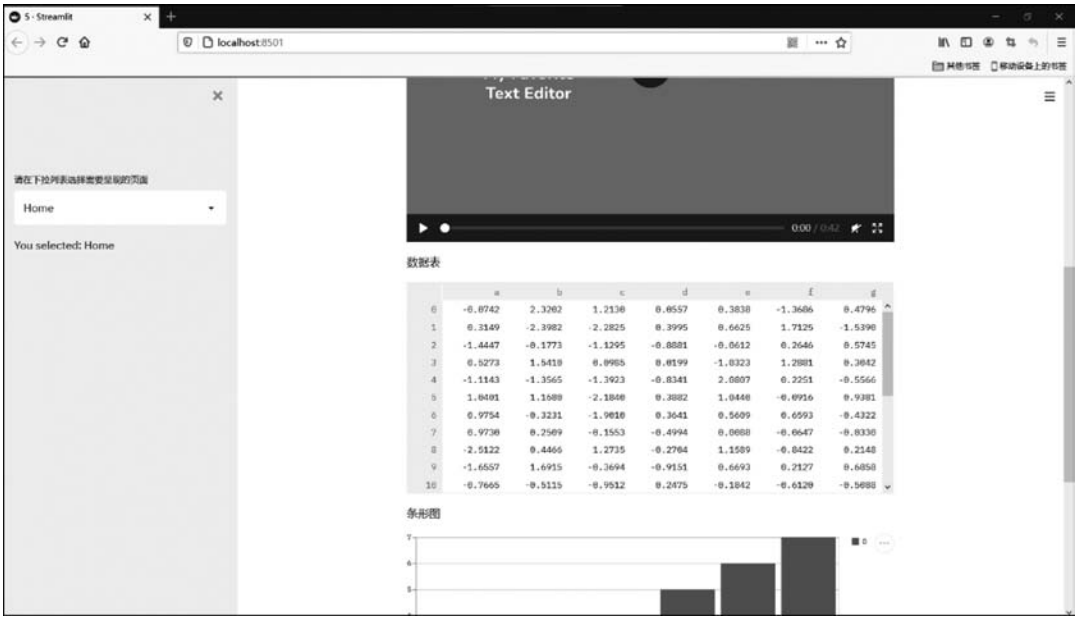

图 5-11　Home 页面

5.3.4　为 Matplotlib 选项制作界面

下面为 Matplotlib 页面编写代码，也需要指定 option 的值。同时为对应下拉列表的功能，使用 if-elif-else 结构。Matplotlib 页面比较简单，此处只设计了一个元素，即简单柱状图，代码如下：

```
//第 5 章/5.3.4/Matplotlib 设计
elif option == 'Matplotlib':
    st.header('Matplotlib 图表')
    a = np.random.rand(100)
    plt.hist(a,bins = 20) #100 个值进行 20 等分
    plt.ylim(0, 15) #限制 y 轴高度:0→15
    st.pyplot()
else:
    pass
```

代码完成后在浏览器中刷新，执行的效果如图 5-12 所示。

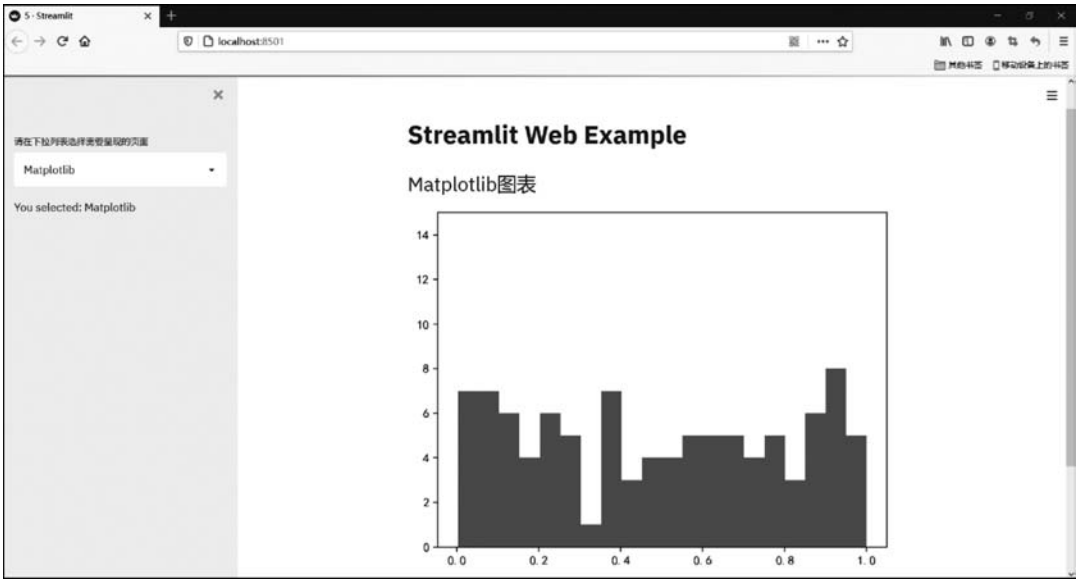

图 5-12　Matplotlib 页面

5.3.5　为 Plotly 选项制作界面

下面为 Plotly 页面编写代码，也要指定 option 的值，同时为对应下拉列表的功能，继续使用 if-elif-else 结构。与 Matplotlib 页面相同，只设计了一个元素，即一幅复合图，代码如下：

```
//第 5 章/5.3.3/Plotly 设计
elif option == 'Plotly':
    st.header('Plotly 图表')
    x1 = np.random.randn(100) - 2
    x2 = np.random.randn(100)
    x3 = np.random.randn(100) + 2
    #分组
    hist_data = [x1, x2, x3]
    group_labels = ['项目 1', '项目 2', '项目 3']
    #创建图表
    fig = ff.create_distplot(
        hist_data, group_labels, bin_size = [.1, .25, .5])
    #Plot!
    st.plotly_chart(fig)
else:
    pass
```

代码完成后在浏览器中刷新，执行的效果如图 5-13 所示。

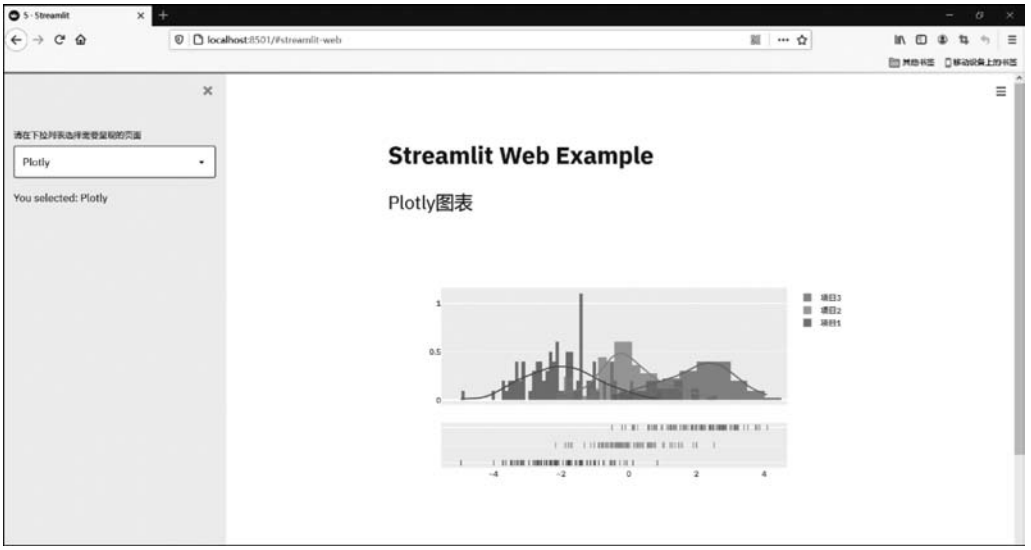

图 5-13　Plotly 页面

5.3.6　为 Altair 选项制作界面

下面为 Altair 页面编写代码，也需要指定 option 的值，同时为对应下拉列表的功能，继续使用 if-elif-else 结构。与 Matplotlib 页面相同，同样只设计了一个元素，即一幅复合图，代码如下：

```
//第 5 章/5.3.4/Altair 设计
else:
    st.header('Altair 图表')
    data = pd.DataFrame({
        'a': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I'],
        'b': [28, 55, 43, 91, 81, 53, 19, 87, 52]
    })
    c1 = alt.Chart(data).mark_bar(
        color = "red"
    ).encode(
        x = 'a',
        y = 'b'
    ).properties(
        width = 500,
        height = 300
    )
    st.altair_chart(c1)
```

代码完成后在浏览器中刷新，执行的效果如图 5-14 所示。

至此，已经借助 Streamlit 完成了一个小型的 Web 数据可视化项目。当然有兴趣的读者可以结合第 4 章的数据分析可视化，尝试进行数据分析 Web 可视化，即在 Web 端单击选项后，后端再进行计算，然后将参数返回前端，以便呈现计算后的效果。不过细心的读者可

以发现，借助 Streamlit 完成的 Web 项目反应相对较慢，虽然 Streamlit 提供了一些方法进行优化，相比纯正的 Web 项目来讲还是略显不足。Streamlit 自带审查系统，每次数据或者后台代码发生更改时，Streamlit 都会重新运行，只不过反应没有那么快速，当然也可以每次手动刷新。虽然 Streamlit 有很多不足，但对于一个简单的数据分析人员来讲，仍不失为一件利器，有兴趣的读者可以仔细阅读 Streamlit 的官方文档，用 Streamlit 构建自己的 Web 应用。如果读者对性能和外观比较看重，则可继续学习后续章节，逐步掌握自主构建一个完整 Web 可视化项目的所有技术。

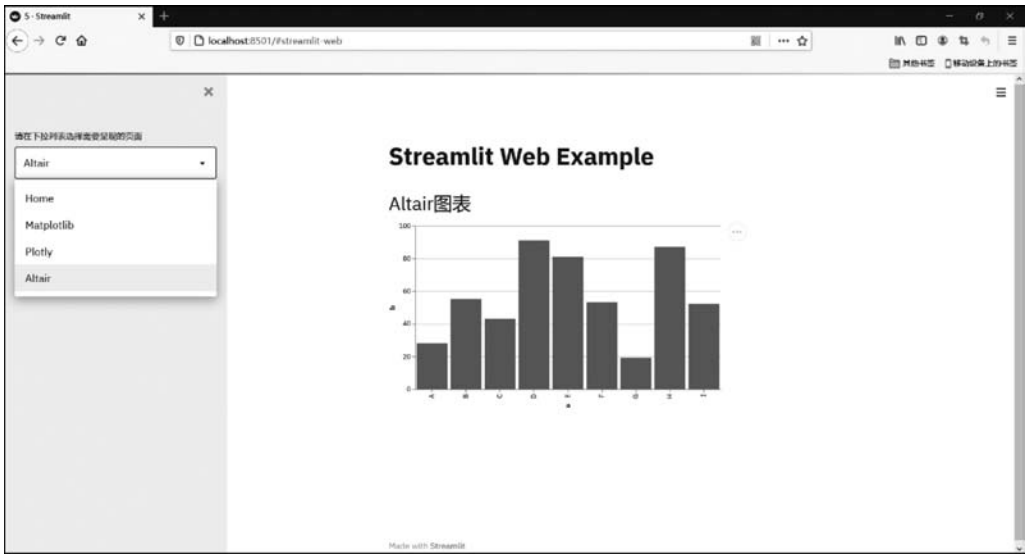

图 5-14　Altair 页面