# PowerBI可视化组件探索（大纲1级）

（个人不需要创建目录）

## PowerBI基础视觉组件探索（宋体小三加粗大纲2级）

1. **视觉组件名称（宋体四号加粗大纲3级）**
   1. **小标题（宋体小四加粗大纲4级）**

正文部分宋体五号，首行缩进。插入的图片全部另起一行并居中对齐。

## PowerBI与Python的可视化工具对比（宋体小三加粗大纲2级）

1. **Plotly库**

**1.1 环形图**

【环形图介绍】

环形图是饼图的延伸，它是由两个及两个以上大小不一的饼图叠在一起，挖去中间的部分所构成的图形。环形图通常用于比较多个样本的结构构成，每个样本用一个环来表示，样本中的每一部分数据用环中的一段表示，因此环形图可显示多个样本各部分所占的相应比例，从而有利于构成的比较研究。

【实例实现】

采用FoodMart数据库中的sales\_fact\_1997和customer两个表，比较不同教育背景（education）消费者的销售额。

首先，导入两张表sales\_fact\_1997和customer，并将这两张表按customer\_id为键进行合并。

import plotly

import pandas as pd

import numpy as np

sales\_fact\_1997 = pd.read\_excel('sales\_fact\_1997.xlsx')

customer = pd.read\_excel(‘customer.xlsx')

data1=pd.merge(left=sales\_fact\_1997, right=customer, how='left', on='customer\_id')

先对数据分组求和，对不同教育背景的消费者分别计算销售额：

labels = np.unique(data1.loc[:,'education'])

sum\_bygroup = data1.groupby(['education'],as\_index=False)['store\_sales'].sum()

values = list(sum\_bygroup.loc[:,'store\_sales'])

其中，labels存储五种不同的教育背景，values存储五种教育背景的消费者销售额。

最后，使用plotly.graph\_objs.Pie画出环形图：

trace0=plotly.graph\_objs.Pie(labels=labels,

values=values,

hole=0.4,

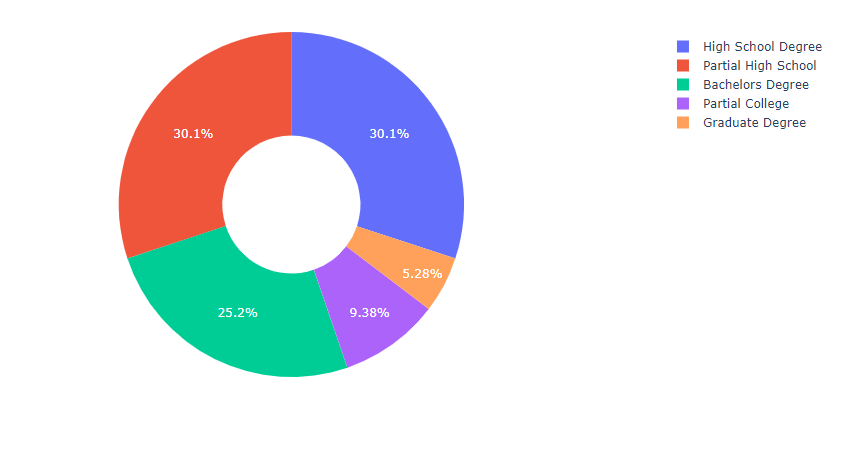
textfont=dict(size=12,color='white'))

data=[trace0]

fig = dict(data=data)

plotly.offline.iplot(fig)

效果图：



* 1. **树形图**

【树形图介绍】

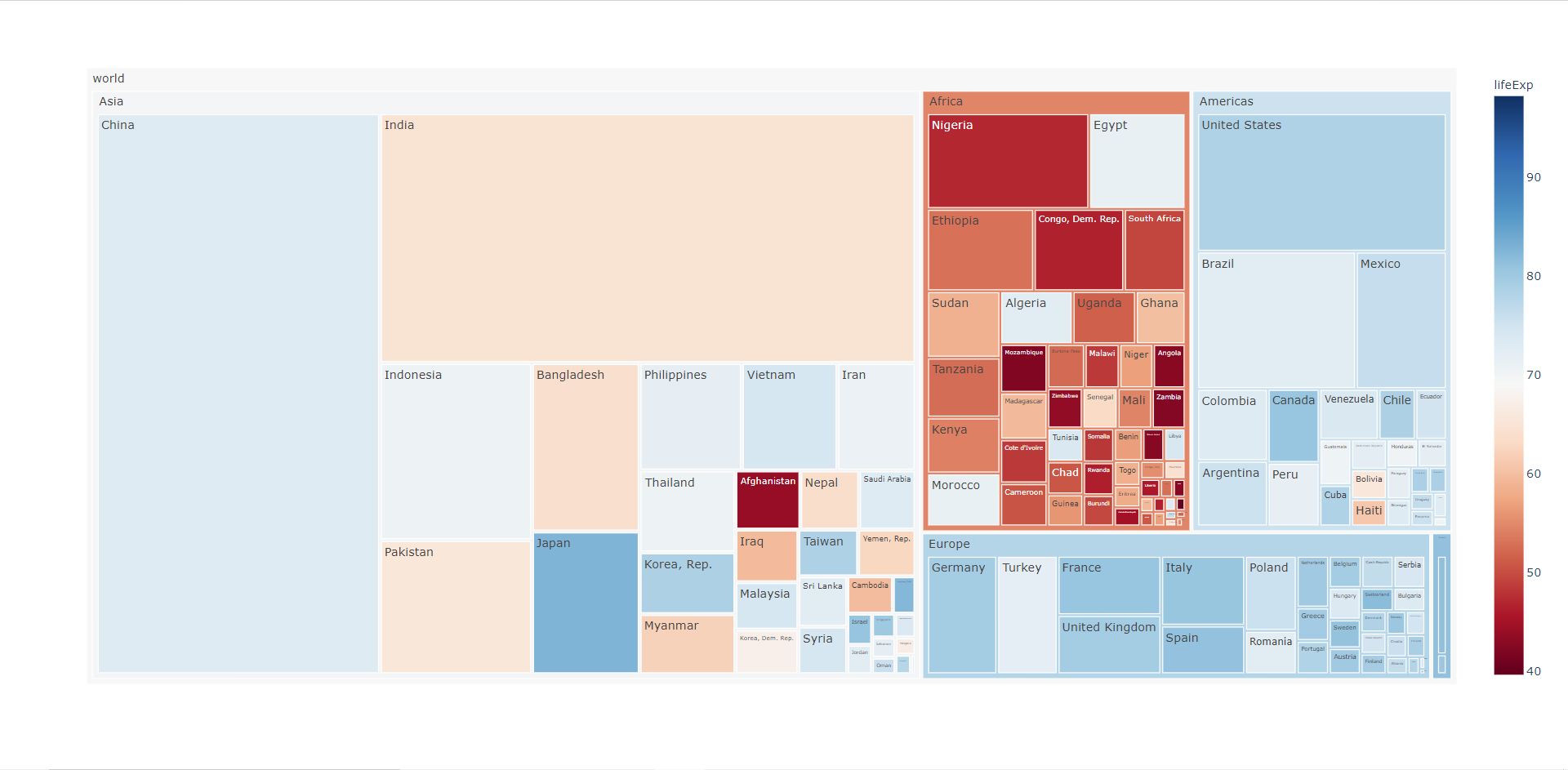
树形图是一种复杂的，基于区域的数据可视化，用于复杂层次结构的数据，可能难以精确解释。在许多情况下，最好使用更简单的可视化效果（例如条形图）。

树形图是用于大型分层数据的数据可视化技术。它们捕获数据中的两种信息：（1）单个数据点的值；（2）层次结构。

树形图是分层数据的可视化。它们由一系列嵌套矩形组成，这些矩形的大小与相应的数据值成比例。大矩形代表数据树的一个分支，然后细分为较小的矩形，这些矩形代表该分支内每个节点的大小。

【实例实现1】

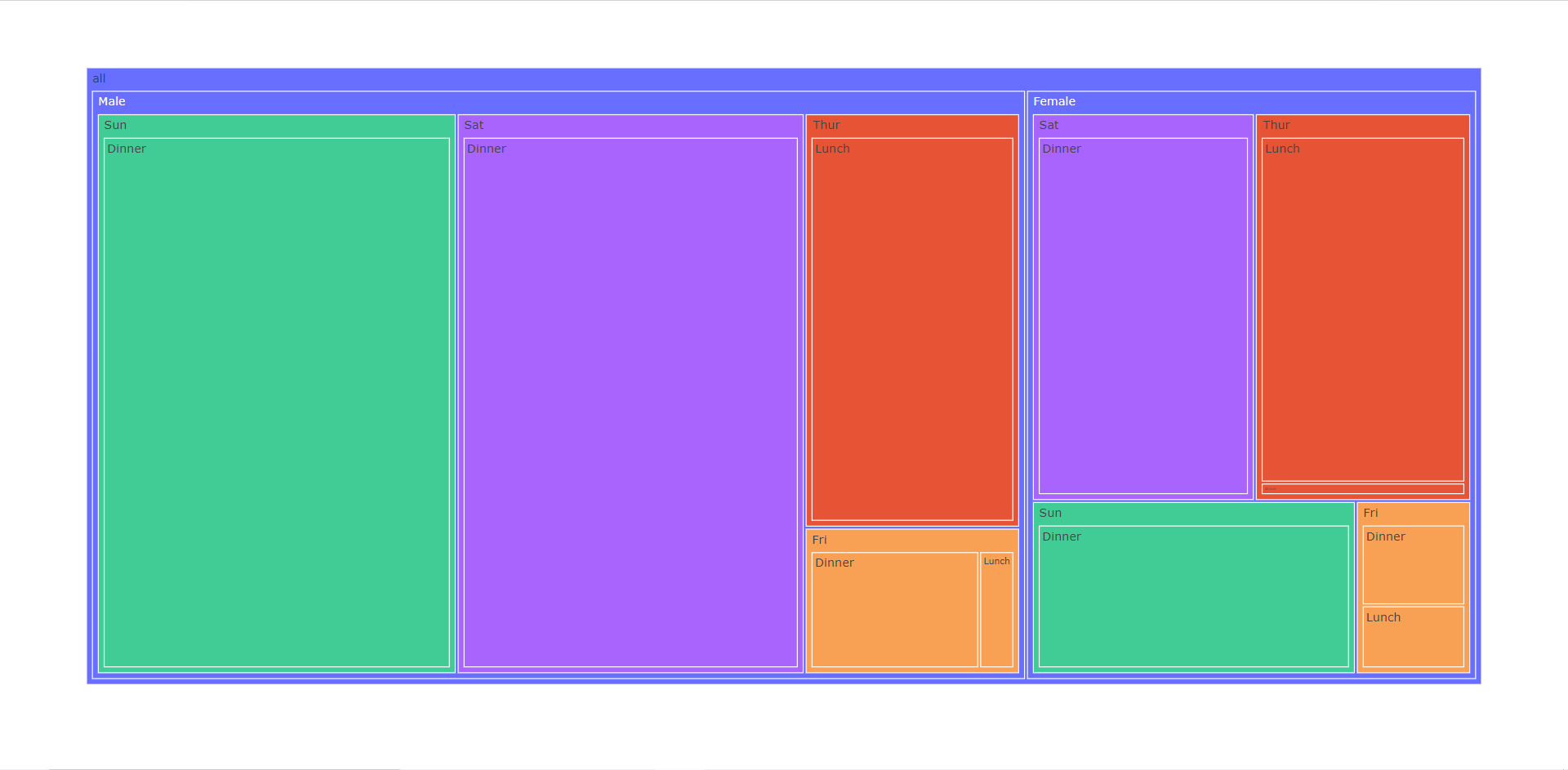
**import** plotly.express **as** px  
**import** numpy **as** np  
df = px.data.gapminder().query(**"year == 2007"**)  
df[**"world"**] = **"world"** *# in order to have a single root node*fig = px.treemap(df, path=[**'world'**, **'continent'**, **'country'**], values=**'pop'**,  
 color=**'lifeExp'**, hover\_data=[**'iso\_alpha'**],  
 color\_continuous\_scale=**'RdBu'**,



【实例实现2】

**import** plotly.express **as** px  
df = px.data.tips()  
df[**"all"**] = **"all"** *# in order to have a single root node*fig = px.treemap(df, path=[**'all'**, **'sex'**, **'day'**, **'time'**], values=**'total\_bill'**, color=**'day'**)  
fig.show()

color\_continuous\_midpoint=np.average(df[**'lifeExp'**], weights=df[**'pop'**]))  
fig.show()



* 1. **雷达图**

【雷达图介绍】

雷达图又叫蜘蛛网图。传统的雷达图被认为是一种表现多维(通常大于等于三维)数据的图表。它将多个维度的数据量映射到坐标轴上，这些坐标轴起始于同一个圆心点，通常结束于圆周边缘，将同一组的点使用线连接起来就称为了雷达图。雷达图在图形表现上通常以线、面或 线面叠加的方式呈现，也可以配置数据点。

【实例实现】

1.2首先我们介绍单一对象的基础雷达图绘制：

### 单一对象的雷达图：

from plotly.graph\_objs import \*

### 数据转换：

data = [Scatterpolar( r = [39, 28, 8, 7, 28, 39], theta = ['A','B','C', 'D', 'E', 'A'],

fill = 'toself',name='对象\_1')]

### 绘图设置：

layout = Layout( title = '单一对象的基础雷达图', polar = dict( radialaxis = dict(

visible = True,range = [0, 50])),showlegend = True)

fig = Figure(data=data, layout=layout)

fig.show()

2.其次我们介绍多对象的交互式雷达图绘制：

### 多对象的交互式雷达图：

from plotly.graph\_objs import \*

data = [Scatterpolar( r = [39, 28, 8, 7, 28, 39],theta = ['A','B','C', 'D', 'E', 'A'],

fill = 'toself', name = '对象\_1',hoverinfo = 'r+theta+name'),

Scatterpolar( r = [1.5, 10, 39, 31, 15, 1.5],theta = ['A','B','C', 'D', 'E', 'A'],

fill = 'toself',name = '对象\_2',hoverinfo = 'r+theta+name' )]

layout = Layout(title = '多对象雷达图',polar = dict(radialaxis = dict(

visible = True, range = [0, 50]) ), showlegend = True,

hovermode = 'closest’ )

fig = Figure(data=data, layout=layout)

fig.show()

【效果图】

