

**《Python编程与实践》**

**期末课程设计**

**——基于Altair的新冠疫情数据可视化处理**

图片包含 游戏机, 标志, 盘子

描述已自动生成

**班 级 2019211120**

**姓 名 李源**

**学 号 2019210621**

1. **年 6 月**

**描述：基于Altair的利用selection和condition架构以及chart对象完成新冠疫情数据动态可视化处理设计**

**一、项目研究目的和框架设计**

### ****1.1研究目的****

自COVID-19疫情出现以来，疫情已经在全球多地爆发。为了能够更好地直观了解和分析新冠肺炎疫情发展状态，出现了很多利用python处理并分析疫情数据的优秀案例，例如专门用于生成covid-19报告和可视化的python包——“covidify”。

本研究是基于Altair数据可视化处理，制作出疫情地图以及主要参数的统计图表以供研究人员更好的了解和掌握疫情有关动态。

### ****1.2整体实现的描述****

由于本项目主要可以分为两大部分即绘制中国疫情图表以及全球疫情分布图，分别绘制了中国累计确诊和疑似人数折线图，以及海外各国相关的柱状图和全球确诊人数热度图。前两个均采用了Altair本身所有的selection和condition架构，使得某些属性的值可以根据不同的condition做出不同的响应。

构建一个selection根据鼠标的移动来做出相应的反应，有一条竖线指示我们当前所在的x轴日期点，并且这个竖线会随我们的鼠标移动而在x轴上移动，并且在竖线于折线的交点上显示对应的人数。并且折线太多时会影响我们查看数据，所以我们还需要一个筛选功能，当我们点击对应图例时，只会显示当前图例对应的折线，其它折线显灰。

这也使得我们所构建的这两个图不仅仅是静态的图表而更是可实现动态的可交互的可视化图表，可以优化使用者的体验并且使得用户可以更好的查看数据。对于最后一个图则是在附录的连接中补充了对应的地图的改动可经行细节处的优化与设计。

**二、项目实现**

### ****2.1实现方式和可行性****

第一步，获得国内以及全球与其相关的准确数据。由于本项目主要研究重点为对数据的可视化处理且主要重点是代码的逻辑性和可重用性，所以直接利用现有的数据库进行有关处理分析。一方面，从国家卫生健康委员会网站提供的接口API可以实时获得国内的疫情情况。另一方面，利用GitHub上有关研究人员从国外网站获取的实时数据成果来完善对全球疫情统计数据的补充。

第二步，使用pandas的DataFrame结构读取并保存已经储存在本地的csv文件中的数据。由于长型DataFrame每有一个观测值就有一行，元数据被作为值被记录在表中。并且每一行只包含一个观测值，还有伴随这个观察值的元数据。在这里观察值是人数，时间和类型是伴随着这个观察值的元数据。最重要的是列和索引标签不再包含任何有用的索引信息,这将会大大的增加我们的效率。最后可以将数据以csv以及获得到的json格式的url字符串储存在本地以供进一步绘图使用。

第三步，根据疫情有关的数据绘制相关折线图、散点图、柱状图等。绘制图表时需要使用具体数据来绘制图表，需要把数据传递给altair.Char。在Altair中，基本的图表对象是altair.Chart。将分别使mark\_line()函数来绘制折线图、mark\_point来绘制散点图、mark\_bar来绘制柱状图、mark\_geoshape()绘制地理地图。

根据Altair的官方文档，所有最高层级的chart对象，都可以接受一个数据集作为其第一个参数。其数据集可以为以下几种类型：

1. pandas 的 DataFrame 类型；
2. Data 或者跟 Data 相关的对象 (例如： UrlData, InlineData, NamedData)；
3. 一个指向 json 或者 csv 格式的url字符串，或者支持 geo\_interface 接口的对象（例如： geopandas 的 GeoDataFrame，shapely 的 Geometries，GeoJson）；

所以在使用pandas 的 DataFrame 类型读入并储存数据后，使用Altair有关的模块经行画图实现可视化。Altair在使用长型结构的DataFrame时可以有更好地表现，推荐使用长型，所以后面我们构建传递给Chart的DataFrame时都是构建长型的。在绘制图时，我们还需要告诉Chart哪些数据是作为x轴，哪些是作为y轴，在Altair中我们可以通过调用Chart对象的encode方法并往里面传递参数来对我们的x轴、y轴等进行设置，并完成绘图和染色。

### ****2.3核心代码****

#### 2.3.1nearest选择器

nearest 是一个selection，字面意思就是选择器，可以给它一个名字方便后续使用。也可以指定类型type，是单选择器还是多选择器选择器可以看作一种选择判断、一种根据情况做出不同显示的类。nearst=True表明选择最近的点，on是指定绑定什么事件，mouseover即鼠标移动，fields指定绑定的字段，指定X轴，即根据X轴取值,根据不同的date值有不同的情况。

nearest = alt.selection\_multi(name='nearest', nearest=True, on='mouseover',fields=['date'],empty='none')

line= alt.Chart(chinaConfAndSusDf,width=800,height=300).mark\_line().encode(

x=alt.X('date', axis=alt.Axis(title='日期',titleFontSize=16,titlePadding=20,grid=True,labelFontSize=14)),

y=alt.Y('count', axis=alt.Axis( title='人数',titleFontSize=16,titleColor='#FF9988',titlePadding=20)),

color=alt.Color('symbol',legend=alt.Legend(title=None,titleFontSize=16,titleOpacity=0.5,titlePadding=20,labelFontSize=16,rowPadding=10,labelOpacity=0.5,offset=60), scale=alt.Scale(domain=['累计确诊','累计疑似'], range=['#FF4433','#FF9933']))

)

points = line.mark\_point(strokeWidth=5).encode(

x=alt.X('date', axis=alt.Axis(title='日期',titleFontSize=16,titlePadding=20,grid=True,labelFontSize=14)),

y=alt.Y('count', axis=alt.Axis( title='人数',titleFontSize=16,titleColor='#FF9988',titlePadding=20)),

opacity=alt.condition(nearest, alt.value(1), alt.value(0)),

).add\_selection(nearest)

text = line.mark\_text(align='left', dx=5, dy=-15,fontSize=15, fontWeight='bold').encode(text=alt.condition(nearest, 'count', alt.value(' ')),)

rules = alt.Chart(chinaConfAndSusDf).mark\_rule(strokeWidth=2,color='gray',opacity=0.2).encode(x=alt.X('date')

).transform\_filter(nearest)

alt.layer(line,rules,points,text)

nearest = alt.selection(name='nearest',type='single', nearest=True, on='mouseover',fields=['date'],empty='none')

#### 2.3.2selCur选择器

selCur 也是一个selection，字面意思就是选择器，可以给它一个名字方便后续使用，fields指定绑定的字段，bind就是这个选择器绑定在legend上，同时根据字段‘symbol’进行区分即是根据不同的symbol值有不同的情况。

selCur = alt.selection\_multi(name='selCur',fields=['symbol'],bind='legend')

#### 2.3.3color类

color类可以为我们的曲线，图例设置不同的颜色，区分曲线，可以用一个legend类构造color的legend。

各种属性名字很好地表现出了它的作用scale属性可以用一个Scale类来实例化，domain指定了颜色作用的“域”，可以根据domain来区分不同的曲线，range可以分别设置颜色。在这里就是累计确诊用#FF4433颜色显示，累计疑似用#FF9933显示。

color = alt.Color('symbol:N',legend=alt.Legend(title=None,titleFontSize=16,titleOpacity=0.5,titlePadding=20,labelFontSize=16,rowPadding=10,labelOpacity=0.5,offset=60),scale=alt.Scale(domain=['累计确诊','累计疑似','累计死亡'], range=['#FF4433','#FF9933','#112233']))

#### 2.3.4color类

encode方法里面面的x属性是设置x轴alt.X是X轴的类，可以指定用那些列作为我们的x轴，axis是对坐标轴进行设计。同理在X中的date是列明，指定date做X轴由于我们的是TimeStamp类型，是一个日期格式，所以可以用monthdata来只显示月日。

line = alt.Chart(chinaConfAndSusDf).mark\_line().encode(

x=alt.X('monthdate(date):T', axis=alt.Axis(title='日期',titleFontSize=16,titlePadding=20,grid=True,labelFontSize=14)),

y=alt.Y('count', axis=alt.Axis( title='人数',titleFontSize=16,titleColor='#FF9988',titlePadding=20)),

# condition 跟selection配合使用，第一个值为selection，第二个值是True时的值，即被选中时的值，第三个Fasle值

color=alt.condition(selCur,color,alt.value('lightgray')),

strokeWidth=alt.condition(selCur,alt.value(5),alt.value(3))

)

#### 2.3.5利用选择器挂载

将选择器挂载到我们的图上

selectors = alt.Chart(chinaConfAndSusDf).mark\_point(strokeWidth=3).encode(

x=alt.X('monthdate(date):T'),

opacity=alt.value(0),

).add\_selection(nearest)

points = line.mark\_point(strokeWidth=5).encode(

opacity=alt.condition(nearest, alt.value(1), alt.value(0)),

size=alt.condition(selCur,alt.value(3),alt.value(0)),

).add\_selection(selCur)

text = line.mark\_text(align='left', dx=5, dy=-15,fontSize=15, fontWeight='bold').encode(

text=alt.condition(nearest, 'count', alt.value(' ')),

opacity=alt.condition(selCur, alt.value(1), alt.value(0)),

)

rules = alt.Chart(chinaConfAndSusDf).mark\_rule(strokeWidth=2,color='gray',opacity=0.2).encode(

x=alt.X('monthdate(date):T')).transform\_filter(nearest)

alt.layer(line, selectors, points, rules, text).properties(

width=1200, height=350,

title= {

"text": "COVID-19 中国 累计确诊人数和累计疑似人数折线图",

"fontSize": 20,

'offset':20

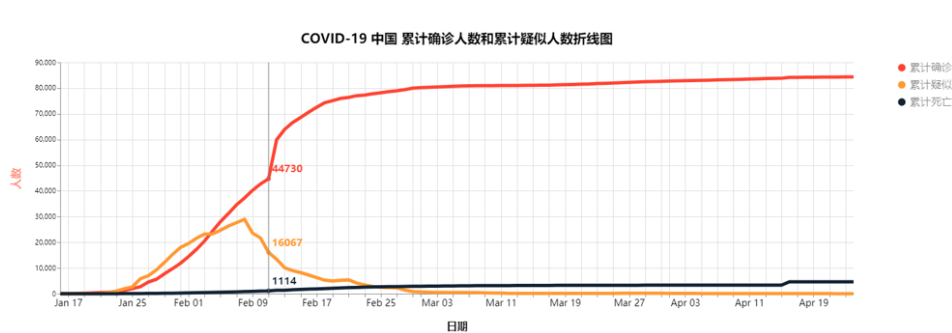
}

).configure\_view(stroke=None)

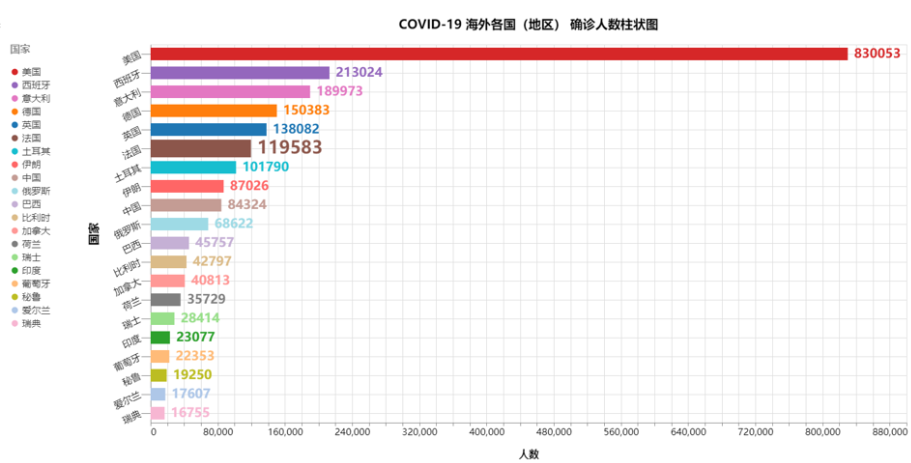
1. **项目结果**

**COVID-19 中国 累计确诊人数和累计疑似人数折线图**

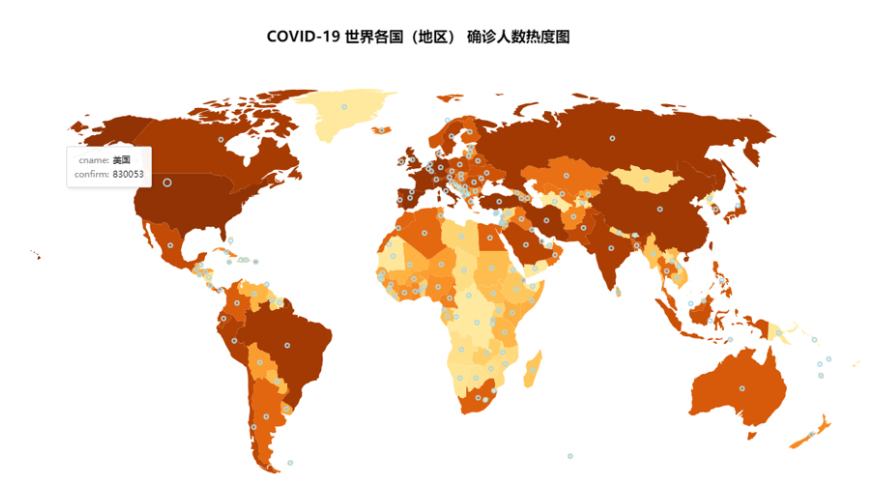
注：使用效果体验类似于百度疫情相关折线图

****

**COVID-19 海外各国（地区）确诊人数柱状图**

****

**COVID-19 世界各国（地区）确诊人数热度图**

****

**四、总结和展望**

本项目基于Altair的利用selection和condition架构以及chart对象完成新冠疫情数据动态可视化处理设计。并成功地绘制出了对应的三大类型的可视化统计图，也对Altair绘画技巧有了一定的掌握。但是一个项目的优劣不仅仅取决于他的本身现在效果，更体现在不断地迭代更新和功能的模块。

在这种数据分析的背后也可以利用着一些数据统计图并结合医学传染病相关原理来更好地建立各种模型来预测疫情的发展以及防控安排。也希望利用一些python编程的技能可以更好地帮助防疫人员更加直观方便地处理数据，来达到更好地推动我们社会的发展。

**附录：**

1. **全球疫情数据库：https://raw.githubusercontent.com/datasets/covid-19/master/data/countries-aggregated.csv**
2. **本项目有关资料的GitHub链接：**