Python For Data Science *Cheat Sheet*

poltnine

EasyCharts团队-张杰出品

Basics

Plotnine是基于图层语法开发的python数据可视化包，主张模块间的协调与分工，各司其职，上帝的归上帝，凯撒的归凯撒：数据集，坐标系和geom可视化数据。

为展示数据，可以通过geom把数据集的变量映射到数据可视化的属性，比如：size(尺寸), color(颜色)和x、y轴的位置。

图表绘制的标准格式：

```
ggplot (data = <DATA> ) +                          必需
<GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),
 stat = <STAT> , position = <POSITION> ) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +                                 可选
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

```
from plotnine import *        #导入plotnine包的绘图函数
from plotnine.data import *   #导入plotnine自带的数据集
```

base_plot=ggplot(data = mpg, **aes(**x = 'cty', y = 'hwy'**)) #**使用ggplot()函数开始绘图，然后通过geom()函数可以添加图层。

print(base_plot) #图表的显示

base_plot.save("plot.png", width = 5, height = 5) #保存绘制的图表为5' x 5' 的png格式的图片 "plot.png"，图片的保存格式也可选择为pdf。

Geoms

使用geom函数绘制数据，使用geom函数的美学属性映射数据的变量，每个函数都返回一个图层。

图元

```
a = ggplot(economics, aes('date', 'unemploy'))
b = ggplot(seals, aes(x = 'long', y = 'lat'))
```

a + geom_blank()
(用于扩大限制)

a + geom_path(lineend="butt", linejoin="round"**)**
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = 'date'**))**
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = 'long', ymin='lat', xmax= 'long + 1', ymax = 'lat + 1'**))**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin='unemploy - 900', ymax= 'unemploy + 900'**))**
x, ymax, ymin, alpha, color, fill, group, linetype, size

线条: 公用的映射属性： x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1**))**
b + geom_hline(aes(yintercept = 'lat'**))**
b + geom_vline(aes(xintercept = 'long'**))**

b + geom_segment(aes(yend='lat+1', xend='long+1'**))**
b + geom_spoke(aes(angle = range(0,seals.shape[0]), radius = 1**))**

1个变量：连续型

```
c = ggplot(mpg, aes('hwy')); c2 = ggplot(mpg)
```

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = 'hwy'**))**
x, y, alpha, color, fill, linetype, size, weight

1个变量：离散型

```
d =ggplot(mpg, aes('fl'))
```

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

2个变量：x-连续型，y-连续型

```
e = ggplot(mpg, aes('cty', 'hwy'))
```

e + geom_label(aes(label = 'cty'), nudge_x = 1, nudge_y = 1**)**
x, y, label, alpha, angle, color, family, fontface, hjust, size, vjust

e + geom_jitter(height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)**
x, y, alpha, color, linetype, size

e + geom_smooth(method = 'lm'**)**
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = 'cty'), nudge_x = 1, nudge_y = 1**)**
x, y, label, alpha, angle, color, family, fontface, hjust, size, vjust

2个变量：x-离散型，y-连续型

```
f=ggplot(mpg, aes('class', 'hwy'))
```

f + geom_col()
x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot()
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center",dotsize=0.1**)**
x, y, alpha, color, fill, group

f + geom_violin(scale = "area"**)**
x, y, alpha, color, fill, group, linetype, size, weight

2个变量：x-离散型，y-离散型

g = ggplot(diamonds, aes('cut', 'color'))

g + geom_count()
x, y, alpha, color, fill, shape, size, stroke

二元连续变量分布

h=ggplot(diamonds, aes('carat', 'price'))

h + geom_bin2d(binwidth = (0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

连续函数

i=ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv"**)**
x, y, alpha, color, group, linetype, size

误差绘制

```
import pandas as pd
df = pd.DataFrame({'grp':  ["A", "B"], 'fit': range(3,5), 'se' : range(0,2)})
j = ggplot(df, aes('grp', 'fit', ymin = 'fit-se', ymax = 'fit+se'))
```

j + geom_crossbar(fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar()
x, ymax, ymin, alpha, color, group, linetype, size

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

地图空间

```
from geopandas import GeoDataFrame
continents = GeoDataFrame.from_file('data/lands-of-ice-and-fire/continents.shp')
westeros = continents.query('name=="Westeros"')
```

ggplot()+ geom_map(westeros, fill=None)

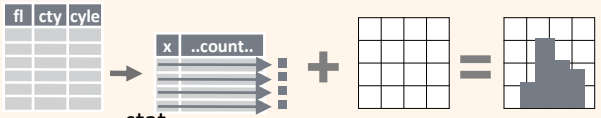
3个变量：x，y，z-连续型

```
seals[ 'z' ] = np.sqrt(pow(seals[ 'delta_long' ],2) + pow( seals['delta_lat'],2))
l=ggplot(seals, aes('long', 'lat'))
```

l + geom_tile(aes(fill = 'z'))

Stats

统计转换函数（stats）在数据被绘制出来之前对数据进行聚合和其他计算。 stat_*确定了数据的计算方法。不同方法的计算会产生不同的结果，所以一个stat()函数u必须与一个geom()函数对应作数据的计算。



c + stat_bin(binwidth = 1, origin = 10**)**
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
c + stat_count(width = 1**) x, y, |** ..count.., ..prop..
c+stat_bindot(width = 1**)**
c + stat_density(adjust = 1, kernel = "gaussian"**)**
x, y, | ..count.., ..density.., ..scaled..
c2+stat_qq(aes(sample='hwy'**))**
c2+stat_qq_line (aes(sample='hwy'**))**
e + stat_bin_2d(bins = 30, drop = True**)**
x, y, fill | ..count.., ..density..
e + stat_density_2d(contour = True, n = 100**)**
x, y, color, size | ..level..
e + stat_ellipse(level = 0.95, segments = 51, type = "t"**)**
f + stat_boxplot(coef = 1.5**) x, y |** ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..
f + stat_ydensity(kernel = "gaussian", scale = "area"**) x, y |** ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..
e + stat_ecdf(n = 40**) x, y |** ..x.., ..y..
e + stat_quantile(quantiles=(0.25, 0.5, 0.75), formula = 'y ~x'**)**
x, y | ..quantile..
e + stat_smooth(method = "lm", se=True, level=0.95**) x, y |** ..se.., ..x.., ..y.., ..ymin.., ..ymax..
ggplot() + stat_function() #自定义函数处理数据
e + stat_unique()
e + stat_identity()
e + stat_sum() x, y, size | ..n.., ..prop..
e + stat_summary(fun_data = "mean_cl_boot"**)**
h + stat_summary_bin(fun_y = " mean_cl_boot ", geom = "bar"**)**

Scales

Scales 映射到图表数据点的属性，比如size，fill，color等。

n = d + geom_bar(aes(fill = 'fl'))

最常见的scale函数
scale_*_continuous() #将连续型数值映射到可视化视觉暗示
scale_*_discrete() #将离散型数值映射到可视化视觉暗示
scale_*_identity() #将时间型数值映射到可视化视觉暗示
scale_*_manual(values = ()**)** #将离散型数值映射到自定义设定的可视化视觉暗示
scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks"**)** #将数据设定为时间型
scale_*_datetime() # 将x轴数据设定为时间型

Scales

X & Y 坐标轴scale设定
scale_x_log10() #将x轴以log10的格式设定
scale_x_reverse() #将x坐标轴反转至y坐标轴
scale_x_sqrt() #将x轴以sqrrt的格式设定

离散型边框颜色和填充颜色的映射

n =d + geom_bar(aes(fill = 'fl'))
n + scale_fill_brewer(palette = "Blues"**)**
#palette的选择可参考: R语言的的RColorBrewer包

n + scale_fill_grey(start = 0.2, end = 0.8)

连续型边框颜色和填充颜色的映射

o = c + geom_dotplot(aes(fill = '..x..'))
o + scale_fill_distiller(palette = "Blues"**)**

o + scale_fill_gradient(low="red", high="yellow"**)**

o + scale_fill_gradient2(low="red", high="blue", mid = "white", midpoint = 25)

o + scale_fill_gradientn(colors= ['red','blue']**)**

形状和尺寸的映射

p = e + geom_point(aes(shape = 'fl', size = 'cyl'))
p + scale_shape() + scale_size()
p + scale_shape_manual(values =('o','v',"s","h","D"))

p + scale_size_area(max_size = 6**)**

坐标系系统

R ggplot可以兼容三大主流可视化坐标系统（笛卡尔坐标系、极坐标系、空间地理信息坐标系），将其整合在一套语法体系之下。Plotnine暂时只能使用笛卡尔坐标系。

r =d + geom_bar()
r + coord_cartesian(xlim = (0, 5))
#默认为笛卡尔坐标系，可调整xlim, ylim

r + coord_fixed(ratio = 1/2**)**
#设定笛卡尔坐标系的x和y轴的单位比例，可调整ratio, xlim, ylim

r + coord_flip()
#对调x和y坐标轴，可调整xlim, ylim

r + coord_trans(y= "sqrt"**)**
#转换的笛卡尔坐标系，可根据函数调整x或者y坐标轴

r+ coord_fixed()
x和y轴同一单位的笛卡尔坐标系

r + coord_polar(theta = "x", direction=1 **)**
#极坐标系，但是此函数plotnine包暂未实现，如需使用，可使用R ggplot2包

位置调整

s = ggplot(mpg, aes('fl', fill = 'drv'))
s + geom_bar(position = "dodge")
#使柱形数据并列排布

s + geom_bar(position = "stack")
#堆积柱形图，使柱形数据堆积排列

s + geom_bar(position = "fill")
#百分比堆积柱形图，使柱形数据堆积排列
#可以调整柱形数据的宽度
s + geom_bar(position = position_dodge(width = 1))

e + geom_point(position = "jitter")
#增加散点数据的扰动，避免重叠

e + geom_label(aes(label='cty'),position = "nudge")
#使数据标签从数据点上移开

主题

plotnine包提供了各种不同图表风格的主题函数，直接调用就可以实现，也可以在此基础上，调用theme()函数再对图表的主题元素修改与调整。

theme_538()
theme_bw()
theme_classic() #R ggplot2的绘图风格，默认
theme_dark()
theme_gray()
theme_light()
theme_linedraw()
theme_matplotlib() #python matplotlib的绘图风格
theme_minimal()
theme_seaborn() #python seaborn的绘图风格
theme_void()
theme_xkcd() #漫画风格的绘图风格

分面

plotnine包Facets函数可以根据数据的离散型变量将图表分成若干个子图表，并按一定的规则排列。

t = ggplot(mpg, aes('cty', 'hwy')) + geom_point()

t + facet_grid('. ~ fl') #根据变量按列排布

t + facet_grid('year ~ .') #根据变量按行排布

t + facet_grid('year ~ fl') #根据两个变量按行列矩阵排布

t + facet_wrap('~ fl') #根据变量按矩形排布

t + facet_grid('drv ~ fl', scales = "free")
#调整x和y坐标轴的取值范围
"free_x" – x坐标轴调整
"free_y" – y坐标轴调整
"fixed"- x和y坐标轴的取值范围统一

标签

t + labs(**x** = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
<aes> = "New <aes> legend title")
t + annotate(geom = "text", x = 8, y = 9, label = "A")

图例

n + theme(legend_position = "bottom"**)**
图例位置的设定："bottom", "top", "left", 或者 "right"
n + scale_fill_discrete(name = "Title",
labels = ("A", "B", "C", "D", "E"**)**
#使用scale函数设定图例的名字和内容

张杰：微信公众号-EasyCharts

欢迎关注