Lineplot with multifacets

In [1]:

```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

nuqta = sns.load_dataset("dots")
nuqta.head()
```

Out[1]:

	align	choice	time	coherence	firing_rate
0	dots	T1	-80	0.0	33.189967
1	dots	T1	-80	3.2	31.691726
2	dots	T1	-80	6.4	34.279840
3	dots	T1	-80	12.8	32.631874
4	dots	T1	-80	25.6	35.060487

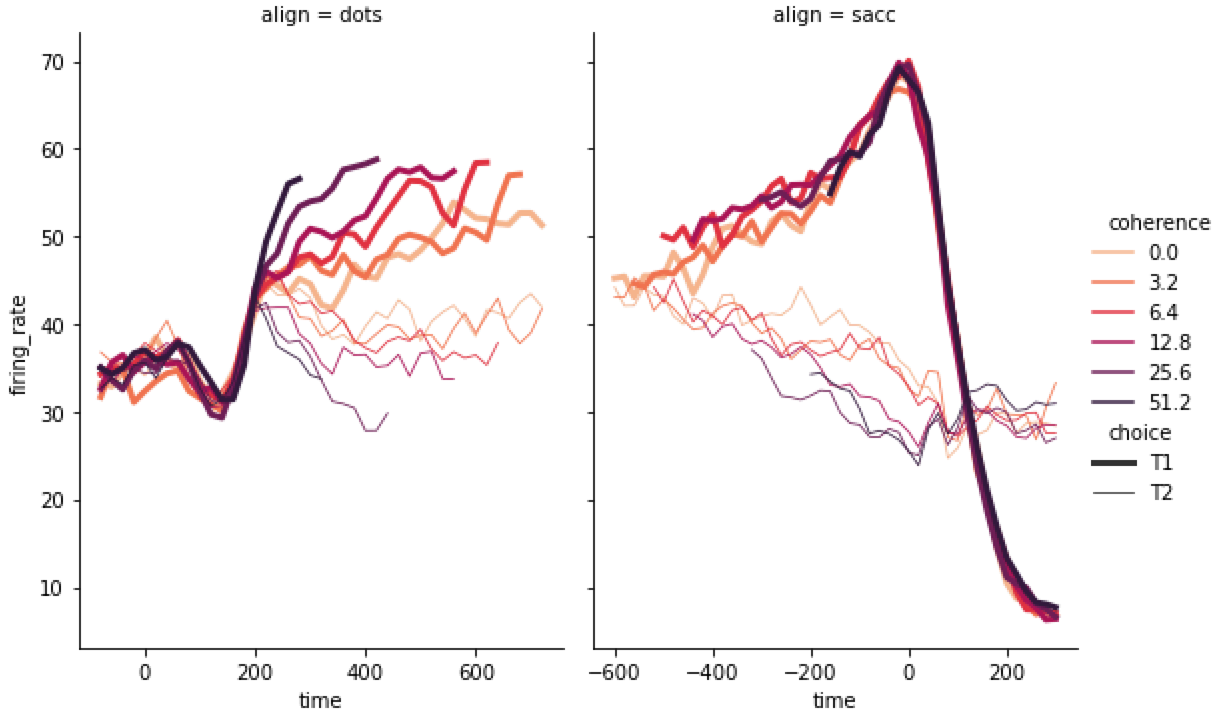
In [2]:

```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

nuqta = sns.load_dataset("dots")
# defining a color pallete
p = sns.color_palette('rocket_r')

# Plot LinePlot

sns.relplot(data=nuqta, x="time", y="firing_rate", hue="coherence",
            size="choice", col="align", kind="line", size_order=["T1","T2"],
            palette=p, height=5, aspect=.75, facet_kws=dict(sharex=False))
```

Out[2]:

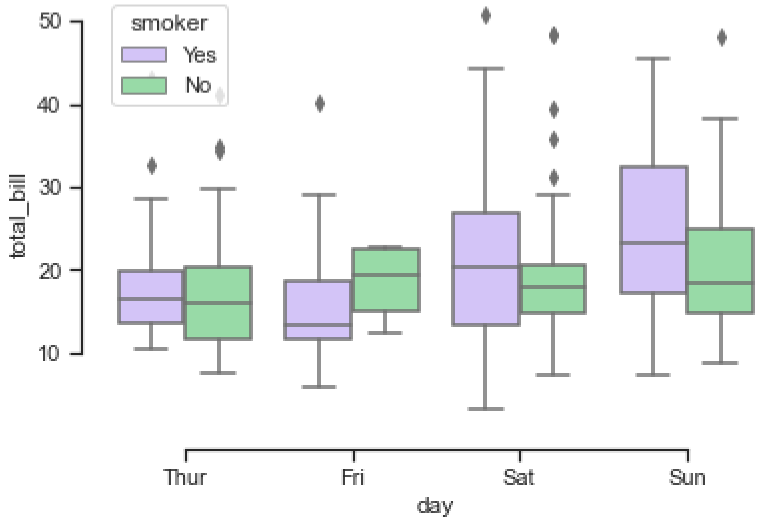
```
<seaborn.axisgrid.FacetGrid at 0x1742230b940>
```

Loading [MathJax]/extensions/Safe.js



In [3]:

```
sns.set_theme(style="ticks", palette="pastel")

# Load the exaple 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=["m","g"], data=tips)
sns.despine(offset=10, trim=True)
```

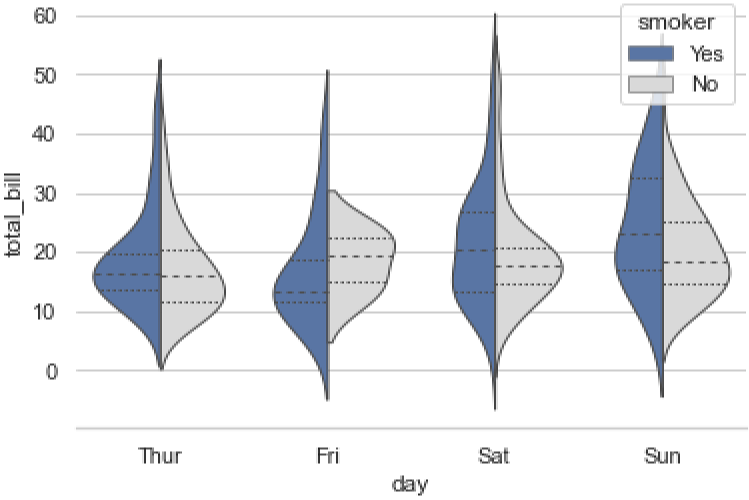


In [4]:

```
sns.set_theme(style="whitegrid")

#load dataset

tips=sns.load_dataset("tips")

# Draw a nested violinplot and split the violins for easier comparison
```

Loading [MathJax]/extensions/Safe.js

```
sns.violinplot(data=tips, x="day", y="total_bill", hue="smoker", split=True, inner="qua
               palette={"Yes": "b", "No": ".85"})
sns.despine(left=True)
```

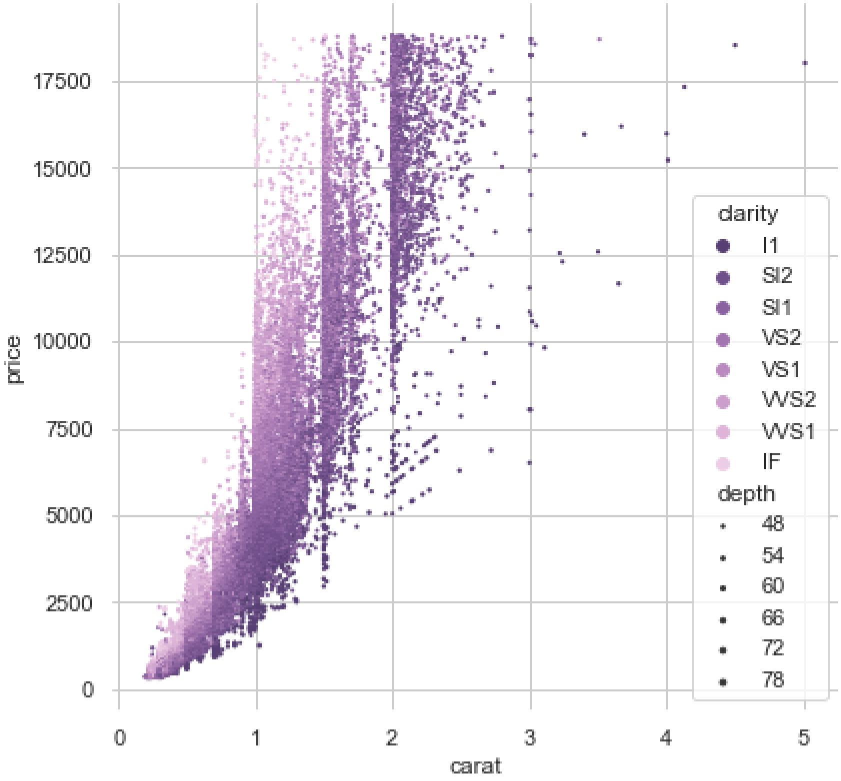


In [5]:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_theme(style="whitegrid")

# Load the dataset
diamonds = sns.load_dataset("diamonds")

# Draw a scatter plot while assigning point colors and sizes to different variables in 

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=.2, d=
                hue_order=clarity_ranking, sizes=(1,8), linewidth=0, data=diamonds, ax=a
```

Out[5]:

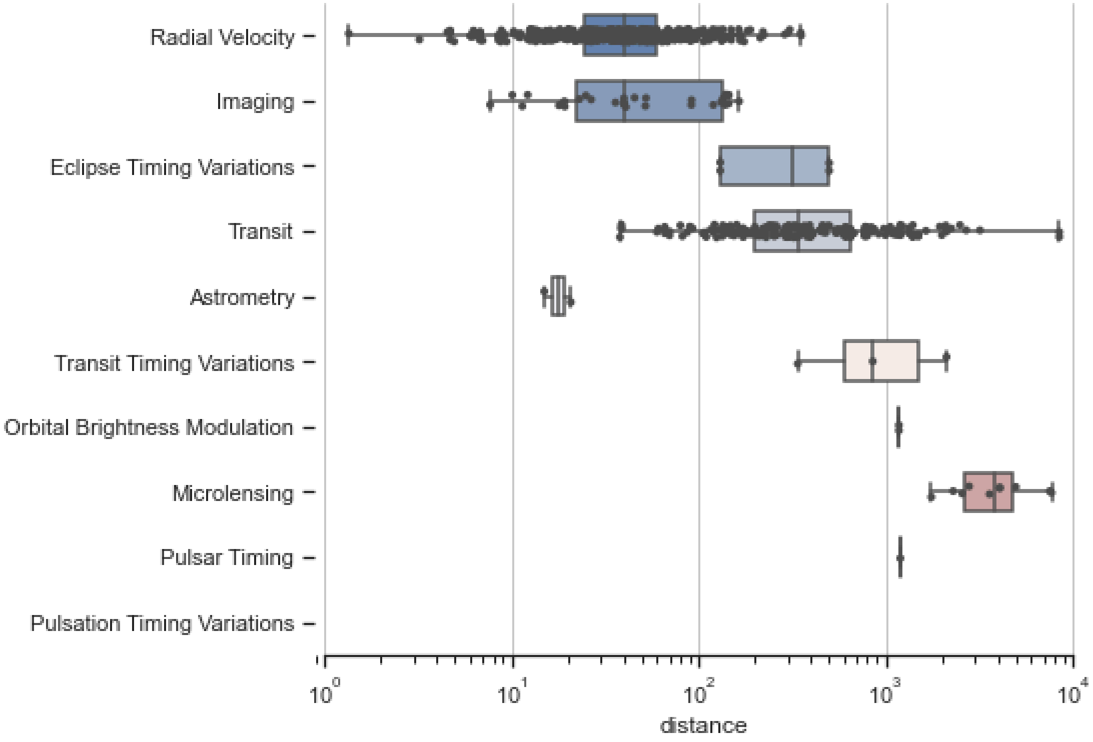
```
<AxesSubplot:xlabel='carat', ylabel='price'>
```



In [6]:

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_theme(style="ticks")

#initialize the figure with a logarithmic x axis
f, ax = plt.subplots(figsize=(7, 6))
ax.set_xscale("log")

#load dataset
planets = sns.load_dataset("planets")

#Plot the orbital period with horizontal axes
sns.boxplot(x="distance", y="method", data=planets,
            whis=[0, 100], width=.6, palette="vlag")

#Add inpoints to show each observation, It has only poitns/dots in a plot
sns.stripplot(x="distance", y="method", data=planets,
              size=4, color =".3", linewidth=0)

#Tweak the visual presentation

ax.xaxis.grid(True)
ax.set(ylabel="")
sns.despine(trim=True, left=True)
```

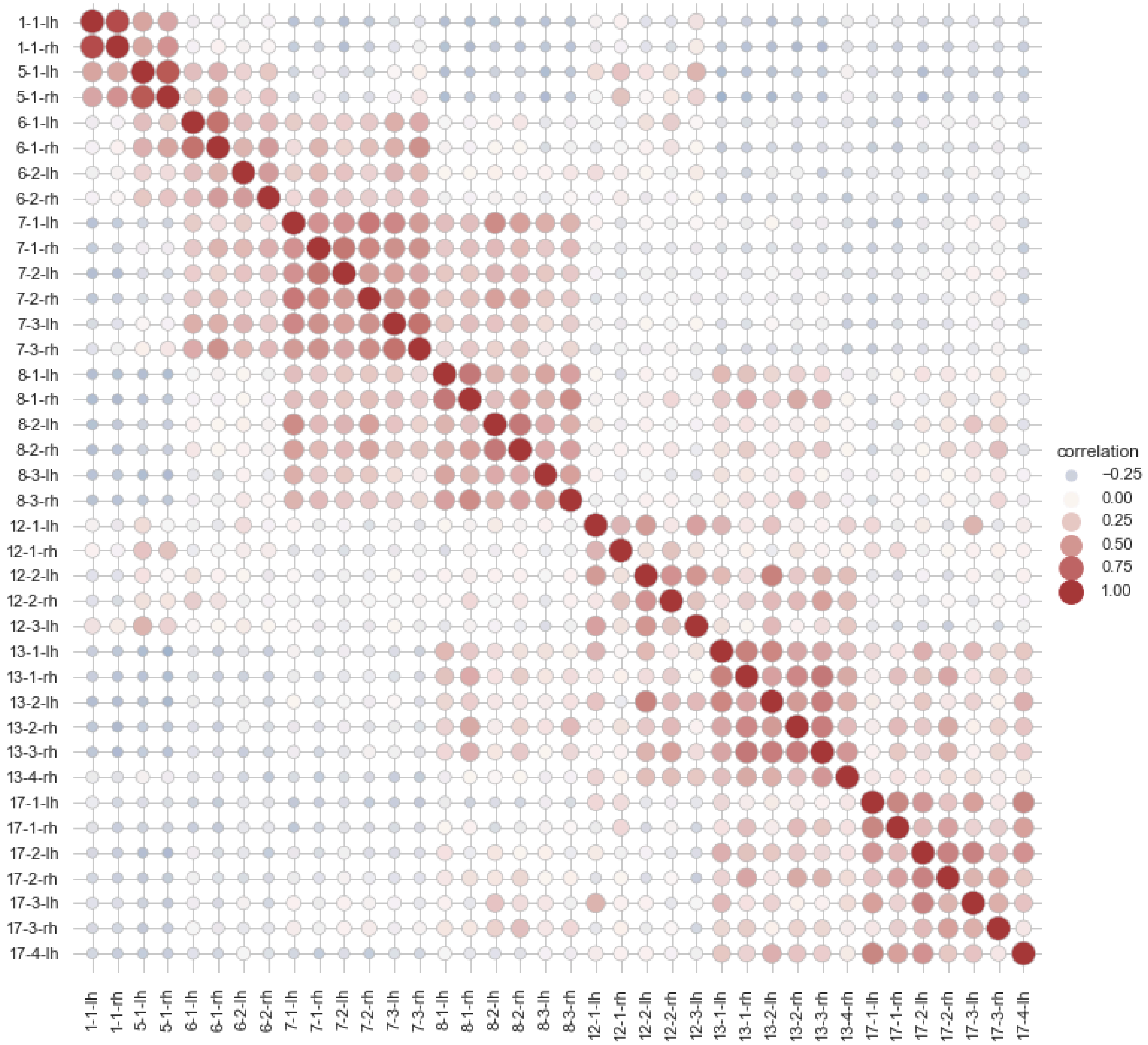
Loading [MathJax]/extensions/Safe.js



In [7]:

```python
import seaborn as sns
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)
```

Loading [MathJax]/extensions/Safe.js



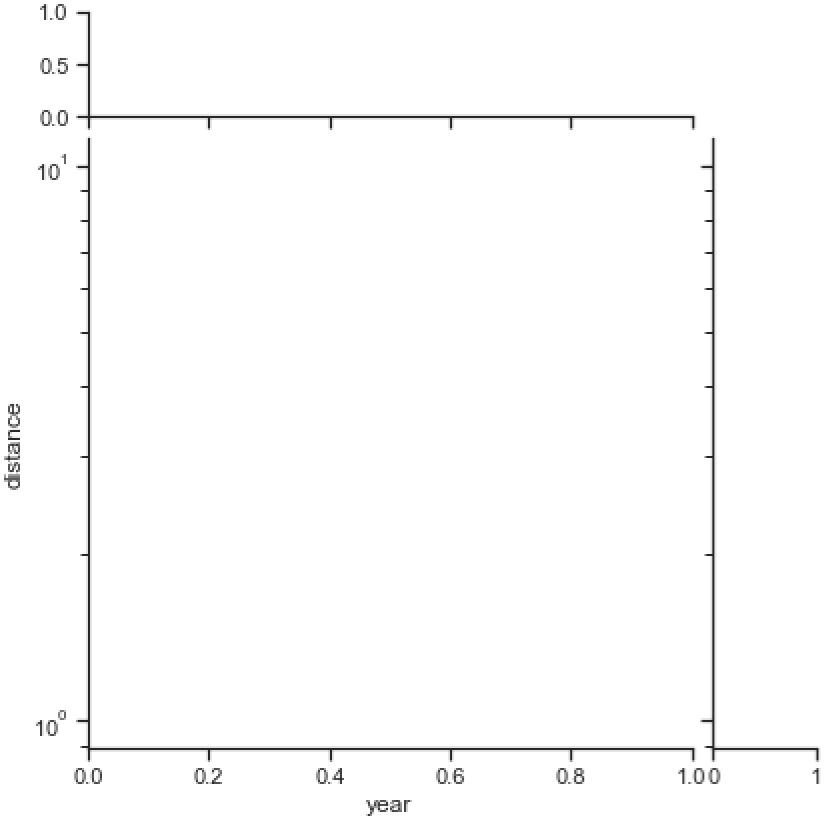
In [8]:

```
import seaborn as sns
import numpy as pd
import matplotlib.pyplot as plt
import pandas as pd

sns.set_theme(style="ticks")

#Load the planets dataset and initialize the figure
planets = sns.load_dataset("planets")
g = sns.JointGrid(data=planets, x="year", y="distance", marginal_ticks=True)

#Set a log scaling on the y axis
g.ax_joint.set(yscale="log")

#Create an insert legend for the histogram colorbar
cax = g.figure.add_axes([.15, .55, .02, .2])

#Add the joint and marginal histogram plots
g.plot_joint(
    sns.histplot, discrete=(True, False),
    cmap="light:#03012d", pmax=.8, cbar=True, cbar_ax=cax
)
```

Loading [MathJax]/extensions/Safe.js

```
g.plot_marginals(sns.histplot, element="step", color="#03012d")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-8-a0c89846ad02> in <module>
     14 
     15 #Create an insert legend for the histogram colorbar
---> 16 cax = g.figure.add_axes([.15, .55, .02, .2])
     17 
     18 #Add the joint and marginal histogram plots

AttributeError: 'JointGrid' object has no attribute 'figure'
```



In [9]:

```
import seaborn as sns
sns.set_theme(style="whitegrid")

diamonds = sns.load_dataset("diamonds")
diamonds.head()
```

Out[9]:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [10]:

```
import seaborn as sns
tyle="whitegrid")
```

Loading [MathJax]/extensions/Safe.js

```
diamonds = sns.load_dataset("diamonds")
clarity_ranking = ["I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF"]

sns.boxenplot(
    diamonds, x="clarity", y="carat",
    color="b", order=clarity_ranking, width_method="linear",
)
```

```
C:\Users\A.S.Pride\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-10-b3ae41adb976> in <module>
      5 clarity_ranking = ["I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF"]
      6 
----> 7 sns.boxenplot(
      8     diamonds, x="clarity", y="carat",
      9     color="b", order=clarity_ranking, width_method="linear",

~\anaconda3\lib\site-packages\seaborn\_decorators.py in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48 

~\anaconda3\lib\site-packages\seaborn\categorical.py in boxenplot(x, y, hue, data, orde
r, hue_order, orient, color, palette, saturation, width, dodge, k_depth, linewidth, scal
e, outlier_prop, trust_alpha, showfliers, ax, **kwargs)
   2619 ):
   2620 
-> 2621     plotter = _LVPlotter(x, y, hue, data, order, hue_order,
   2622                          orient, color, palette, saturation,
   2623                          width, dodge, k_depth, linewidth, scale,

~\anaconda3\lib\site-packages\seaborn\categorical.py in __init__(self, x, y, hue, data,
order, hue_order, orient, color, palette, saturation, width, dodge, k_depth, linewidth,
scale, outlier_prop, trust_alpha, showfliers)
   1837         self.showfliers = showfliers
   1838 
-> 1839         self.establish_variables(x, y, hue, data, orient, order, hue_order)
   1840         self.establish_colors(color, palette, saturation)
   1841 

~\anaconda3\lib\site-packages\seaborn\categorical.py in establish_variables(self, x, y,
hue, data, orient, order, hue_order, units)
    151                 if isinstance(var, str):
    152                     err = "Could not interpret input '{}'".format(var)
--> 153                     raise ValueError(err)
    154 
    155             # Figure out the plotting orientation

ValueError: Could not interpret input 'carat'
```

In [14]:

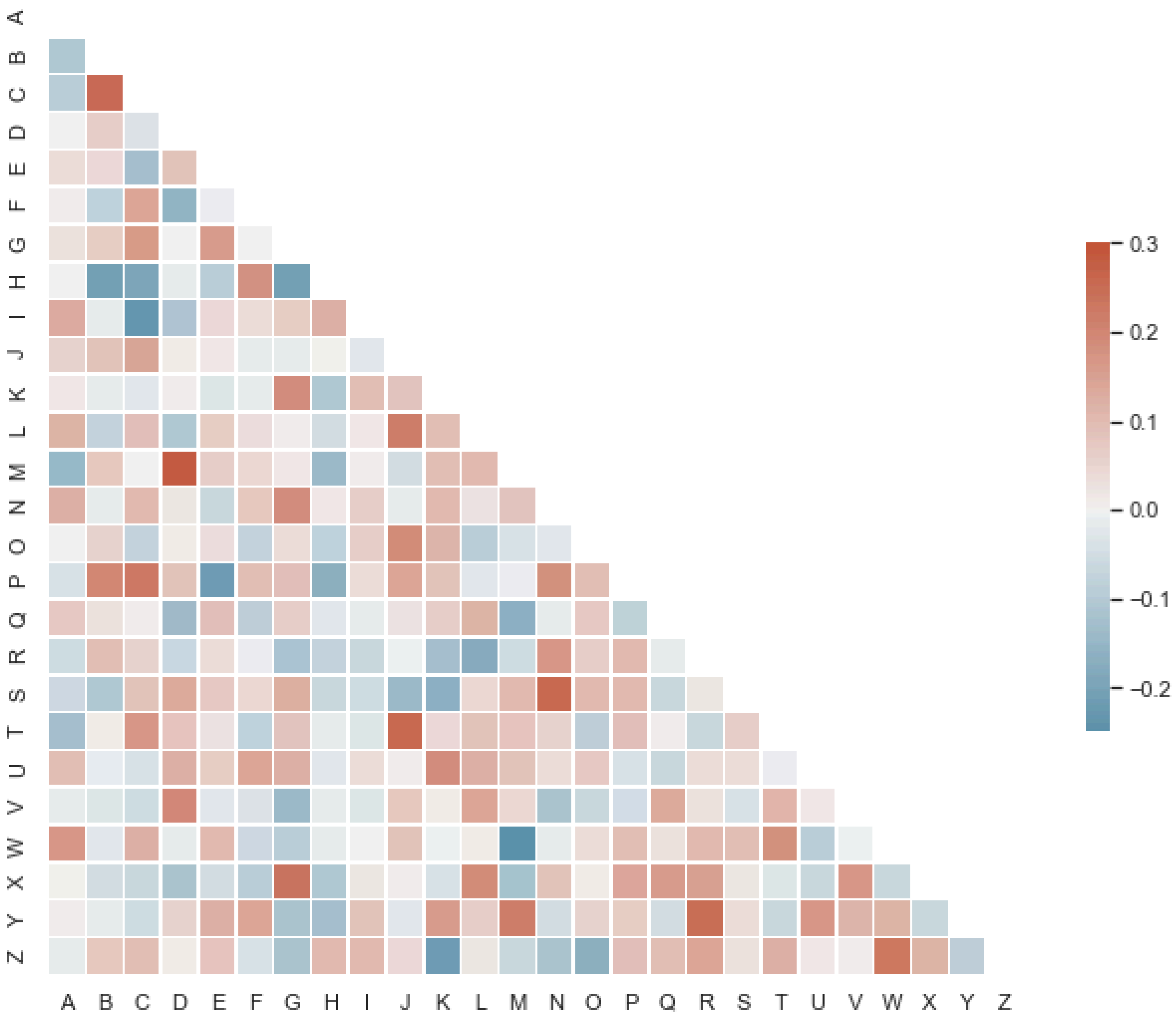
```
from string import ascii_letters
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading [MathJax]/extensions/Safe.js

```
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[14]:

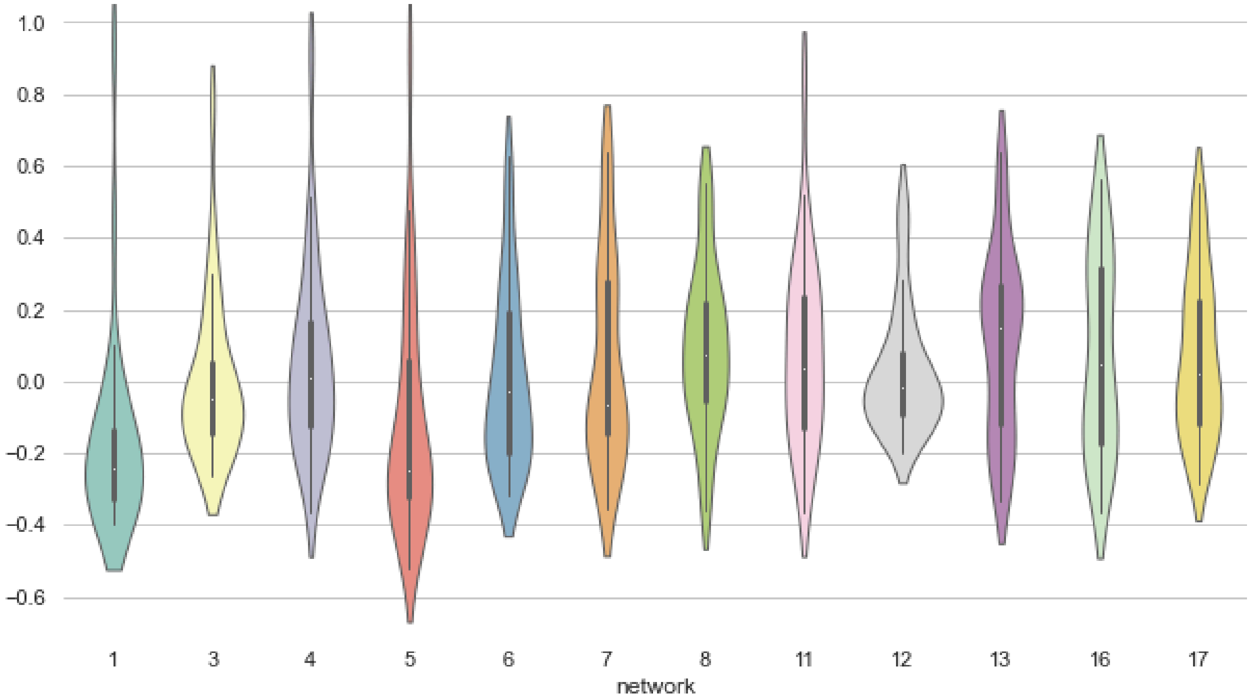
```
<AxesSubplot:>
```



Loading [MathJax]/extensions/Safe.js

```
as sns
import matplotlib.pyplot as plt
```

```
sns.set_theme(style="whitegrid")

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

# Pull out a specific subset of networks
used_networks = [1, 3, 4, 5, 6, 7, 8, 11, 12, 13, 16, 17]
used_columns = (df.columns.get_level_values("network")
                          .astype(int)
                          .isin(used_networks))
df = df.loc[:, used_columns]

# Compute the correlation matrix and average over networks
corr_df = df.corr().groupby(level="network").mean()
corr_df.index = corr_df.index.astype(int)
corr_df = corr_df.sort_index().T

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 6))

# Draw a violinplot with a narrower bandwidth than the default
sns.violinplot(data=corr_df, bw_adjust=.5, cut=1, linewidth=1, palette="Set3")

# Finalize the figure
ax.set(ylim=(-.7, 1.05))
sns.despine(left=True, bottom=True)
```



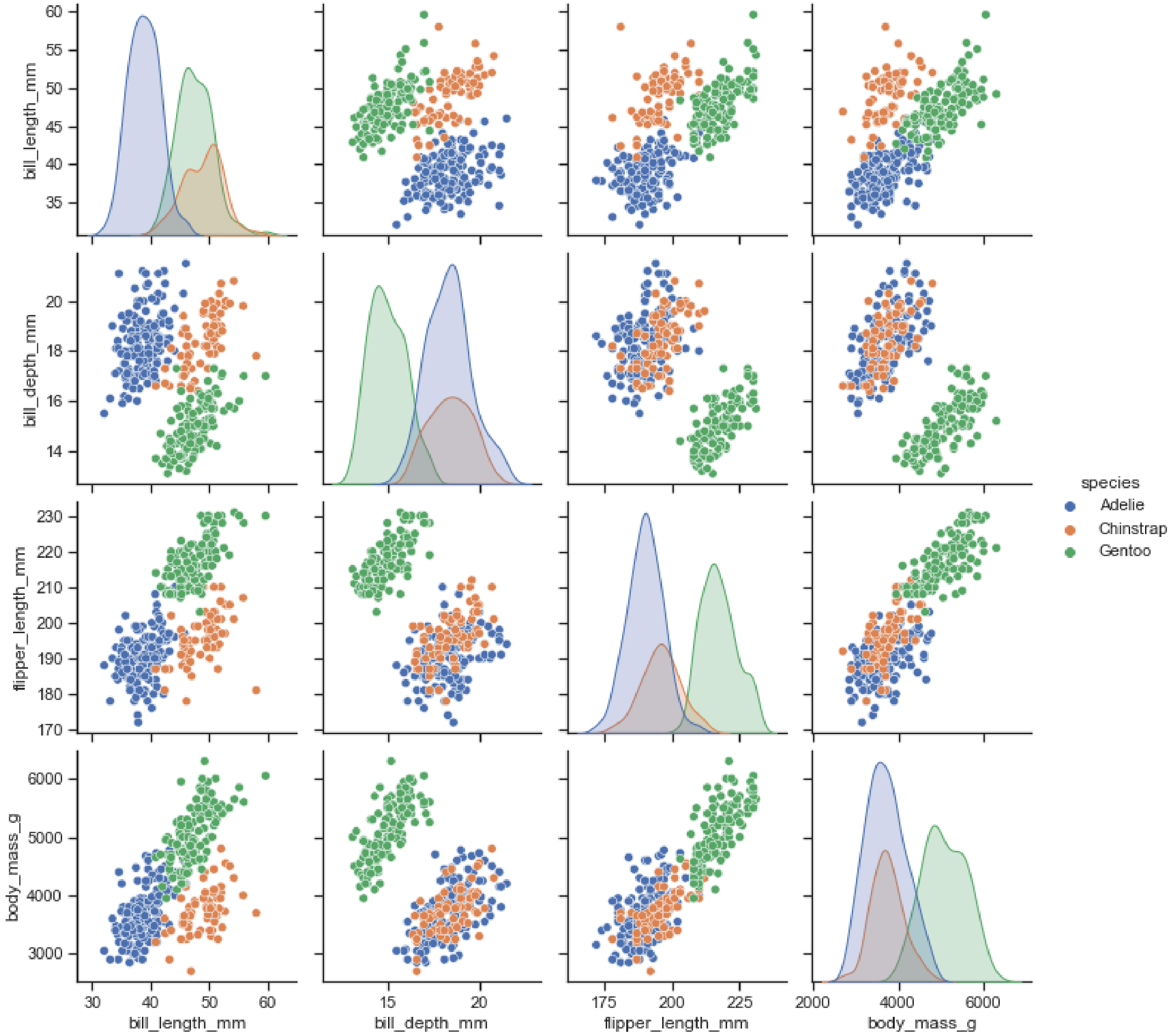
In [16]:

```
import seaborn as sns
sns.set_theme(style="ticks")

df = sns.load_dataset("penguins")
sns.pairplot(df, hue="species")
```

Out[16]:

```
<seaborn.axisgrid.PairGrid at 0x174268616a0>
```

Loading [MathJax]/extensions/Safe.js



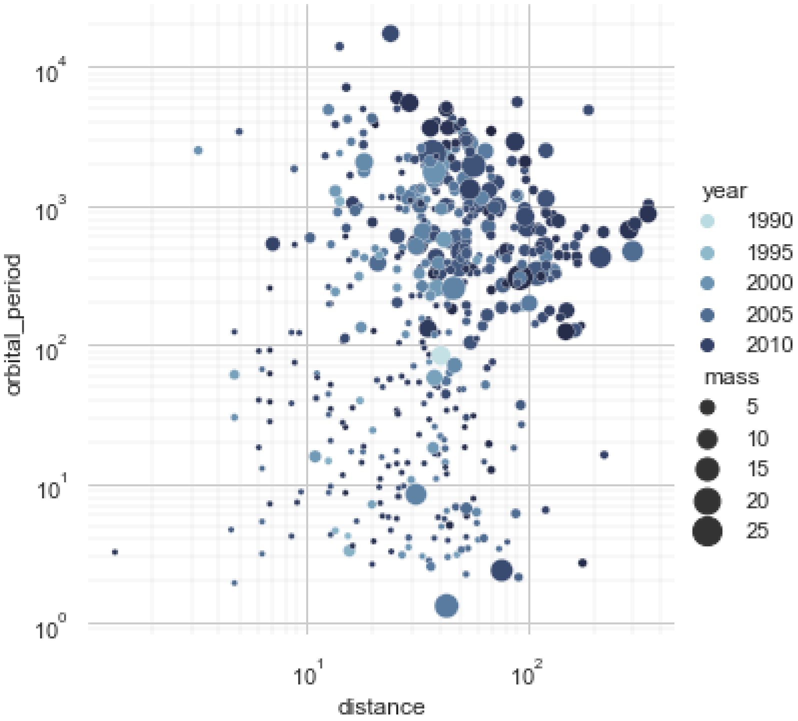
In [17]:

```
import seaborn as sns
sns.set_theme(style="whitegrid")

# Load the example planets dataset
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[17]:

```
<seaborn.axisgrid.FacetGrid at 0x17424d64cd0>
```

Loading [MathJax]/extensions/Safe.js

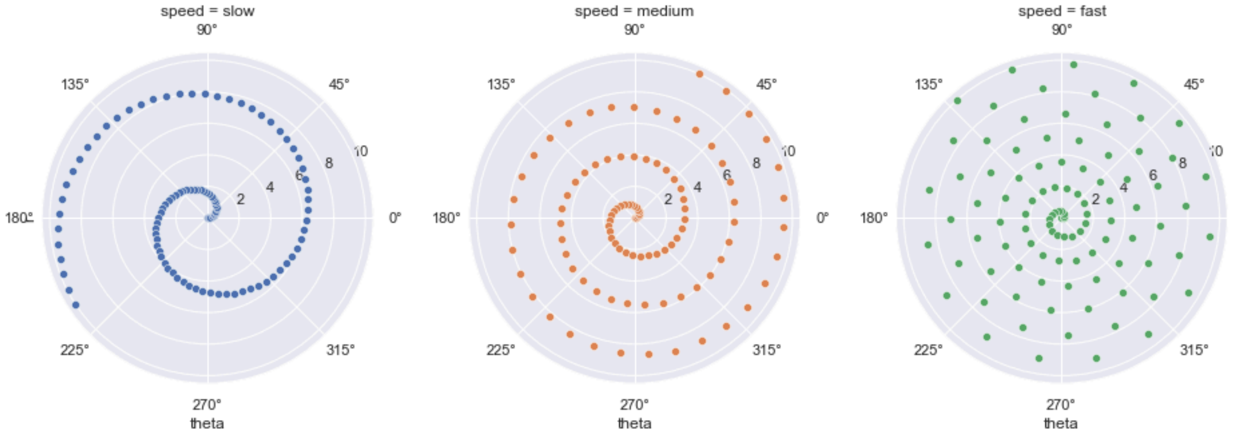


In [18]:

```
import numpy as np
import pandas as pd
import seaborn as sns

sns.set_theme()

# Generate an example radial datast
r = np.linspace(0, 10, num=100)
df = pd.DataFrame({'r': r, 'slow': r, 'medium': 2 * r, 'fast': 4 * r})

# Convert the dataframe to long-form or "tidy" format
df = pd.melt(df, id_vars=['r'], var_name='speed', value_name='theta')

# Set up a grid of axes with a polar projection
g = sns.FacetGrid(df, col="speed", hue="speed",
                  subplot_kws=dict(projection='polar'), height=4.5,
                  sharex=False, sharey=False, despine=False)

# Draw a scatterplot onto each axes in the grid
g.map(sns.scatterplot, "theta", "r")
```

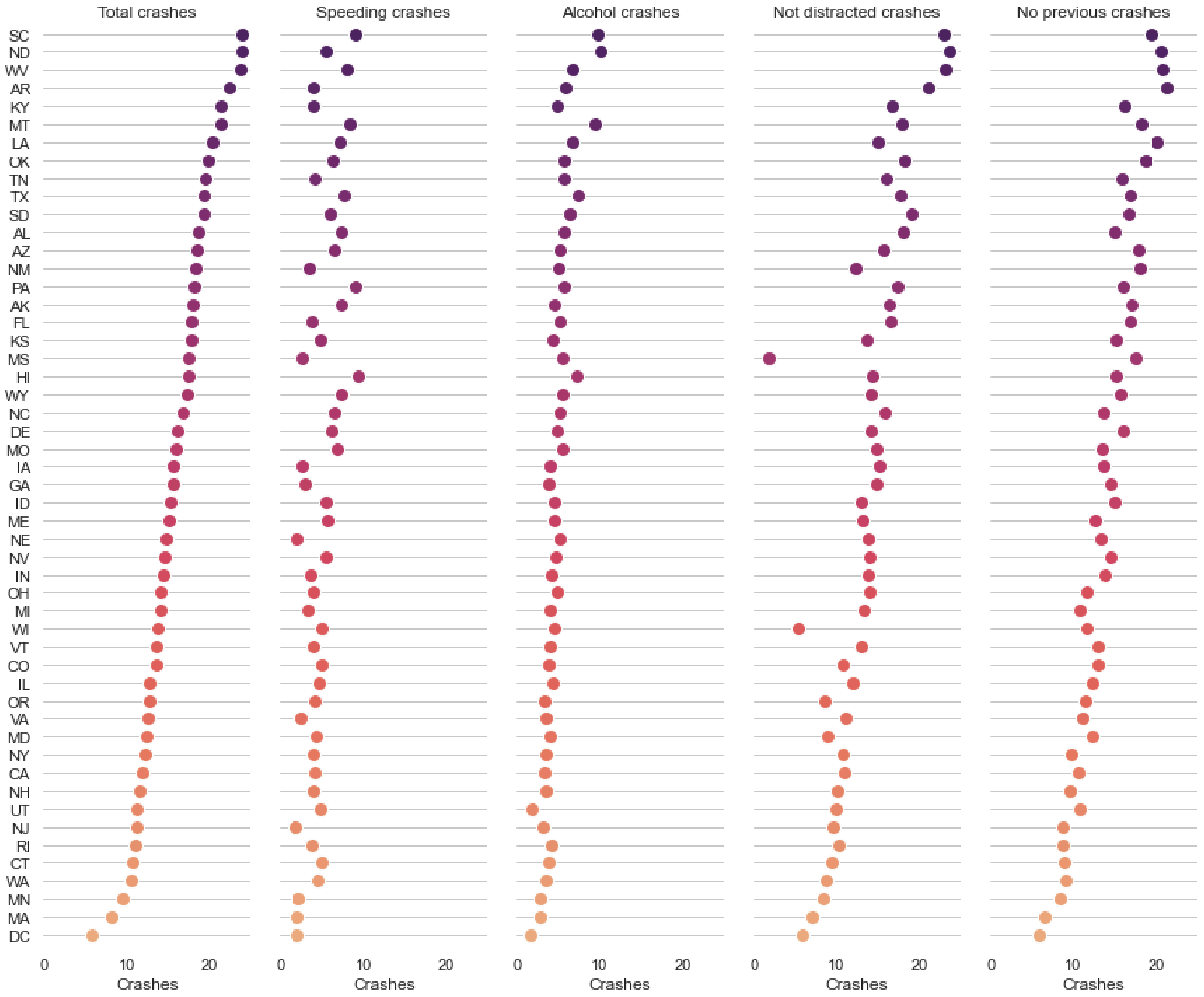
Loading [MathJax]/extensions/Safe.js

```
Out[18]: <seaborn.axisgrid.FacetGrid at 0x17424c09700>
```



```
In [19]: import seaborn as sns
sns.set_theme(style="whitegrid")

# Load the dataset
crashes = sns.load_dataset("car_crashes")

# Make the PairGrid
g = sns.PairGrid(crashes.sort_values("total", ascending=False),
                 x_vars=crashes.columns[:-3], y_vars=["abbrev"],
                 height=10, aspect=.25)

# Draw a dot plot using the stripplot function
g.map(sns.stripplot, size=10, orient="h", jitter=False,
      palette="flare_r", linewidth=1, edgecolor="w")

# Use the same x axis limits on all columns and add better labels
g.set(xlim=(0, 25), xlabel="Crashes", ylabel="")

# Use semantically meaningful titles for the columns
titles = ["Total crashes", "Speeding crashes", "Alcohol crashes",
          "Not distracted crashes", "No previous crashes"]

for ax, title in zip(g.axes.flat, titles):

    # Set a different title for each axes
    ax.set(title=title)

    # Make the grid horizontal instead of vertical
    ax.xaxis.grid(False)
    ax.yaxis.grid(True)

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

Loading [MathJax]/extensions/Safe.js



In [21]:
```
import seaborn as sns
import numpy as pd
import matplotlib.pyplot as plt
import pandas as pd

nuqta = sns.load_dataset("dots")
nuqta.head()
```

Out[21]:

	align	choice	time	coherence	firing_rate
0	dots	T1	-80	0.0	33.189967
1	dots	T1	-80	3.2	31.691726
2	dots	T1	-80	6.4	34.279840
3	dots	T1	-80	12.8	32.631874
4	dots	T1	-80	25.6	35.060487

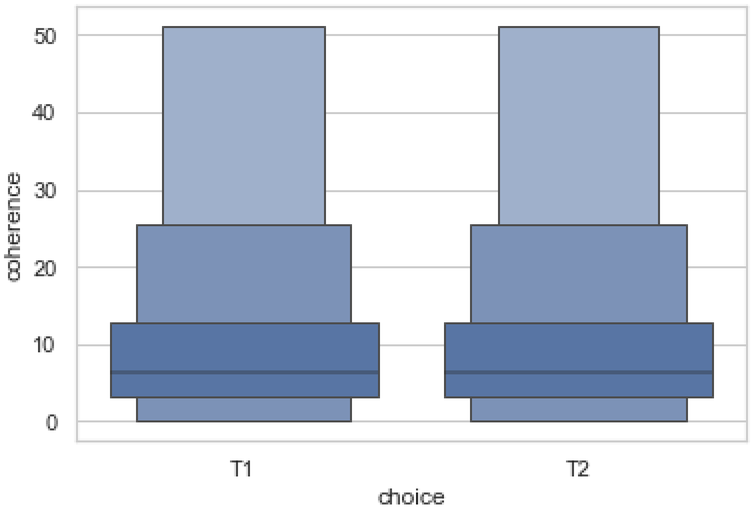
In [31]:
```
import seaborn as sns
import numpy as pd
import matplotlib.pyplot as plt
import pandas as pd

```

Loading [MathJax]/extensions/Safe.js

```
d_dataset("dots")
```

```
sns.set_theme(style="whitegrid")

sns.boxenplot(x="choice", y="coherence", color="b",
              scale="linear", data =nuqta)
```

```
Out[31]: <AxesSubplot:xlabel='choice', ylabel='coherence'>
```


coherence
T1
T2
choice