**分析方法：**

**分析指标（度量）**

分析指标

数据库表中的数值字段可以聚合成原子指标

原子指标也叫基础指标表达业务实体原子量化属性，不可再分的概念集合

派生指标：基础指标加修饰词

衍生指标也叫复合指标

数理衍生： 比率，最大 最小 平均 累计

时间衍生：同比 环比 累计

进度衍生： 完成计划比

 ---------1.原子指标---------

SELECT SUM(A) FROM Z GROUP BY C;

SELECT SUM(B) FROM Z GROUP BY C;

---------2.派生指标---------

SELECT SUM(A) AS SUM\_A FROM Z WHERE D = 'X' GROUP BY C;

SELECT SUM(B) AS SUM\_B FROM Z WHERE D = 'X' AND E = 'Y' GROUP BY C;

---------3.衍生指标---------

SELECT SUM\_A/SUM\_B FROM TEMP;

**统计分析方法**

单分类变量 频数 频率统计

单连续变量 分布情况 均值 方差 等

多个分类变量 联列表，卡方检验，组合频率

连续变量+连续变量 相关性分析 R

二分类+连续变量 配对样本检测 双样本T检验

多分类+连续变量 方差分析

**作图分析**

单度量（指标）

和维度无关：指标分布，标准值（警戒值）。

单维度：纵向（时间）折线， 横向（维度）条柱形，比例分析饼图，累计分析-二八法则帕累托图（单位个数与份额做累计图），

多维度：多色多符号折线，多族柱形图，雷达图。

多度量（指标）：

两度量相关散点图，波士顿矩阵，双轴折线图。多度量用相关矩阵，矩阵散点图，

杜邦指标分解（正影响度指标）。

多维度（包含索引维度）：多色多符号相关散点图。 多行多列组合折线柱形图

ggplot作图原理：维度和度量都算一列，x轴，y轴各一个列，以后一个特征一个维度。

**MODELER:**

读取excel中日期格式要现在excel中设置号日期，否则在modeler中很难再认定日期。Modeler中的数据类型是自动判断的。

**POWERBI：**

**发布看板：**

Powerbi网页版，文件菜单—嵌入报表—发布到web（公共）

如果无法发布。使用管理员账号去到设置—管理门户—租户设置—发布到WEB启用

**日期表：**

日期表不是任意的时间列，他是单独的每个日期具有唯一值的日期的表。

1.制作日期表，将需要的月度季度年度维度建立，日期表日期设置为日期格式

2.设置日期表

请在“字段”窗格中选择要用作日期表的表，然后右键单击该表，在出现的菜单中选择“标记为日期表”>“标记为日期表”，

指定自己的日期表时，Power BI Desktop 会对该列及其数据执行以下验证，以确保数据：

包含唯一值

不包含任何 null 值

包含连续的日期值（从开头到末尾）

如果它是日期/时间数据类型，则它在每个值间具有相同的时间戳

3.将数据表日期与日期表关联

设置为日期表后所有系统不再为任意表的日期列生成层级关系。自己再日期表中建立日期层级关系。涉及日期字段筛选切片均采用日期表日期字段

日期轴受筛选期控制 通过日期智能函数突破日期筛选限制

**度量值的移动：**

1.在菜单栏中选择主表移动

2.在数据模型视图中同时选择多个度量值然后输入显示文件夹

3.在数据模型视图中直接拖动度量值到某个文件夹

**python脚本获取数据到pbi中处理：**

**Powerbi 使用anaconda中的python必须要添加anaconda的PATH变量不然无法找到python**

主页选择“获取数据/更多…”，随后依次选择“其它/Python脚本”，点击确定按钮，在其他编辑器中测试好以后粘贴到pbi的编辑窗口，运行后通过导航选择df，或者是运行python.excute()查询后选择表右键添加到新查询。

Python生成多个数据框，如果导航中选择多个数据表，生成多个新查询，都不会让各表同源，多次执行python脚本，效率低。如果想多个表同源。

可以用：1.python构建初始查询（不用导航和更改数据类型等）上点右键-选择“引用”，然后重命名。

2.如果已有多表的修改：点选源，在上面公式输入框中输入源=初始查询名称。

3.主要特别注意选择某个步骤右键选择“提取之前的步骤”生成的新查询表，会引用提取出来的步骤的所有过程作为源。即源=提取出来的步骤查询名。这样也可实现带步骤的源的引用，本质上同第一个方法。但是注意源中的步骤是否符合新表。

对于有小时分秒粒度的日期时间格式，无法系统设置声明日期表。建立时期时间表，与其他表的日期时间建立关系即可。

**对于查询中间数据集运用python处理。**

选中查询后选择运行python脚本，对dataset使用python处理即可。python有多少个df，处理结果就会呈现多少个表，通过右键来选择df。继续pbi的后续处理。

**SUM,SUMX函数区别**

SUM语法：= SUM（<列名>）

示例：总销售额= SUM（销售表[销售额]）

SUMX语法：= SUMX（<Table>，<expression>）

示例：总销售额SUMX = SUMX（销售表，销售表[数量] \*销售表[单位价格]）

先逐行计算出结果再相加，适用于平均单价，平均压力这种加权平均数。

每次访问平均花费= DIVIDE（[总花费SUMX]， [总访问量]）

**Caculate函数**

CALCULATE([产品数量],'产品明细'[品牌]="苹果")搭配计算用度量值用，主要在于外部上下文条件删选。类似于pandas query一个小dataframe，PBI的筛选框属于内部上下文，是在外部上下文的范围内再进行筛选。

CALCULATE(度量值,ALL(表名))加ALL可以扩大上下文，让内部上下文失效。想计算每个产品数量占总产品数量的比重，对总产品数量使用ALL，即可让内部上下文影响失效

产品数量4 = CALCULATE([产品数量],

all('产品明细'[产品名称]),

'产品明细'[类别]="手机")

被ALL清除行标签的外部筛选后，从全部产品中统计品类为"手机"的产品的数量，和直接写'产品明细'[类别]="手机"的区别再，一旦加上了ALL，则内部上下文失效。是统计所有符合外部上下文条件的数量。及类别=手机的数量。

FILTER(<table>,<filter>) 用于caculate的条件是用另外一个条件筛选出来的，受上下文筛选影响。

CALCULATE([销售总额],

FILTER(ALL('门店城市'),

[销售总额]>20000000))

不再扩大上下文，外上下文使用filter筛选出来的销售总额超过2000万的城市。

Values函数如果是列则返回所有唯一值，如果是表则返回唯一行？类似distinct但是distinct不返回空白行。受上下文筛选影响。

Australian Sales Tax =

IF(

HASONEVALUE(Customer[Country-Region]),

IF(

VALUES(Customer[Country-Region]) = "Australia",

[Sales] \* 0.10

)

) 是 Australian Sales Tax =

IF(

SELECTEDVALUE(Customer[Country-Region]) = "Australia",

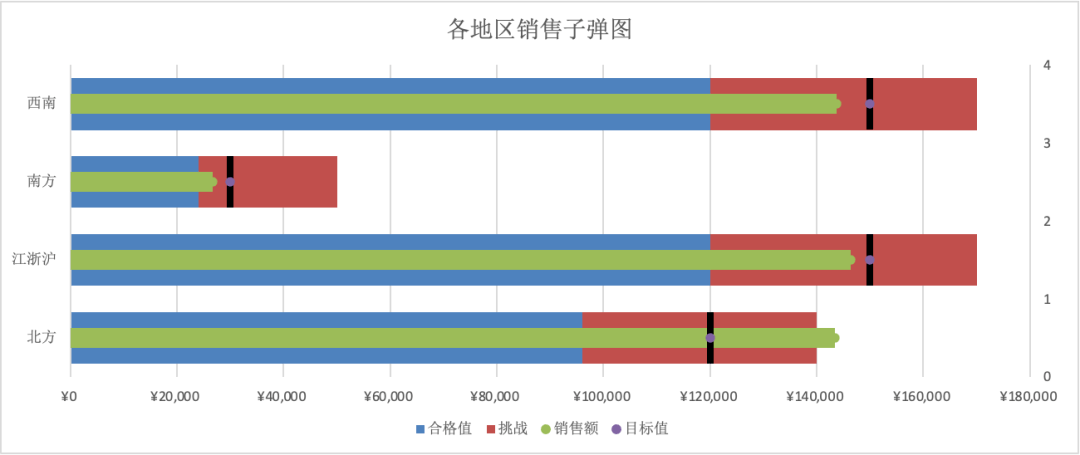
[Sales] \* 0.10

) 的等效表达式。是对上下文过滤的一个判断

**可视化：**

**EXCEL:**

做子弹图



技巧（WPS）

1.正常做出单位、合格值、挑战增量的横向堆积条形图

实际值和目标值需要一个一个增加（切记别一起增加会次轴冲突）

2.先增加实际值的条形图，然后选择条形图变成散点图会自动选择次坐标轴。然后修改散点图的x轴为实际值，y轴为自己做的等差辅助数列。**然后删掉次x轴。散点图会自动适配下面的主x轴。**调整辅助数列的数值和辅助轴的范围让散点居中。适当调整蓝色柱子的分类间距让柱子粗细合适。

3.然后增加第二个目标值散点图。先默认先择数值即可。这时候散点不再需要的位置上。此时需要在选择数据那里调整目标值的x和Y轴。调整方法如上，数值为x轴，辅助列为Y轴。

4.用误差线的特性做出子弹的内芯，和目标的竖线。选择实际值的系列，点添加标准误差线，误差线会动，没关系删除垂直误差线即可恢复。选择水平误差线设置误差线格式，方向为【负偏差】，末端样式为【无线端】，误差量为【自定义】，【指定值】数据来源为实际值单元格。先选择实际值水平误差线，增加线条的粗细至合适宽度，颜色改为同色，保留标记点更好看。

再按照上一步方法设置目标值，添加误差线，必须是百分比误差线，删掉水平方面方向，无线端，（此处有bug）选择负误差，误差量固定值调整至合适宽度，再改为正负误差。标记点设为无。

Offset （，，， counta（$A：$A），counta（$1：$1）） 可以自动选择变化的数据源

表格 复制 选择性粘贴 有锁链 可以变成自动更新的图片。

**ANACONDA 与R 结合：**

1. 正常安装R和Rstudio，安装anaconda不要选择注册path变量。
2. 给R注册环境变量path
3. 在电脑里自己下载的r（rstudio）中的Console中输入：

install.packages('IRkernel')

4. 随后在电脑里找到这个包的地址，我的是 "F:/R-3.6.2/library/IRkernel"

然后打开anaconda prompt，输入：

jupyter kernelspec install F:/R-3.6.2/library/IRkernel #安装R内核

5. 接着还是在anaconda prompt里输入：R，回车，接着输入：

IRkernel::installspec(user = FALSE) #注册R内核

6. pip install rpy2 不要用conda，会安装一个新内核，带来麻烦。

7.%load\_ext rpy2.ipython #jupyter调用R命令

**8.spyder和jupyter中用R的编码问题。**

前两者的编码大多用utf-8，但是R一般会跟中文操作系统用GBK。

在R代码开始中加入Sys.setlocale('LC\_ALL', locale = "English\_United States.1252")将中文编码换成utf-8。以防R发出的中文编码提示，在python界面里无法解码显示。？是否一定？

R在读取sqlserver数据库时，中文采用gbk编码，如果想在spyder，jupyter打印或者传出数据时，会被错误解码。因此要转码。

fenqushuiliang$fName <- iconv(as.vector(fenqushuiliang$fName), 'GB2312', 'UTF-8')

注意Encoding(someX)<-'UTF-8'，只是告诉R用什么编码来解码这个变量，并非改变编码。

iconv是R的转码命令。最多只能转向量。不能转dataframe

%R 当前行切换R语言模式

%%R 当前cell切换R语言模式

-o data\_out # jupyter的R输出数据

-i data #向R输入数据

**9.Spyder中的操作：**

import rpy2.robjects as robjects

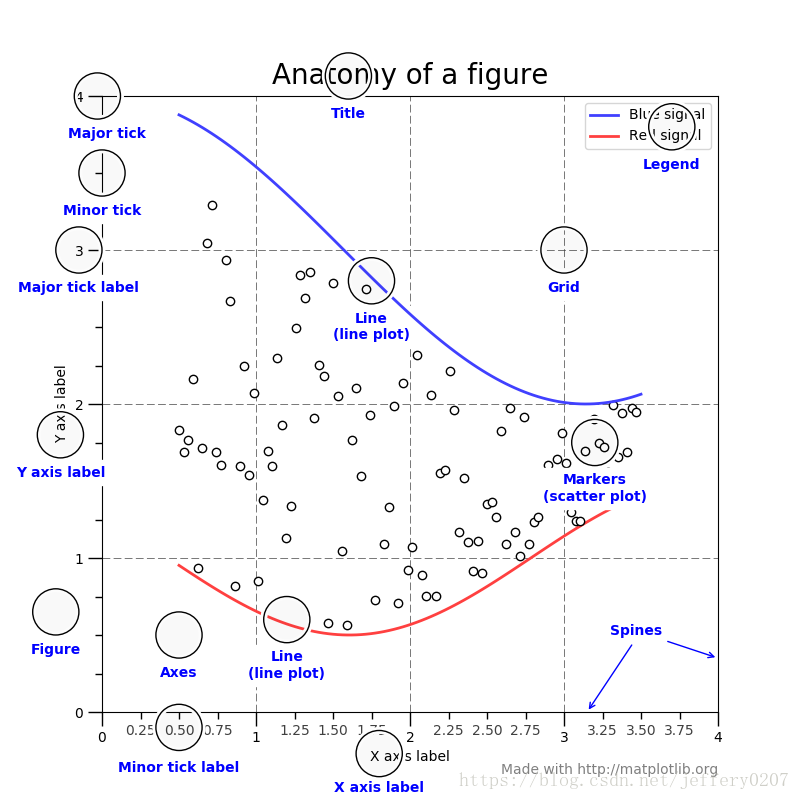
robjects.r(r\_script)

Spyder的R转出dataframe用array当中介。

matrix = robjects.r['fenqushuiliang']#将dataframe变量通过运行变量名调出来

a = np.array(matrix)

**MATPLOTLIB:**

****

matplotlib.pyplot.plot()参数详解 plt.plot([x], y, [fmt]= '[color][marker][line]', [x2], y2, [fmt2], ..., \*\*kwargs)

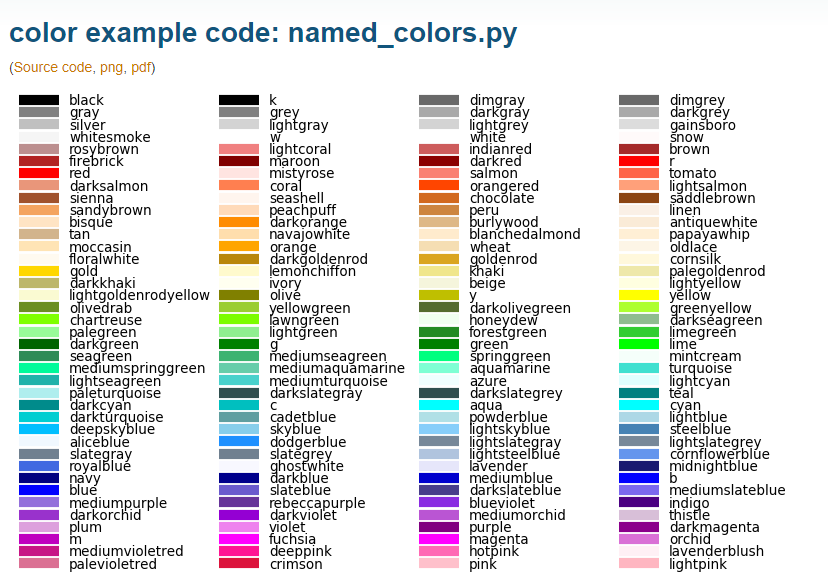
实例：plt.plot(x,y2,color='green', marker='o', linestyle='dashed', linewidth=1, markersize=6,zorder=5,)

**zorder调整图层显示顺序。**

**dashes=[2, 1] 数字list，表示虚线实虚各段长度。从实开始。**

各个绘图类型参数不同函数名称不同，比如plt.scatter()增加点大小的s参数

**Matplotlib 所有颜色名称。也可以用ps拾色器拾取现成颜色来配色**



**绘图原理**

**1.函数命令式作图：**

通过plt.plot或者plt.subplot(2，2，1)，定义ax布局和激活当前ax。当参数小于10可以不用逗号；

plt.plot()自动生成fig和ax，多次运行也只作用在同一个绘图区。plt.close（）会在一次命令执行中删除画布，让下一个画图命令新开画板。

作图命令直接作用在当前激活ax绘图区上。命令多次运行在同一个ax绘图区画多个图。命令执行完figure自动回收，无法再调用，命令一次运行更新一次内存。

同时显示多个画布，一定要用plt.figrue()创建新画布，并将新建画布作为当前画布。plt.plot或plt.subplot(221)运行在当前激活的画布的ax上。

多块画布中一块画布的重复使用通过plt.figure（1）给数字编号进行重复调用。如果新建画布没有指定编号，则按照建立顺序作为数字编号来复用。多个axes通过plt.csa(ax1)来切换复用，首先还是要定义ax对象。

plt.pie（z输入的数组，labels每一块的类别名，explode每一块的间距，autopct数据输出。格式如“%1.1f”则输出不含百分比的数值。格式如“%1.1f%%”则连接百分号）

**2.面向对象式做图：**

手动创建fig和axes对象，图必须用ax的方法绘制.创建的图形可以重复调用.

fig和axes可以用fig, axes = plt.subplots（figsize=（，），nrows=2, ncols=2，sharey=True，facecolor=“y”，#sharex和sharey：表示坐标轴的属性是否相同，可选的参数：True，False，row，col，默认值均为False，表示画布中的四个ax是相互独立的；

gridspec\_kw=dict(width\_ratios=[4, 3])）

#gridspec\_kw:字典类型，可选参数。把字典的关键字传递给GridSpec构造函数创建子图放在网格里(grid)。

单子图不用加nrows，ncols参数，一次生成画布尺寸和子图布局,选择某一个子图绘图则通过axes[1,1]来选择，单列只需一个位置参数，单子图不需要加位置参数，也可以用ax0,ax1=axes.flatten()。也可以一个一个ax创建，fig=plt.figure(figsize=(a, b), dpi=dpi),figsize 设置图形的大小，a 为图形的宽， b 为图形的高，单位为英寸,dpi 为设置图形每英寸的点数

生成图画对象后用ax1=fig.add\_subplot(211)来创建布局并指定一个一个ax。

fig.suptitle（“我是画布的标题”，fontsize=20）

fig.text（0.45，0.9，“画布副标题”）

fig.tight\_layout() #子图紧凑输出

**画布中调整子图位置**

fig.subplots\_adjust(left=0.2,bottom=0.1,right=0.8,top=0.8,hspace=0.5)。以fig左下角为原点，建立直角坐标系，0.2表示整个fig的20%。hspace子图的高度间距，wspace子图的宽度间隙，数字大小是子图的比例。

ax.set\_position((0.05,0.05,0.95,0.95))

**添加嵌套子图**

fig.add\_axes([left以fig的左边缘开始的fig比例,bottom以fig下缘开始的fig的比例,width子图的宽,height子图的高]，facecolor=“y”)。此方法的函数方法plt.axes（[]）

**对象式画图ax.plot()等等（不同图形画法自行查阅）**

规内容调整

plt.xlim() 限制数轴范围。.axis([xmin，xmax，ymin，ymax]）同时指定xylim

另有title() xlabel()

plt.legend()#除'best'，另外loc属性有：'upper right', 'upper left', 'lower left', 'lower right', 'right', 'center left', 'center right', 'lower center', 'upper center', 'center'

**对象式常规内容调整**

ax.set\_xlim（） ax.set\_xticks() ax.set\_xticklabels(),ax.set\_title(),ax.set\_xlabel(),

一次定义方法

ax.set(xlim=(0, 24), ylabel="",

xlabel="Automobile collisions per billion miles")

**图例的设置**

ax级别

ax.legend(wedges, ingredients,

title="Isoform type",

loc="center left",

bbox\_to\_anchor=(1, 0, 0.5, 1),

fontsize=12, ##设置图例文字大小

title\_fontsize=14 ##设置图例title大小

)

画布级别

fig.legend(frameon=False,scatterpoints=1,labels = Label\_Com, labelspacing=0.4,columnspacing=0.4,markerscale=2,bbox\_to\_anchor=(0.9, 0),ncol=12,prop=font1,handletextpad=0.1)

bbox\_to\_anchor=(0.9, 0) 通过调节此参数放置图例，(0.9,0) 分别是横向相对位置和纵向

相对位置

scatterpoints=1：设置图例中对应图像只出现一次

labels= Label\_Com ：设置图例中的名称

labelspacing=0.4 调整图例中标签的距离

columnspacing=0.4 调整图例中不同列之间的间距

handletextpad=0.1 用此参数调节图例和标签之间的距离

prop=font1 图例的字体格式设置，赋值font1可以是一个字典，包含各个属性及其对应值，属性包括family（字体）、size（字体大小）等常用属性，更详细的解释可参考matplotlib手册中关于legend prop的解释。一种比较简单的设置为：

font1 = {'family' : 'Times New Roman',

'weight' : 'normal',

'size' : 23,}

**#图例调整显示顺序**

handles, labels = plt.gca().get\_legend\_handles\_labels()

order = [0,2,1]

plt.legend([handles[idx] for idx in order],[labels[idx] for idx in order])

或者

handles, labels = ax.get\_legend\_handles\_labels()

# sort both labels and handles by labels

labels, handles = zip(\*sorted(zip(labels, handles), key=lambda t: t[0]))

ax.legend(handles, labels)

**函数式添加文字一般用来遍历添加数据标签**

matplotlib.pyplot.text(x, y, s, fontdict=None, withdash=False, \*\*kwargs)

x, y：表示坐标；

s：字符串文本；

fontdict：字典，可选；

kw：

fontsize=12,

horizontalalignment=‘center’、ha=’cener’

verticalalignment=’center’、va=’center’

**设置数据刻度和标签值**

plt.xticks(ticks=刻度数值列表，labels=刻度名列表，fontsize=字体大小,,rotation=0)

ax.xaxis.set\_tick\_params(rotation=45)

**设置数轴标签格式调整**

from matplotlib import ticker

通过子模块ticker可以对坐标轴刻度的位置和样式进行设置。刻度线分为major和minor ticks, 通过以下4个函数可以对其位置和样式进行设置

**1. .set\_major\_locator()**

**2. .set\_minor\_locator()**

**3. .set\_major\_formatter()**

**4. .set\_minor\_formatter()**

**一、 locator** ticker模块中提供了多种locator类，部分列表如下

1. AutoLocator, 默认值，自动对刻度线的位置进行设置

2. MaxNLocator, 根据提供的刻度线的最大个数，自动设置

ax.xaxis.set\_major\_locator(ticker.MaxNLocator(5))#指定最大刻度数5

1. IndexLocator, 根据起始位置和间隔来设置刻度线

ax.xaxis.set\_major\_locator(ticker.IndexLocator(2,0)#开始刻度为0，间隔2.

5. MultipleLocator, 根据指定的间隔来设置刻度线

ax.xaxis.set\_major\_locator(ticker.MultipleLocator(1.5))#刻度间隔设置为1.5

6. FixedLocator, 根据提供的列表元素来设置刻度线

ax.xaxis.set\_major\_locator(ticker.FixedLocator([0, 2, 4, 6]))#设置刻度0，2，4，6

7. NullLocator,不显示刻度线

ax.xaxis.set\_major\_locator(ticker.NullLocator())#取消刻度线

**二、 formatter**

和locator类相似，formatter也是有很多的类，部分列表如下

1. PercentFormatter，标签显示成百分比

ax.yaxis.set\_major\_formatter(ticker.PercentFormatter(xmax=100#如果Y已经是百分数的分子，那么化成百分数会扩大，因此需要缩小100倍。小数就是放大, decimals=None, symbol='%'#规定百分号怎么写，在这里可以将百分号写成“万”，配合xmax缩放百万倍，巧妙的变成缩放Y轴数字大小。

, is\_latex=False))

2. StrMethodFormatter，根据字符串格式化语法{}进行格式化

ax.xaxis.set\_major\_formatter(ticker.StrMethodFormatter('{x:.2f}'))

3. FormatStrFormatter，根据字符串格式化语法%进行格式化

ax.xaxis.set\_major\_formatter(ticker.FormatStrFormatter('%.2f'))

ticker.FormatStrFormatter('%.2f%%')

5. NullFormatter，不显示标签

#刻度日期调整

ax1.xaxis.set\_major\_formatter(mpl.dates.DateFormatter('%Y/%m/%d'))

def to\_percent(temp, position):

return '%1.0f'%(10\*temp) + '%'

ax.yaxis.set\_major\_formatter(ticker.FuncFormatter(to\_percent))

#**数轴其他格式设定**

**刻度线调整**

ax.tick\_params(axis=u'both', which=u'both',length=0) 显示数据轴和label不显示刻度线

ax.yaxis.set\_ticks\_position('left'|’both’) #指定tick在左边或者右边或者十字

**边框数轴调整**

ax.spines['right'].set\_visible(False) 不要右边的边框

ax.spines['bottom'].set\_visible(False) #把上边的边框颜色设置为无色,隐藏上边框

ax.spines['right'].set\_color('none') # 设置上边框为无色，效果一样

ax.spines['bottom'].set\_linewidth(bwith)#设置边框粗细

ax.spines['left'].set\_position(('data', 0)) 将左边框设置在数据0处

ax.axes.get\_yaxis().set\_visible(False) 获取对象方法不显示Y轴

ax.xaxis.set\_visible(False) # 属性方法不显示x轴

ax1.axis[:].set\_visible(False) 不要所有的轴

**调整Y轴显示在左边或者右边一个ax只有一个x，y轴，双xy轴其实是新建一个ax**

ax.yaxis.tick\_right()

ax2.yaxis.tick\_left()

ax.yaxis.set\_label\_position("right")#数轴移动后也要调整标签的位置处于外侧

**网格线格式调整**

plt.grid( color = 'black',linestyle='-.',linewidth = 1,alpha=0.4)

**#填充单个曲线下部**

ax.fill\_between(x, f(x), color='green', alpha=0.5)

**#填充两个曲线之间的面积**

x=numpy.linspace(-3,3,100)

f1=func1(x)

f2=func2(x)

ax.fill\_between(x,f1,f2,where=f1>f2,color='g',alpha=0.5)

plt.show（）只显示当前内存中的所有图像，不能重复调用。

**函数式与面向对象画图也可以混合用，函数式画图默认作用在当前激活的ax上.**

**函数式显性转对象**用

ax=plt.gca()获取当前ax对象，fig=gcf()获取当前画布对象

plt. plot(xyz三维图)scatte() bar()contourf()热图，每个图画类型的参数不尽相同需要单独查询。

**#axvline/axhline 与vlines/hlines的区别**

axvline/axhline绘制一条线，返回值为Line2D对象。vlines/hlines绘制一组线返回值为LineCollection对象，相当于Line2D对象的集合。

axvline/axhline绘制的直线默认是跨越整个子图的，即便更改线的长度，单位是子图高度/宽度的比例（[0,1]），是相对值。vlines/hlines绘制的直线需要指定长度，长度单位是对应坐标轴的单位，是绝对值。

t = np.arange(0.0, 5.0, 0.1)

s = np.exp(-t) + np.sin(2 \* np.pi \* t) + 1

nse = np.random.normal(0.0, 0.3, t.shape) \* s

fig, (vax, hax) = plt.subplots(1, 2, figsize=(12, 6))

vax.plot(t, s + nse, '^')

vax.vlines(t, [0], s)

vax.vlines([1, 2], 0, 1, transform=vax.get\_xaxis\_transform(), colors='r')

vax.set\_title('竖线')

hax.plot(s + nse, t, '^')

hax.hlines(t, [0], s, lw=2)

hax.set\_title('横线'）

在Transformation Object 列中，ax是一个Axes实例，fig是一个Figure实例。



**Matplotlib可以无缝的处理LaTex字体，在图中加入数学公式**

# 添加数学公式和坐标轴标签

plt.text(0.5 \* (a+b), 1, r"$\int\_a^b f(x)\mathrm{d}x$",

horizontalalignment='center', fontsize=20)

# 前两个参数是放置文本的坐标

plt.figtext(0.9, 0.075,'$x$')

plt.figtext(0.075,0.9,'$f(x)$' )

行内公式用 $...$ 也有用\(...\),让公式处于文字行内。

行间公式用\[...\]也有用$$...$$,让公式处于单独一行。

**巧用数据大小自定义颜色**

col = np.where(x<1,'k',np.where(y<5,'b','r'))

plt.scatter(x, y, c=col, s=5, linewidth=0)

**#十字箭头坐标**

1.创建画布并引入axisartist工具。

import mpl\_toolkits.axisartist as axisartist

#创建画布

fig = plt.figure(figsize=(8, 8))

#使用axisartist.Subplot方法创建一个绘图区对象ax

ax = axisartist.Subplot(fig, 111)

#将绘图区对象添加到画布中

fig.add\_axes(ax)

2.绘制带箭头的x-y坐标轴

#通过set\_visible方法设置绘图区所有坐标轴隐藏

ax.axis[:].set\_visible(False)

#ax.new\_floating\_axis代表添加新的坐标轴

ax.axis["x"] = ax.new\_floating\_axis(0,0)

#给x坐标轴加上箭头

ax.axis["x"].set\_axisline\_style("->", size = 1.0)

#添加y坐标轴，且加上箭头

ax.axis["y"] = ax.new\_floating\_axis(1,0)

ax.axis["y"].set\_axisline\_style("-|>", size = 1.0)

#设置x、y轴上刻度显示方向

ax.axis["x"].set\_axis\_direction("top")

ax.axis["y"].set\_axis\_direction("right")

我们先把原始的如上图的所有坐标轴隐藏，即长方形的四个边。

然后用ax.new\_floating\_axis在绘图区添加坐标轴x、y，这里的ax.new\_floating\_axis(0,0)，第一个0代表平行直线，第二个0代表该直线经过0点。同样，ax.axis["y"] = ax.new\_floating\_axis(1,0)，则代表竖直曲线且经过0点。

再次，x.axis["x"].set\_axisline\_style("->", size = 1.0)表示给x轴加上箭头，"->"表示是空箭头，size = 1.0表示箭头大小。ax.axis["y"].set\_axisline\_style("-|>", size = 1.0)中"-|>"则是实心箭头。

最后，设置x、y轴上刻度显示方向，对于x轴是刻度标签在上面还是下面，y轴则是刻度标签在左边还是右边。

3.在带箭头的x-y坐标轴背景下，绘制函数图像

#生成x步长为0.1的列表数据

x = np.arange(-15,15,0.1)

#生成sigmiod形式的y数据

y=1/(1+np.exp(-x))

#设置x、y坐标轴的范围

plt.xlim(-12,12)

plt.ylim(-1, 1)

#绘制图形

plt.plot(x,y, c='b')

**双Y轴画图**

双Y轴ax2 = ax.twinx()

双x轴ax2 = ax.twiny()

新的ax2对象已经建立一个ax的X轴镜像，在ax2上画图则是使用ax的x轴，ax2的y轴。默认是在右边。如果画使用左边Y轴的图形则用ax对象来画。同一个ax中的先后顺序按照画图函数的顺序或者zorder参数的大小顺序。但是镜像ax内的所有图层都在前一个图层之上。

**Pyecharts：**

**在用+号连接js代码时需要打引号的地方，就算连接的时python中的文本，引号也要保留。否则当成变量出错。**

用snapshot截取图片比组件下降图片更清晰。

#make\_snapshot(snapshot,page.render(), "bar0.png",

#is\_remove\_html=True

# )

注意事项：1.按要求安装了pip install snapshot-phantomjs-->下载并解压phantomjs-->将phantomjs.exe路径添加到环境变量中。如果python提示找不到程序，则把phantomjs.exe复制到了c:/windows根目录中

2.运行make\_snapshot的程序文件必须纯英文的目录下。不然会报Can’t find variable: echarts

formatter=JsCode("""function(params) {return Math.abs(params.data).toFixed(2);} """ )

#params是根据js下echarts中series中的字典式的变量名来确定的。具体属性再下面展示其中data属性一般是一个数组或者列表，他是一个一个进入函数处理的。如果每一项是字典，则继续选择属性。比如.data.color。

#params.dataIndex 表示数据位置序号。条件判断params.value> 0 ? ('+' + params.value) : ('-' + params.value); formatter={value}也可以

#params 是 formatter 需要的数据集。格式如下：(文字解释在上)

componentType: 'series',

// 系列类型

seriesType: string,

// 系列在传入的 option.series 中的 index

seriesIndex: number,

// 系列名称

seriesName: string,

// 数据名，类目名

name: string,

// 数据在传入的 data 数组中的 index

dataIndex: number,

// 传入的原始数据项

data: Object,

// 传入的数据值。在多数系列下它和 data 相同。在一些系列下是 data 中的字典分量（如 map、radar 中）

value: number|Array|Object,

// 坐标轴 encode 映射信息，

// key 为坐标轴（如 'x' 'y' 'radius' 'angle' 等）

// value 必然为数组，不会为 null/undefied，表示 dimension index 。

// 其内容如：

// {

// x: [2] // dimension index 为 2 的数据映射到 x 轴

// y: [0] // dimension index 为 0 的数据映射到 y 轴

// }

encode: Object,

// 维度名列表

dimensionNames: Array<String>,

// 数据的维度 index，如 0 或 1 或 2 ...

// 仅在雷达图中使用。

dimensionIndex: number,

// 数据图形的颜色

color: string,

// 饼图的百分比

percent: number,

**#对于坐标轴的axislabel\_opts设置 ：params 直接传刻度值**

#富文本设置 一个标题代表一个框块。 <br/>表示换行’:’ ’%’ 直接可加入

#回调函数设置富文本

formatter=JsCode(

"""function(params) {

str = '{a|'+params.data.value.toFixed(2)+'}';

return str }

""" ),

rich={"a": {"color": "red","lineHeight": 22, "align": "center"}},

#格式函数的富文本

formatter="{a|{a}}{abg|}\n{hr|}\n {b|{b}: }{c} {per|{d}%} ",

rich={"a": {"color": "#999", "lineHeight": 22, #表示文字行高。小于文字大小会叠字

"align": "center"},

"abg": { "backgroundColor": "#e3e3e3",

"width": "100%",

"align": "right",#在框的左右中对其。选右会左出框，做文字背景有用。

"height": 22,#框高

"borderRadius": [4, 4, 0, 0], },

"hr": { "borderColor": "#aaa","width": "100%",

"borderWidth": 0.5,

"height": 0,},#划线其实就是做一个0高的框。

"b": {"fontSize": 16, "lineHeight": 33,

backgroundColor: { image: 'xxx/xxx.jpg' },#直接用电脑里的图片背景

},

"per":{"color":"#eee","backgroundColor":"#334455", "padding": [2, 4], "borderRadius": 2, },}

echarts 提供了丰富的文本标签配置项，包括：

字体基本样式设置：fontStyle、fontWeight、fontSize、fontFamily。

文字颜色：color。

文字描边：textBorderColor、textBorderWidth。

文字阴影：textShadowColor、textShadowBlur、textShadowOffsetX、textShadowOffsetY。

文本块或文本片段大小：lineHeight、width、height、padding。

文本块或文本片段的对齐：align、verticalAlign。

文本块或文本片段的边框、背景（颜色或图片）：backgroundColor、borderColor、borderWidth、borderRadius。

文本块或文本片段的阴影：shadowColor、shadowBlur、shadowOffsetX、shadowOffsetY。

文本块的位置和旋转：position、distance、rotate。

#自定义图形组件

opts.GraphicGroup 图形组

graphic\_item 组件位置

children 列表，组件的列表

opts.GraphicText 文本组件 里面有graphic\_item(通用) graphic\_textstyle\_opts 样式设置专用

样式设置中有graphic\_basicstyle\_opts 通用这是来设置颜色 位置 等。

**BAR图注意要点**

bar\_width=规定柱子之间的距离比category\_gap自由度更高，但category\_gap是最所有柱子制定比例宽度。Bar\_width可以设置不同宽度。

gap = '-100%' 可以控制不同系列的柱子进行重合，要设置在最后的作图序列函数中，对所有系列生效。但是如果只有一个x轴，他们无法完全居中重合，只能靠左重合。因此需要给每个系列建立一个x轴，并且将该系列的图画在这个x轴上。通过is\_show = False,来隐藏他们。即可达到完全重合。

**多X、Y轴**

通过在图形对象中通过增加如下方法实现：

extend\_axis(

yaxis=opts.AxisOpts(

name="总分表差率(%)",

type\_="value",

#min\_=0,

#max\_=45,

)

此方法同时柔和轴刻度数据，还有轴格式。而在普通画图中轴刻度数据通过.add\_xaxis(xaxis\_data=x\_data)方法添加，其中只有xaxis\_data而没有，yaxis\_data，因而y轴的添加不能一步完成， 因为数据是绑定到y轴的。

轴格式则在set\_global\_opts中的xaxis\_opts=opts.AxisOpts()参数设置。

可以预先建立多条轴，通过 position="right",参数安排出现的位置。按照建立的顺便自动编制index序号。在添加画图序列函数时通过xaxis\_index=1, yaxis\_index=1,参数分别指定使用的x轴y轴的index编号即可。

z和zlevel来控制所有图层顺序，

# 柱状图组件的所有图形的z值。控制图形的前后顺序。

# z值小的图形会被z值大的图形覆盖。

# z相比zlevel优先级更低，而且不会创建新的 Canvas。

如果是不同图形共用多数轴，则新建图形对象line 通过bar.overlap(line)，整合后Line中可以使用bar中的多数轴序号。但是单独画图是没办法显示出来的。

**SEABORN :**

#sns专用分面函数作图

g = sns.FacetGrid(zone3,col="分公司",hue="月份", height=10)

g.map(sns.kdeplot, "累计产销差率")

#类别画图函数可以直接分面。细项绘图函数不行。

g=sns.displot(data=zone3 , x="累计产销差率", hue="月份",col="分公司" ,kind="kde",col\_wrap=2,height=5,aspect=1.5)

g.set\_axis\_labels('累计产销差率','二级分区数量密度')

g.set(ylabel='二级分区集中度')

设置legend名称

g = sns.factorplot(...)

legend = g.\_legend

legend.set\_title("Sex")

for t, l in zip(legend.texts,("Male", "Female")):

t.set\_text(l)

Sns调出fig

g.fig.subplots\_adjust(top=0.9,right=0.7)

#细项绘图函数没有尺寸设置？细项绘图函数分面用matplot

f, axes = plt.subplots(2,2, figsize=(12, 10),dpi=100)

ax1,ax2,ax3,ax4=axes.flatten()

#必须用细项函数才可以写到axes里面,一个图不加ax索引，单行列图只加一个索引。

sns.histplot(data=zone3.query("分公司 == '中区'") , x="累计产销差率", bins=10,

hue="月份",multiple='dodge',shrink=.8,kde=True,ax=ax1)

ax1.set\_xticks([xticks])#刻度位置

ax1.set\_xticklabels([xlabel])#刻度文字位置

ax1.set\_xlabel('')#坐标轴名称

ax1.set\_ylabel('二级分区数量')

ax1.set\_title('中区')

#绘制直方图bins可以按照分界点。shrink可以控制控柱子之间的间隙。Kde可以加分布曲线。

g=sns.histplot(data=zone3 , x="累计产销差率", bins=[-20,-10,0,10,20,30,40,50],hue="月份",multiple='dodge',shrink=.8,kde=True)

sns.despine(left=True) #移除边框

**VIZ社交网络图：**