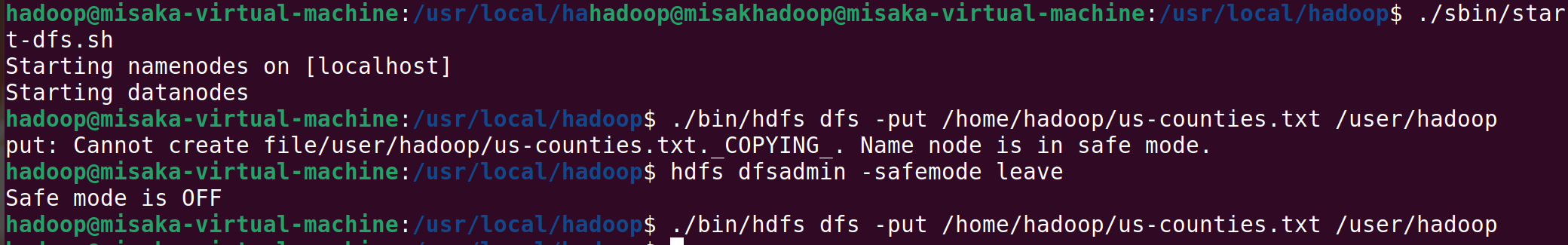
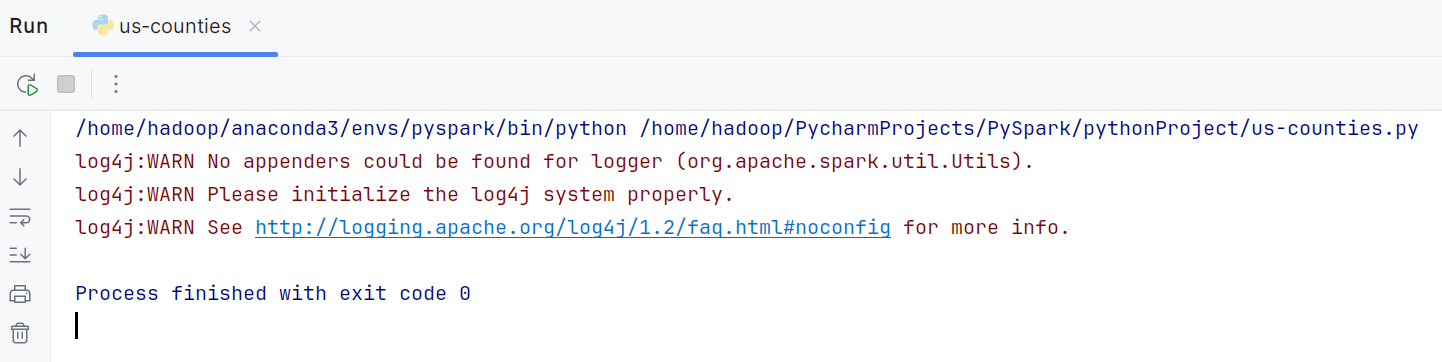
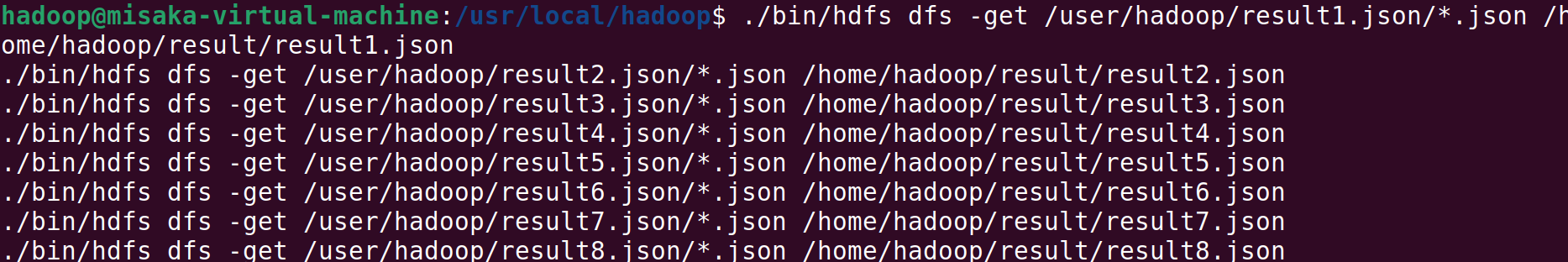
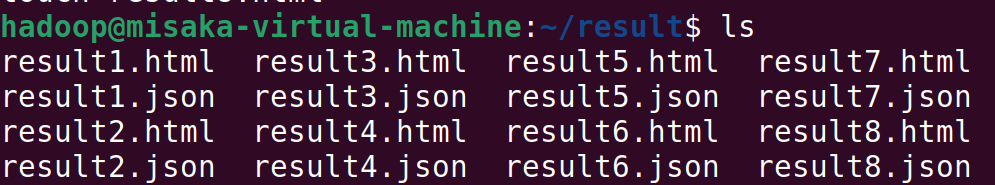
# 作业5---2020年美国新冠肺炎疫情数据分析

将厦门大学数据库实验室的博客文章 <https://dblab.xmu.edu.cn/blog/2636/> 里的代码运行通过，将数据可视化的结果分析截图贴上，要求不能跟文章的图片完全一样。

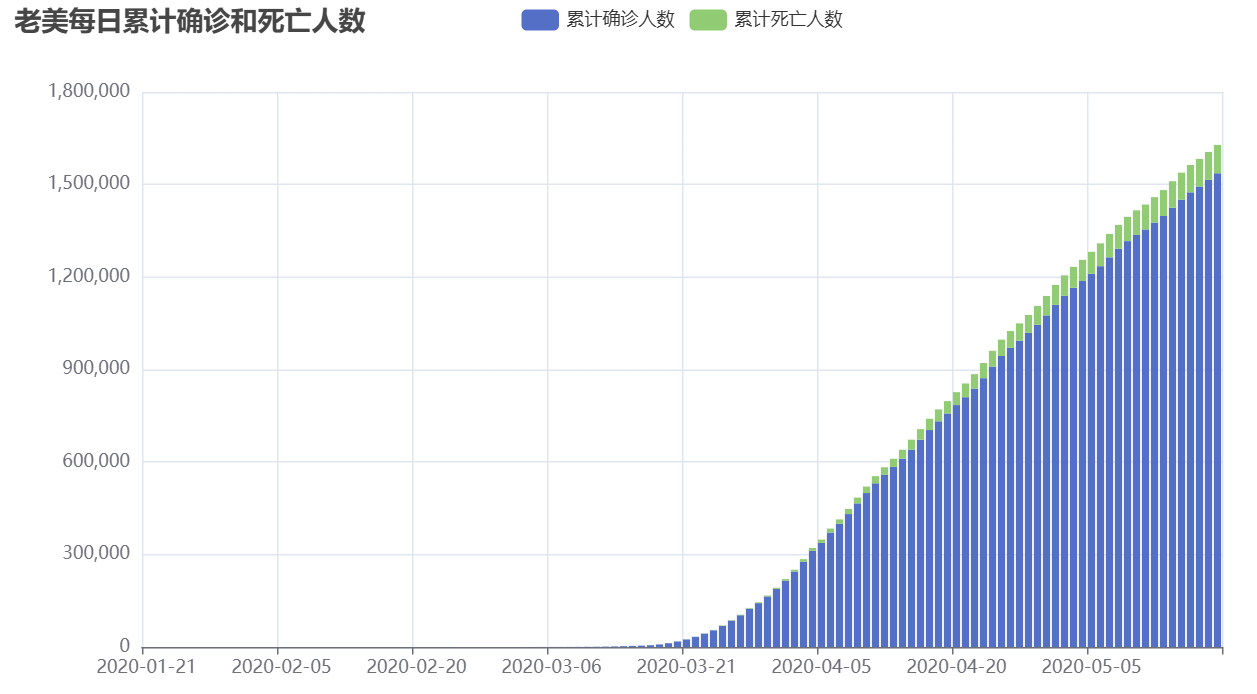
过程：

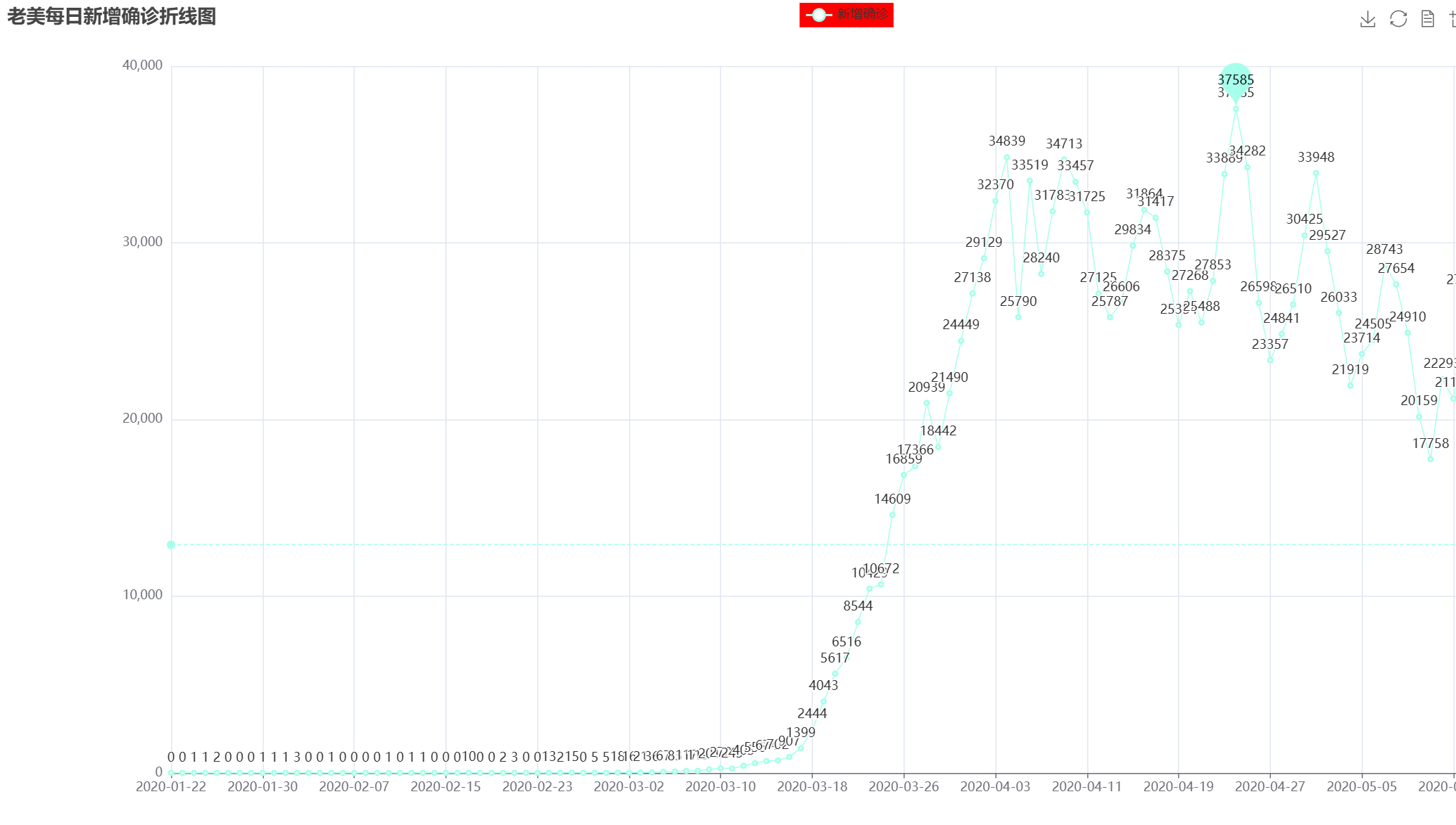
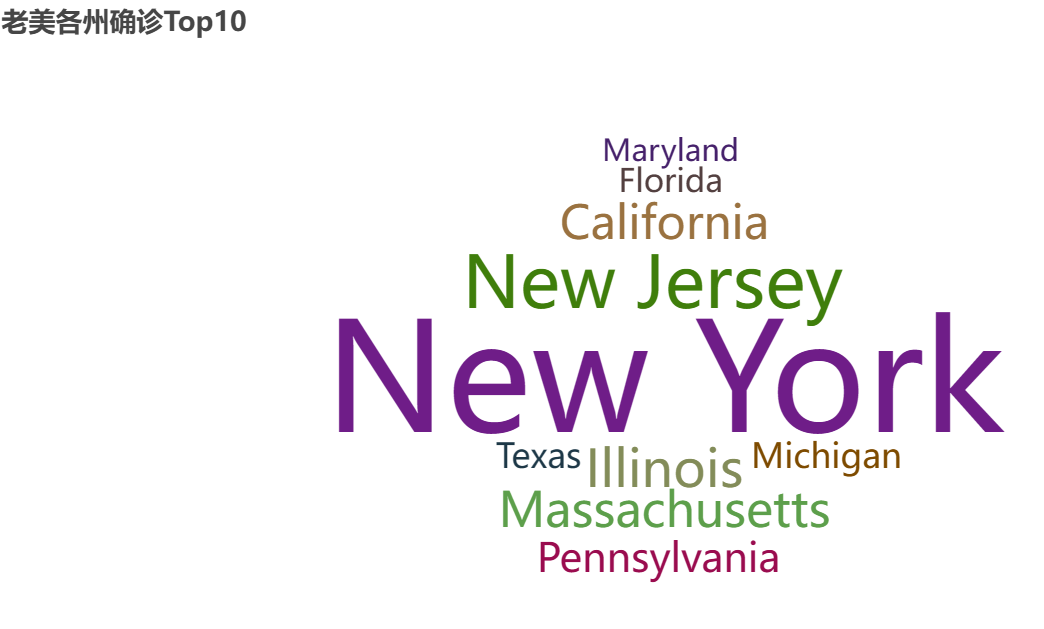


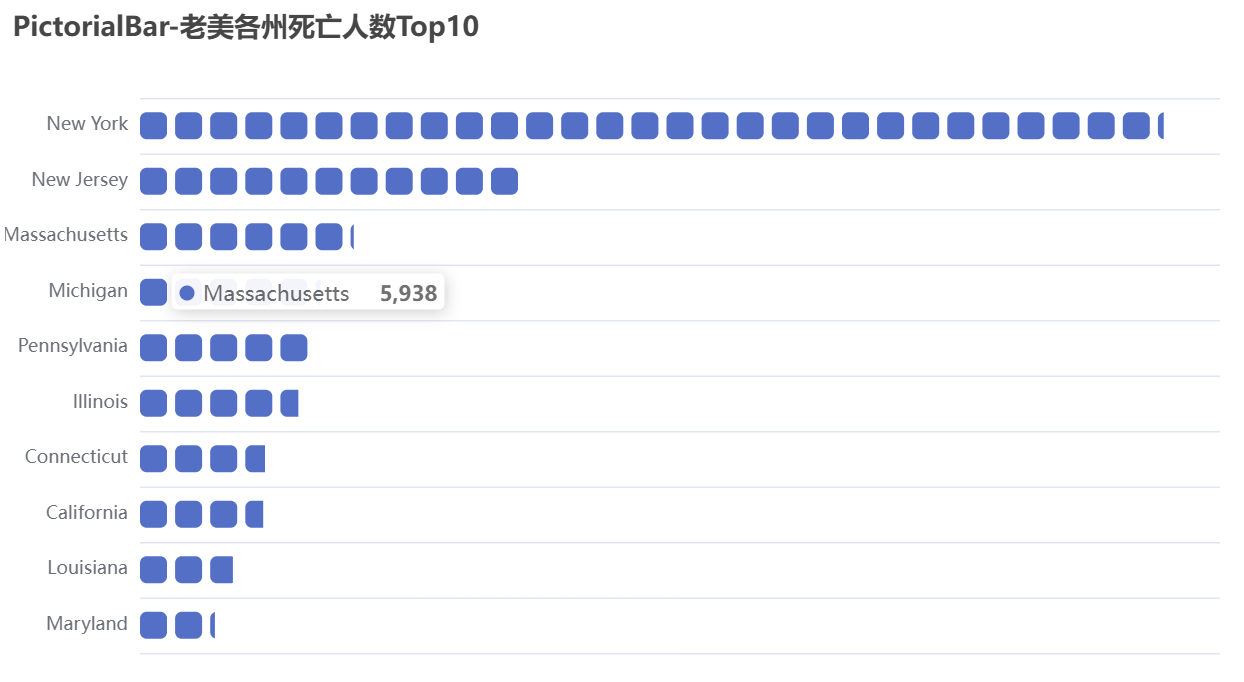
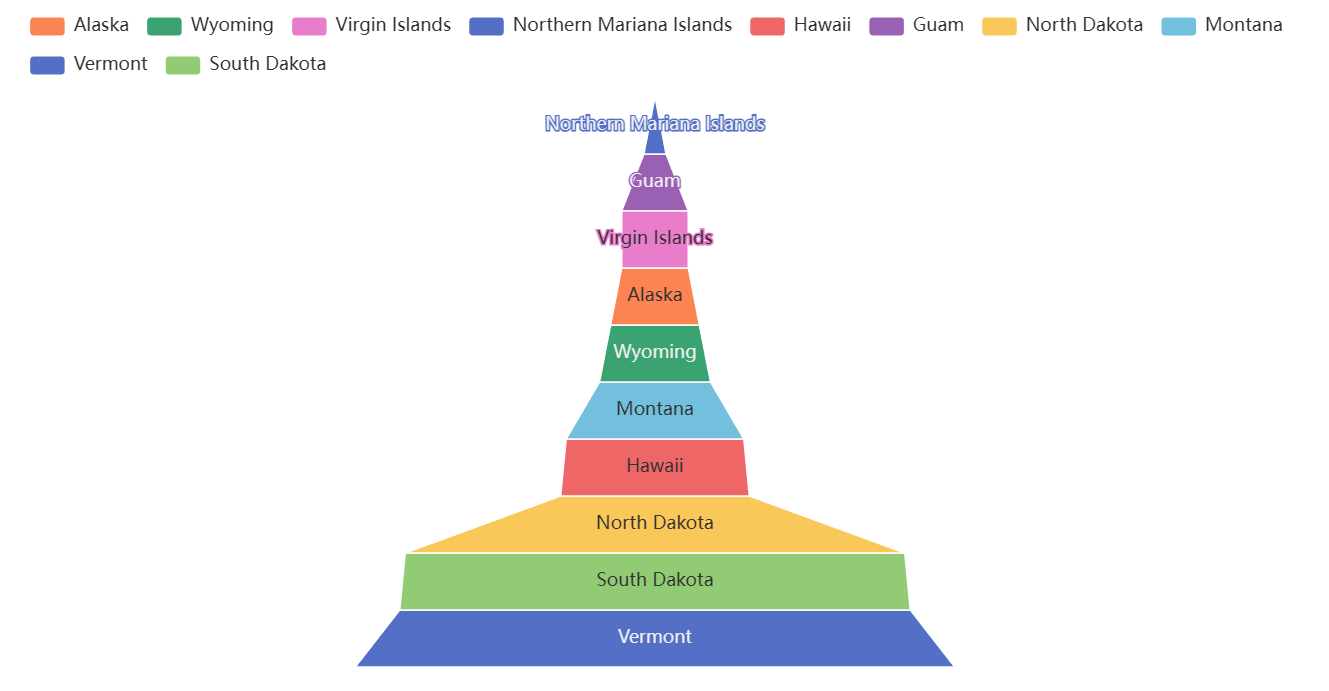


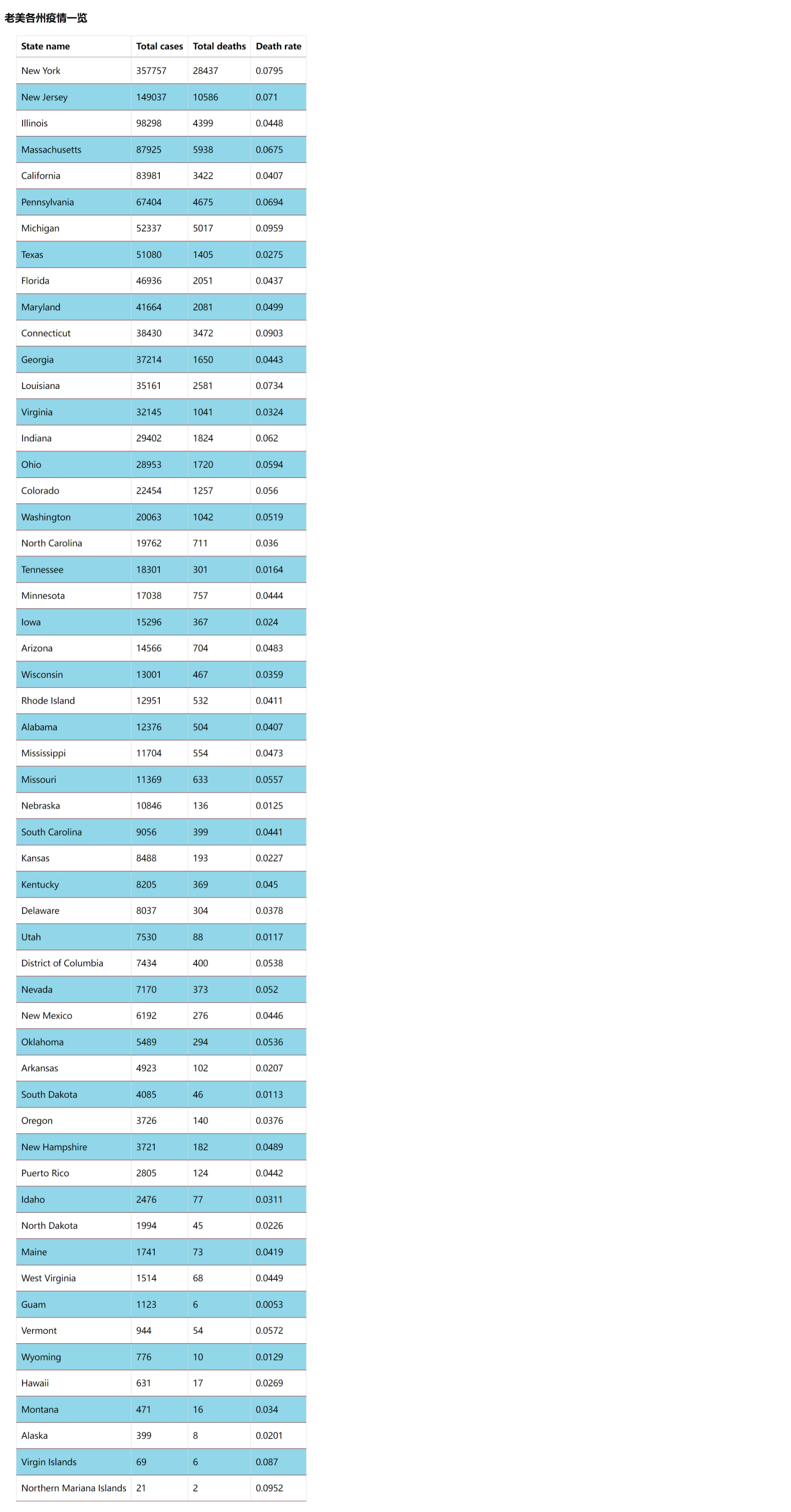


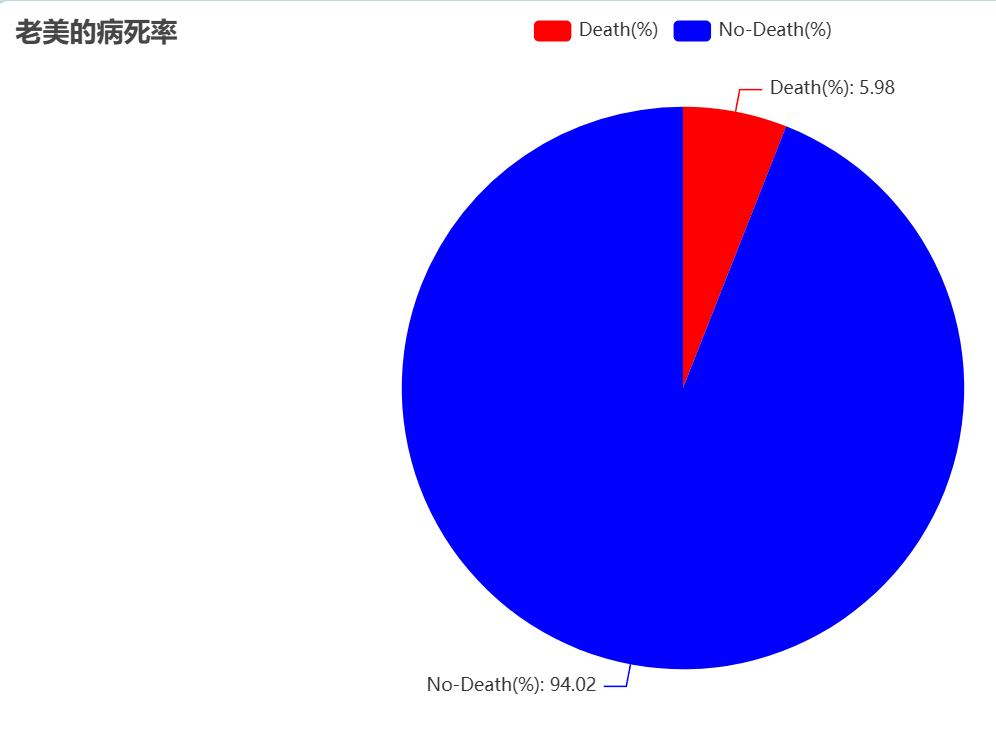
结果分析图：







代码：

分析：

1. **import** pandas as pd

4. # .csv->.txt
5. **def** csv2txt():
6. data = pd.read\_csv('/home/hadoop/us-counties.csv')
7. with open('/home/hadoop/us-counties.txt', 'a+', encoding='utf-8') as f:
8. **for** line **in** data.values:
9. f.write((str(line[0]) + '\t' + str(line[1]) + '\t'
10. + str(line[2]) + '\t' + str(line[3]) + '\t' + str(line[4]) + '\n'))

13. csv2txt()
15. **from** pyspark **import** SparkConf
16. **from** pyspark.sql.types **import** \*
17. **from** pyspark.sql **import** SparkSession
18. **from** datetime **import** datetime
19. **import** pyspark.sql.functions as func

22. **def** toDate(inputStr):
23. newStr = ""
24. **if** len(inputStr) == 8:
25. s1 = inputStr[0:4]
26. s2 = inputStr[5:6]
27. s3 = inputStr[7]
28. newStr = s1 + "-" + "0" + s2 + "-" + "0" + s3
29. **else**:
30. s1 = inputStr[0:4]
31. s2 = inputStr[5:6]
32. s3 = inputStr[7:]
33. newStr = s1 + "-" + "0" + s2 + "-" + s3
34. date = datetime.strptime(newStr, "%Y-%m-%d")
35. **return** date

38. # 主程序:
39. spark = SparkSession.builder.config(conf=SparkConf()).getOrCreate()
41. fields = [StructField("date", DateType(), False), StructField("county", StringType(), False),
42. StructField("state", StringType(), False),
43. StructField("cases", IntegerType(), False), StructField("deaths", IntegerType(), False), ]
44. schema = StructType(fields)
46. rdd0 = spark.sparkContext.textFile("/user/hadoop/us-counties.txt")
47. rdd1 = rdd0.map(**lambda** x: x.split("\t")).map(**lambda** p: Row(toDate(p[0]), p[1], p[2], int(p[3]), int(p[4])))
49. shemaUsInfo = spark.createDataFrame(rdd1, schema)
51. shemaUsInfo.createOrReplaceTempView("usInfo")
53. # 1.计算每日的累计确诊病例数和死亡数
54. df = shemaUsInfo.groupBy("date").agg(func.sum("cases"), func.sum("deaths")).sort(shemaUsInfo["date"].asc())
56. # 列重命名
57. df1 = df.withColumnRenamed("sum(cases)", "cases").withColumnRenamed("sum(deaths)", "deaths")
58. df1.repartition(1).write.json("result1.json", mode="overwrite")  # 写入hdfs
60. # 注册为临时表供下一步使用
61. df1.createOrReplaceTempView("ustotal")
63. # 2.计算每日较昨日的新增确诊病例数和死亡病例数
64. df2 = spark.sql(
65. "select t1.date,t1.cases-t2.cases as caseIncrease,t1.deaths-t2.deaths as deathIncrease from ustotal t1,ustotal t2 where t1.date = date\_add(t2.date,1)")
67. df2.sort(df2["date"].asc()).repartition(1).write.json("result2.json")  # 写入hdfs
69. # 3.统计截止5.19日 美国各州的累计确诊人数和死亡人数
70. df3 = spark.sql(
71. "select date,state,sum(cases) as totalCases,sum(deaths) as totalDeaths,round(sum(deaths)/sum(cases),4) as deathRate from usInfo  where date = to\_date('2020-05-19','yyyy-MM-dd') group by date,state")
73. df3.sort(df3["totalCases"].desc()).repartition(1).write.json("result3.json")  # 写入hdfs
75. df3.createOrReplaceTempView("eachStateInfo")
77. # 4.找出美国确诊最多的10个州
78. df4 = spark.sql("select date,state,totalCases from eachStateInfo  order by totalCases desc limit 10")
79. df4.repartition(1).write.json("result4.json")
81. # 5.找出美国死亡最多的10个州
82. df5 = spark.sql("select date,state,totalDeaths from eachStateInfo  order by totalDeaths desc limit 10")
83. df5.repartition(1).write.json("result5.json")
85. # 6.找出美国确诊最少的10个州
86. df6 = spark.sql("select date,state,totalCases from eachStateInfo  order by totalCases asc limit 10")
87. df6.repartition(1).write.json("result6.json")
89. # 7.找出美国死亡最少的10个州
90. df7 = spark.sql("select date,state,totalDeaths from eachStateInfo  order by totalDeaths asc limit 10")
91. df7.repartition(1).write.json("result7.json")
93. # 8.统计截止5.19全美和各州的病死率
94. df8 = spark.sql(
95. "select 1 as sign,date,'USA' as state,round(sum(totalDeaths)/sum(totalCases),4) as deathRate from eachStateInfo group by date union select 2 as sign,date,state,deathRate from eachStateInfo").cache()
96. df8.sort(df8["sign"].asc(), df8["deathRate"].desc()).repartition(1).write.json("result8.json")

可视化：

1. **from** pyecharts **import** options as opts
2. **from** pyecharts.charts **import** Bar
3. **from** pyecharts.charts **import** Line
4. **from** pyecharts.components **import** Table
5. **from** pyecharts.charts **import** WordCloud
6. **from** pyecharts.charts **import** Pie
7. **from** pyecharts.charts **import** Funnel
8. **from** pyecharts.charts **import** Scatter
9. **from** pyecharts.charts **import** PictorialBar
10. **from** pyecharts.options **import** ComponentTitleOpts
11. **from** pyecharts.globals **import** SymbolType
12. **import** json

15. # 1.画出每日的累计确诊病例数和死亡数——>双柱状图
16. **def** drawChart\_1(index):
17. root = f"result/result{index}.json"
18. date = []
19. cases = []
20. deaths = []
21. with open(root, 'r') as f:
22. **while** True:
23. line = f.readline()
24. **if** **not** line:  # 到 EOF，返回空字符串，则终止循环
25. **break**
26. js = json.loads(line)
27. date.append(str(js['date']))
28. cases.append(int(js['cases']))
29. deaths.append(int(js['deaths']))
31. d = (
32. Bar()
33. .add\_xaxis(date)
34. .add\_yaxis("累计确诊人数", cases, stack="stack1")
35. .add\_yaxis("累计死亡人数", deaths, stack="stack1")
36. .set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))
37. .set\_global\_opts(title\_opts=opts.TitleOpts(title="老美每日累计确诊和死亡人数"))
38. .render("result/result1.html")
39. )

42. # 2.画出每日的新增确诊病例数和死亡数——>折线图
43. **def** drawChart\_2(index):
44. root = f"result/result{index}.json"
45. date = []
46. cases = []
47. deaths = []
48. with open(root, 'r') as f:
49. **while** True:
50. line = f.readline()
51. **if** **not** line:  # 到 EOF，返回空字符串，则终止循环
52. **break**
53. js = json.loads(line)
54. date.append(str(js['date']))
55. cases.append(int(js['caseIncrease']))
56. deaths.append(int(js['deathIncrease']))
58. (
59. Line(init\_opts=opts.InitOpts(width="1600px", height="800px"))
60. .add\_xaxis(xaxis\_data=date)
61. .add\_yaxis(
62. series\_name="新增确诊",
63. y\_axis=cases,
64. markpoint\_opts=opts.MarkPointOpts(
65. data=[
66. opts.MarkPointItem(type\_="max", name="最大值")
68. ]
69. ),
70. markline\_opts=opts.MarkLineOpts(
71. data=[opts.MarkLineItem(type\_="average", name="平均值")]
72. ),
73. )
74. .set\_global\_opts(
75. title\_opts=opts.TitleOpts(title="老美每日新增确诊折线图", subtitle=""),
76. tooltip\_opts=opts.TooltipOpts(trigger="axis"),
77. toolbox\_opts=opts.ToolboxOpts(is\_show=True),
78. xaxis\_opts=opts.AxisOpts(type\_="category", boundary\_gap=False),
79. )
80. .render("result/result2.html")
81. )
82. (
83. Line(init\_opts=opts.InitOpts(width="1600px", height="800px"))
84. .add\_xaxis(xaxis\_data=date)
85. .add\_yaxis(
86. series\_name="新增死亡",
87. y\_axis=deaths,
88. markpoint\_opts=opts.MarkPointOpts(
89. data=[opts.MarkPointItem(type\_="max", name="最大值")]
90. ),
91. markline\_opts=opts.MarkLineOpts(
92. data=[
93. opts.MarkLineItem(type\_="average", name="平均值"),
94. opts.MarkLineItem(symbol="none", x="90%", y="max"),
95. opts.MarkLineItem(symbol="circle", type\_="max", name="最高点"),
96. ]
97. ),
98. )
99. .set\_global\_opts(
100. title\_opts=opts.TitleOpts(title="老美每日新增死亡折线图", subtitle=""),
101. tooltip\_opts=opts.TooltipOpts(trigger="axis"),
102. toolbox\_opts=opts.ToolboxOpts(is\_show=True),
103. xaxis\_opts=opts.AxisOpts(type\_="category", boundary\_gap=False),
104. )
105. .render("result/result2.html")
106. )

109. # 3.画出截止5.19，老美各州累计确诊、死亡人数和病死率--->表格
110. **def** drawChart\_3(index):
111. root = f"result/result{index}.json"
112. allState = []
113. with open(root, 'r') as f:
114. **while** True:
115. line = f.readline()
116. **if** **not** line:  # 到 EOF，返回空字符串，则终止循环
117. **break**
118. js = json.loads(line)
119. row = []
120. row.append(str(js['state']))
121. row.append(int(js['totalCases']))
122. row.append(int(js['totalDeaths']))
123. row.append(float(js['deathRate']))
124. allState.append(row)
126. table = Table()
128. headers = ["State name", "Total cases", "Total deaths", "Death rate"]
129. rows = allState
130. table.add(headers, rows)
131. table.set\_global\_opts(
132. title\_opts=ComponentTitleOpts(title="老美各州疫情一览", subtitle="")
133. )
134. table.render("result/result3.html")

137. # 4.画出老美确诊最多的10个州——>词云图
138. **def** drawChart\_4(index):
139. root = f"result/result{index}.json"
140. data = []
141. with open(root, 'r') as f:
142. **while** True:
143. line = f.readline()
144. **if** **not** line:  # 到 EOF，返回空字符串，则终止循环
145. **break**
146. js = json.loads(line)
147. row = (str(js['state']), int(js['totalCases']))
148. data.append(row)
150. c = (
151. WordCloud()
152. .add("", data, word\_size\_range=[20, 100], shape=SymbolType.DIAMOND)
153. .set\_global\_opts(title\_opts=opts.TitleOpts(title="老美各州确诊Top10"))
154. .render("result/result4.html")
155. )

158. # 5.画出老美死亡最多的10个州——>象柱状图
159. **def** drawChart\_5(index):
160. root = f"result/result{index}.json"
161. state = []
162. totalDeath = []
163. with open(root, 'r') as f:
164. **while** True:
165. line = f.readline()
166. **if** **not** line:  # 到 EOF，返回空字符串，则终止循环
167. **break**
168. js = json.loads(line)
169. state.insert(0, str(js['state']))
170. totalDeath.insert(0, int(js['totalDeaths']))
172. c = (
173. PictorialBar()
174. .add\_xaxis(state)
175. .add\_yaxis(
176. "",
177. totalDeath,
178. label\_opts=opts.LabelOpts(is\_show=False),
179. symbol\_size=18,
180. symbol\_repeat="fixed",
181. symbol\_offset=[0, 0],
182. is\_symbol\_clip=True,
183. symbol=SymbolType.ROUND\_RECT,
184. )
185. .reversal\_axis()
186. .set\_global\_opts(
187. title\_opts=opts.TitleOpts(title="PictorialBar-老美各州死亡人数Top10"),
188. xaxis\_opts=opts.AxisOpts(is\_show=False),
189. yaxis\_opts=opts.AxisOpts(
190. axistick\_opts=opts.AxisTickOpts(is\_show=False),
191. axisline\_opts=opts.AxisLineOpts(
192. linestyle\_opts=opts.LineStyleOpts(opacity=0)
193. ),
194. ),
195. )
196. .render("result/result5.html")
197. )

200. # 6.找出老美确诊最少的10个州——>词云图
201. **def** drawChart\_6(index):
202. root = f"result/result{index}.json"
203. data = []
204. with open(root, 'r') as f:
205. **while** True:
206. line = f.readline()
207. **if** **not** line:  # 到 EOF，返回空字符串，则终止循环
208. **break**
209. js = json.loads(line)
210. row = (str(js['state']), int(js['totalCases']))
211. data.append(row)
213. c = (
214. WordCloud()
215. .add("", data, word\_size\_range=[100, 20], shape=SymbolType.DIAMOND)
216. .set\_global\_opts(title\_opts=opts.TitleOpts(title="老美各州确诊最少的10个州"))
217. .render("result/result6.html")
218. )

221. # 7.找出老美死亡最少的10个州——>漏斗图
222. **def** drawChart\_7(index):
223. root = f"result/result{index}.json"
224. data = []
225. with open(root, 'r') as f:
226. **while** True:
227. line = f.readline()
228. **if** **not** line:  # 到 EOF，返回空字符串，则终止循环
229. **break**
230. js = json.loads(line)
231. data.insert(0, [str(js['state']), int(js['totalDeaths'])])
233. c = (
234. Funnel()
235. .add(
236. "State",
237. data,
238. sort\_="ascending",
239. label\_opts=opts.LabelOpts(position="inside"),
240. )
241. .set\_global\_opts(title\_opts=opts.TitleOpts(title=""))
242. .render("result/result7.html")
243. )

246. # 8.老美的病死率--->饼状图
247. **def** drawChart\_8(index):
248. root = f"result/result{index}.json"
249. values = []
250. with open(root, 'r') as f:
251. **while** True:
252. line = f.readline()
253. **if** **not** line:  # 到 EOF，返回空字符串，则终止循环
254. **break**
255. js = json.loads(line)
256. **if** str(js['state']) == "USA":
257. values.append(["Death(%)", round(float(js['deathRate']) \* 100, 2)])
258. values.append(["No-Death(%)", 100 - round(float(js['deathRate']) \* 100, 2)])
259. c = (
260. Pie()
261. .add("", values)
262. .set\_colors(["blcak", "orange"])
263. .set\_global\_opts(title\_opts=opts.TitleOpts(title="老美的病死率"))
264. .set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {c}"))
265. .render("result/result8.html")
266. )

269. # 可视化主程序：
270. index = 1
271. **while** index < 9:
272. funcStr = "drawChart\_" + str(index)
273. eval(funcStr)(index)
274. index += 1