In [ ]:

```
# 一个复杂的不等距网格案例
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec

# 自定义函数，隐藏数轴刻度
def make_ticklabels_invisible(fig):
    for i, ax in enumerate(fig.axes):
        ax.text(0.5, 0.5, "ax%d" % (i+1), va="center", ha="center")
        ax.tick_params(labelbottom=False, labelleft=False)

fig = plt.figure()

gs = GridSpec(3, 3)
ax1 = plt.subplot(gs[0, :])
# identical to ax1 = plt.subplot(gs.new_subplotspec((0, 0), colspan=3))
ax2 = plt.subplot(gs[1, :-1])
ax3 = plt.subplot(gs[1:, -1])
ax4 = plt.subplot(gs[-1, 0])
ax5 = plt.subplot(gs[-1, -2])

fig.suptitle("$GridSpec$")
make_ticklabels_invisible(fig)

plt.show()
```

7.4 实战练习

寻找手边做的比较好的图形叠加/图中图实例，并尝试在python中实现。

请尝试使用子图方式实现6.4节中的面板图形，并思考面板方式和子图方式各自的优缺点与适用环境是什么。

请自行设计一个面板，在其中同时呈现第5章中使用过的anscombe四个数据集各自拟合最佳回归模型的散点图，并使其显示效果达到最佳。

8 色彩搭配

8.1 色彩搭配的基本原则

8.2 如何自定义理想的色系

8.3 色板的指定方式

8.3.1 color_palette函数

除单独指定颜色外，seaborn的优势在于可以将颜色根据数据呈现的需求编组成恰当的色板/色环加以使用。

color_palette函数可以接受任何seaborn或者matplotlib颜色表中的颜色名称（除了jet），也可以接受任何有效的matplotlib形式的颜色列表（比如RGB元组，hex颜色代码，或者HTML颜色名称）。

seaborn.color_palette(

palette = None : 希望使用的色板名称或者色彩序列，None时返回当前色板
    None, string, or sequence, optional
n_colors : int, 色板中需要使用的颜色数量
desat : float, 色彩饱和度

)返回：RGB元组组成的色彩列表

In [ ]:

```
# 返回当前色板
sns.color_palette()
```

seaborn.palplot()可以直接显示以RGB十进制代码方式给定的颜色。

In [ ]:

```
sns.palplot(((0.5, 1, 0), (0, 0, 1)))
```

8.3.2 指定具体颜色

常规的颜色指定方式

In [ ]:

```
# 使用颜色名称
sns.scatterplot(ccss.s3, ccss.index1b, hue = ccss.s2,
                palette = ['red','blue'])
```

In [ ]:

```
# 使用HTML颜色代码
sns.scatterplot(ccss.s3, ccss.index1b, hue = ccss.s2,
                palette = ['#ff0000','#0000ff'])
```

In [ ]:

```
# 使用RGB代码
sns.scatterplot(ccss.s3, ccss.index1b, hue = ccss.s2,
                palette = ((1.0,0.0,0.0),(0.0,0.0,1.0)))
```

Color Brewer调色板

Color Brewer预定义了一批调色板，可以直接按对应的名称加以调用。

具体名称请参考：ColorBrewer颜色速查表.xlsx

In [ ]:

```
print(sns.color_palette("Set1"))
sns.palplot(sns.color_palette("Set1"))
```

seaborn.choose_colorbrewer_palette()函数可用来手工调整具体的色板定义

In [ ]:

```
sns_type = ["qualitative", "sequential", "diverging"]
for elem in sns_type:
    sns.choose_colorbrewer_palette(elem)
```

使用xkcd颜色名称系统

xkcd系统对各类可能出现的RGB颜色均给出了命名，一共有954个颜色，可以随时通过xkcd_rgb字典调用。

    http://xkcd.com/color/rgb/

In [ ]:

```
plt.plot([0, 1], [0, 1], sns.xkcd_rgb["pale red"], lw = 3)
```

In [ ]:

```
# 可以将一组选择好的xkcd颜色放到 xkcd_palette 函数中
colors = ["windows blue", "amber", "greyish",
          "faded green", "dusty purple"]
sns.palplot(sns.xkcd_palette(colors))
```

8.3.3 应用色板

seaborn.set_palette()用于设定默认使用的色环，它接受与color_palette一样的参数，并会对所有的绘图的默认色环进行设置。

    注意：它会更改默认的matplotlib参数，从而成为默认的调色板配置
    也可以在with语句中使用color_palette来临时的改变默认颜色

In [ ]:

```
# 示例用绘图函数
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 7):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

sns.set_palette("husl")
sinplot()
```

In [ ]:

```
# color_palette()函数也可以在一个with块中使用，以达到临时更改调色板的目的。
with sns.color_palette("PuBuGn_d"):
    sinplot()
```

In [ ]:

```
# 再次绘图时调色板回到原先设定状态
sinplot()
```

In [ ]:

```
# 在默认设定的基础上修改个别参数
# set_style()为风格设定，包括了颜色，同时可以调整个别参数
sns.set_style("darkgrid", {"axes.facecolor": "white"})
sinplot()
```

In [ ]:

```
# 将所有设定还原至默认状态
sns.set()
sinplot()
```

8.4 分类色板

分类色板（quanlitative）对于分类数据的显示很有帮助。当想要区别不连续的且内在没有顺序关系的数据时，这个方式是最好的。

seaborn中默认使用的调色板实际上是标准的matplolib色环。

In [ ]:

```
sns.palplot(sns.color_palette())
```

默认的色环主题

有6种不同的默认主题，它们分别是：deep，muted，pastel，birght，dark，colorblind。

In [ ]:

```
themes = ['deep', 'muted', 'pastel', 'bright', 'dark', 'colorblind']
for theme in themes:
    sns.palplot(sns.color_palette(theme))
```

使用颜色空间（色圈）

当有超过6种类型的数据要区分时，最简单的方法就是在一个色圈内使用均匀分布的颜色。这也是当需要使用更多颜色时大多数seaborn函数的默认方式。

最常用的方法就是使用hls颜色空间，它是一种简单的RGB值的转换。

In [ ]:

```
sns.palplot(sns.color_palette("hls", 10))
```

In [ ]:

```
# hls_palette函数，用于调节hls颜色的亮度和饱和度。
sns.palplot(sns.hls_palette(10, l = .8, s = .8))
```

然而，由于人类视觉系统工作的原因，根据RGB颜色产生的平均视觉强度的颜色，从视觉上看起来并不是相同的强度。如果你观察仔细，就会察觉到，黄色和绿色会更亮一些，而蓝色则相对暗一些。因此，如果你想用hls系统达到一致性的效果，就会出现上面的问题。

为了修补这个问题，seaborn给hls系统提供了一个接口，可以让操作者简单容易的选择均匀分布，且亮度和饱和度看上去明显一致的色调。

In [ ]:

```
sns.palplot(sns.color_palette("husl", 10))
```

In [ ]:

```
# hls_palette函数，用于调节hls颜色的亮度和饱和度。
sns.palplot(sns.husl_palette(10, l = .8, s = .8))
```

使用分类Color Brewer调色板

Color Brewer中预定义了一批分类调色板，但颜色数量有限，可能会出现循环使用。

In [ ]:

```
print(sns.color_palette("Set1", 10))
sns.palplot(sns.color_palette("Set1", 10))
```

In [ ]:

```
sns.palplot(sns.color_palette("Set1", 10, 0.5))
```

完全使用自选颜色定义调色板

In [ ]:

```
flatui = ["#9b59b6", "#3498db", "#95a5a6", "#e74c3c",
          "#34495e", "#2ecc71"]
sns.palplot(sns.color_palette(flatui))
```

8.5 连续色板

连续色板（sequential）对于有从低（无意义）到高（有意义）范围过度的数据非常适合。

过大的色调变化会带来数据本身不连续的错觉，对于连续的数据，最好是使用那些在色调上有相对细微变化的调色板，同时在亮度和饱和度上有很大的变化。这种方法将自然地将数据中相对重要的部分成为关注点。

使用Color Brewer预定义色板

Color Brewer 的字典中就有一组很好的调色板。它们是以在调色板中的主导颜色命名的。

> 具体名称请参考：ColorBrewer颜色速查表.xlsx

In [ ]:

```
sns.palplot(sns.color_palette("Blues"))
```

和在matplotlib中一样，如果想要翻转渐变，可以在面板名称中添加一个_r后缀。

In [ ]:

```
sns.palplot(sns.color_palette("BuGn_r"))
```

seaborn还增加了一个允许创建没有动态范围的"dark"面板。如果想按顺序画线或点，这可能比较有用，因为颜色鲜艳的线可能很难区分。

类似的，这种暗处理的颜色，需要在面板名称中添加一个_d后缀。

In [ ]:

```
sns.palplot(sns.color_palette("GnBu_d"))
```

如果想返回一个变量当做颜色映射传入seaborn或matplotlib的函数中，可以设置 as_cmap 参数为True。

In [ ]:

```
cmap = sns.cubehelix_palette(light = 1, as_cmap = True)
cmap.colors
```

In [ ]:

```
x, y = np.random.multivariate_normal([0, 0], [[1, -.5], [-.5, 1]],
                                     size=300).T
cmap = sns.cubehelix_palette(light = 1, as_cmap = True)
sns.kdeplot(x, y, cmap = cmap, shade = True)
```

定制连续调色板

可以使用light_palette() 或者 dark_palette()函数，用更简单的方式定制连续色板。这两个函数可以产生从亮值或者暗去饱和的值到这个颜色的调色板。

同样可以使用choose_light_palette和choose_dark_palette两个函数来交互式的调节创建调色板。

In [ ]:

```
sns.palplot(sns.light_palette("green"))
```

In [ ]:

```
sns.palplot(sns.dark_palette("purple"))
```

In [ ]:

```
sns.palplot(sns.light_palette("navy", reverse = True))
```

In [ ]:

```
# 也可以创建为一个颜色映射对象，而不仅仅是颜色列表。
pal = sns.dark_palette("palegreen", as_cmap = True)
sns.kdeplot(x, y, cmap = pal)
```

In [ ]:

```
# 使用默认的hsl格式交互创建调色板
sns.choose_light_palette()
```

In [ ]:

```
sns.choose_light_palette('rgb')
```

8.6 离散色板

这类色板适用于数据需要呈现从最低值到最高值的数值变化情况，且数据中通常有一个意义明确的中点。例如，如果想从某个基线时间点绘制温度变化，最好使用离散的颜色表显示相对降低和相对增加面积的地区。

除了需要选择中点色，以及两个方向的起始颜色外，选择离散色板的基本规则类似于顺序色板。

使用Color Brewer预定义色板

Color Brewer颜色字典里也拥有一套精心挑选的离散颜色映射可供使用。

> 具体名称请参考：ColorBrewer颜色速查表.xlsx

In [ ]:

```
sns.palplot(sns.color_palette("BrBG", 7))
```

定制离散色板

也可以使用seaborn.diverging_palette()为离散的数据创建一个定制的色板。

> 该函数使用husl颜色系统，需要给出两端的颜色，并可选择性的设定明度和饱和度
> choose_diverging_palette()则可以进行色板的交互设定

In [ ]:

```
# husl颜色取值范围：[0, 359]
sns.palplot(sns.diverging_palette(220, 20, n = 7))
```

In [ ]:

```
# sep参数控制面板中间区域的两个渐变的宽度。
sns.palplot(sns.diverging_palette(10, 220, sep = 80, n = 7))
```

In [ ]:

```
# 使用中间的色调而不是亮度来调色，center : {“light”, “dark”}
sns.palplot(sns.diverging_palette(255, 133, l = 60, n = 7,
                                  center = "dark"))
```

In [ ]:

```
# 交互式定制离散色板
sns.choose_diverging_palette()
```

8.7 实战练习

请自行判断下列数据可视化需求应当使用哪种类型的色板进行修饰：

比较不同职业的人群其当前家庭经济状况信心值（QA3）的差异，该数值以100为中值，上下波动范围为0~200。
比较不同职业的人群其受教育状况的构成比有怎样的差异。
比较不同职业的人群其家庭平均收入有怎样的差异。

尝试使用choose_light_palette函数配制自定义的连续色板，然后利用适当的函数在绘图中使用该色板。

9 统计图的进一步美化与修饰

9.1 专业图表的风格应当是怎样的?

9.2 Seaborn的样式管理

Seaborn 将 matplotlib 的参数划分为两个独立的组合：

第一组是设置绘图Axes对象的整体外观风格
    axes_style()
    set_style()
第二组主要设定图中各种图形元素的大小
    plotting_context()
    set_context()

每对方法中的第一个方法（axes_style(), plotting_context()）会返回一组字典参数。

第二个方法（set_style(), set_context()）会设置matplotlib中对应的默认参数。

但是用户只能通过这个方法来覆盖matplotlib的风格定义中的部分参数。

9.2.1 设定绘图外观风格

axes_style()返回的相关参数设置会影响Axes对象颜色设定、网格设定和其他视觉设定。

直接使用预设主题模板

seaborn.set_style()中提供了一些预设主题模板：

暗网格(darkgrid)
白网格(whitegrid)
全黑(dark)
全白(white)
全刻度(ticks)