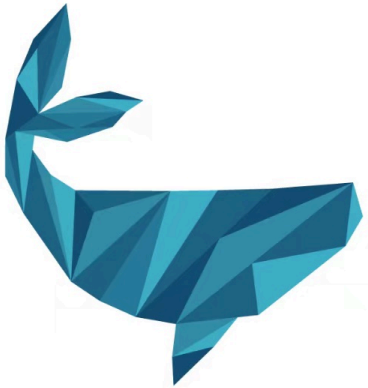


鲸析

Seaborn简易教程

基本信息

Seaborn是一个用Python制作统计图形的库。它建立在matplotlib之上，并与panda数据结构紧密集成

以下是seaborn提供的一些功能:

- 一个面向数据集的API，用于检查多个变量之间的关系
- 专门支持使用分类变量来显示观察结果或汇总统计数据
- 用于可视化单变量或双变量分布以及在数据子集之间进行比较的选项
- 各类因变量线性回归模型的自动估计与作图
- 方便查看复杂数据集的整体结构
- 用于构建多图块网格的高级抽象，使您可以轻松地构建复杂的可视化
- 对matplotlib图形样式与几个内置主题的简洁控制
- 选择调色板的工具，忠实地揭示您的数据模式
- Seaborn的目标是使可视化成为探索和理解数据的核心部分。它的面向数据集的绘图功能对包含整个数据集的数据流和数组进行操作，并在内部执行必要的语义映射和统计聚合以生成信息图。

---

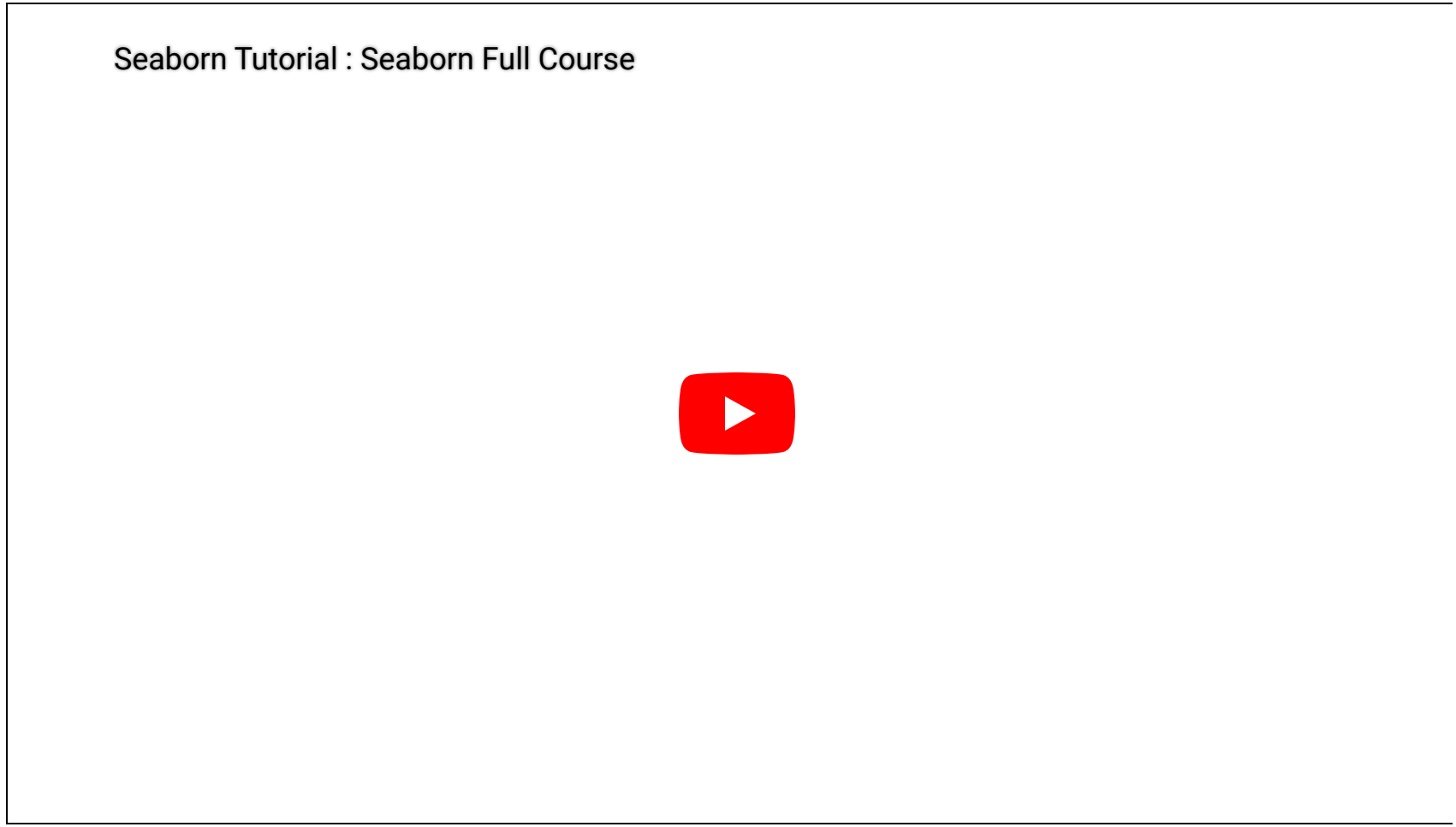
1. 线型
2. 点型
3. 柱型
4. 密度型
5. 矩阵型

推荐视频🤩🤩🤩

In [1]:
```
from IPython.display import IFrame
```

In [2]:
```
IFrame(width="853",height="480",src = "https://www.youtube.com/embed/6GUZXDef2U0")
```

Out[2]:



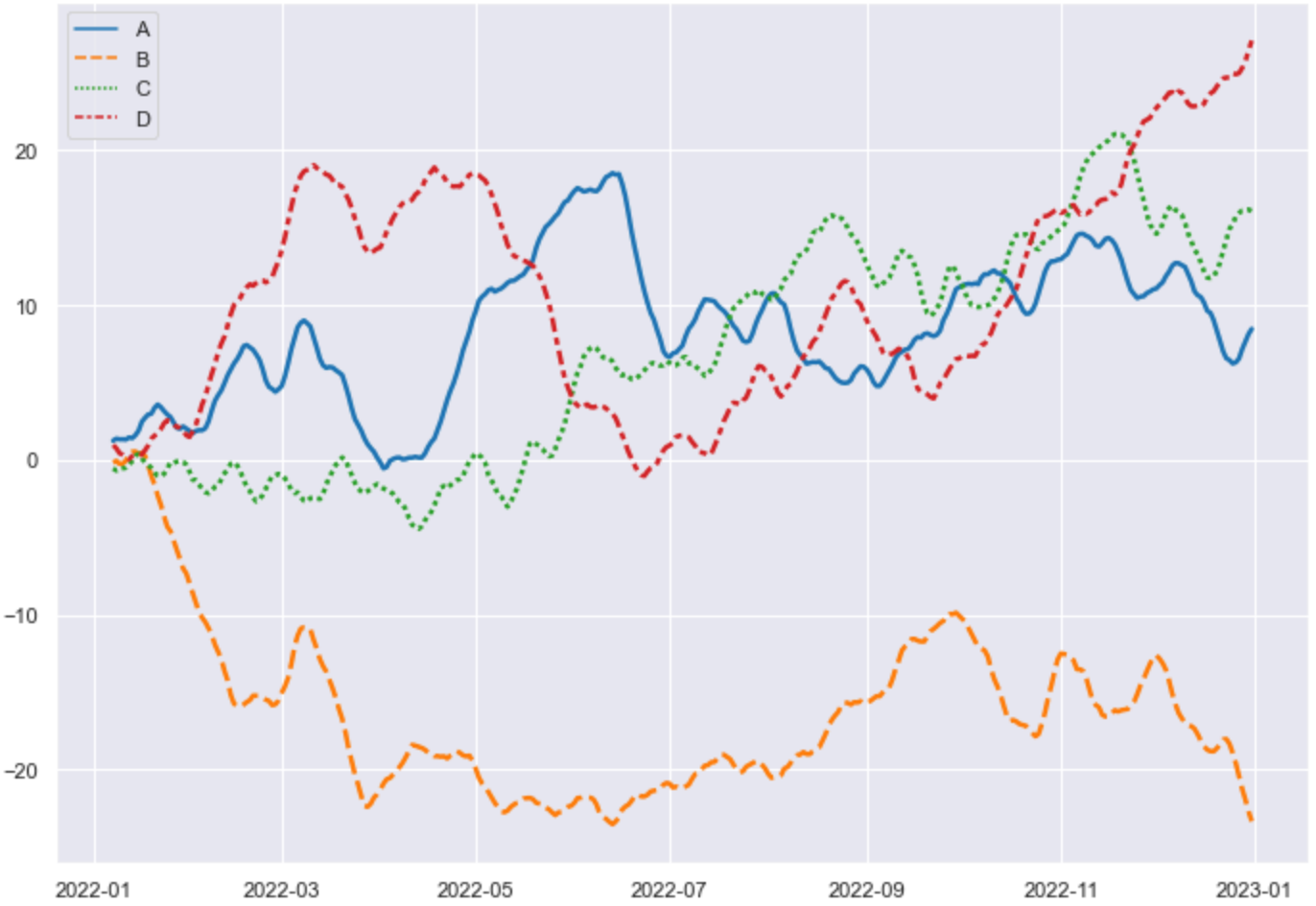
线型

In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="whitegrid")
sns.set(rc={'figure.figsize':(11.7,8.27)})
```

Lineplot from a wide-form dataset

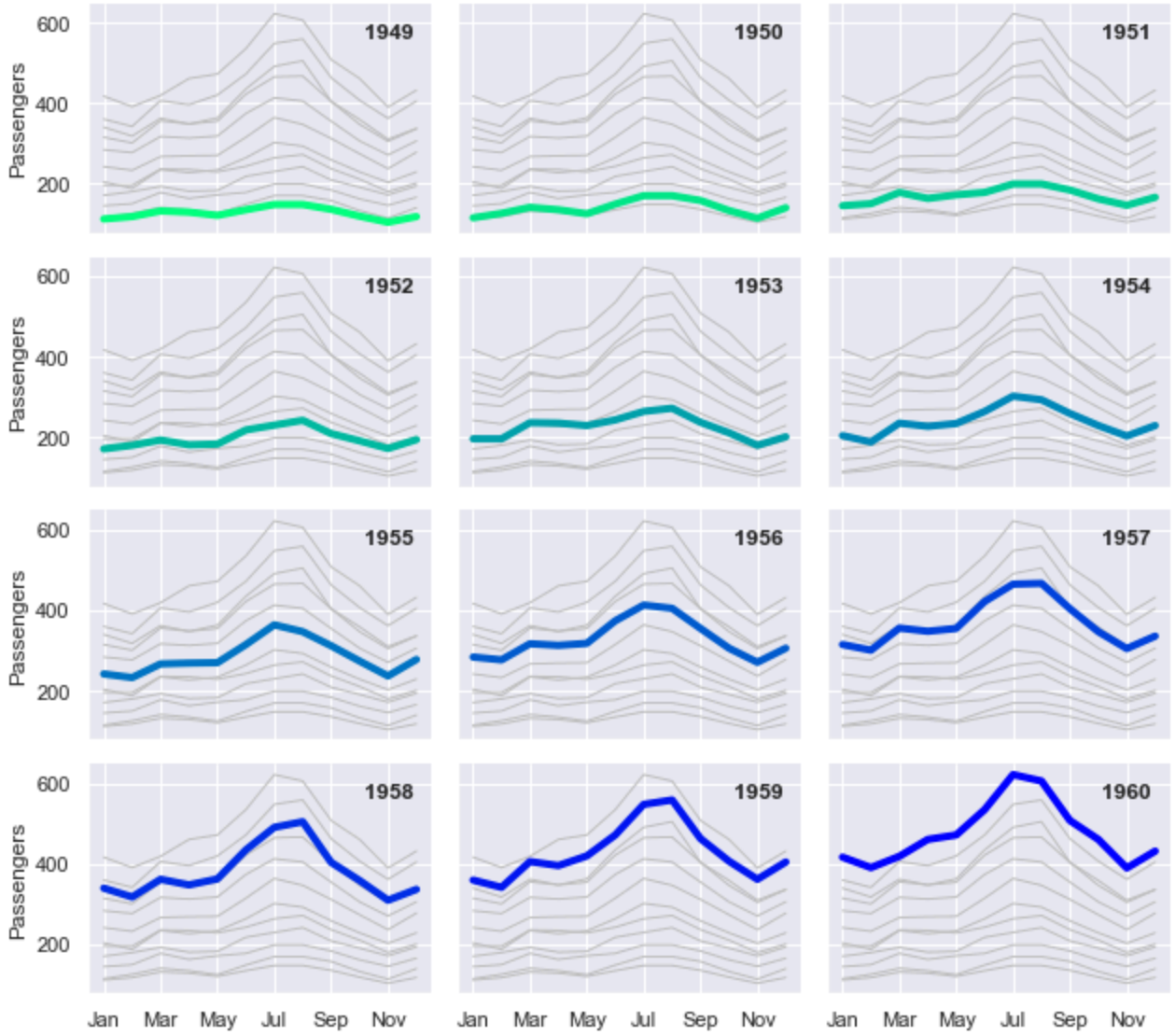
In [4]:
```
rs = np.random.RandomState(2022)
values = rs.randn(365, 4).cumsum(axis=0)
dates = pd.date_range("1 1 2022", periods=365, freq="D")
data = pd.DataFrame(values, dates, columns=["A", "B", "C", "D"])
data = data.rolling(7).mean()

sns.lineplot(data=data, palette="tab10", linewidth=2.5)
```

Out[4]: <AxesSubplot:>



multiple time series

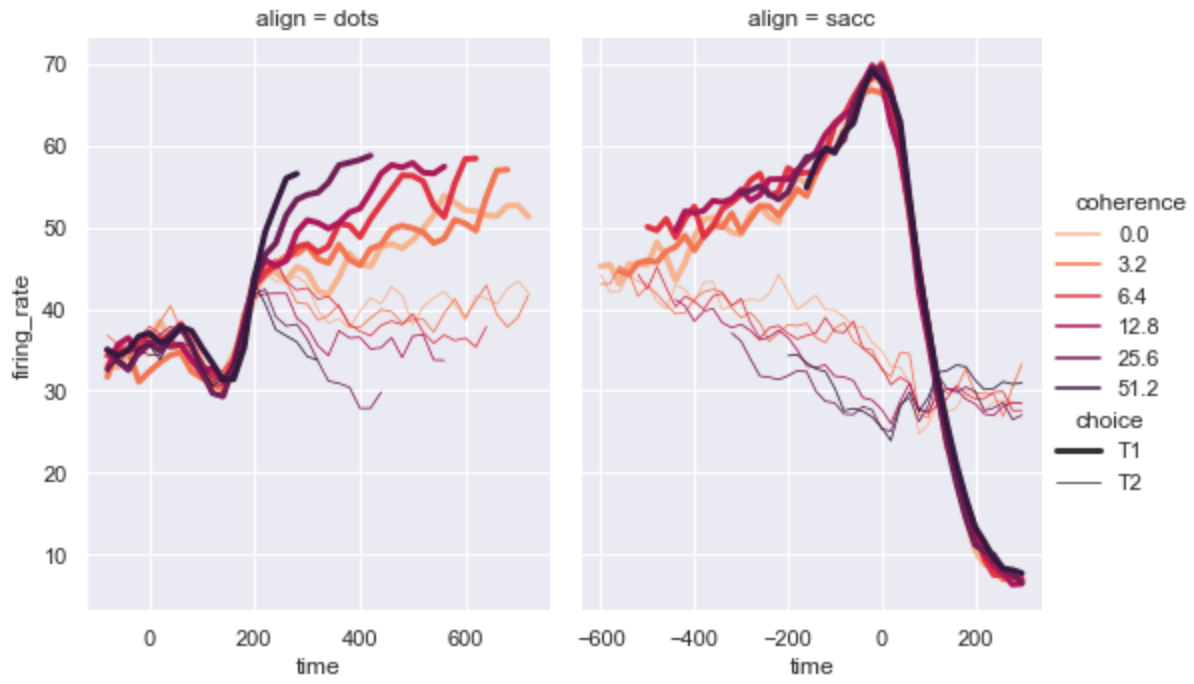
In [5]:
```
flights = sns.load_dataset("flights")

# Plot each year's time series in its own facet
g = sns.relplot(
    data=flights,
    x="month", y="passengers", col="year", hue="year",
    kind="line", palette="winter_r", linewidth=4, zorder=5,
    col_wrap=3, height=2, aspect=1.5, legend=False,
)

# Iterate over each subplot to customize further
for year, ax in g.axes_dict.items():

    # Add the title as an annotation within the plot
    ax.text(.8, .85, year, transform=ax.transAxes, fontweight="bold")

    # Plot every year's time series in the background
    sns.lineplot(
        data=flights, x="month", y="passengers", units="year",
        estimator=None, color=".75", linewidth=1, ax=ax,
    )

# Reduce the frequency of the x axis ticks
ax.set_xticks(ax.get_xticks()[::2])

# Tweak the supporting aspects of the plot
g.set_titles("")
g.set_axis_labels("", "Passengers")
g.tight_layout()
```



Line plots on multiple facets

In [6]:
```
dots = sns.load_dataset("dots")

# Define the palette as a list to specify exact values
palette = sns.color_palette("rocket_r")

# Plot the lines on two facets
sns.relplot(
    data=dots,
    x="time", y="firing_rate",
    hue="coherence", size="choice", col="align",
    kind="line", size_order=["T1", "T2"], palette=palette,
    height=5, aspect=.75, facet_kws=dict(sharex=False),
)
```

Out[6]: <seaborn.axisgrid.FacetGrid at 0x1f796d29fc8>

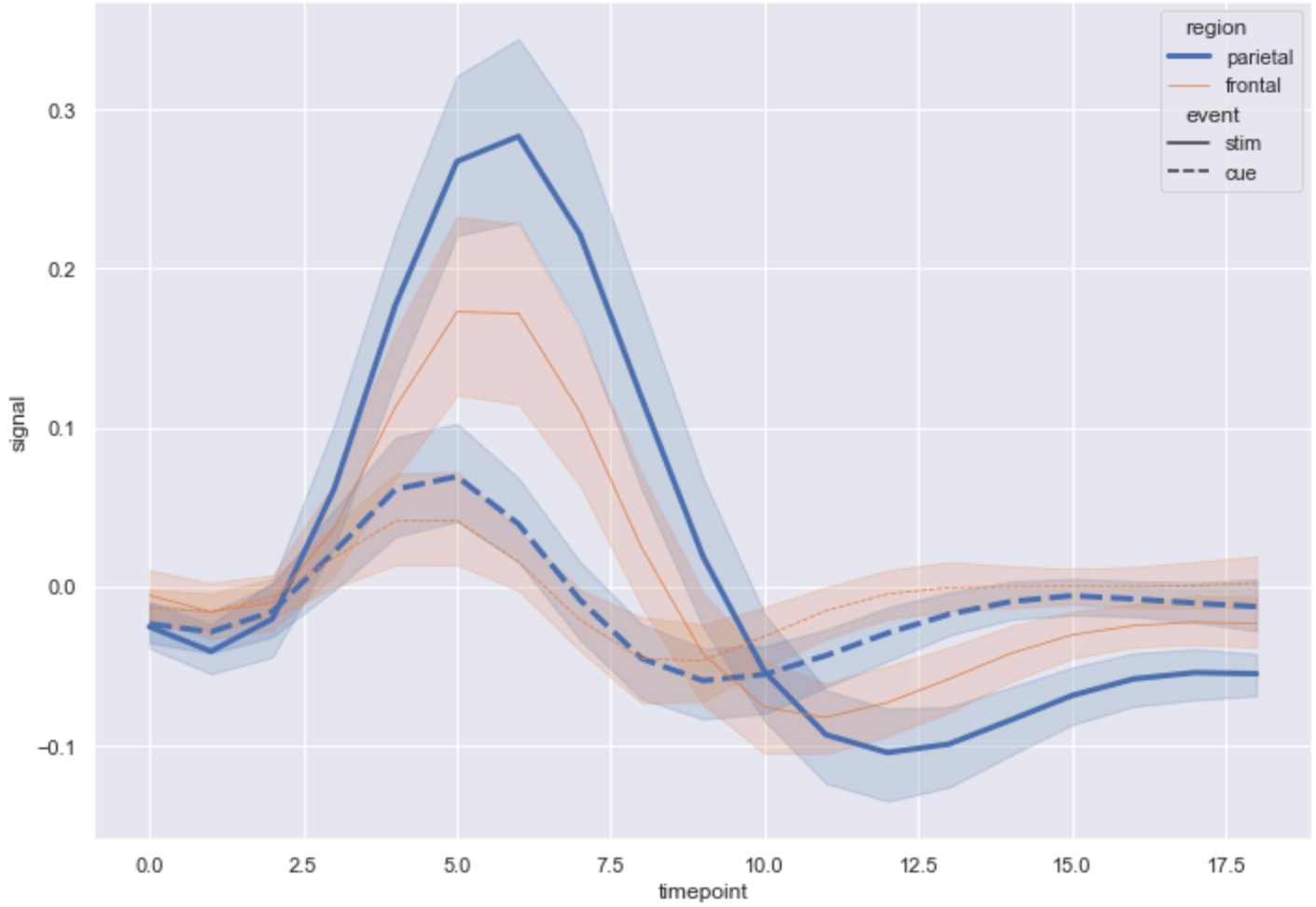


Timeseries plot with error bands

Timeseries plot with error bands

In [7]:
```
fmri = sns.load_dataset("fmri")

# Plot the responses for different events and regions
sns.lineplot(x="timepoint", y="signal",
             hue="region", style="event",size = "region",
             data=fmri)
```

Out[7]: <AxesSubplot:xlabel='timepoint', ylabel='signal'>

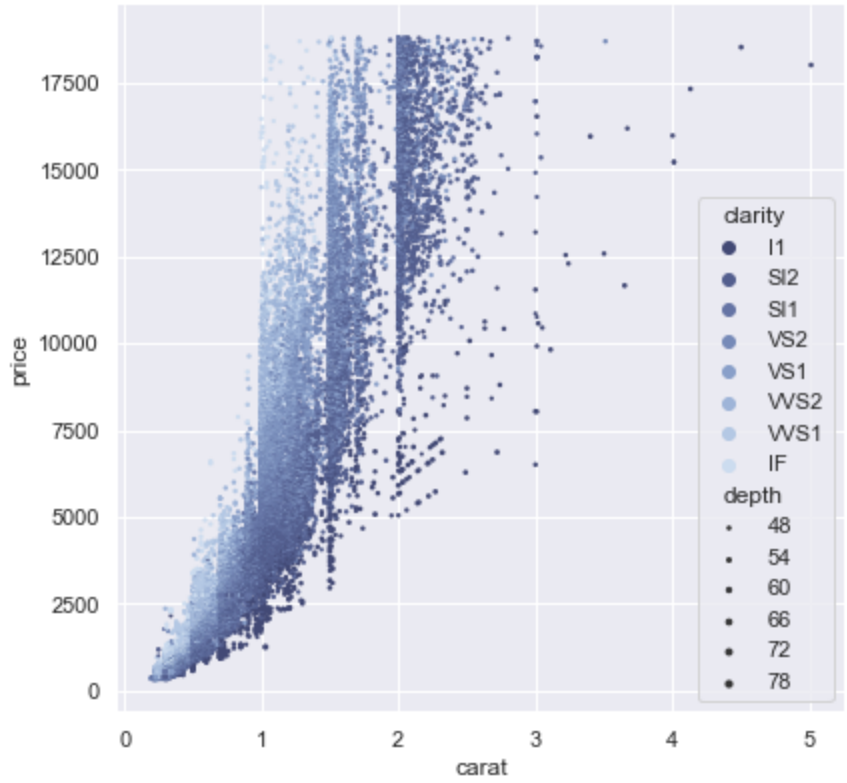


点型

Scatterplot with multiple semantics

In [8]:
```
diamonds = sns.load_dataset("diamonds")

# Draw a scatter plot while assigning point colors and sizes to different
# variables in the dataset
f, ax = plt.subplots(figsize=(6.5, 6.5))
sns.despine(f, left=True, bottom=True)
clarity_ranking = ["I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF"]
sns.scatterplot(x="carat", y="price",
                hue="clarity", size="depth",
                palette="ch:r=-.1,d=.3_r",
                hue_order=clarity_ranking,
                sizes=(1,10),
                linewidth=0,
                data=diamonds, ax=ax)
```

Out[8]: <AxesSubplot:xlabel='carat', ylabel='price'>



Conditional means with observations

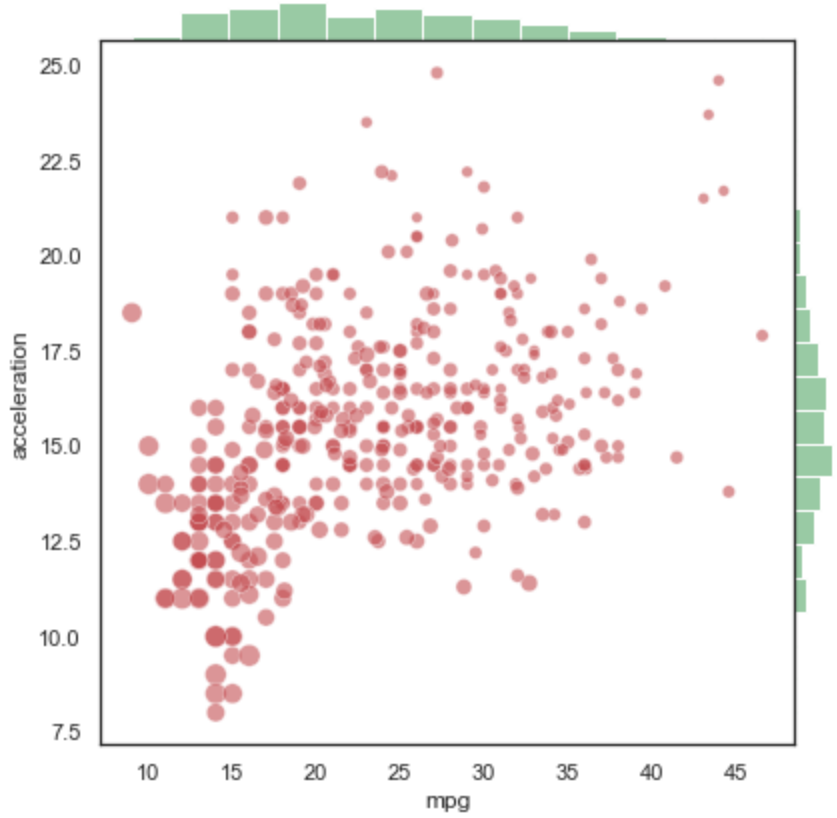
In [9]:
```
sns.set_theme(style="darkgrid")
iris = sns.load_dataset("iris")

# "Melt" the dataset to "long-form" or "tidy" representation
iris = pd.melt(iris, "species", var_name="measurement")

# Initialize the figure
f, ax = plt.subplots()
sns.despine(bottom=True, left=True)

# Show each observation with a scatterplot
sns.stripplot(x="value", y="measurement", hue="species",
              data=iris, dodge=True, alpha=.25, zorder=1)

# Show the conditional means, aligning each pointplot in the
# center of the strips by adjusting the width allotted to each
# category (.8 by default) by the number of hue levels
sns.pointplot(x="value", y="measurement", hue="species",
              data=iris, dodge=.8 - .8 / 3,
              join=False, palette="dark",
              markers="d", scale=.75, ci=None)

# Improve the legend
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles[3:], labels[3:], title="species",
          handletextpad=0, columnspacing=1,
          loc="lower right", ncol=3, frameon=True)
```

Out[9]: <matplotlib.legend.Legend at 0x1f79714bc88>

Scatterplot with marginal ticks

In [10]:
```
sns.set_theme(style="white", color_codes=True)
mpg = sns.load_dataset("mpg")

# Use JointGrid directly to draw a custom plot
g = sns.JointGrid(data=mpg, x="mpg", y="acceleration", space=0, ratio=17)
g.plot_joint(sns.scatterplot, size=mpg["horsepower"], sizes=(30, 120),
             color="r", alpha=.6, legend=False)
g.plot_marginals(sns.histplot,color="g", alpha=.6)
```

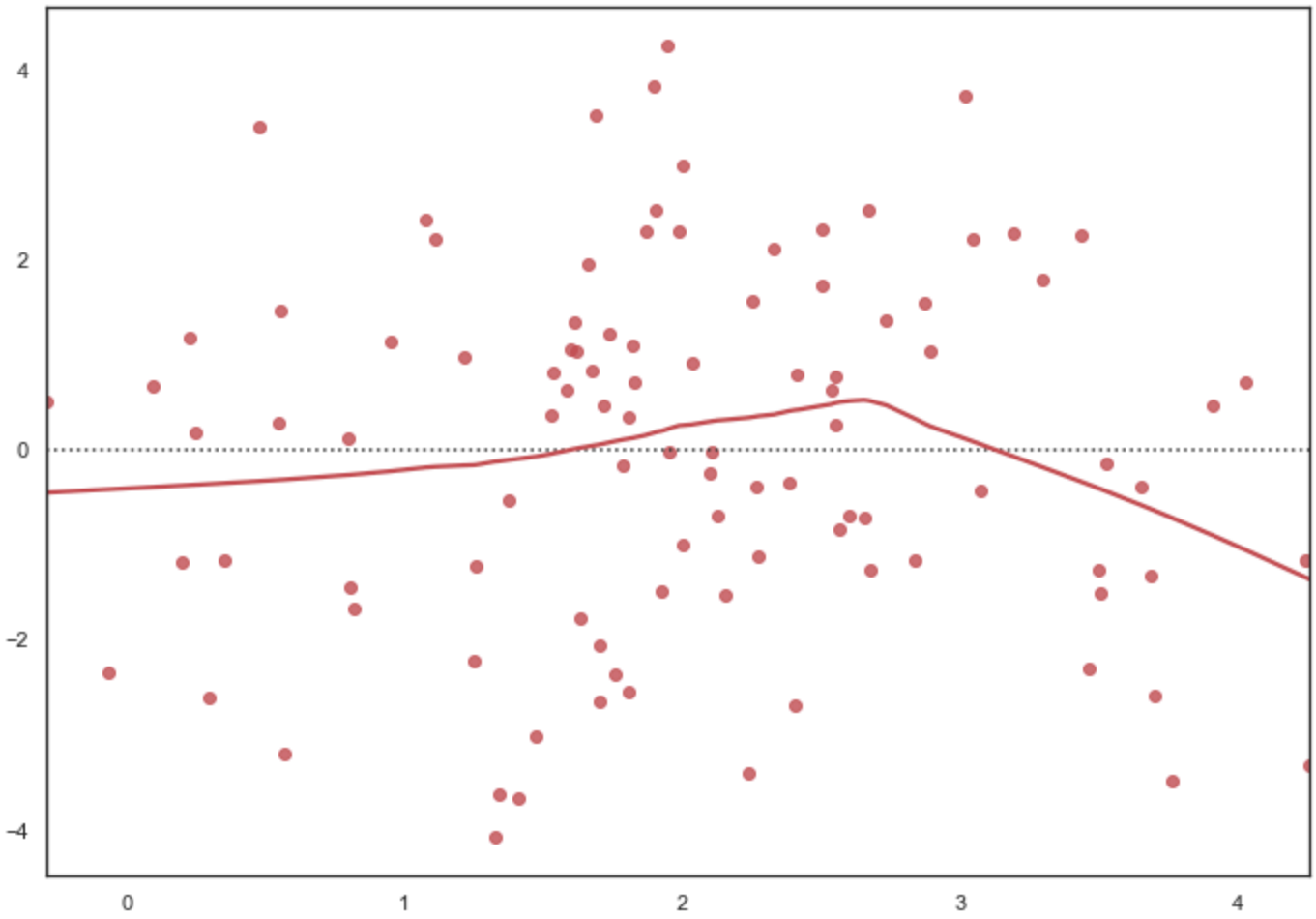
Out[10]: <seaborn.axisgrid.JointGrid at 0x1f797259ec8>



Plotting model residuals

In [11]:
```
rs = np.random.RandomState(7)
x = rs.normal(2, 1, 100)
y = 20 + 1.5 * x + rs.normal(0, 2, 100)

# Plot the residuals after fitting a linear model
sns.residplot(x=x, y=y, lowess=True, color="r")
```

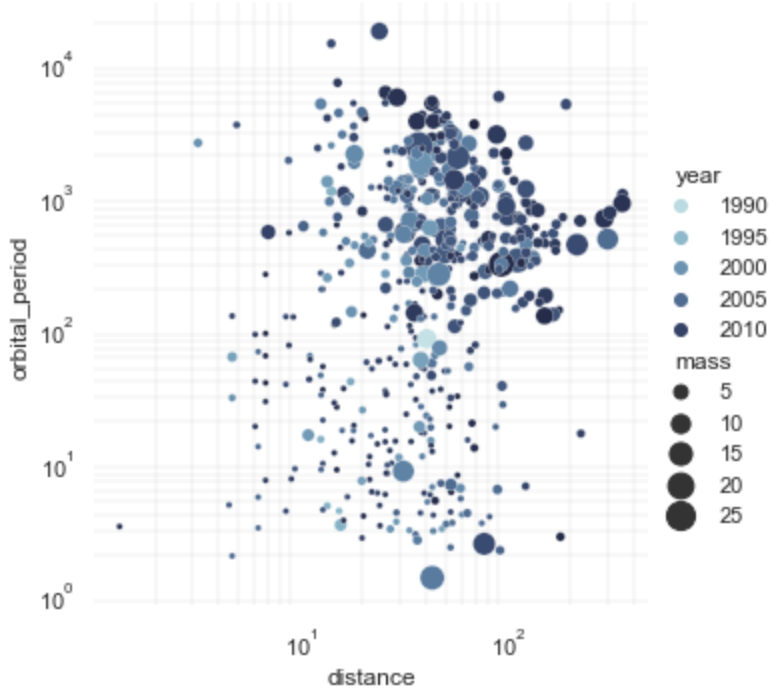
Out[11]: <AxesSubplot:>



Scatterplot with continuous hues and sizes

In [12]:
```
planets = sns.load_dataset("planets")

cmap = sns.cubehelix_palette(rot=-.2, as_cmap=True)
g = sns.relplot(
    data=planets,
    x="distance", y="orbital_period",
    hue="year", size="mass",
    palette=cmap, sizes=(10, 200),
)
g.set(xscale="log", yscale="log")
g.ax.xaxis.grid(True, "minor", linewidth=.25)
g.ax.yaxis.grid(True, "minor", linewidth=.25)
g.despine(left=True, bottom=True)
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0x1f79699c3c8>

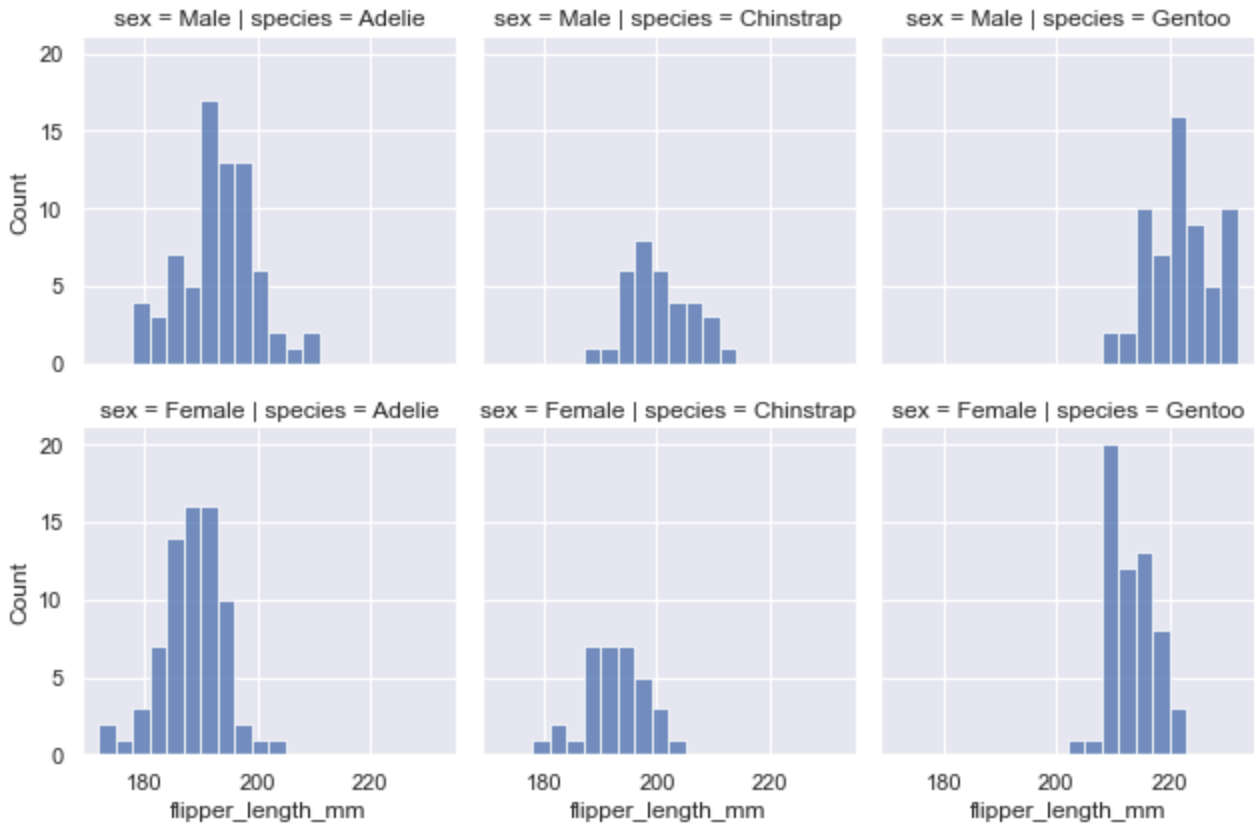


柱型

Facetting histograms by subsets of data

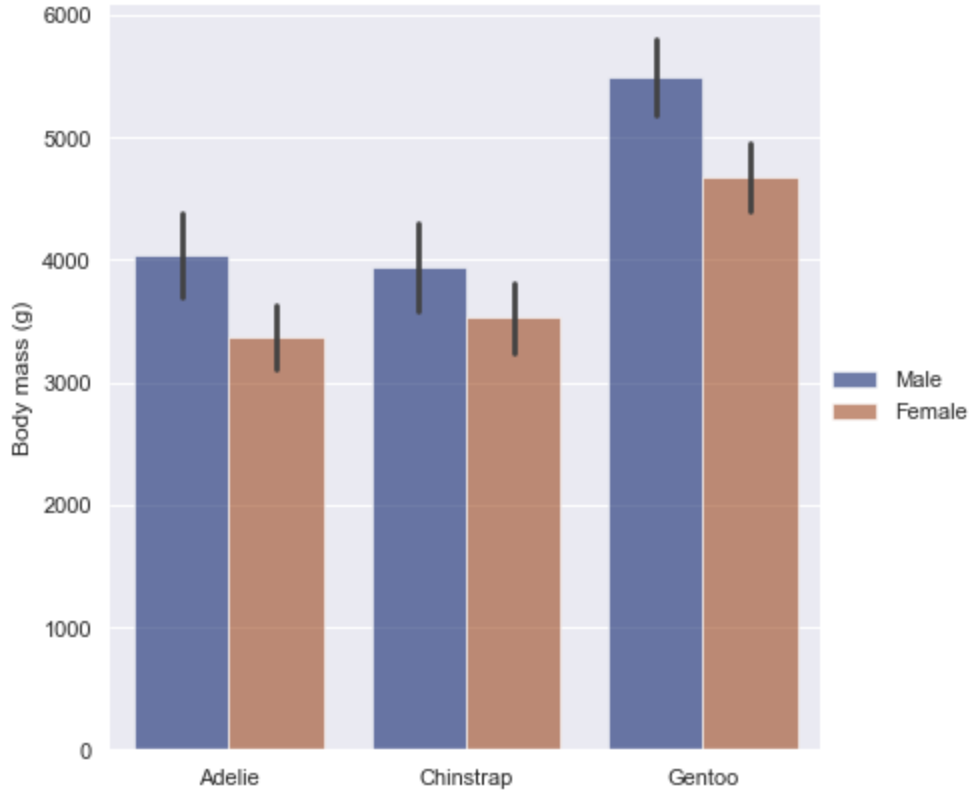
In [13]:
```
sns.set_theme(style="darkgrid")
df = sns.load_dataset("penguins")
sns.displot(
    df, x="flipper_length_mm", col="species", row="sex",
    binwidth=3, height=3, facet_kws=dict(margin_titles=False),
)
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x1f7966a8408>



Grouped barplots

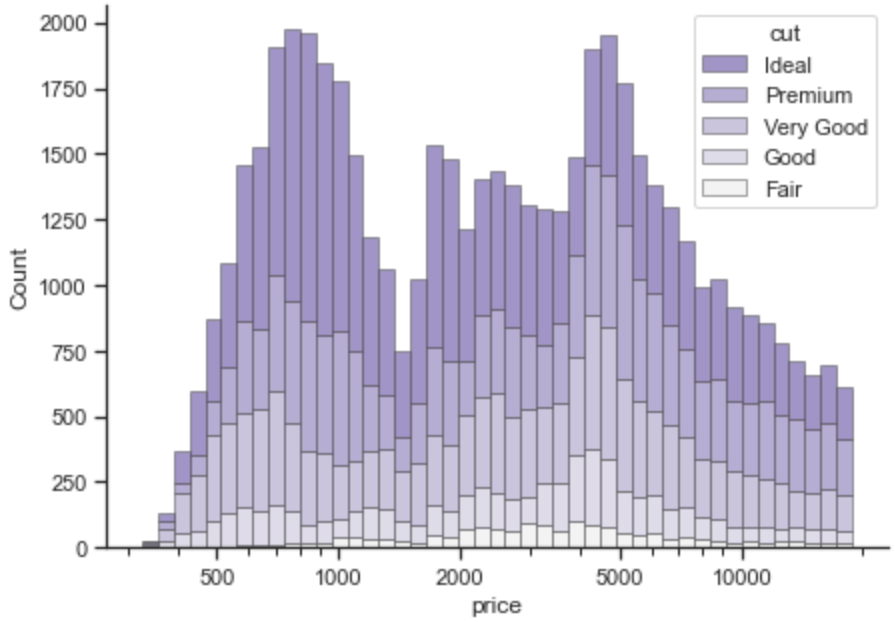
In [14]:
```
penguins = sns.load_dataset("penguins")

# Draw a nested barplot by species and sex
g = sns.catplot(
    data=penguins, kind="bar",
    x="species", y="body_mass_g", hue="sex",
    ci="sd", palette="dark", alpha=.6, height=6
)
g.despine(left=True)
g.set_axis_labels("", "Body mass (g)")
g.legend.set_title("")
```



Stacked histogram on a log scale

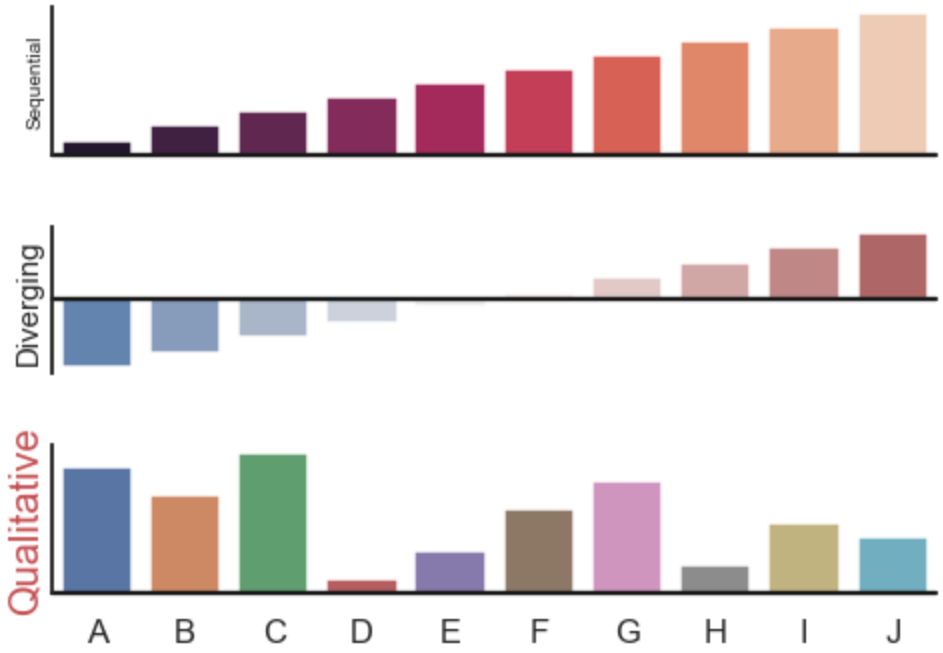
In [15]:
```
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt

sns.set_theme(style="ticks")

diamonds = sns.load_dataset("diamonds")

f, ax = plt.subplots(figsize=(7, 5))
sns.despine(f)

sns.histplot(
    diamonds,
    x="price", hue="cut",
    multiple="stack",
    palette="light:m_r",
    edgecolor=".3",
    linewidth=.5,
    log_scale=True,
)
ax.xaxis.set_major_formatter(mpl.ticker.ScalarFormatter())
ax.set_xticks([500, 1000, 2000, 5000, 10000])
```

Out[15]:
```
[<matplotlib.axis.XTick at 0x1f796c66e48>,
 <matplotlib.axis.XTick at 0x1f796c0c708>,
 <matplotlib.axis.XTick at 0x1f798873e08>,
 <matplotlib.axis.XTick at 0x1f798dba4c8>,
 <matplotlib.axis.XTick at 0x1f798dbf448>]
```

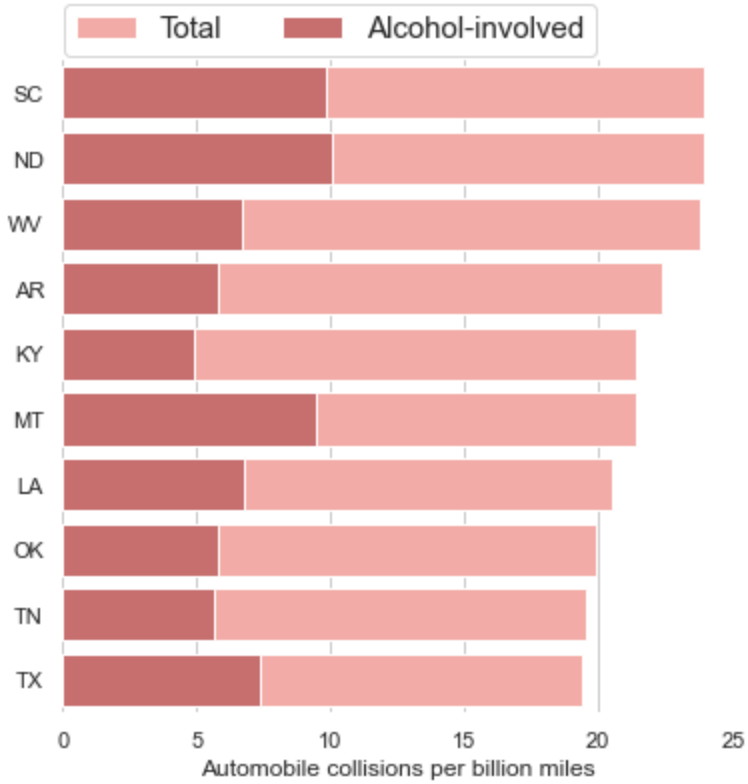


Color palette choices

In [16]:
```
sns.set_theme(style="white", context="talk")
rs = np.random.RandomState(8)

# Set up the matplotlib figure
f, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(7, 5), sharex=True)

# Generate some sequential data
x = np.array(list("ABCDEFGHIJ"))
y1 = np.arange(1, 11)
sns.barplot(x=x, y=y1, palette="rocket", ax=ax1)
ax1.axhline(0, color="k", clip_on=False)
ax1.set_ylabel("Sequential",size = 10)

# Center the data to make it diverging
y2 = y1 - 5.5
sns.barplot(x=x, y=y2, palette="vlag", ax=ax2)
ax2.axhline(0, color="k", clip_on=False)
ax2.set_ylabel("Diverging",size = 15)

# Randomly reorder the data to make it qualitative
y3 = rs.choice(y1, len(y1), replace=False)
sns.barplot(x=x, y=y3, palette="deep", ax=ax3)
ax3.axhline(0, color="k", clip_on=False)
ax3.set_ylabel("Qualitative",size = 20, color = 'r')

# Finalize the plot
sns.despine(bottom=True)
plt.setp(f.axes, yticks=[])
plt.tight_layout(h_pad=2)
```


Sequential
Diverging
Qualitative
A
B
C
D
E
F
G
H
I
J

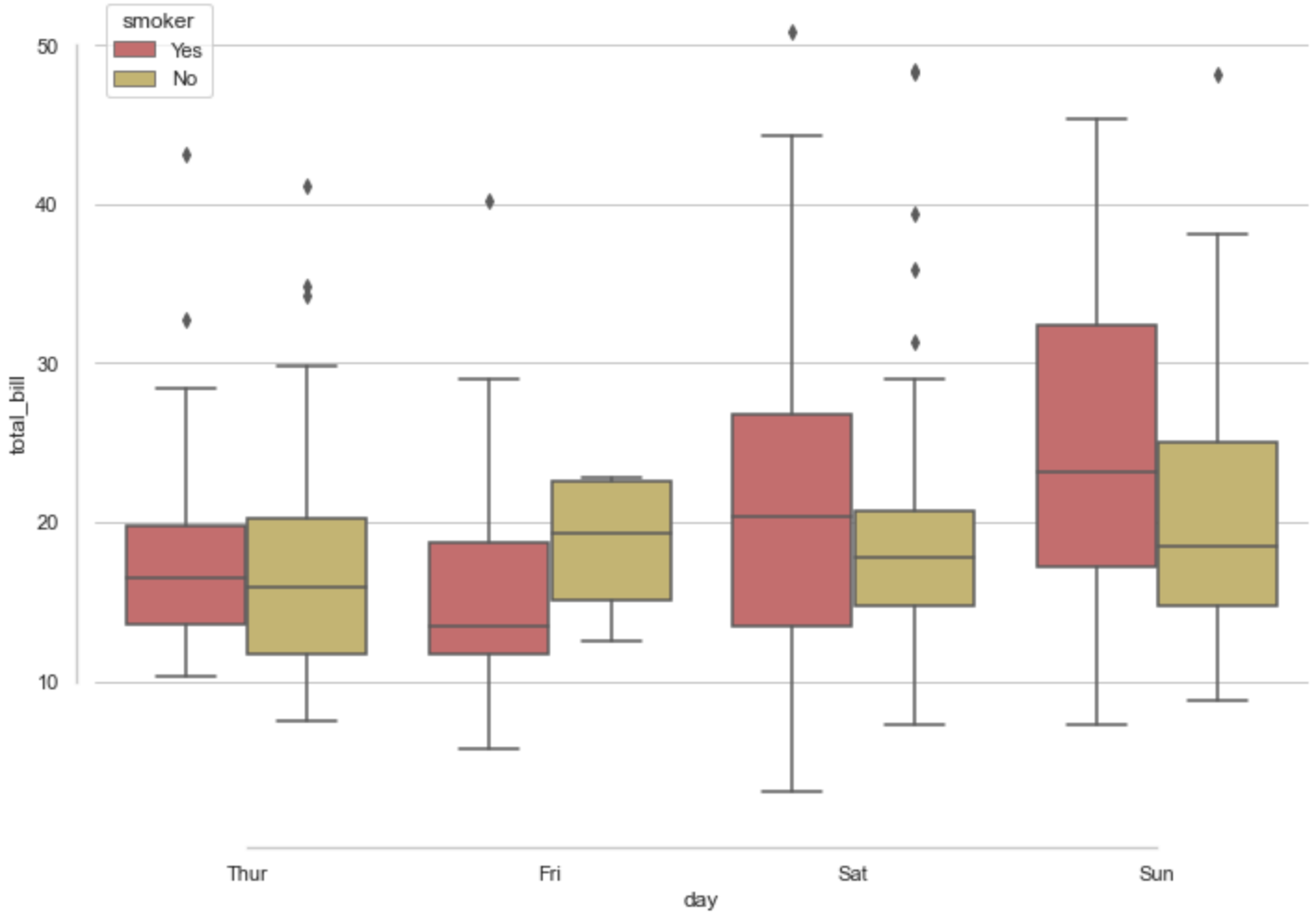
In [17]:
```
# Initialize the matplotlib figure
sns.set_theme(style="whitegrid")
f, ax = plt.subplots(figsize=(6, 6))

# Load the example car crash dataset
crashes = sns.load_dataset("car_crashes").sort_values("total", ascending=False).head(10)

# Plot the total crashes
sns.set_color_codes("pastel")
sns.barplot(x="total", y="abbrev", data=crashes,
            label="Total", color="r")

# Plot the crashes where alcohol was involved
sns.set_color_codes("muted")
sns.barplot(x="alcohol", y="abbrev", data=crashes,
            label="Alcohol-involved", color="r")

# Add a legend and informative axis label
ax.legend(ncol=2, frameon=True, fontsize = 15, bbox_to_anchor=(.82, 1.11))
ax.set(xlim=(0, 25), ylabel="",
       xlabel="Automobile collisions per billion miles")
sns.despine(left=True, bottom=True)
```



密度型

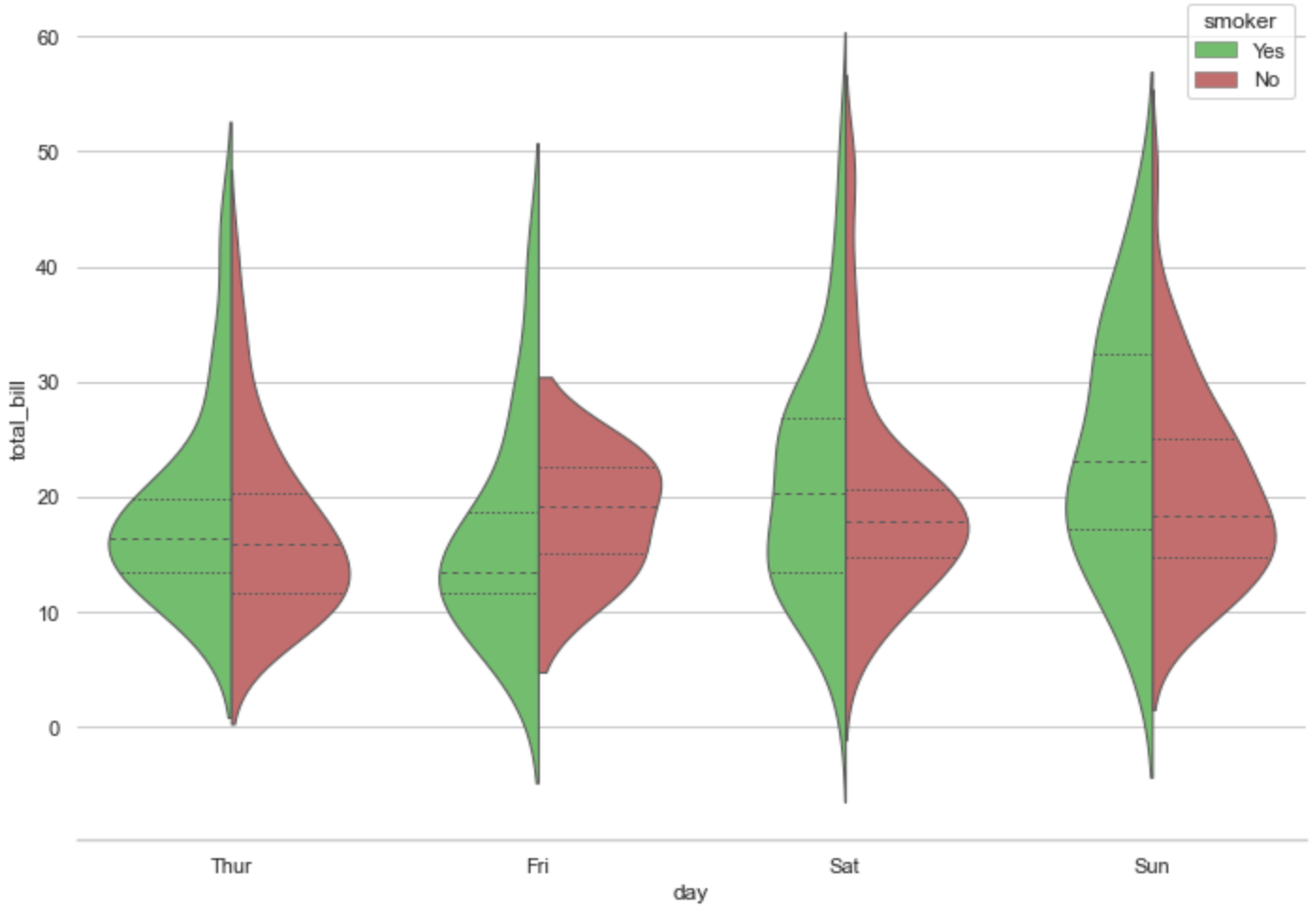
Grouped boxplots

In [18]:
```
# Load the example tips dataset
tips = sns.load_dataset("tips")

# Draw a nested boxplot to show bills by day and time
sns.boxplot(x="day", y="total_bill",
            hue="smoker", palette=["r", "y"],
            data=tips)
sns.despine(offset=10, trim=True)
```



Grouped violinplots with split violins

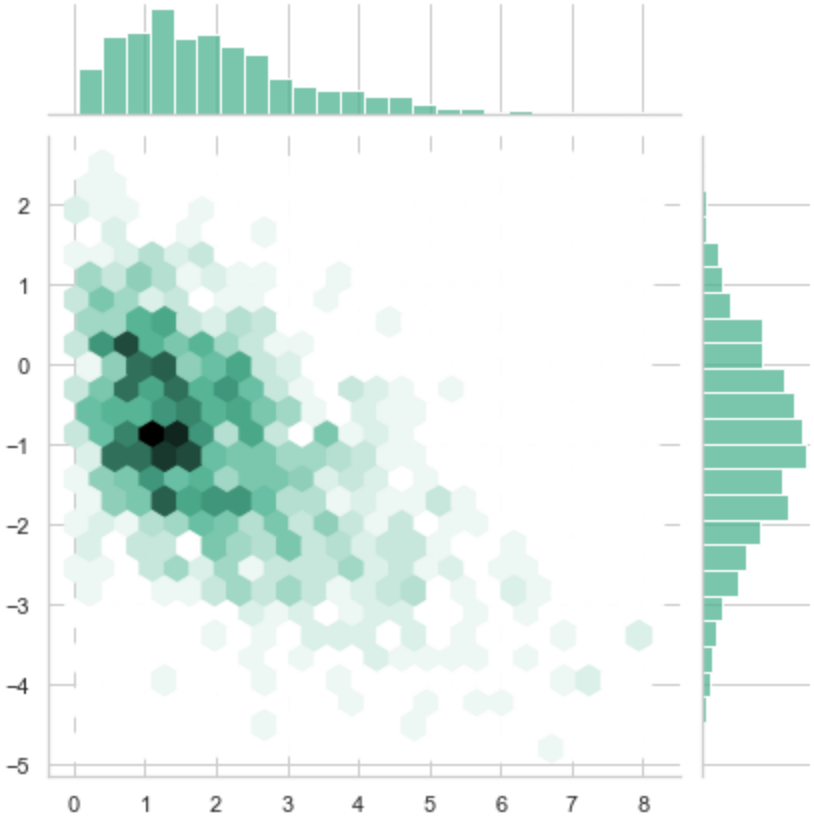
In [19]:
```
# Draw a nested violinplot and split the violins for easier comparison
sns.violinplot(data=tips, x="day", y="total_bill", hue="smoker",
               split=True, inner="quart", linewidth=1,
               palette={"Yes": "g", "No": "r"})
sns.despine(left=True)
# Remove the top and right spines from plot(s).
```



Hexbin plot with marginal distributions

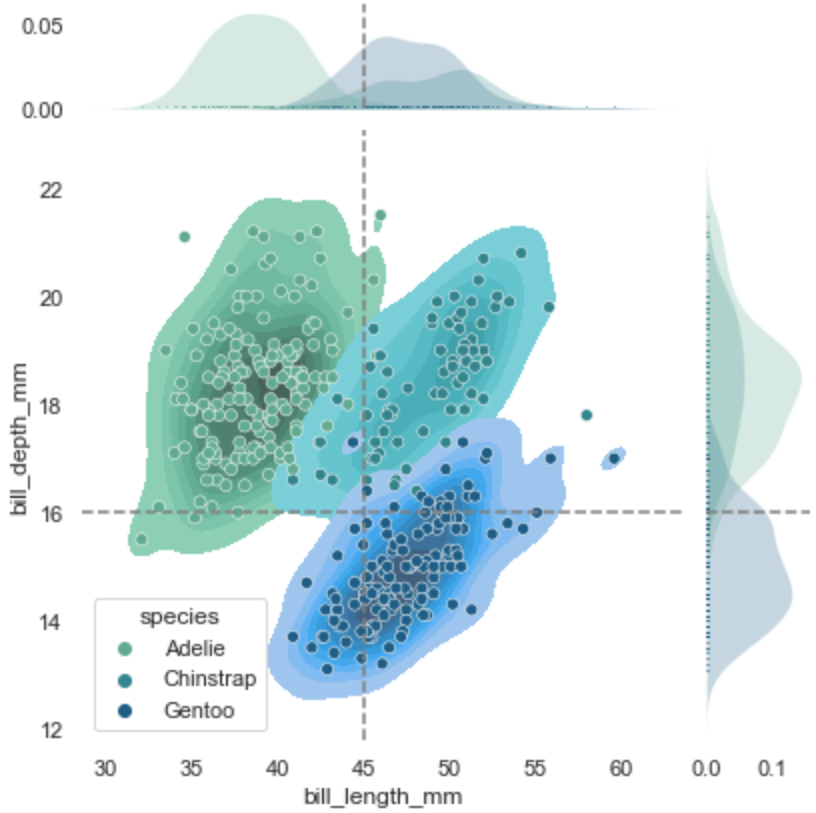
In [20]:
```
rs = np.random.RandomState(11)
x = rs.gamma(2, size=1000)
y = -.5 * x + rs.normal(size=1000)

sns.jointplot(x=x, y=y, kind="hex", color="#4CB391")
```

Out[20]: <seaborn.axisgrid.JointGrid at 0x1f7996c13c8>



Joint kernel density estimate

In [21]:
```
sns.set_theme(style="white", context="notebook")
sns.set_context()
penguins = sns.load_dataset("penguins")

# Show the joint distribution using kernel density estimation
g = sns.jointplot(
    data=penguins,
    x="bill_length_mm", y="bill_depth_mm", hue="species",
    kind="kde",
    shade = True,
    palette = 'crest',
    marginal_ticks = True,
    marginal_kws = {'shade':True,
                    'linewidth':0}
)

g.plot(sns.scatterplot, sns.rugplot)
g.refline(x=45, y=16)

sns.despine(left=True,bottom = True)
```

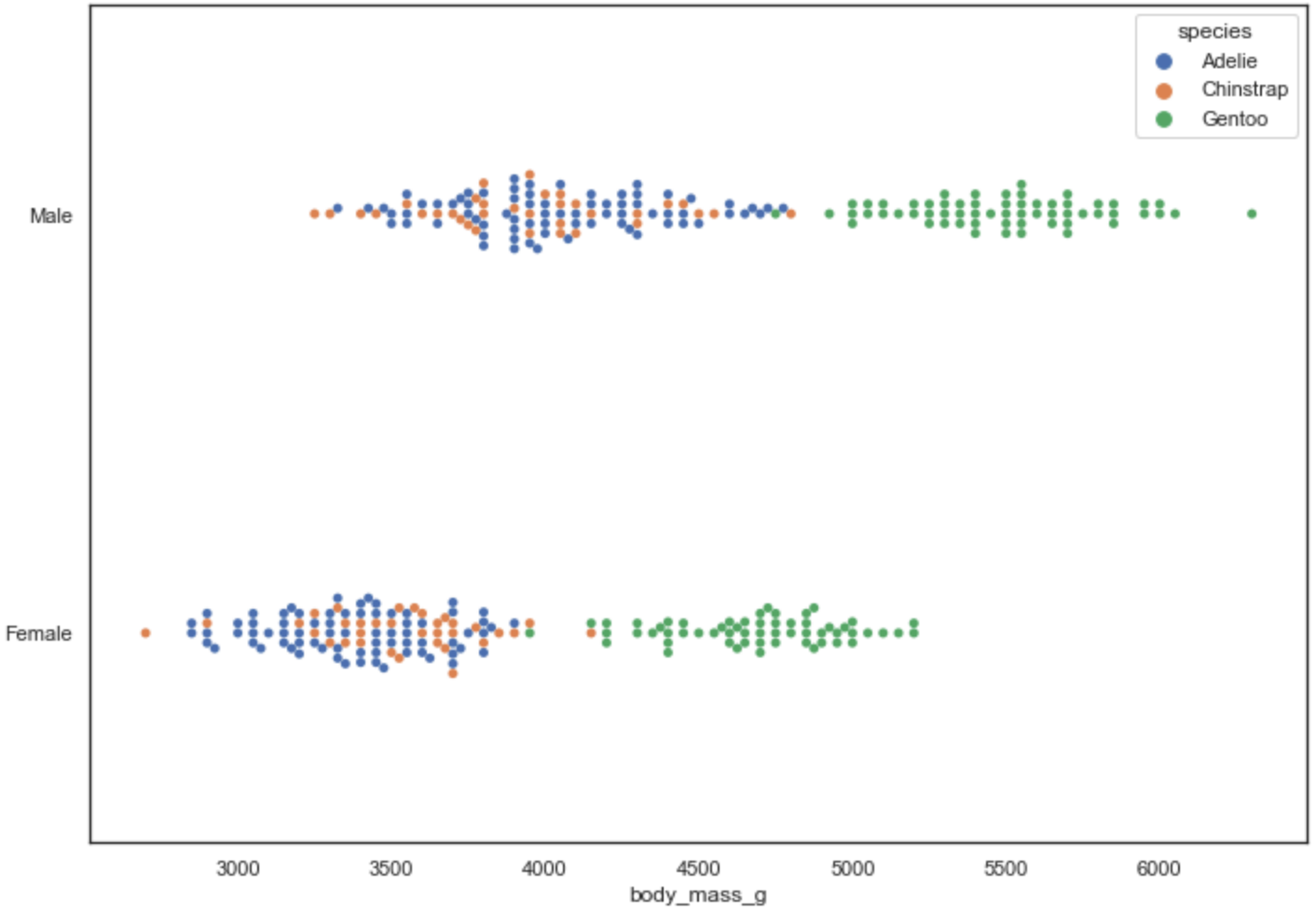


In [22]:
```
# plt.rcParams.keys()
```

Scatterplot with categorical variables

In [23]:
```
# Load the penguins dataset
df = sns.load_dataset("penguins")

# Draw a categorical scatterplot to show each observation
ax = sns.swarmplot(data=df, x="body_mass_g", y="sex", hue="species")
ax.set(ylabel="")
```

Out[23]:
```
[Text(0, 0.5, '')]
```

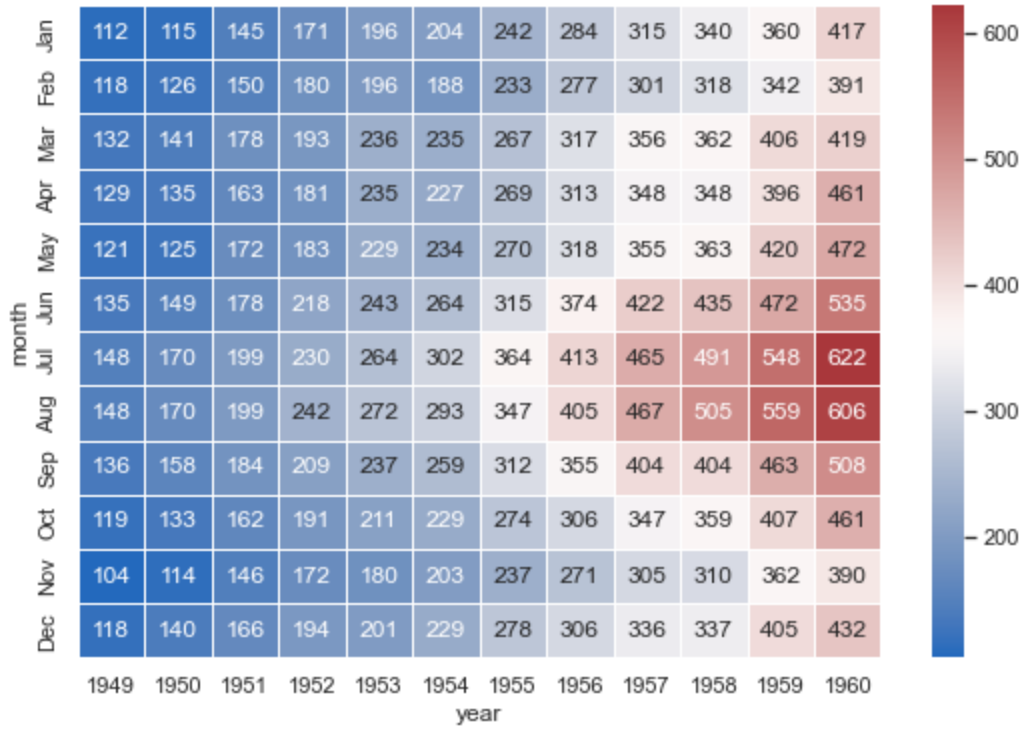


矩阵型

Annotated heatmaps

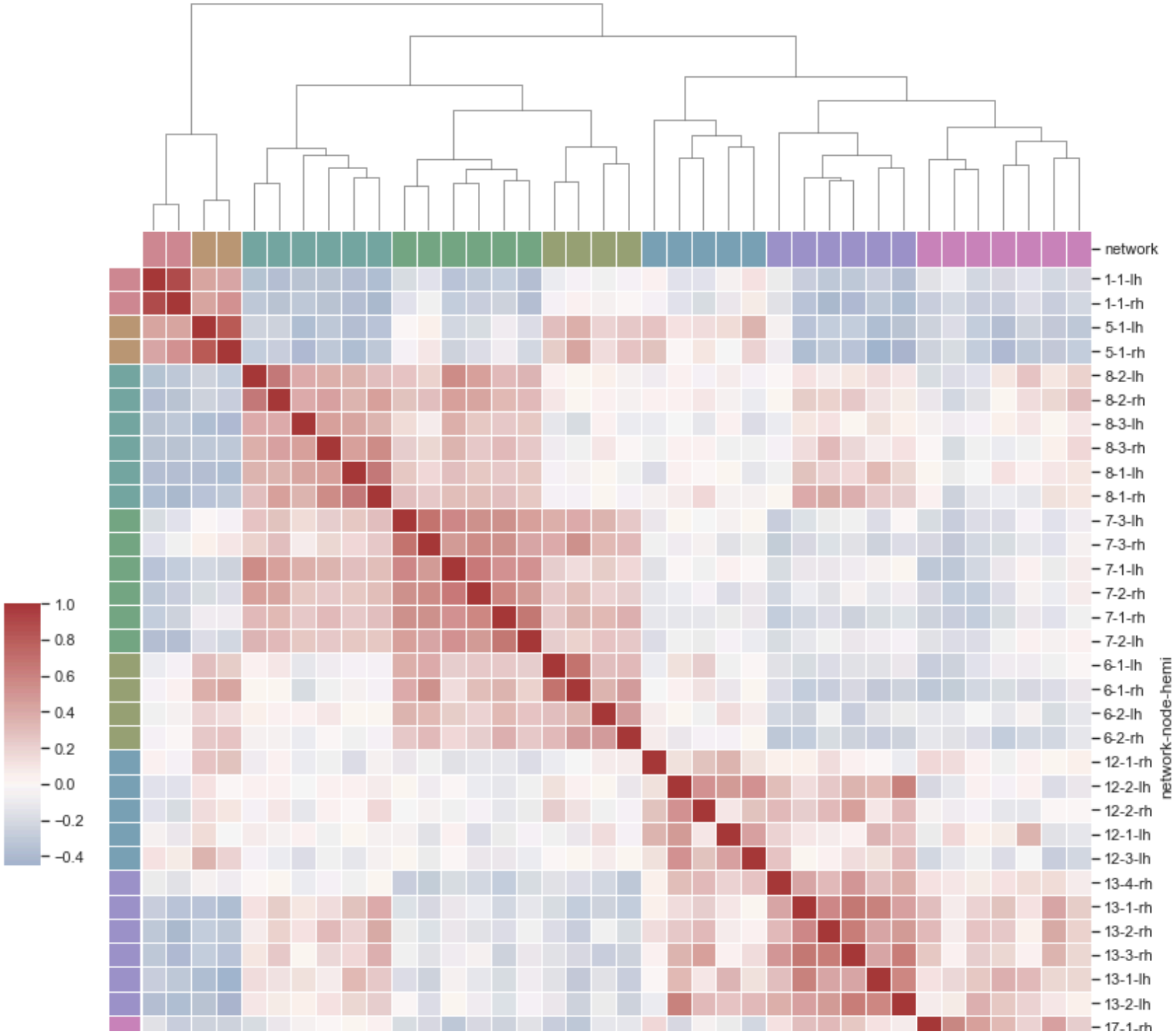
In [24]:
```
sns.set_theme()

# Load the example flights dataset and convert to long-form
flights_long = sns.load_dataset("flights")
flights = flights_long.pivot("month", "year", "passengers")

# Draw a heatmap with the numeric values in each cell
f, ax = plt.subplots(figsize=(9, 6))
sns.heatmap(flights, annot=True, fmt="d", linewidths=.5, ax=ax, cmap = 'vlag')
```

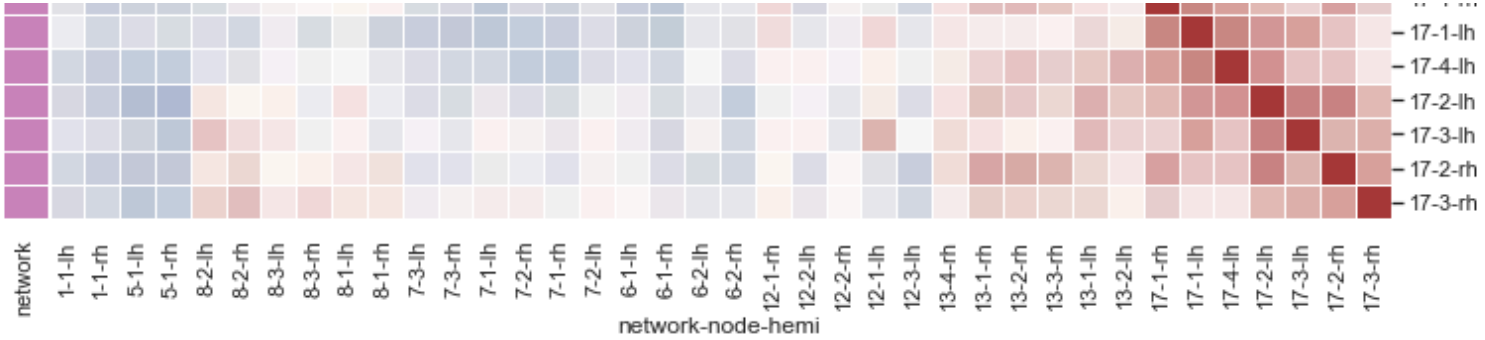
Out[24]: <AxesSubplot:xlabel='year', ylabel='month'>



Discovering structure in heatmap data

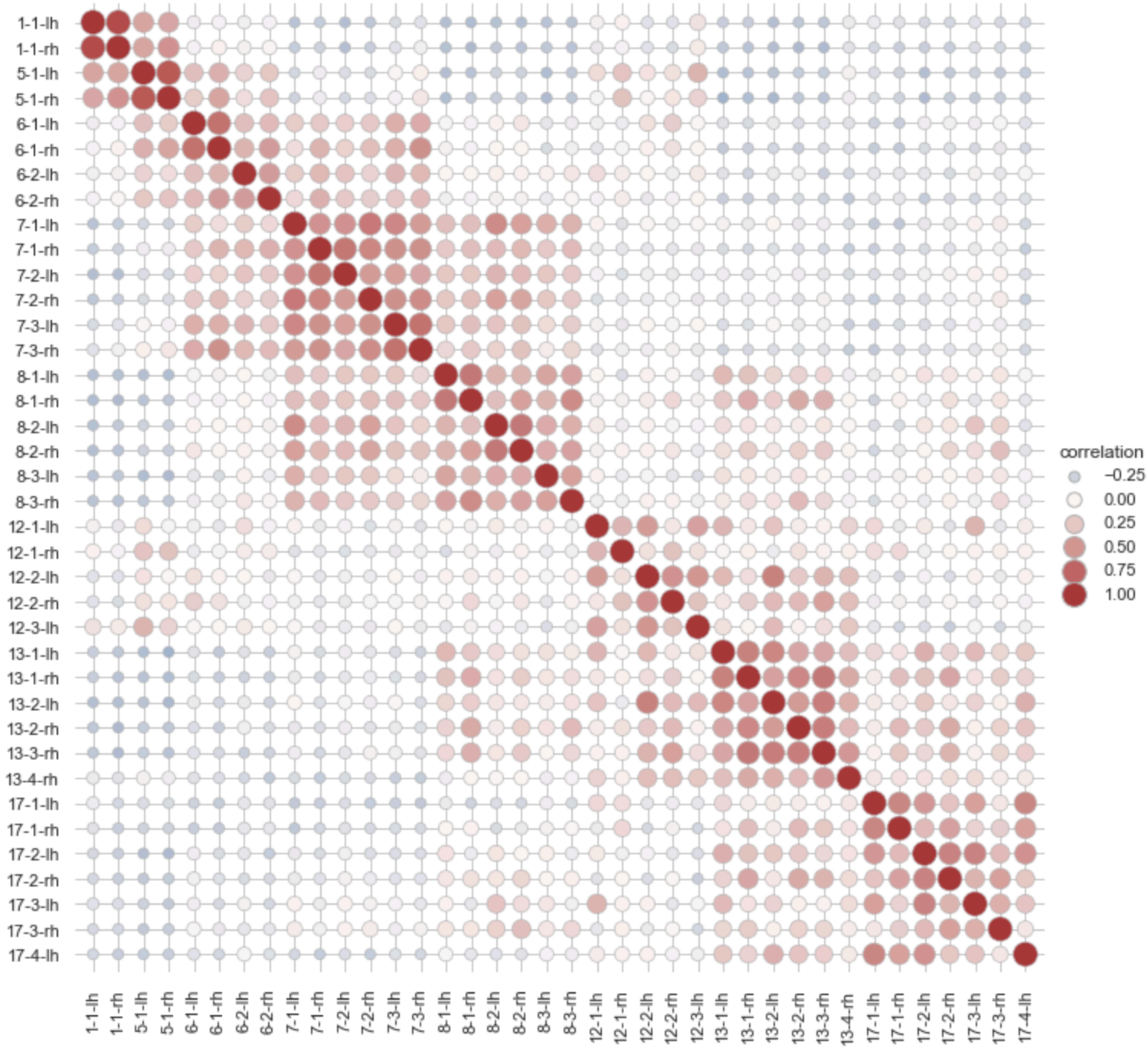
In [25]:
```
sns.set_theme()

# Load the brain networks example dataset
df = sns.load_dataset("brain_networks", header=[0, 1, 2], index_col=0)

# Select a subset of the networks
used_networks = [1, 5, 6, 7, 8, 12, 13, 17]
used_columns = (df.columns.get_level_values("network")
                          .astype(int)
                          .isin(used_networks))
df = df.loc[:, used_columns]

# Create a categorical palette to identify the networks
network_pal = sns.husl_palette(8, s=.45)
network_lut = dict(zip(map(str, used_networks), network_pal))

# Convert the palette to vectors that will be drawn on the side of the matrix
networks = df.columns.get_level_values("network")
network_colors = pd.Series(networks, index=df.columns).map(network_lut)

# Draw the full plot
g = sns.clustermap(df.corr(), center=0, cmap="vlag",
                   row_colors=network_colors, col_colors=network_colors,
                   dendrogram_ratio=(.1, .2),
                   cbar_pos=(.02, .32, .03, .2),
                   linewidths=.75, figsize=(12, 13))

g.ax_row_dendrogram.remove()
```


network
1-1-lh
1-1-rh
5-1-lh
5-1-rh
8-2-lh
8-2-rh
8-3-lh
8-3-rh
8-1-lh
8-1-rh
7-3-lh
7-3-rh
7-1-lh
7-2-rh
7-1-rh
7-2-lh
6-1-lh
6-1-rh
6-2-lh
6-2-rh
12-1-rh
12-2-lh
12-2-rh
12-1-lh
12-3-lh
13-4-rh
13-1-rh
13-2-rh
13-3-rh
13-1-lh
13-2-lh
17-1-rh
network-node-hemi
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4



Scatterplot heatmap

In [26]:
```
sns.set_theme(style="whitegrid")

# Load the brain networks dataset, select subset, and collapse the multi-index
df = sns.load_dataset("brain_networks", header=[0, 1, 2], index_col=0)

used_networks = [1, 5, 6, 7, 8, 12, 13, 17]
used_columns = (df.columns
                  .get_level_values("network")
                  .astype(int)
                  .isin(used_networks))
df = df.loc[:, used_columns]

df.columns = df.columns.map("-".join)

# Compute a correlation matrix and convert to long-form
corr_mat = df.corr().stack().reset_index(name="correlation")

# Draw each cell as a scatter point with varying size and color
g = sns.relplot(
    data=corr_mat,
    x="level_0", y="level_1", hue="correlation", size="correlation",
    palette="vlag", hue_norm=(-1, 1), edgecolor=".7",
    height=10, sizes=(50, 250), size_norm=(-.2, .8),
)

# Tweak the figure to finalize
g.set(xlabel="", ylabel="", aspect="equal")
g.despine(left=True, bottom=True)
g.ax.margins(.02)
for label in g.ax.get_xticklabels():
    label.set_rotation(90)
for artist in g.legend.legendHandles:
    artist.set_edgecolor(".7")
```


1-1-lh
1-1-rh
5-1-lh
5-1-rh
6-1-lh
6-1-rh
6-2-lh
6-2-rh
7-1-lh
7-1-rh
7-2-lh
7-2-rh
7-3-lh
7-3-rh
8-1-lh
8-1-rh
8-2-lh
8-2-rh
8-3-lh
8-3-rh
12-1-lh
12-1-rh
12-2-lh
12-2-rh
12-3-lh
13-1-lh
13-1-rh
13-2-lh
13-2-rh
13-3-rh
13-4-rh
17-1-lh
17-1-rh
17-2-lh
17-2-rh
17-3-lh
17-3-rh
17-4-lh
correlation
−0.25
0.00
0.25
0.50
0.75
1.00

Plotting a diagonal correlation matrix

In [27]:
```
from string import ascii_letters
sns.set_theme(style="white")

# Generate a large random dataset
rs = np.random.RandomState(33)
d = pd.DataFrame(data=rs.normal(size=(100, 26)),
                 columns=list(ascii_letters[26:]))

# Compute the correlation matrix
corr = d.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[27]: <AxesSubplot:>