**import seaborn as sns**

**配色方案：<http://seaborn.pydata.org/generated/seaborn.color_palette.html>**

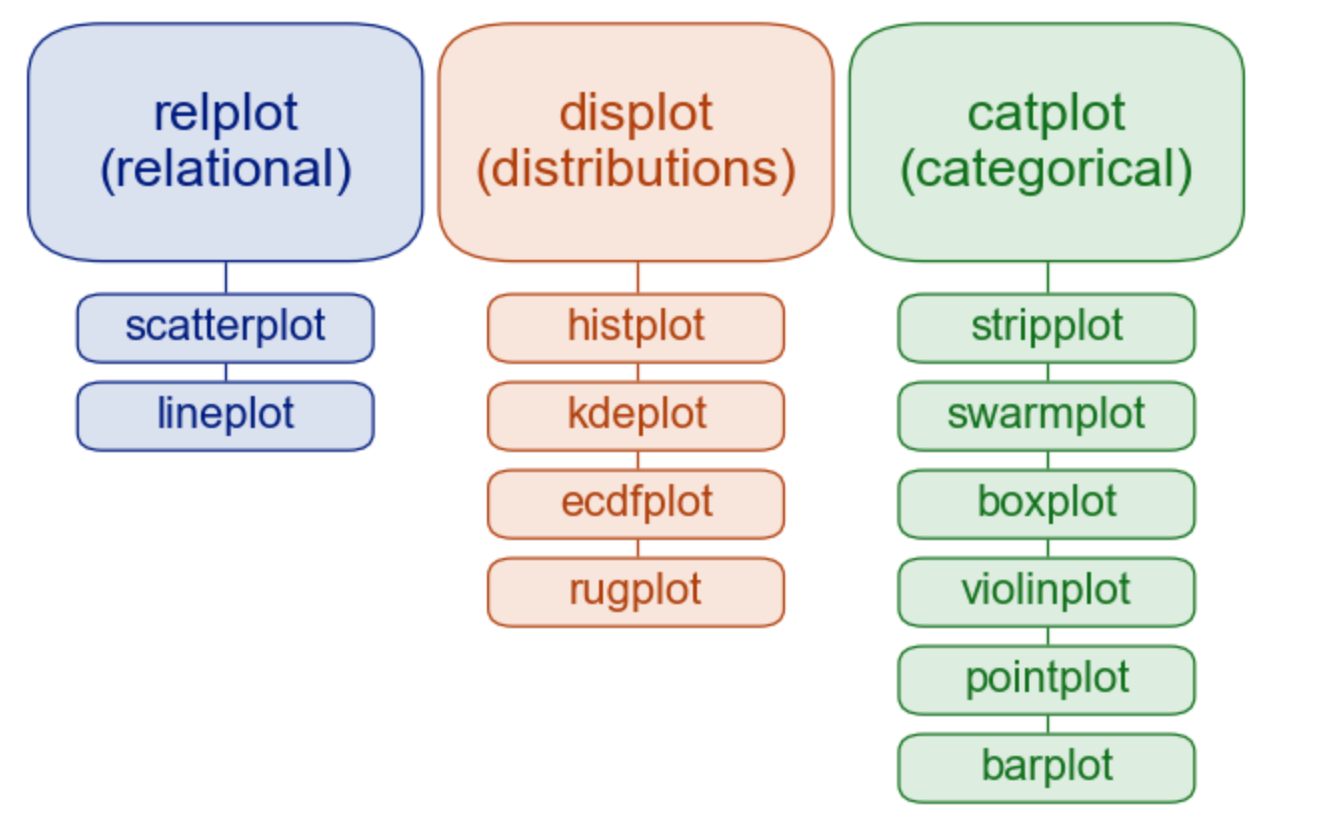
**背景色设置：**

sns.set\_style(主题名)

#主题名=”darkgrid”,”whitegrid”,”dark”,”white”,”ticks”

plt.xticks(rotation=角度) #倾斜坐标轴字体

g.set\_xticklables(rotation=角度)



**figure-level:**

**relplot(kind=**"**scatter**"/"**line**"**,row=行分面元素列,col=列分面元素列),**

**displot(kind="hist"/"kde"/"ecdf",row=行分面元素列,col=列分面元素列),**

**catplot(kind="strip"/"swarm"/"box"/"violin"/"boxen"/ "point"/"bar"/ "count",**

**row=行分面元素列,col=列分面元素列)**

**lmplot()**

**直方图:**

sns.distplot(x,hist=直方图,kde=密度曲线,rug=分布指示线,)

sns.histplot(stat=’count’/’ frequency’/’probability’/’percent’/’density’)

**二维散点图:**

sns.scatterplot(data=df,x="x", y="y", hue=”颜色分类变量”,style=”形状分类变量”,alpha=透明度,size=‘点大小变量’)

sns.regplot(x="x", y="y",data=df) #此函数还会附加散点图的线性回归曲线及置信区间

sns.regplot(x="x", y="y", data=df,ci=None,order = n) #order=n设置拟合曲线的阶数

sns.regplot(x="x", y="y", data=df,robust=True) #robust参数使拟合时忽略异常点

sns.relplot(x="x", y="y", data=df)

**分类散点图:**

sns.catplot(x="x", y="y", data=df,jitter=True) #jitter参数使得点之间不会重叠从而展现分布

sns.catplot(x="x", y="y", data=df,kind=”swarm”) #蠕虫图

**!** catplot不支持subplots

**散点图带回归线**

sns.lmplot(x="x", y="y", data=df line\_kws={'color': 回归线颜色})

sns.regplot(x="x", y="y", data=df, ci=是否绘制置信区间) #如果回归线没有显示，就是数据有问题

**箱型图:**

sns.catplot(x="x", y="y", data=df, kind="box")

sns.catplot(x="x", y="y", data=df, kind="violin") #提琴图

sns.catplot(x="x", y="y", data=df, kind=”violin”,hue=颜色分类变量,split=True) #拆分提琴图

**密度图:**

sns.kdeplot(x,shade=True,color=”r”) #利用shade参数和color参数设置阴影，x只接受数组

sns.kdeplot(x,y) #默认为等高线图,可用shade加阴影

**!** kdeplot不能添加aes对象

sns.displot(kind=’kde’,multiple : {{"layer", "stack", "fill"}})

手动实现

density = np.histogram(degrees, density=True,bins=len(degrees))

x=[density[1][i]+density[1][i+1] for i in range(len(density[0])) if density[0][i]!=0] #区间中值

y=[density[0][i] for i in range(len(density[0])) if density[0][i]!=0] #密度

sns.scatterplot(x,y)

**CDF图**

sns.displot(data=df, kind="ecdf", x , hue, rug=True,row=行分面元素,col=列分面元素)

sns.ecdfplot(x, \*log\_scale=(True,True) ,complementary=False)

手动实现

prob= np.linspace(10e-6,1-10e-6,len(X))

sns.scatterplot(sorted(X),prob)

**二维密度图:**

sns.jointplot(x="x", y="y", data=df, kind="kde")

**条形图:**

bars=sns.barplot(x=x, y=y,data=data,hue=颜色分类变量,orient=”v”/”h”,order=顺序,palette=颜色,dodge=是否绘制分堆)

#返回的指针可用bars.patches来调用各个方块的数据

**统计条形图:**

sns.catplot(x="x", data=df, hue=颜色分类变量, kind="count")

sns.countplot(x="x", data=df)

**pair-plot:**

sns.pairplot(data,hue)

**自定义pair-plot**

g = sns.PairGrid(df, hue, corner=是否不画上三角)

g.map\_upper(sns.kdeplot, hue=None, levels=5) #对角线上二维密度图

g.map\_lower(sns.scatterplot, marker="+") #对角线下添加散点图

g.map\_diag(sns.histplot, element="step", linewidth=0, kde=True) #对角线直方图+密度曲线

g.add\_legend(frameon=True)

g.legend.set\_bbox\_to\_anchor((0.61, 0.6)) #设定图例位置

**折线图:**

sns.relplot(x="x”, y="y", data=df, kind="line",sort=True, estimator=None)

#sort参数对x进行排序; estimator参数将对曲线进行平滑并附加置信区间

sns.lineplot(x="x”, y="y", data=df)

**热力图:**

#常用DataFrame.pivot(“A”,”B”,”C“)来生成交叉表,或用DataFrame.corr(“A”,”B”)生成相关矩阵

heatmap = sns.heatmap(data,

vmin=数据下限, vmax=数据上限,cmap=颜色‘Oranges’

annot=True/False #若data中含有第三个维度的数据，则将其标记在图上；无数据时，标记每格的数据量

linewidth=间隔宽度, # linewidth 使得格与格之间有距离

linecolor=间隔颜色,

center=数据色彩中心值, #调整色彩中心值可以调整热力图的色彩分布

robust=True/False, #True则开启鲁棒区间画图（抑制极端值影响）

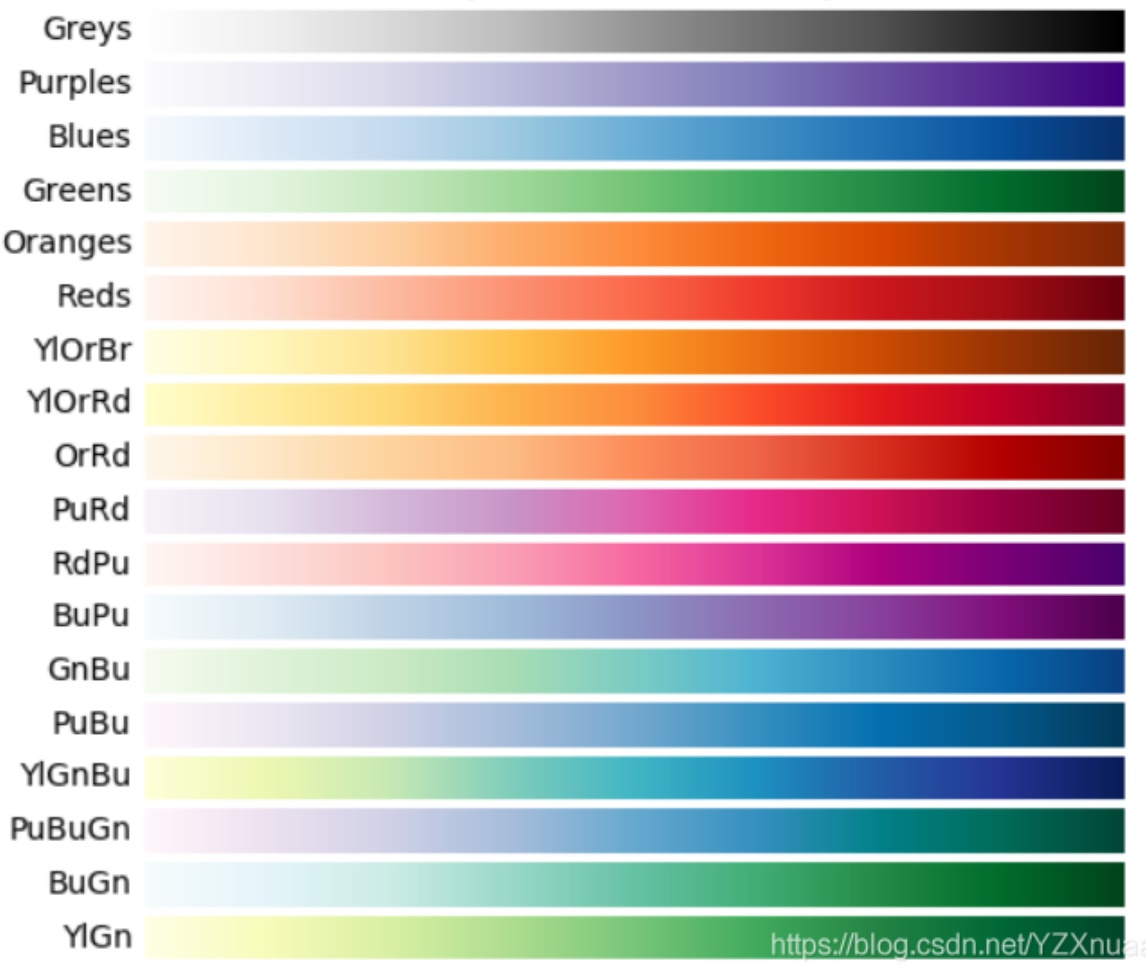
fmt=‘d’, #控制标注数据的数据格式,’d’可取消科学计数法

cbar=True/False, #是否绘制颜色刻度条

xticklabels=X轴标签名, yticklabels=Y轴标签名

)

# cmap = sns.diverging\_palette(220, 20, as\_cmap=True) 红-蓝色板



**关联图**

**relplot(**

**x,y,data=数据,**

**hue=分色特征,**

**markers=点类型,size=分尺寸特征,sizes=(点尺寸下限,点尺寸上限),**

**style=分点型/线型特征,**

**kind=’scatter’/’line’,legend=’brief’/’full’/False)**

**设置主题**

**sns.set\_theme(style="whitegrid", palette="Set1",font=字体,font\_scale)**

**sns.set\_palette("Pastel1")**

**sns.set\_style("whitegrid"/"darkgrid"/"dark"/"white"/"choice"…)**

**移除边框：**sns.despine(top=True, right=True, left=False, bottom=False, trim=False)

**一个给特征barplot上数据标签的函数**

def show\_values(axs, orient="v", space=.01):

def \_single(ax):

if orient == "v":

for p in ax.patches:

\_x = p.get\_x() + p.get\_width() / 2

\_y = p.get\_y() + p.get\_height() + (p.get\_height()\*0.01)

value = '{:.1f}'.format(p.get\_height())

ax.text(\_x, \_y, value, ha="center")

elif orient == "h":

for p in ax.patches:

\_x = p.get\_x() + p.get\_width() + float(space)

\_y = p.get\_y() + p.get\_height() - (p.get\_height()\*0.5)

value = '{:.1f}'.format(p.get\_width())

ax.text(\_x, \_y, value, ha="left")

if isinstance(axs, np.ndarray):

for idx, ax in np.ndenumerate(axs):

\_single(ax)

else:

\_single(axs)

show\_values(p)