# 《高级编程技术实践》大作业

题目：基于streamlit的2021年度steam游戏发售可视化数据

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 学号 | 班级 | 最终总成绩  (100） |
| 王一博 | 20212251115 | 国软2101 |  |

大连理工大学国际学院

2023年7月

大作业要求

此作业为独立完成的作业，学生独立完成作业要求。

（1）题目：

根据自己对国内外感兴趣的方向进行数据分析，比如经济数据，金融数据，重疾数据等等，可以自行选择。

1. 自行获取相关数据（日期维度至少20天以上）
2. 要求必须实现类别发展情况变化（使用折线图和地图），地图要按照日期进行自动变化；
3. 其它功能自行扩展，扩展功能实现好的可以加分；
4. 使用技术，python相关技术不限其他技术。
5. 需要提交电子版和纸质版作业；电子版包括代码、数据、结果演示的录屏；纸质版就是打印的大作业；

（2）作业内容包括：

根据学生自选题目，就选定题目，进行可视化实践，对该题目深入探讨研究，并实施，并进行汇报，并提交大作业。

（3）要求：

独立完成，对选择课题进行分析、编码及可视化呈现，最终能够保证对要求的课题有效可视化。要有可视化结果展示截图等。

学生要求提交大作业报告，对可视化分析设计过程说明，把数据进行可视化展示，并给出课程学习总结体会。

选题目的，背景，使用的什么样图表以及使用方法。碰到的困难点，如何克服，最后总结。

（4）打印要求：

正文使用宋体小四号字，1.25倍行间距； 反正面或正面打印均可。请使用统一给定的分组报告的封皮。

评分标准

1. 大作业成绩

1）总分100分，根据作业完成情况给分；分为两部分：其中作业文档撰写情况30分，代码及视频成果物提交及完成情况70分。

2）文档30分评分要求：

主要分为：内容完整10分；分析理解15分；结果正确性5分

具体评分标准如下：

1. 内容完整：10分
2. 层次分明，内容完整，调理清晰：8-10分
3. 层次一般，内容一般，调理不顺：5-8分
4. 论文层次不合理，内容不全面：5分以下
5. 分析理解：15分
6. 分析深入，理解深刻：13-15分
7. 有一定分析，理解到位：8-13分
8. 缺乏必要分析，理解有问题：8分以下
9. 结果正确5分
10. 结果正确，体会真实，总结到位：4-5分
11. 结果基本正确，有一定的总结：3-4分
12. 缺乏总结：3分以下

3）成果物提交及完成情况70分评分要求：

a、成果物完整，代码及初始化数据完整，视频完整并内容清晰正确：55-70分；

b、成果物基本完整，代码及初始化数据部分缺失，视频及内容有部分缺陷：40-55分；

c、成果物缺失严重，代码及初始化数据缺失严重，视频及内容很少或者没有：30-40分；

# 项目需求简介

项目背景： 在2021中，Steam 平台上推出了众多令人兴奋且创意新颖的游戏。为了深入了解这些游戏的销售情况、用户反馈以及市场趋势，我们打算开发一款可视化程序，将 Steam 平台在 2021 年度发售的游戏数据进行分析和展示。该报告将有助于游戏开发者、投资者以及游戏爱好者乃至外行人更好地了解 Steam 平台上的游戏市场。

项目目标： 本项目的目标是编写基于streamlit的可视化程序，能够对 Steam 平台在 2021 年度发售的游戏数据进行可视化分析。通过该程序，我们可以直观地了解游戏的销售情况、用户评价、流行度等关键指标。同时，该程序还应支持对游戏数据进行多维度的筛选，以便用户可以根据自己的需求进行定制化的分析。

项目要求：数据获取：程序需要从 Steam 平台获取 2021 年度发售的游戏数据。这包括游戏的名称、发行日期、销售额、评分等信息。数据的获取通过爬虫程序爬取。在获取到数据后进行数据清洗。

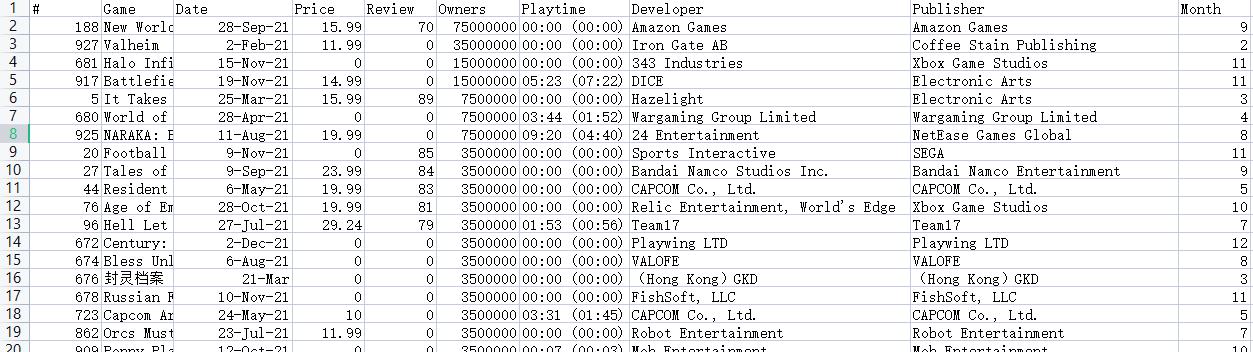
数据分析与可视化：程序应该能够对获取到的游戏数据进行分析和可视化。用户可以通过直观的图表和图形了解游戏的销售趋势、用户评价分布、游戏类型分布等信息。程序支持多种图表类型，如柱状图、折线图、饼图等，以便根据自己的需求选择最合适的展示方式。

交互功能：由于引入了streamlit，程序提供交互功能，允许使用者对数据进行选择，可以根据时间，图表等关键指标对数据进行筛选。此外，使用者还可以通过与图表的交互来进一步探索数据，例如鼠标悬停显示详细信息等。

导出和分享功能：程序应支持将分析结果导出为常见的文件格式（如图像文件、PDF 等），以便用户可以将报告分享给其他人或保存为备份。详细操作在Page中进行

# 项目数据说明

清洗后的数据示例，csv格式：



总共有10项数据，从左到右分别是：

游戏编号，游戏名，发行日期，发售价格，拥有人数(截至到爬取当天)，好评率，平均游戏时间，开发商，发行商，发行月份

说明：

游戏编号（＃）：即该游戏在steamspy中的编号，object格式（默认）

游戏名称：游戏的命名，str格式

发行日期：年/月/日，实际上是Datetime格式

发售价格：发售时的售价，float格式

拥有人数：截止到爬取时间的玩家数，int64格式

好评率：好评数/总评论数，int64格式

平均游戏时间：由于网站限制，爬取的这项数据有缺失，后续也不将使用这项数据，object格式（默认）

开发商：开发该游戏的公司，str格式

发行商：负责产品宣发的公司，str格式

发行月份：数据清洗中单独列出的月份数据，方便后续代码操作，int64格式

# 项目详细设计及成果展示

## 首先进行成果展示：请首先阅读readme.txt

程序运行后弹出的Page



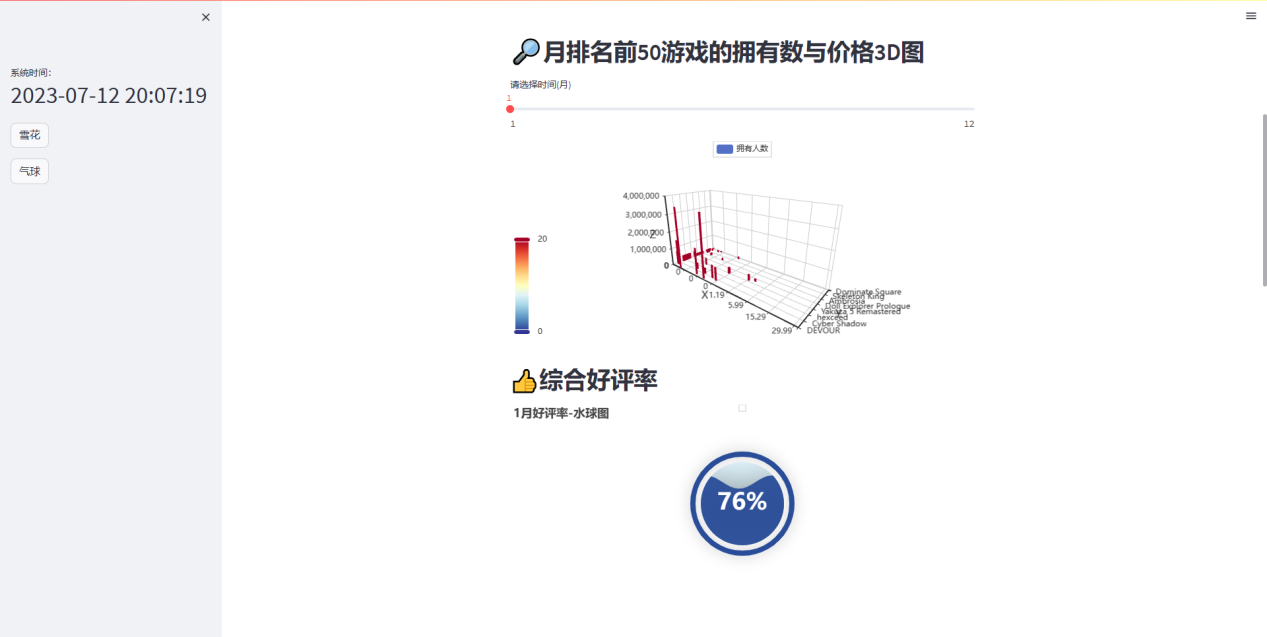
#图太大了，为了方便移到下一页去了

月排名前50游戏拥有数与价格，以及月综合好评率，随月份变化

通过3D图上方的滑块选择要显示的月份，默认显示1月信息，3D图和水滴图同时变化。

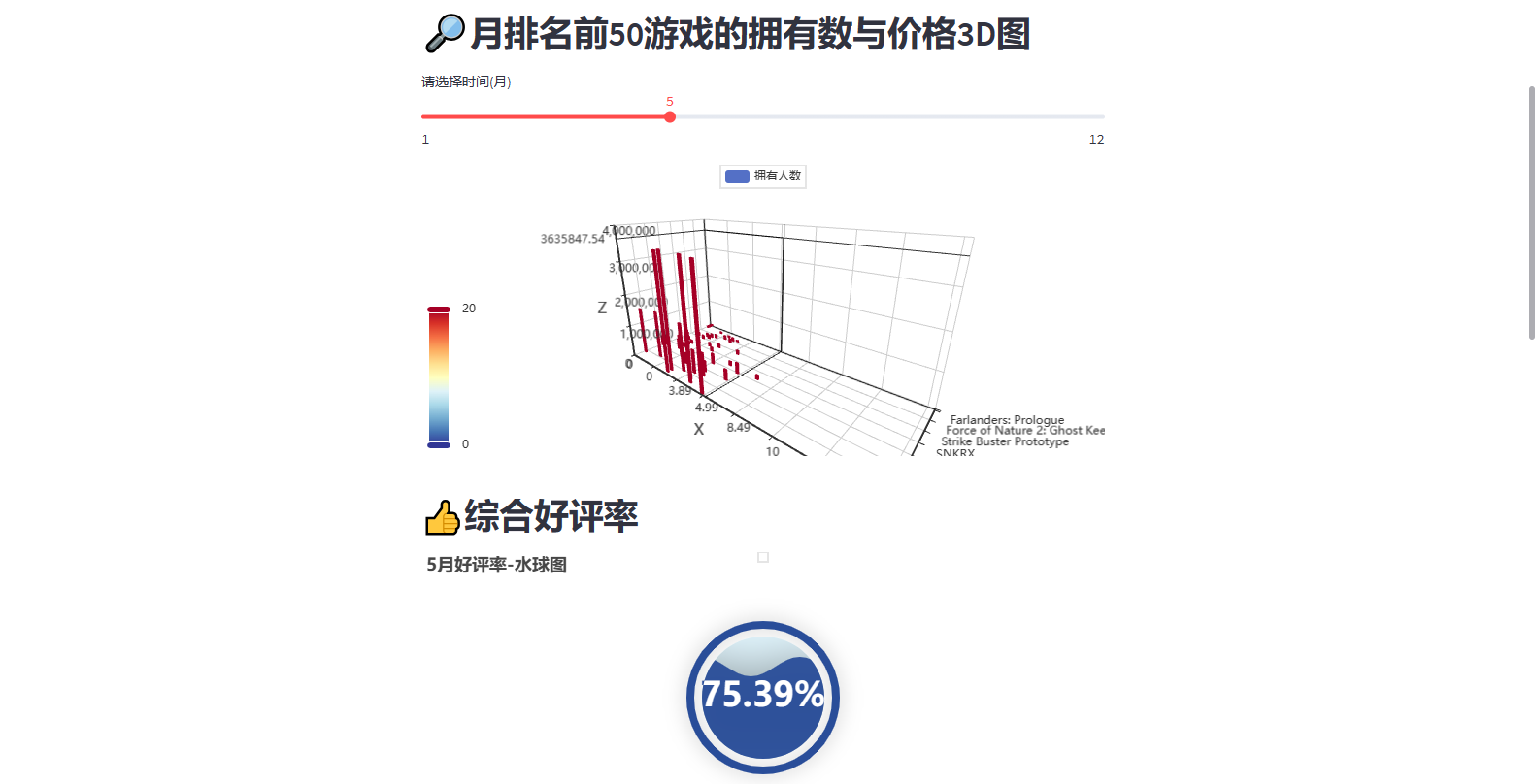
直观展示了每月前50的游戏的售价与购买数的关系，以及各个月的平均好评率

当前为一月数据



滑动滑块：

当前为五月数据



#本来是想用pyecharts的时间轴模块的，但是发现pyecharts的Tab模块和时间轴模块都没法和Page联用，导入即报错，去官方Github上看QA得知官方不想开前端这个头，明确表示不会加入这个功能，只好接着用streamlit实现

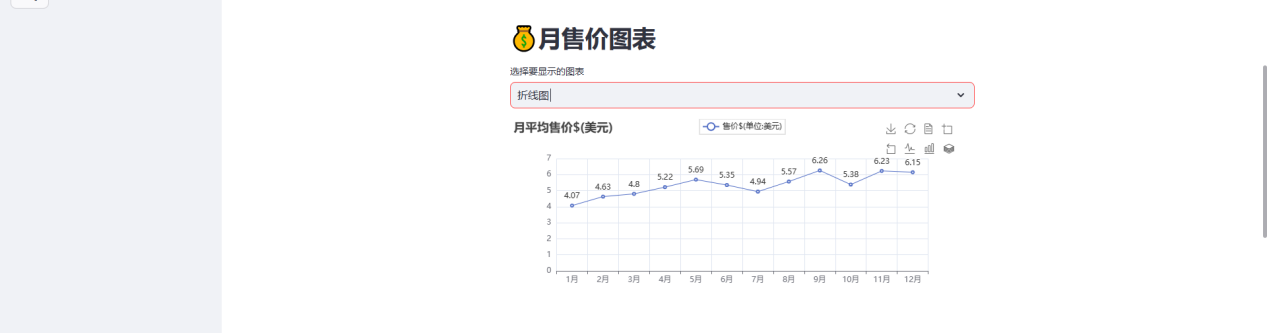
月售价图表



直观展示了每个月的平均售价和变化趋势，可以看出9月份价格最高而1月最低

共3个图，第一个图为柱状图，通过图标上方选择块进行切换

例如：



切换折线图



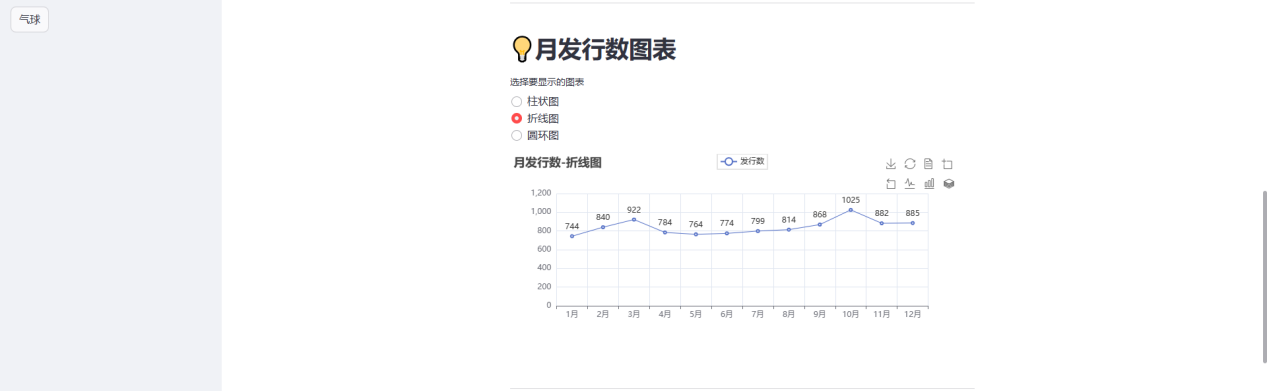
切换散点图

每月游戏发行数可视化图表

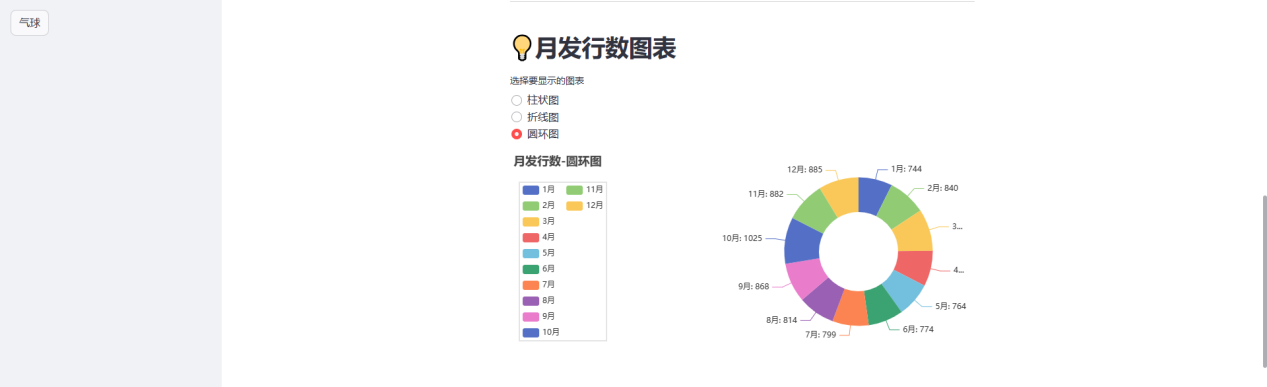


展示了各个月的发行总量信息和变化趋势，总体来讲春秋季节是发行旺季

同样是三个图，第一个为柱状图，通过广播(radio)选择枝进行切换

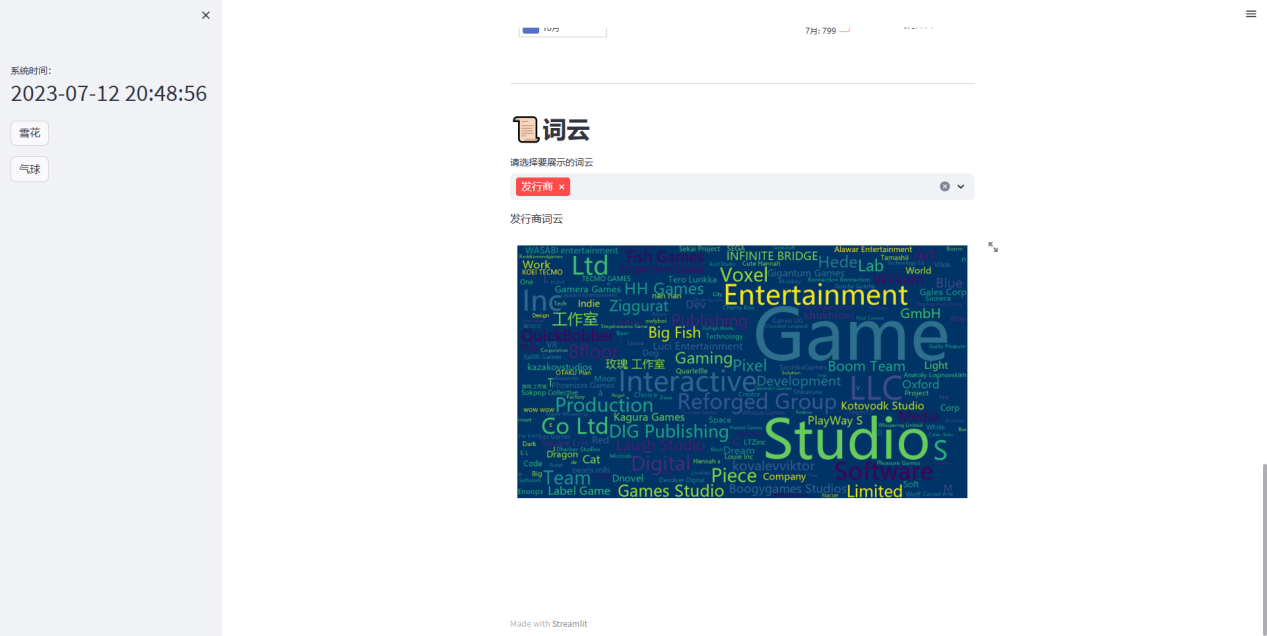


切换折线图



切换圆环（饼状）图

开发商/发行商/游戏名 词云

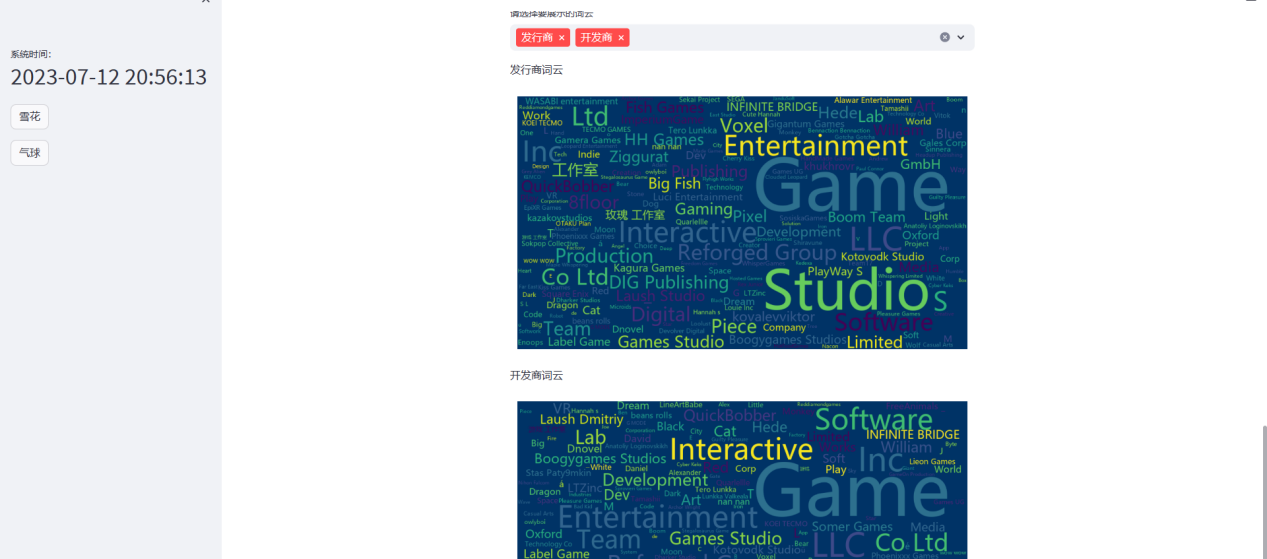


直观展示了开发商、发行商、游戏名中的高频词，了解在这1年内的热门厂商和游戏话题

共三个词云，可利用多项选择块进行选择性或同时展示

例如：





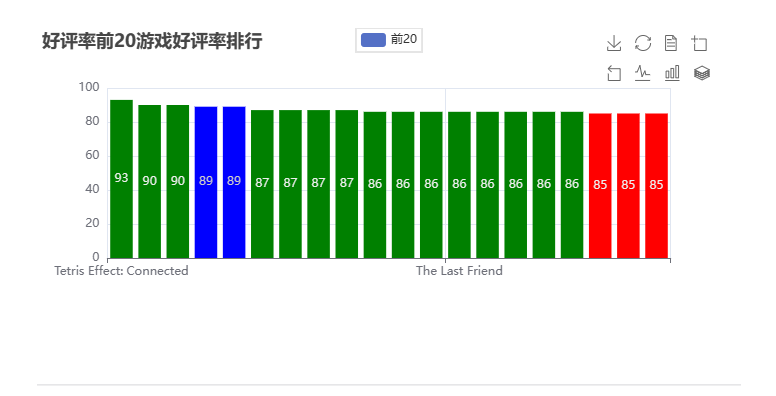
选择发行商和开发商词云



仅选择游戏名



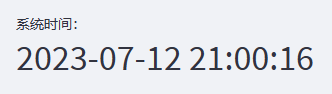
三个都选



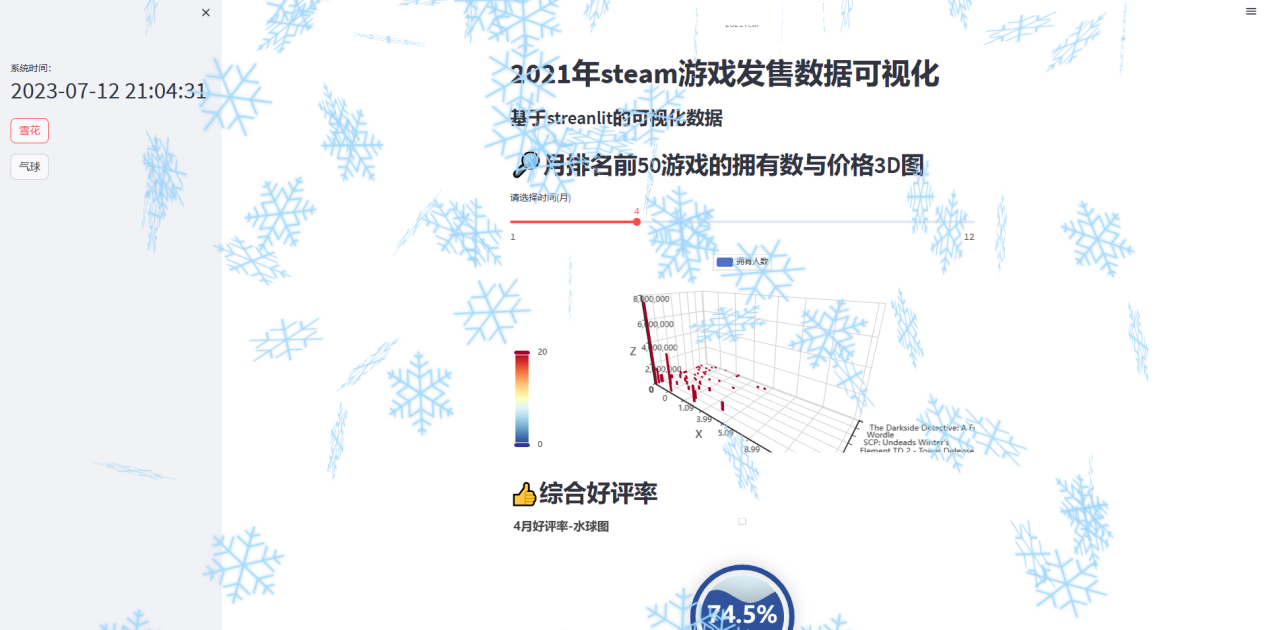
好评率折线图

除此之外，为了页面的美观加入了侧边栏和一些小组件

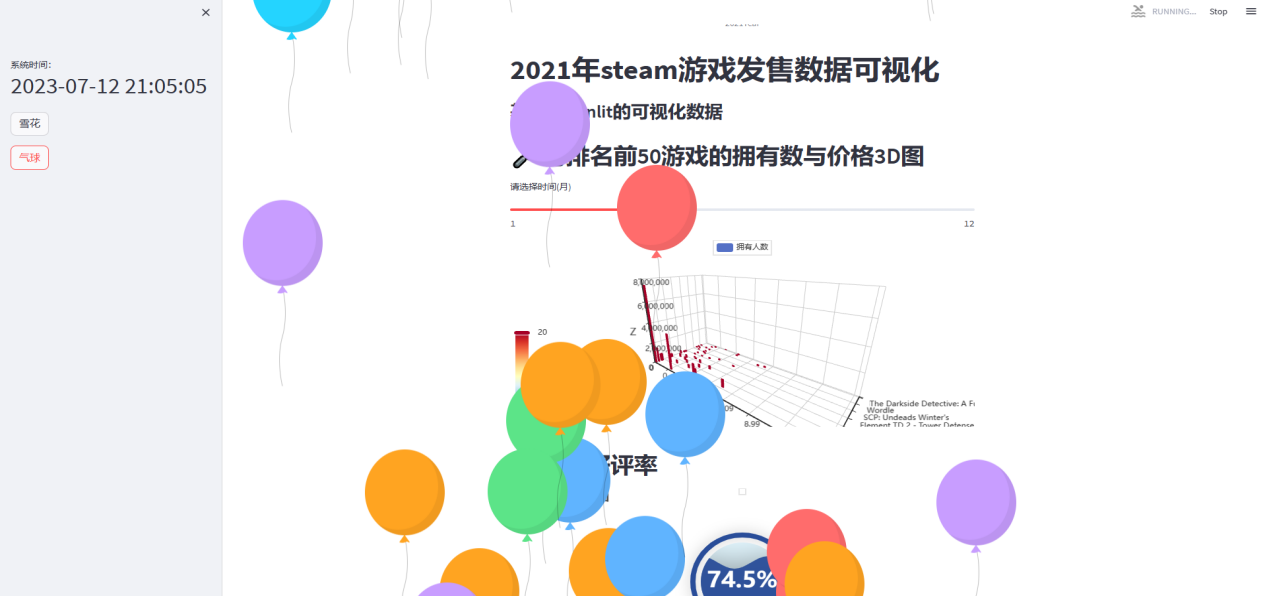
时间



雪花



气球



附上代码：

with st.sidebar:

    st.metric(*label*="系统时间：", *value*=time.strftime('%Y-%m-%d %H:%M:%S', time.localtime()))

    if st.button('雪花'):

        st.snow()

    if st.button('气球'):

        st.balloons()

## 接下来就是项目详细设计（代码详解）

从上至下，仅解释重要代码

利用爬虫程序爬取https://www.steampy.com/的数据

首先是数据清洗部分，wash.py

path = 'D:/pycode/Pratice/steam\_game\_visualization/2021 - Year Stats.csv'

df = pd.read\_csv(path,*index\_col*=0,*encoding*='gbk')

Pandas读取csv文件，编码格式为gbk

*#处理价格数据*

*def* price(*x*):

    try:

        pricenum=*int*(x['Price'].replace('$',''))

    except:

        pricenum=0

    return pricenum

df['Price'] = df.apply(*lambda* *x*:price(x),*axis*=1)

df['Price'] = pd.to\_numeric(df['Price'])

处理价格数据，用replace()方法将数据中的美元符号去掉，除此之外，通过抛出异常将免费游戏，即数据为”Free”的数据直接置0。随后遍历，写入Price列中，最后使用to\_numeric方法进行格式转换，转为int64格式

*#处理好评率*

*def* review(*x*):

    try:

        reviewnum=*int*(x['Review'].replace('N/A (N/A/','').replace('%)',''))

    except:

        reviewnum=0

    return reviewnum

df['Review'] = df.apply(*lambda* *x*:review(x),*axis*=1)

df['Review'] = pd.to\_numeric(df['Review'])

处理好评率，由于得到的原始数据中有许多的干扰字符，如N/A，用replace()函数替换为’’，没有评价数据的异常抛出则直接置0，随后依旧是遍历，写入Review列，使用to\_numeric()方法格式转换

*#玩家数*

*def* owners(*x*):

    List = x['Owners'].replace(',', '')

    cut = List.index("?")  *# 查找第一个 "?" 的索引位置*

    n\_str = List[:cut]  *# 提取 "?" 之前的子字符串*

    len1 = len(n\_str)

    y=List.replace("?..?",'')

    minnum = *int*(y[:len1])

    maxnum = *int*(y[len1:])

    average = (minnum + maxnum) / 2

    return average

df['Owners']=df.apply(*lambda* *x*:owners(x),*axis*=1)

df['Owners']=pd.to\_numeric(df['Owners'])

处理玩家数数据，得到的原始数据中以逗号分隔千分位、十万分位，并且?..?两边的数字为最大最小值（即范围），我们首先用replace()方法去除逗号，写入一个列表List中，然后设计一个算法：用index方法查找第一个？的索引位置，提取？之前的字符串，len()方法获取该串长度，使用切片获取两边的有效数据，最后求平均，得到大致的购买数

数据清洗结束

主程序main.py:

*#获取路径*

subpath=os.path.abspath(".")+"/2021 - Year Stats.csv"

df = pd.read\_csv(subpath, *index\_col*=0, *encoding*='gbk')

df['Price'] = pd.to\_numeric(df['Price'])

df\_re = df[df['Review'].values!=0].copy()  *#拷贝一份好评不为零(存在评价)的*

df\_mon=df.groupby(['Month'])

获取路径，为了方便在不同系统环境下导入csv，利用os.path.abspath获取根目录路径，以便读取。以Month月份数据进行分组赋给df\_mon，备用，copy()方法拷贝一份好评不为零的dataframe，备用。

*#排名前50的游戏拥有数与价格随时间变化*

*def* tDBar(*df*):

*#mon = st.slider('请选择时间', 1, 12, 1)*

    get=df.groupby(['Month']).get\_group(mon).head(50)

    game=get['Game']

    game\_list=game.values.tolist()

    owners=get['Owners']

    owner\_list=owners.values.tolist()

    price=get['Price']

    price\_list=sorted(price.values.tolist())

    data=[]

    for index, row in get.iterrows():

        x = row['Price']

        y = row['Game']

        z = row['Owners']

        data.append([x, y, z])

    bar3d=Bar3D()

    bar3d.add(

*series\_name*="拥有人数",

*data*=data,

*xaxis3d\_opts*=opts.Axis3DOpts(*type\_*="category", *data*=price\_list),

*yaxis3d\_opts*=opts.Axis3DOpts(*type\_*="category", *data*=game\_list),

*zaxis3d\_opts*=opts.Axis3DOpts(*type\_*="value"),

    )

    bar3d.set\_global\_opts(

*visualmap\_opts*=opts.VisualMapOpts(

*max\_*=20,

*range\_color*=[

                "#313695",

                "#4575b4",

                "#74add1",

                "#abd9e9",

                "#e0f3f8",

                "#ffffbf",

                "#fee090",

                "#fdae61",

                "#f46d43",

                "#d73027",

                "#a50026",

            ],))

绘制前50游戏的购买量与价格关系的3D图，通过 groupby 方法，根据选择的月份（mon）从数据框中筛选出相应的数据，并获取排名前50的游戏。其中滑动块获取的mon提取特定组，取前50行数据。创建一个空列表 data 用于存储游戏的价格、名称和拥有数信息。分别以列表储存Game、Price、Owners三项数据作为XYZ轴，使用 Bar3D 创建一个3D柱状图对象 bar3d。使用 bar3d.set\_global\_opts() 方法设置全局选项，包括可视化映射选项，用于设置颜色渐变范围。设置颜色，绘制图表，随时间变化数据。

日期滑块：

mon = st.slider('请选择时间(月)', 1, 12, 1)

*#评价信息*

*def* Review\_Timeline(*df*):

    re\_liquid=df.groupby(['Month'])['Review'].mean()

    alist=re\_liquid.values.tolist()

    new\_list = [x/100 for x in alist]

    mid\_np = np.array(new\_list)

    mid\_np\_4f = np.round(mid\_np,4)

    n2\_list = *list*(mid\_np\_4f)     *#列表处理，列表转np数组再转列表*

*#mon = st.slider('请选择时间', 1, 12, 1)*

    want=n2\_list[mon-1]

    liquid=Liquid()

    liquid.add("好评率", [want],

*label\_opts*=opts.LabelOpts(

*font\_size*=35,

*formatter*=JsCode(

                """function (param) {

                    return (Math.floor(param.value \* 10000) / 100) + '%';

                }"""            *#设置精度*

            )))

    liquid.set\_global\_opts(*title\_opts*=opts.TitleOpts(*title*="{}月好评率-水球图".format(mon)))

    st\_pyecharts(liquid)

评价信息的水滴图，同样根据滑块改变月份，随时间变化数据

使用groupby()方法根据Month分组，取Review列求均值，写入列表，将数据进行相应的处理，先将每个月的平均评价信息除以100，然后使用 NumPy 进行精度处理，将小数位数限制为4位。使用 Liquid 创建一个水球图对象 liquid。

使用 liquid.add() 方法将处理后的评价信息添加到图表中，并设置标签选项，包括字体大小和格式化函数，用于显示百分比形式的好评率。

使用 liquid.set\_global\_opts() 方法设置全局选项，包括标题选项，用于设置图表的标题。

其中通过滑块组件得到的mon值改变图表和标题

*#月平均价格柱状图*

*def* Price\_Bar(*df*):

    title\_len = df.groupby(['Month'])['Price'].mean().round(2)  *#设置小数点后两位*

*#转化为列表*

    agg\_price=title\_len.values.tolist()

    bar = Bar()

    bar.add\_xaxis(['1月','2月','3月','4月','5月','6月','7月','8月','9月','10月','11月','12月'])

    bar.add\_yaxis("售价$(单位:美元)",agg\_price)

    bar.set\_global\_opts(

*title\_opts*=opts.TitleOpts(

*title*="月平均售价$(美元)"),

*toolbox\_opts*=opts.ToolboxOpts(),

    )

    return bar

月平均价格柱状图，使用groupby方法以Month分组，选取Price，求均值，也就是平均价格，用tolist()方法转换为列表。使用bar.add\_xaxis() 方法设置 x 轴的数据，即月份。

使用 bar.add\_yaxis() 方法将月平均价格的数据添加到图表中，设置图例名称为 "售价$(单位:美元)"。

使用 bar.set\_global\_opts() 方法设置全局选项，包括标题选项和工具箱选项。

返回生成的柱状图对象 bar。

设置x轴为1到12月，y轴即为转换后的列表数据，最后return bar对象方便组件调用

折线图同理，仅仅是修改了一些变量

*#月价格散点图*

*def* Price\_sca(*df*):

    X=df['Date']

    Y=df['Price']

    plt.style.use('seaborn')

    plt.rcParams['font.family'] = ['Microsoft YaHei']

    plt.figure(*figsize*=(15, 5))

    plt.scatter(X,Y, *s*=12000/(df.index+200), *alpha*=.7,*cmap*=plt.get\_cmap('RdYlBu'))

    ax = plt.gca()

    tick\_spacing = 30

    ax.xaxis.set\_major\_locator(ticker.MultipleLocator(tick\_spacing))

    plt.xlabel('时间轴(1~12月)',*fontsize*=20)

    plt.ylabel('价格',*fontsize*=20)

    st.pyplot()

月价格散点图，以Date列数据作为X轴，Price数据作为Y轴，分别对应Datetime格式的日期和平均售价，style设置主题，设置字体，figsize设置画幅大小使用 scatter() 方法绘制散点图，其中 X 是横坐标数据，Y 是纵坐标数据。根据数据点的索引值进行 s 参数的计算，用于设置散点的大小。使用 alpha 参数设置散点的透明度，使用 cmap 参数设置散点的颜色映射设置 x 轴主刻度的间隔为 30 天，并设置 x 轴和 y 轴的标签。。最后设置XY轴标签，作图

option1 = st.selectbox('选择要显示的图表',('柱状图', '折线图','散点图'))

if option1=="柱状图":

    st\_pyecharts(Price\_Bar(df))

elif option1=="折线图":

    st\_pyecharts(Price\_Line(df))

elif option1=="散点图":

    Price\_sca(df)

单选组件selectbox，通过if和elif检查输入，并调用函数输出对应图像

月发行量折线图和柱状图与 月平均售价的两张图类似，为节省篇幅不贴出

下面是圆环图

*def* Game\_Pie(*df*):

    game\_count=df.groupby(['Month'])['Game'].count()

    counts=game\_count.values.tolist()

    mon=['1月','2月','3月','4月','5月','6月','7月','8月','9月','10月','11月','12月']

    pie=Pie()

    pie.add("",[*list*(z) for z in zip(mon,counts)],*center*=["75%", "50%"],*radius*=["40%", "75%"])

    pie.set\_global\_opts(*title\_opts*=opts.TitleOpts(*title*="月发行数-圆环图"),

*legend\_opts*=opts.LegendOpts(*orient*="vertical", *pos\_top*="15%", *pos\_left*="2%"))

    pie.set\_series\_opts(*label\_opts*=opts.LabelOpts(*formatter*="{b}: {c}"))

    return pie

月圆环图（饼图），使用groupby方法对Month分组，指定Game列并count()计数，获取发行总量。mon设置各块标签，add中以for循环读取数据，center设置位置,radius设置半径大小，通过set\_global\_opts的orient和pos参数决定圆环，最后返回pie对象方便后续组件调用

Radio组件和单选组件代码类似，仅在附录中贴出

*#发行商与开发商词云*

*def* Dev\_Wordcloud(*df*):

    text=''

    for line in df['Developer']:

        text += ' '.join(jieba.cut(*str*(line), *cut\_all*=False))

        text +=' '

    wc = WordCloud(

*background\_color*='#003366',

*font\_path*='C:\Windows\Fonts\msyh.ttc',

*max\_words*=1000,

*max\_font\_size*=150,

*min\_font\_size*=10,

*width*=960,

*height*=540,

*prefer\_horizontal*=1,

*random\_state*=50,

    )

    wc.generate\_from\_text(text)

    plt.imshow(wc,*interpolation*='bilinear')

    plt.axis('off')

    st.markdown("开发商词云")

    st.pyplot()

词云，仅展示开发商词云，其他两个词云仅修改了[‘Developer’]参数

将Dateframe中的Developer数据循环写入字符串，用jieba库分词遍历数据框中的开发商信息，将每个开发商的文本进行分词处理，并拼接到 text 字符串中。

Font\_path设置字体路径，max\_words设置字数上限,max\_font\_size和min设置字体大小范围，width和height设置画幅大小，使用 imshow() 方法绘制词云图，并设置插值方法为双线性插值。使用 axis('off') 方法关闭坐标轴的显示。最后绘图

option2 = st.multiselect(

    '请选择要展示的词云',

    ['开发商', '发行商', '游戏'],

    ['发行商'])

多选组件multiselect，由于if和elif太多了，完整代码在附录

function (params) {

            if (params.value > 0 && params.value < 86) {

                return 'red';

            } else if (params.value > 87 && params.value < 90) {

                return 'blue';

            }

            return 'green';

        }

        """

前面柱状图的颜色变化设置函数，分别为三个等级，小于86的设置红色，87和90之间的设置蓝色，大于90的设置绿色，其它2个同理

,*itemstyle\_opts*=opts.ItemStyleOpts(*color*=JsCode(color\_function))

设置颜色函数参数

## 总结

这次大作业不仅考验python可视化相关的技术和能力，同样也帮助同学们学习了python爬虫和数据清洗相关知识，更是使我深入理解了python的数据结构和文件操作。此次课题我挑选了自己感兴趣的方面进行工作，由于对js等前端技术使用较少，故选取了streamlit库和适配pyecharts的streamlit\_echarts库进行Web的制作，遇到了很多困难、报错等，花费了大量时间在论坛求助、在搜索引擎查询。但最终得到的成果给与了我很大的成就感，本次小学期的课程使我受益匪浅。

# 附录

from pyecharts.charts import Bar,Pie,Line,Liquid,Bar3D

from pyecharts.commons.utils import JsCode

from pyecharts import options as opts

import streamlit\_echarts as ste

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.ticker as ticker

import numpy as np

import streamlit as st

import os

from streamlit\_echarts import st\_pyecharts

from PIL import Image

from wordcloud import WordCloud,ImageColorGenerator

import jieba

import time

*#一些没用的侧边栏小组件,只是为了让页面布局更好看*

with st.sidebar:

    st.metric(*label*="系统时间：", *value*=time.strftime('%Y-%m-%d %H:%M:%S', time.localtime()))

    if st.button('雪花'):

        st.snow()

    if st.button('气球'):

        st.balloons()

*#We will remove the ability to do this as it requires the use of Matplotlib's global figure object, which is not thread-safe.*

st.set\_option('deprecation.showPyplotGlobalUse', False)  *#消除警报*

*#设置页面图片*

image = Image.open('steam.jpg')

st.image(image, *caption*='2021Year')

st.title('2021年steam游戏发售数据可视化')

st.subheader("基于streanlit的可视化数据")

*#获取路径*

subpath=os.path.abspath(".")+"/2021 - Year Stats.csv"

df = pd.read\_csv(subpath, *index\_col*=0, *encoding*='gbk')

df['Price'] = pd.to\_numeric(df['Price'])

df\_re = df[df['Review'].values!=0].copy()  *#拷贝一份好评不为零(存在评价)的*

df\_mon=df.groupby(['Month'])

*#排名前50的游戏拥有数与价格随时间变化*

*def* tDBar(*df*):

*#mon = st.slider('请选择时间', 1, 12, 1)*

    get=df.groupby(['Month']).get\_group(mon).head(50)

    game=get['Game']

    game\_list=game.values.tolist()

    owners=get['Owners']

    owner\_list=owners.values.tolist()

    price=get['Price']

    price\_list=sorted(price.values.tolist())

    data=[]

    for index, row in get.iterrows():

        x = row['Price']

        y = row['Game']

        z = row['Owners']

        data.append([x, y, z])

    bar3d=Bar3D()

    bar3d.add(

*series\_name*="拥有人数",

*data*=data,

*xaxis3d\_opts*=opts.Axis3DOpts(*type\_*="category", *data*=price\_list),

*yaxis3d\_opts*=opts.Axis3DOpts(*type\_*="category", *data*=game\_list),

*zaxis3d\_opts*=opts.Axis3DOpts(*type\_*="value"),

    )

    bar3d.set\_global\_opts(

*visualmap\_opts*=opts.VisualMapOpts(

*max\_*=20,

*range\_color*=[

                "#313695",

                "#4575b4",

                "#74add1",

                "#abd9e9",

                "#e0f3f8",

                "#ffffbf",

                "#fee090",

                "#fdae61",

                "#f46d43",

                "#d73027",

                "#a50026",

            ],))

    st\_pyecharts(bar3d)

st.markdown("## :mag\_right:月排名前50游戏的拥有数与价格3D图")

mon = st.slider('请选择时间(月)', 1, 12, 1)

tDBar(df)

*#评价信息*

*def* Review\_Timeline(*df*):

    re\_liquid=df.groupby(['Month'])['Review'].mean()

    alist=re\_liquid.values.tolist()

    new\_list = [x/100 for x in alist]

    mid\_np = np.array(new\_list)

    mid\_np\_4f = np.round(mid\_np,4)

    n2\_list = *list*(mid\_np\_4f)     *#列表处理，列表转np数组再转列表*

*#mon = st.slider('请选择时间', 1, 12, 1)*

    want=n2\_list[mon-1]

    liquid=Liquid()

    liquid.add("好评率", [want],

*label\_opts*=opts.LabelOpts(

*font\_size*=35,

*formatter*=JsCode(

                """function (param) {

                    return (Math.floor(param.value \* 10000) / 100) + '%';

                }"""            *#设置精度*

            )))

    liquid.set\_global\_opts(*title\_opts*=opts.TitleOpts(*title*="{}月好评率-水球图".format(mon)))

    st\_pyecharts(liquid)

st.markdown("## :+1:综合好评率")

Review\_Timeline(df\_re)

st.divider()

*#月平均价格柱状图*

*def* Price\_Bar(*df*):

    title\_len = df.groupby(['Month'])['Price'].mean().round(2)  *#设置小数点后两位*

*#转化为列表*

    agg\_price=title\_len.values.tolist()

    bar = Bar()

    bar.add\_xaxis(['1月','2月','3月','4月','5月','6月','7月','8月','9月','10月','11月','12月'])

    bar.add\_yaxis("售价$(单位:美元)",agg\_price)

    bar.set\_global\_opts(

*title\_opts*=opts.TitleOpts(

*title*="月平均售价$(美元)"),

*toolbox\_opts*=opts.ToolboxOpts(),

    )

    return bar

*#月平均价格折线图*

*def* Price\_Line(*df*):

    title\_len = df.groupby(['Month'])['Price'].mean().round(2)  *#设置小数点后两位*

*#转化为列表*

    agg\_price=title\_len.values.tolist()

    line = Line()

    line.add\_xaxis(['1月','2月','3月','4月','5月','6月','7月','8月','9月','10月','11月','12月'])

    line.add\_yaxis("售价$(单位:美元)",agg\_price)

    line.set\_global\_opts(

*title\_opts*=opts.TitleOpts(

*title*="月平均售价$(美元)"),

*toolbox\_opts*=opts.ToolboxOpts(),

    )

    return line

*#月价格散点图*

*def* Price\_sca(*df*):

    X=df['Date']

    Y=df['Price']

    plt.style.use('seaborn')

    plt.rcParams['font.family'] = ['Microsoft YaHei']

    plt.figure(*figsize*=(15, 5))

    plt.scatter(X,Y, *s*=12000/(df.index+200), *alpha*=.7,*cmap*=plt.get\_cmap('RdYlBu'))

    ax = plt.gca()

    tick\_spacing = 30

    ax.xaxis.set\_major\_locator(ticker.MultipleLocator(tick\_spacing))

    plt.xlabel('时间轴(1~12月)',*fontsize*=20)

    plt.ylabel('价格',*fontsize*=20)

    st.pyplot()

st.markdown("## :moneybag:月售价图表")

option1 = st.selectbox('选择要显示的图表',('柱状图', '折线图','散点图'))

if option1=="柱状图":

    st\_pyecharts(Price\_Bar(df))

elif option1=="折线图":

    st\_pyecharts(Price\_Line(df))

elif option1=="散点图":

    Price\_sca(df)

*#分线*

st.divider()

*#月游戏发行数量*

*def* Game\_Bar(*df*):

    game\_count=df.groupby(['Month'])['Game'].count()

    counts=game\_count.values.tolist()

    bar = Bar()

    bar.add\_xaxis(['1月','2月','3月','4月','5月','6月','7月','8月','9月','10月','11月','12月'])

    bar.add\_yaxis("发行数",counts)

    bar.set\_global\_opts(

*title\_opts*=opts.TitleOpts(

*title*="月发行数-柱状图"),

*toolbox\_opts*=opts.ToolboxOpts(),

    )

    return bar

*def* Game\_Line(*df*):

    game\_count = df.groupby(['Month'])['Game'].count()

    counts=game\_count.values.tolist()

    line = Line()

    line.add\_xaxis(['1月','2月','3月','4月','5月','6月','7月','8月','9月','10月','11月','12月'])

    line.add\_yaxis("发行数",counts)

    line.set\_global\_opts(

*title\_opts*=opts.TitleOpts(

*title*="月发行数-折线图"),

*toolbox\_opts*=opts.ToolboxOpts(),

    )

    return line

*def* Game\_Pie(*df*):

    game\_count=df.groupby(['Month'])['Game'].count()

    counts=game\_count.values.tolist()

    mon=['1月','2月','3月','4月','5月','6月','7月','8月','9月','10月','11月','12月']

    pie=Pie()

    pie.add("",[*list*(z) for z in zip(mon,counts)],*center*=["75%", "50%"],*radius*=["40%", "75%"])

    pie.set\_global\_opts(*title\_opts*=opts.TitleOpts(*title*="月发行数-圆环图"),

*legend\_opts*=opts.LegendOpts(*orient*="vertical", *pos\_top*="15%", *pos\_left*="2%"))

    pie.set\_series\_opts(*label\_opts*=opts.LabelOpts(*formatter*="{b}: {c}"))

    return pie

st.markdown("## :bulb:月发行数图表")

genre = st.radio('选择要显示的图表',('柱状图', '折线图','圆环图'))

if genre=="柱状图":

    st\_pyecharts(Game\_Bar(df))

elif genre=="折线图":

    st\_pyecharts(Game\_Line(df))

elif genre=="圆环图":

    st\_pyecharts(Game\_Pie(df))

st.divider()

*#发行商与开发商词云*

*def* Dev\_Wordcloud(*df*):

    text=''

    for line in df['Developer']:

        text += ' '.join(jieba.cut(*str*(line), *cut\_all*=False))

        text +=' '

    wc = WordCloud(

*background\_color*='#003366',

*font\_path*='C:\Windows\Fonts\msyh.ttc',

*max\_words*=1000,

*max\_font\_size*=150,

*min\_font\_size*=10,

*width*=960,

*height*=540,

*prefer\_horizontal*=1,

*random\_state*=50,

    )

    wc.generate\_from\_text(text)

    plt.imshow(wc,*interpolation*='bilinear')

    plt.axis('off')

    st.markdown("开发商词云")

    st.pyplot()

*def* Pub\_Wordcloud(*df*):

    text=''

    for line in df['Publisher']:

        text += ' '.join(jieba.cut(*str*(line), *cut\_all*=False))

        text +=' '

    wc = WordCloud(

*background\_color*='#003366',

*font\_path*='C:\Windows\Fonts\msyh.ttc',

*max\_words*=1000,

*max\_font\_size*=150,

*min\_font\_size*=10,

*width*=960,

*height*=540,

*prefer\_horizontal*=1,

*random\_state*=50,

    )

    wc.generate\_from\_text(text)

    plt.imshow(wc,*interpolation*='bilinear')

    plt.axis('off')

    st.markdown("发行商词云")

    st.pyplot()

*#游戏词云*

*def* Game\_Wordcloud(*df*):

    text=''

    for line in df['Game']:

        text += ' '.join(jieba.cut(*str*(line), *cut\_all*=False))

        text +=' '

    wc = WordCloud(

*background\_color*='#003366',

*font\_path*='C:\Windows\Fonts\msyh.ttc',

*max\_words*=1000,

*max\_font\_size*=150,

*min\_font\_size*=10,

*width*=960,

*height*=540,

*prefer\_horizontal*=1,

*random\_state*=50,

    )

    wc.generate\_from\_text(text)

    plt.imshow(wc,*interpolation*='bilinear')

    plt.axis('off')

    st.markdown("游戏词云")

    st.pyplot()

st.markdown("## :scroll:词云")

option2 = st.multiselect(

    '请选择要展示的词云',

    ['开发商', '发行商', '游戏'],

    ['发行商'])

if option2==['开发商']:

    Dev\_Wordcloud(df)

elif option2==['发行商']:

    Pub\_Wordcloud(df)

elif option2==['游戏']:

    Game\_Wordcloud(df)

elif option2==['开发商','发行商']:

    Dev\_Wordcloud(df)

    Pub\_Wordcloud(df)

elif option2==['发行商','开发商']:

    Pub\_Wordcloud(df)

    Dev\_Wordcloud(df)

elif option2==['发行商','游戏']:

    Pub\_Wordcloud(df)

    Game\_Wordcloud(df)

elif option2==['游戏','发行商']:

    Game\_Wordcloud(df)

    Pub\_Wordcloud(df)

elif option2==['开发商','游戏']:

    Dev\_Wordcloud(df)

    Game\_Wordcloud(df)

elif option2==['游戏','开发商']:

    Game\_Wordcloud(df)

    Dev\_Wordcloud(df)

else:

    Dev\_Wordcloud(df)

    Pub\_Wordcloud(df)

    Game\_Wordcloud(df)