4 - Seaborn

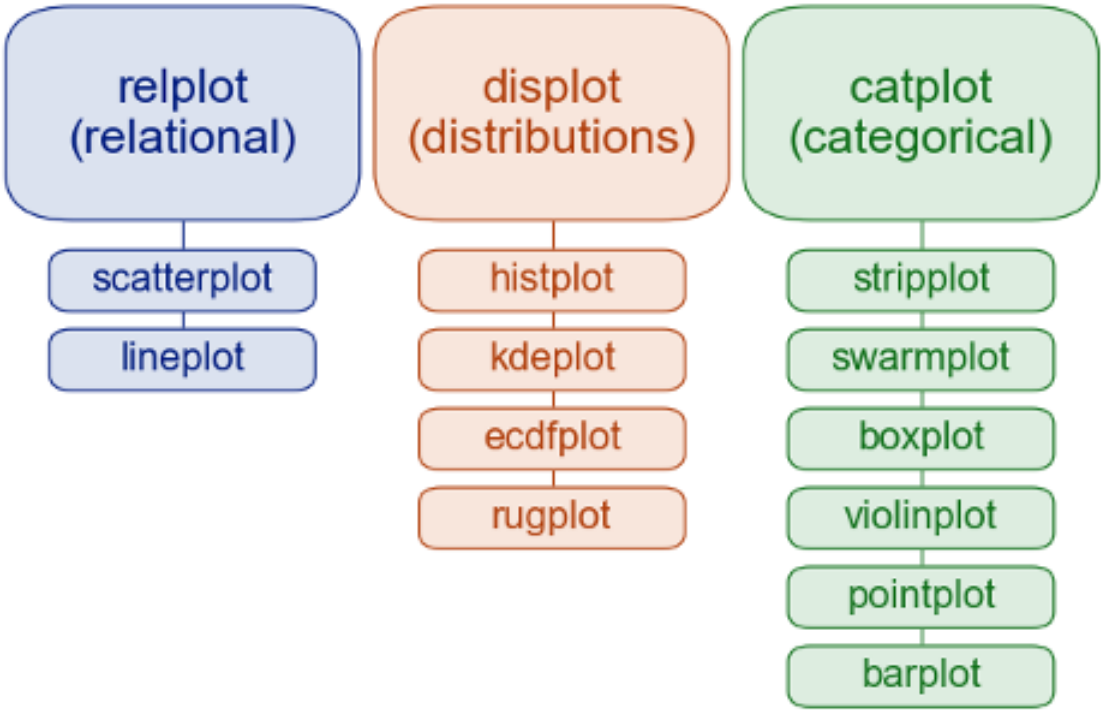
October 14, 2021

Table of Contents: Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

0.1 Figure & Axes Level Plotting Functions



0.2 Figure level Functions

- relplot
- displot - default behavior is histplot
- catplot

0.3 Axes level Functions

- scatterplot
- lineplot
- histplot
- etc

[10]:
```
import numpy as py
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

#import os
#for dirname, _, filenames in os.walk('/kaggle/input'):
#    for filename in filenames:
#        print(os.path.join(dirname, filename))
```

```
sns.set_theme()

#df = pd.read_csv('/kaggle/input/seaborn-practice/diamonds.csv')
#tips = pd.read_csv('/kaggle/input/seaborn-practice/tips.csv')
#penguins = pd.read_csv('/kaggle/input/seaborn-practice/penguins.csv')
#flights = pd.read_csv('/kaggle/input/seaborn-practice/flights.csv')

df = sns.load_dataset("diamonds")
tips = sns.load_dataset("tips")
penguins = sns.load_dataset("penguins")
flights = sns.load_dataset("flights")
```

[11]:
```
# sns.get_dataset_names()
```

[12]:
```
df.head()
```

[12]:

	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

[13]:
```
tips.head()
```

[13]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

[14]:
```
penguins.head()
```

[14]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0		

```
4       3450.0  Female
```

[15]:
```
flights.head()
```

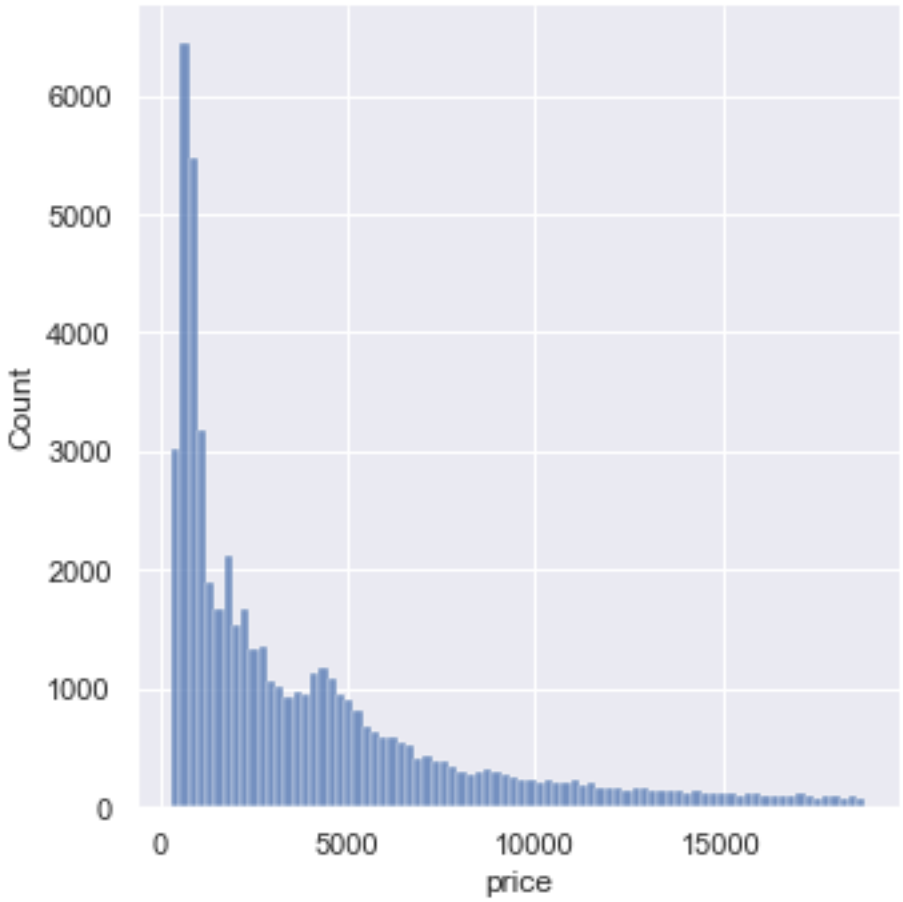
[15]:
```
   year month  passengers
0  1949   Jan         112
1  1949   Feb         118
2  1949   Mar         132
3  1949   Apr         129
4  1949   May         121
```

0.4 http://seaborn.pydata.org/tutorial.html

0.5 Figure level

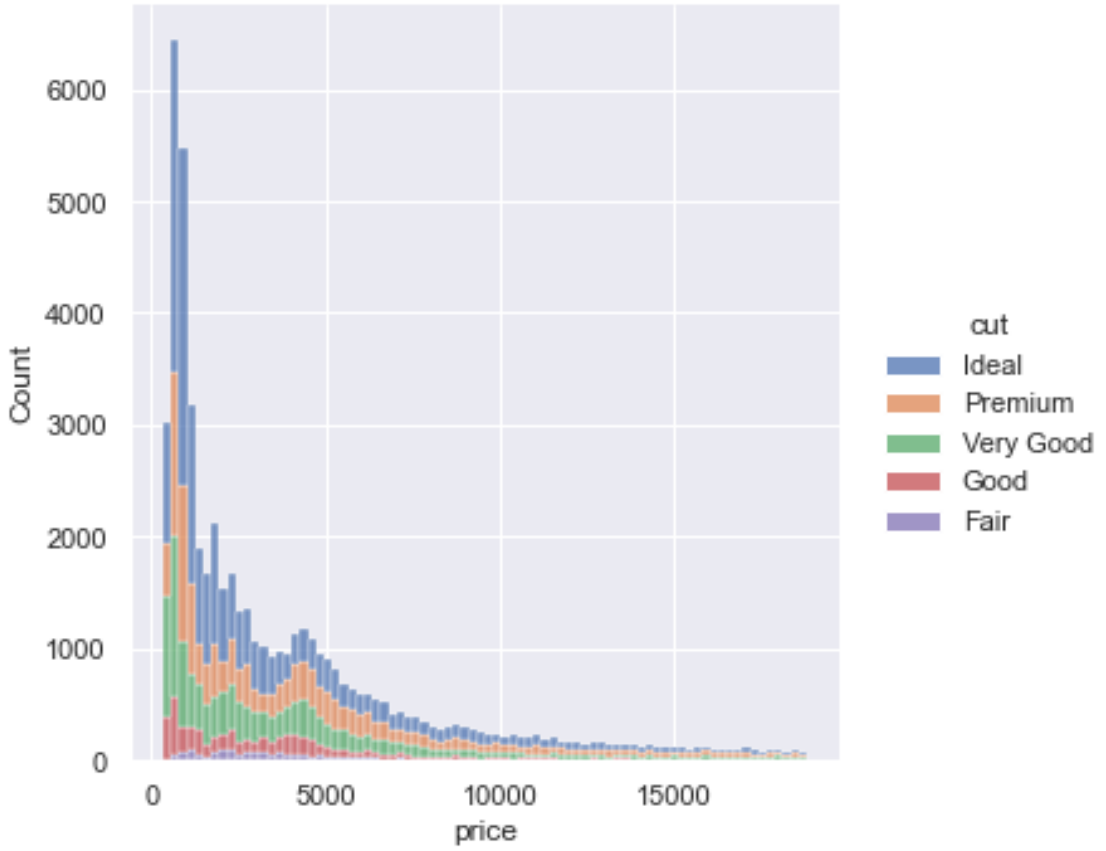
- Figure-level functions interface with matplotlib through a seaborn object, usually a FacetGrid
- Each module (relational, distributions, categorical) has a single figure-level function

[16]:
```
# The default for distplot is a histogram
sns.displot(data=df, x="price")
# plt.show() # removes the 'output' text
plt.savefig('save_as_a_png.png')
```



[17]:
```
sns.displot(data=df, x="price", hue="cut", multiple="stack")
```

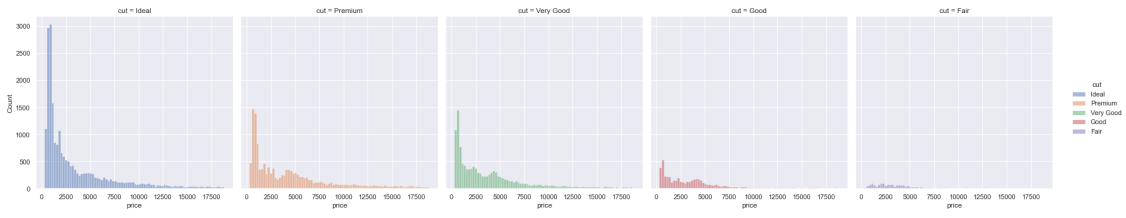
[17]: <seaborn.axisgrid.FacetGrid at 0x7ff9469ccbb0>



0.6 Using the 'kind' kwarg

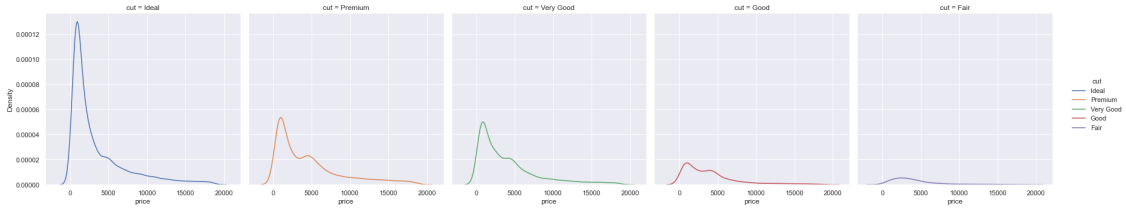
[18]:
```
sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'hist')
```

[18]: <seaborn.axisgrid.FacetGrid at 0x7ff94ecb22b0>



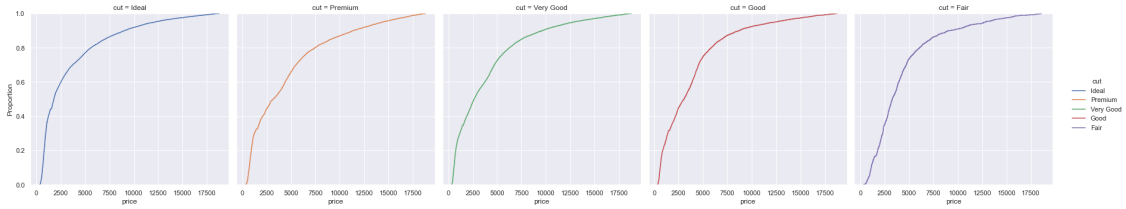
[19]:
```
sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'kde')

# kernel density estimation
```

[19]: <seaborn.axisgrid.FacetGrid at 0x7ff94e04e250>



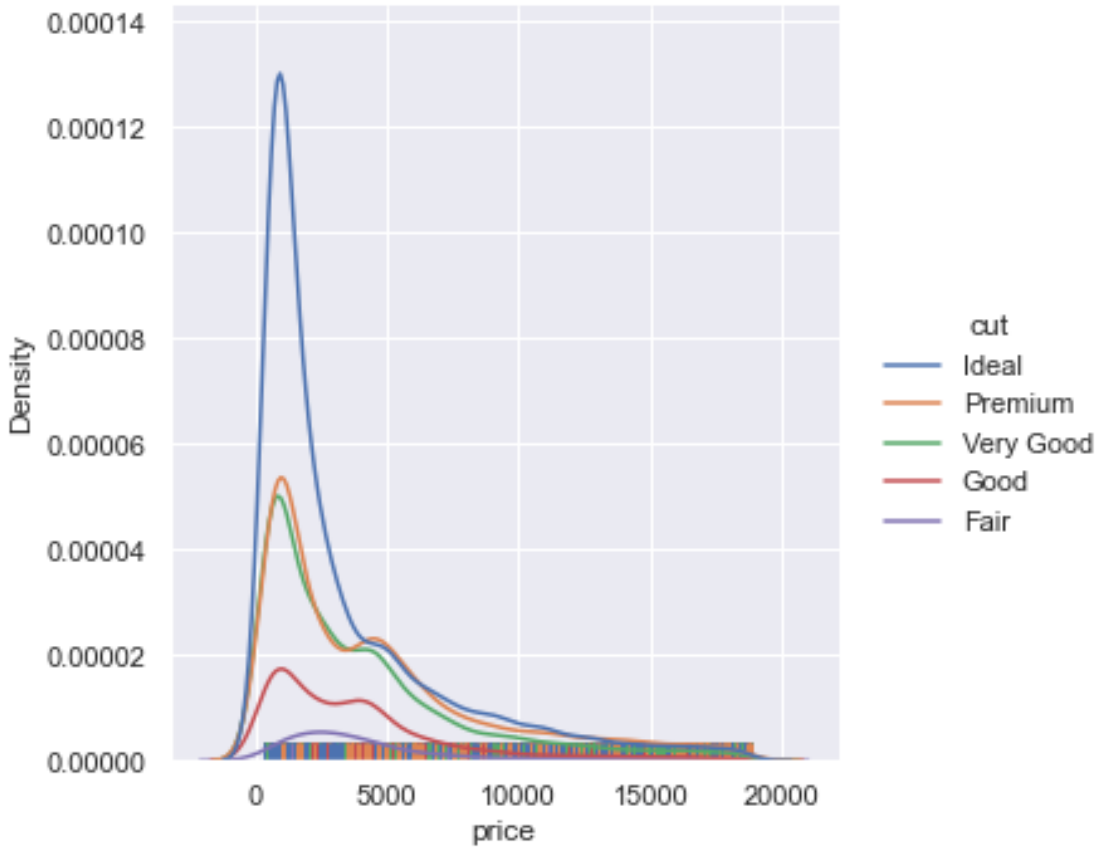
[20]:
```
sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'ecdf')

# empirical cumulative distribution functions
```

[20]: <seaborn.axisgrid.FacetGrid at 0x7ff9325cb130>



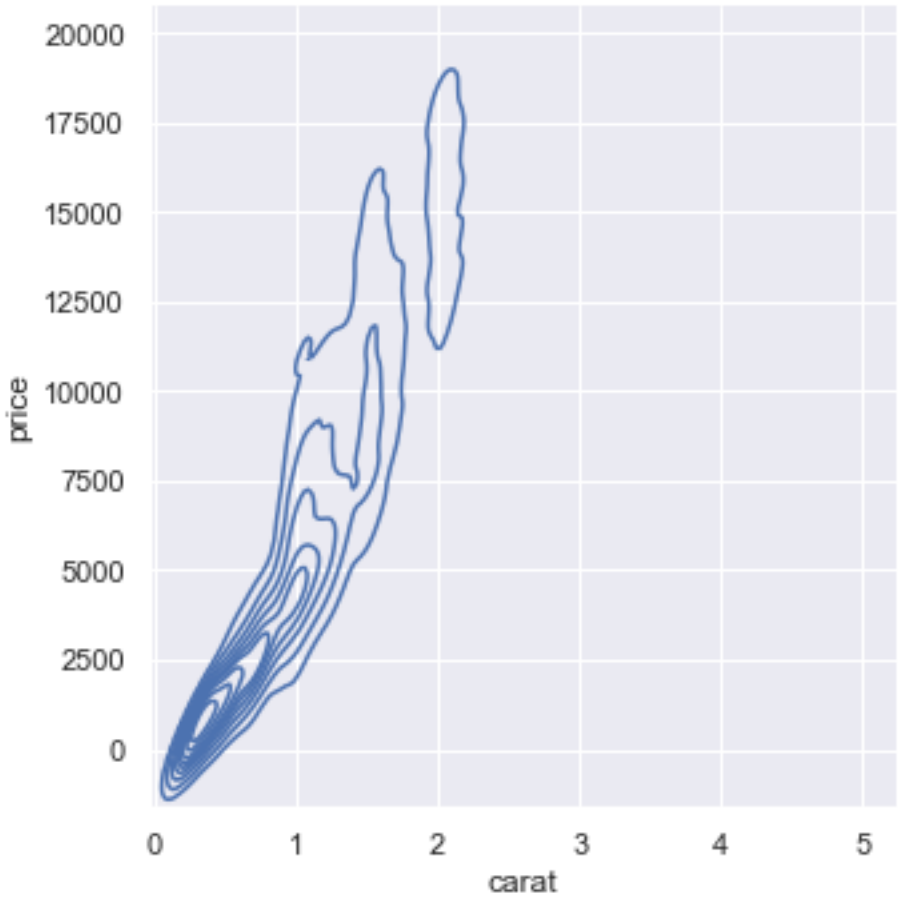
[21]:
```
sns.displot(data=df, x="price", hue="cut", kind = 'kde', rug = True)
```

[21]: <seaborn.axisgrid.FacetGrid at 0x7ff933211e80>



[22]:
```
# This one might take a minute to run.

sns.displot(data=df, x="carat", y='price', kind ='kde')
```

[22]: <seaborn.axisgrid.FacetGrid at 0x7ff9335414c0>



0.7 Seaborn Exercise 1 - 10 minutes

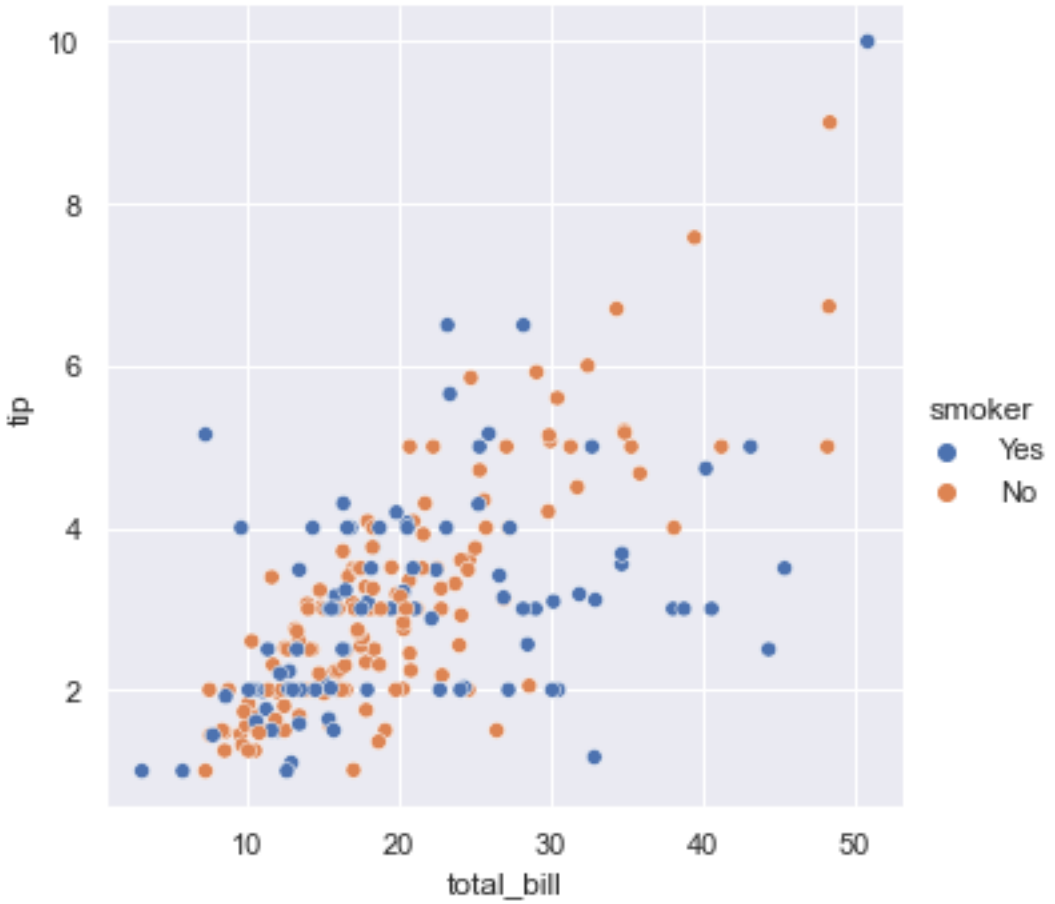
- Use the relational (relplot) figure-level function to create two charts. First a scatterplot and second a line chart.
- Use the 'tips' data set.
- For the scatterplot, determine if tips increasewith the bill amount. Try to show a distinction between data points based on time of day.
- For the line chart, show how tips change based on size of the party.

[23]:
```
tips.head()
```

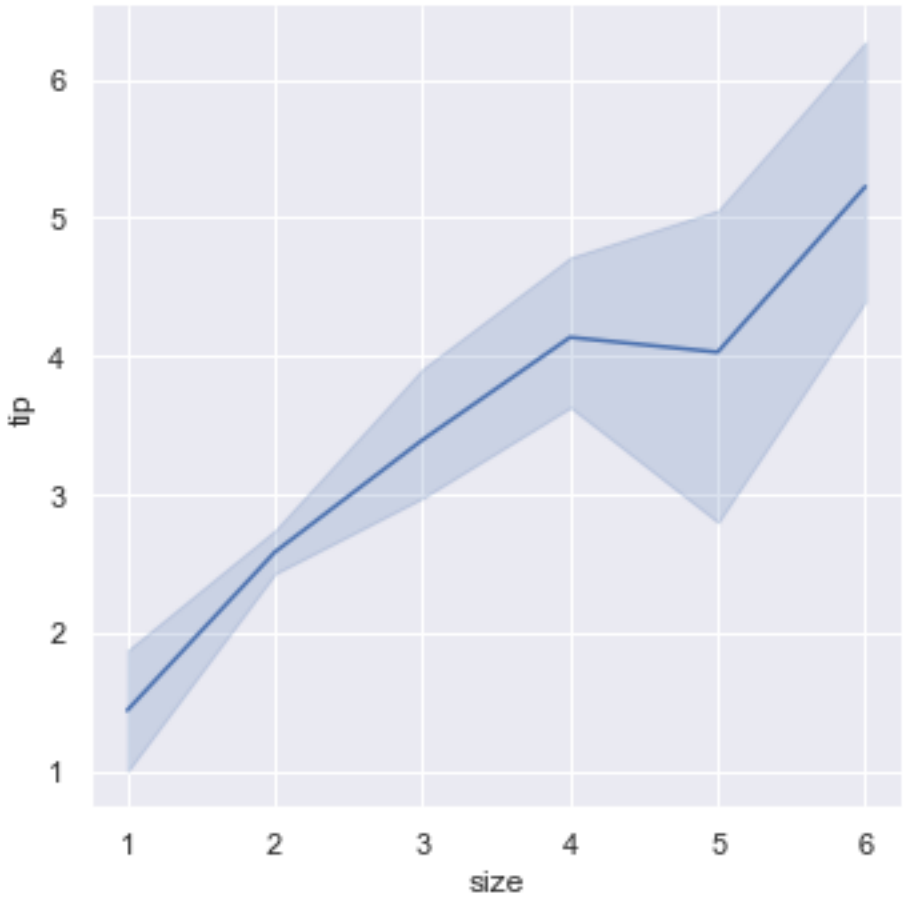
[23]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

[24]:
```
# Place scatterplot here
tips = sns.load_dataset("tips")
sns.relplot(x="total_bill", y="tip", data=tips, kind = 'scatter', hue =␣
↪'smoker');
```



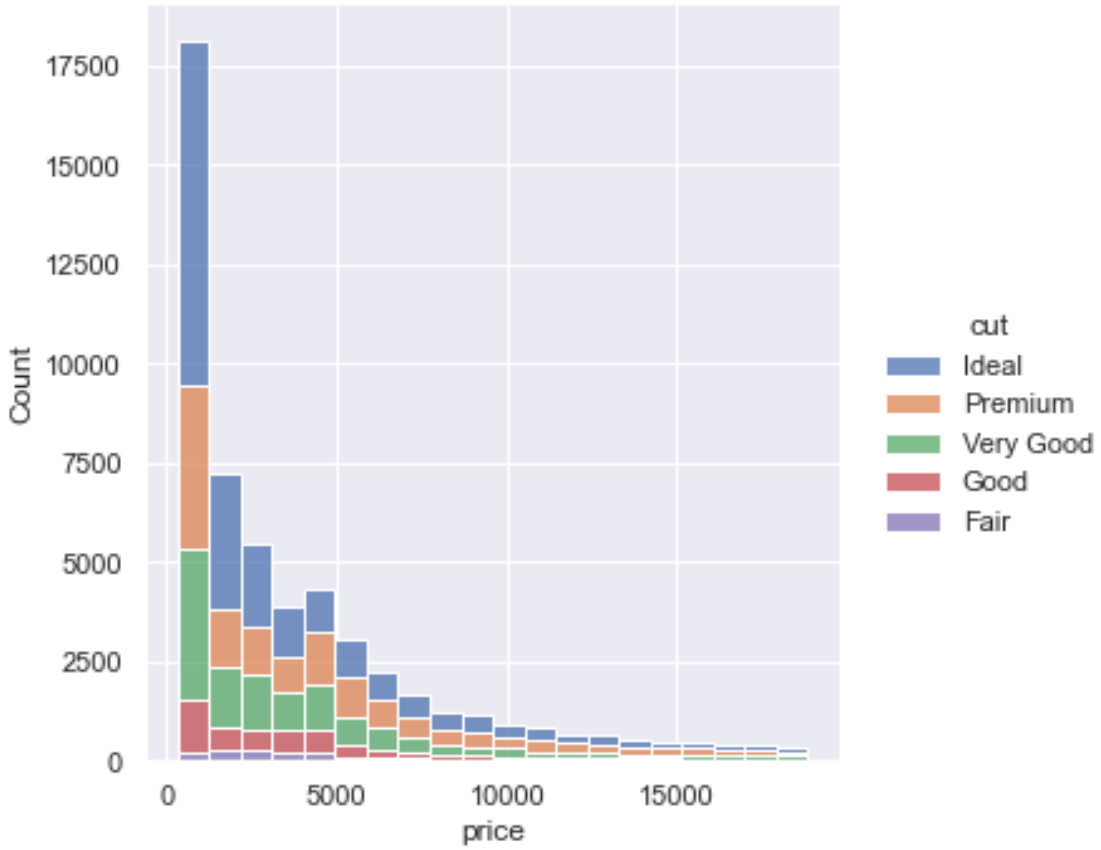
[25]:
```
# Place line chart here
sns.relplot(x="size", y="tip", data=tips, kind = 'line');
```



0.8 What's the difference?

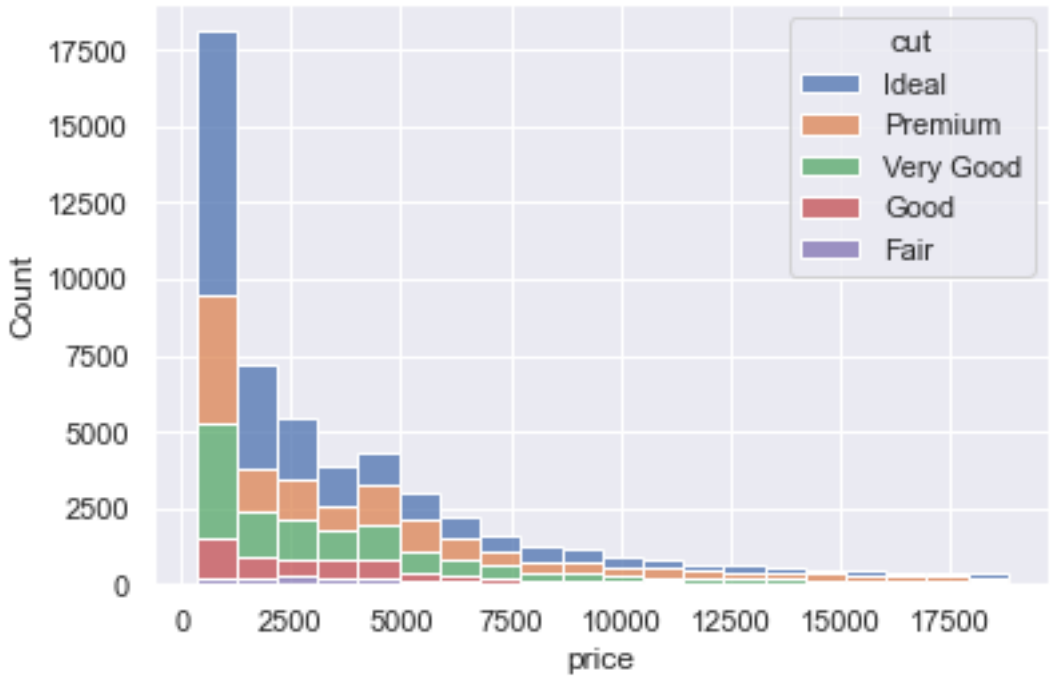
[26]:
```
sns.displot(data=df, x="price", hue="cut", multiple="stack", kind = 'hist',␣
↪bins = 20)
```

[26]:
```
<seaborn.axisgrid.FacetGrid at 0x7ff933537e80>
```



[27]:
```
sns.histplot(data=df, x='price', hue='cut', multiple = 'stack', bins = 20)
```

[27]:
```
<AxesSubplot:xlabel='price', ylabel='Count'>
```



Axes-level functions make self-contained plots The axes-level functions are written to act like drop-in replacements for matplotlib functions. While they add axis labels and legends automatically, they don't modify anything beyond the axes that they are drawn into. That means they can be composed into arbitrarily-complex matplotlib figures with predictable results.

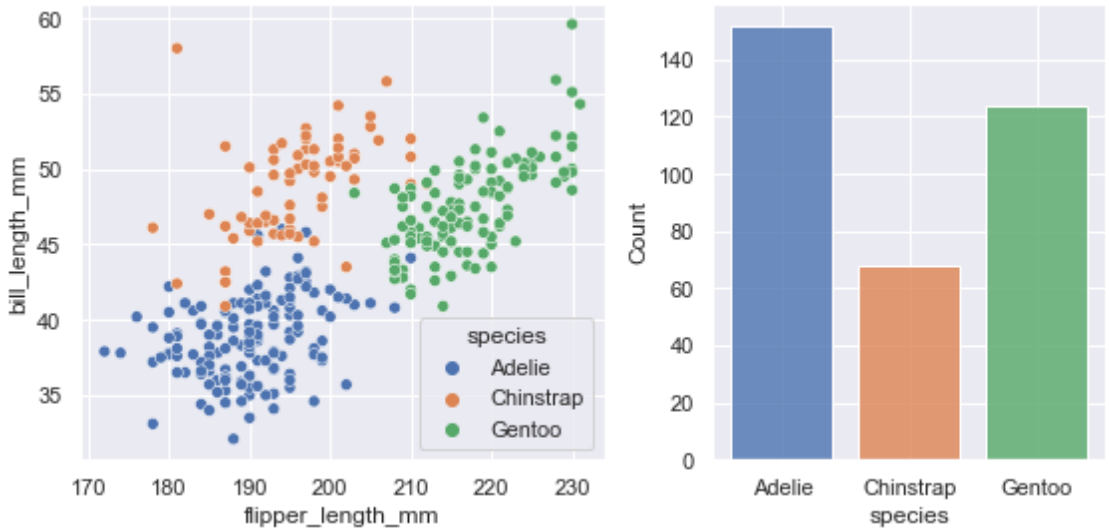
0.8.1 Combining matplotlib & seaborn syntax

[28]:
```
penguins.head()
```

[28]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

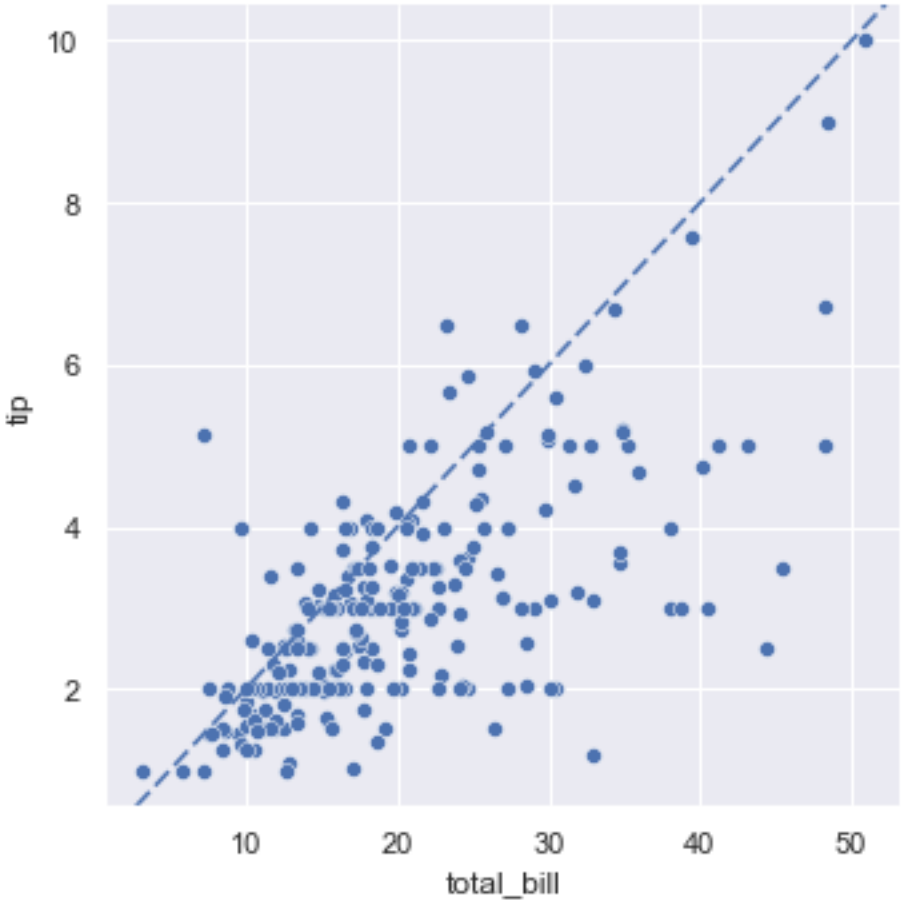
[29]:
```
# Example taken from Seaborn documentation

# Use penguins dataset

f, axs = plt.subplots(1, 2, figsize=(8, 4), gridspec_kw=dict(width_ratios=[4,␣
↪3]))

sns.scatterplot(data=penguins,
                x="flipper_length_mm",
                y="bill_length_mm",
                hue="species",
                ax=axs[0])

sns.histplot(data=penguins,
             x="species",
             hue="species",
             shrink=.8,
             alpha=.8,
             legend=False,
             ax=axs[1])

f.tight_layout() # adjusts the space between subplots & around  figure edge to␣
↪accomodate labels and content.
```



[30]:
```
# Example taken from Seaborn documentation
# Use tips dataset

g = sns.relplot(data=tips, x="total_bill", y="tip")
```

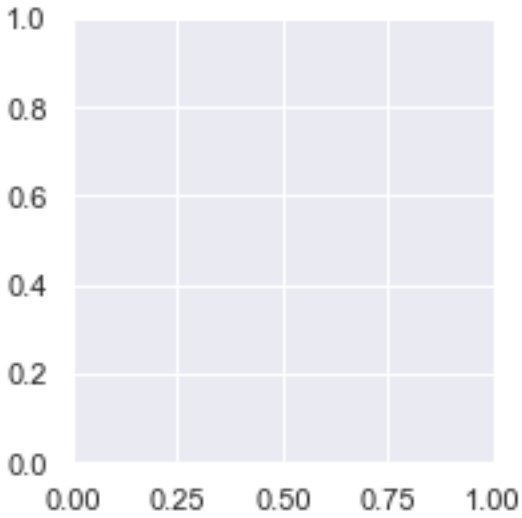
```
g.ax.axline(xy1=(10, 2), slope=.2, color="b", dashes=(5, 2))
```

[30]: <matplotlib.lines._AxLine at 0x7ff9385c46d0>

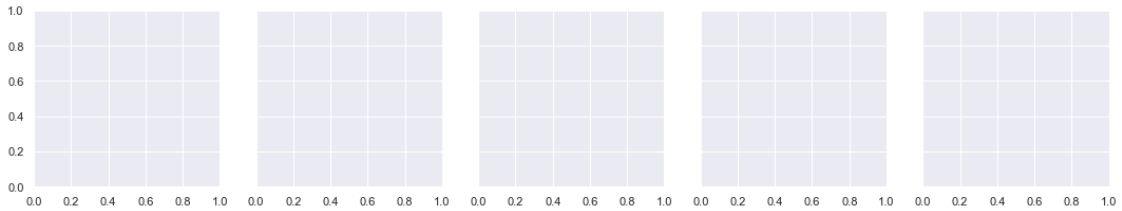


0.9 Facet Grid

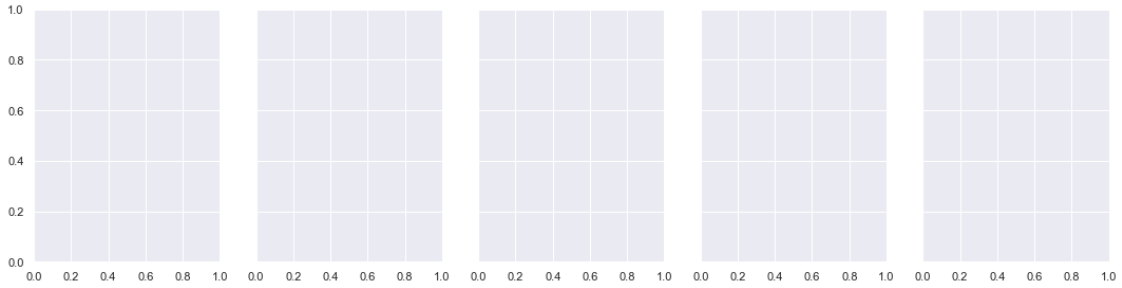
[31]:
```
p = sns.FacetGrid(df)
```



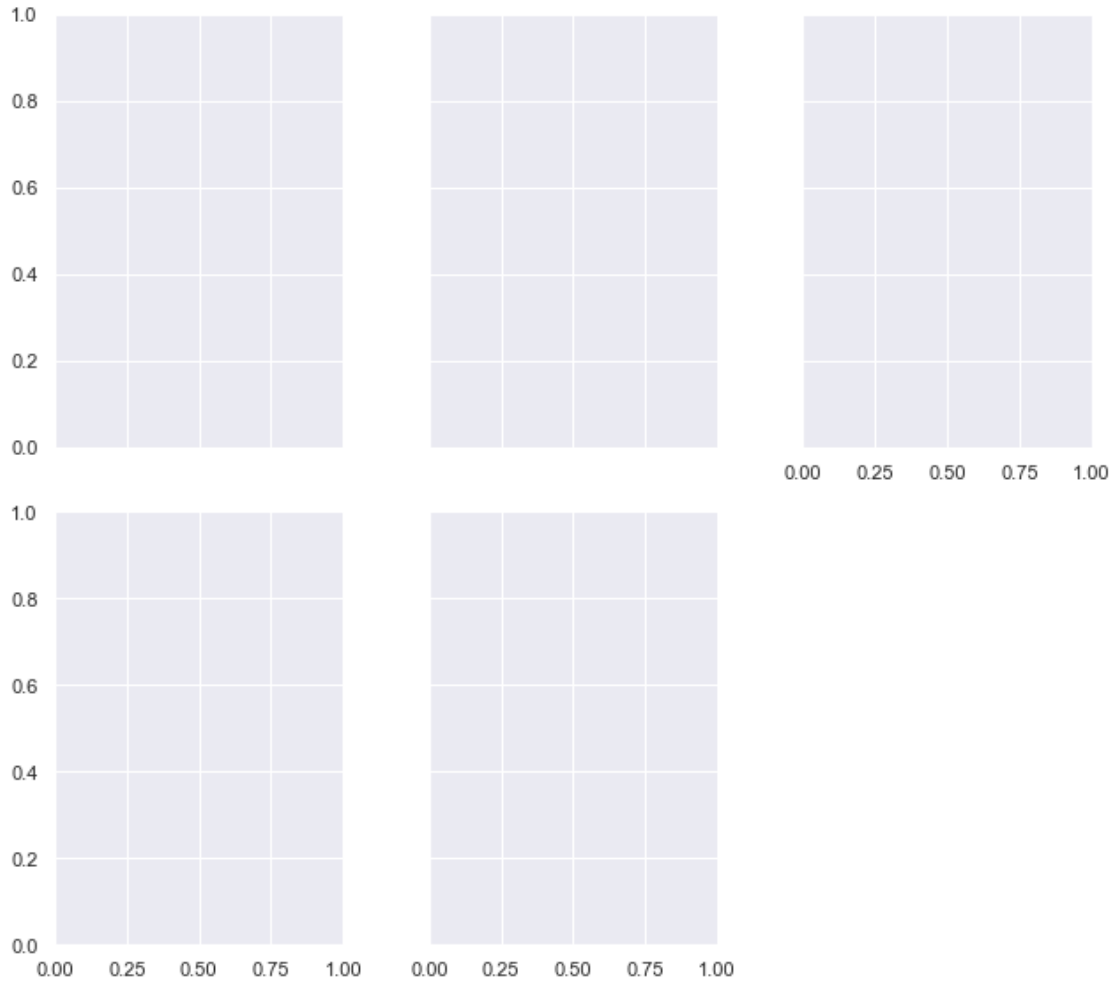
[32]:
```
p = sns.FacetGrid(df, col = 'cut')

# matplotlib will squeeze the 5 plots into the orginal size.
```



[33]:
```
p = sns.FacetGrid(df, col = 'cut', height = 4, aspect = 0.75)
# Aspect ratio of each facet, so that aspect * height gives the width of each␣
 ↪facet.
```

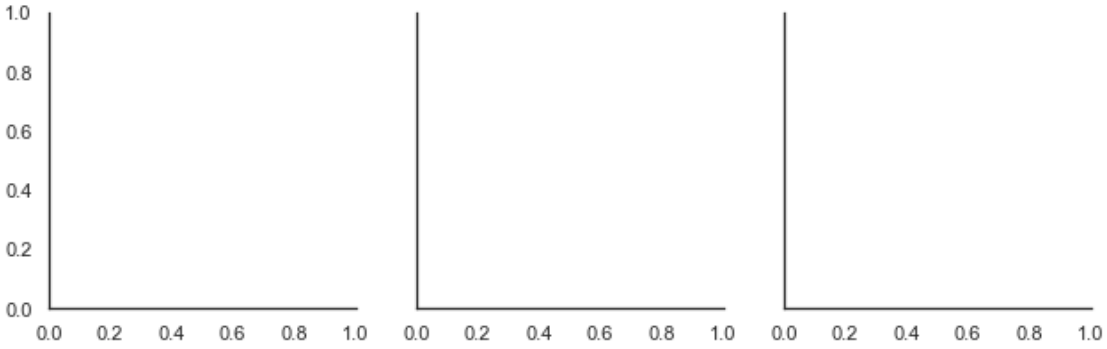


[34]:
```
p = sns.FacetGrid(df, col = 'cut', height = 4, aspect = 0.75, col_wrap = 3)
# Aspect ratio of each facet, so that aspect * height gives the width of each␣
 ↪facet.
```



[35]:
```
sns.set_style('white')
penguins = sns.load_dataset("penguins")
```

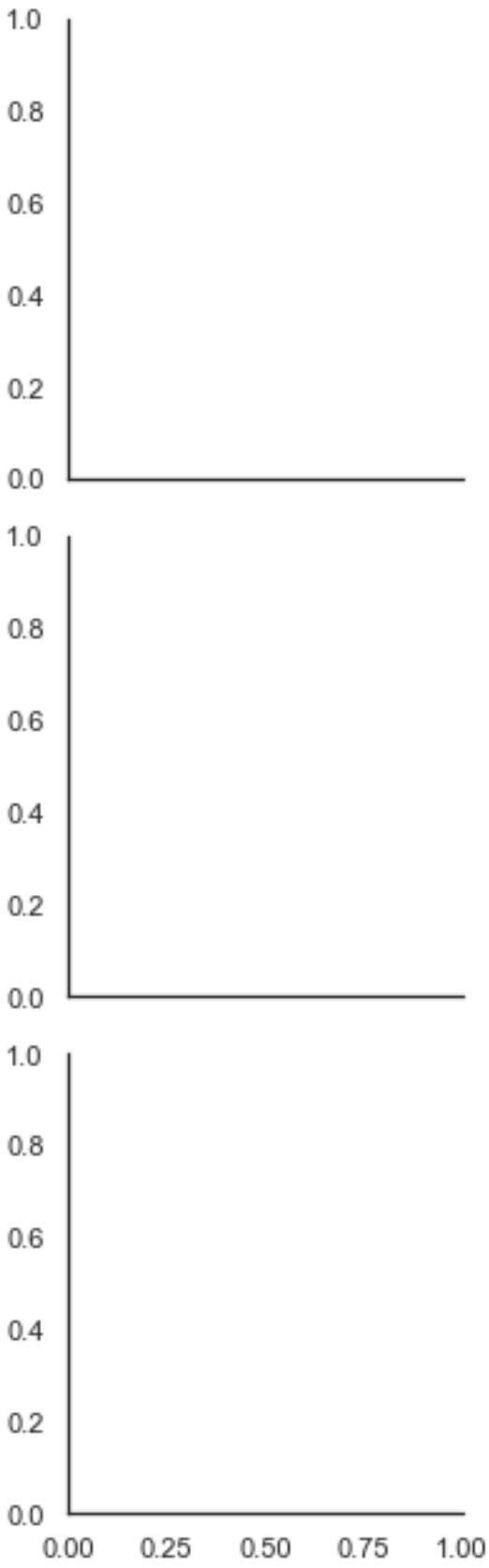
[36]:
```
p = sns.FacetGrid(penguins, col='island');
```



[37]:
```
type(p)
```

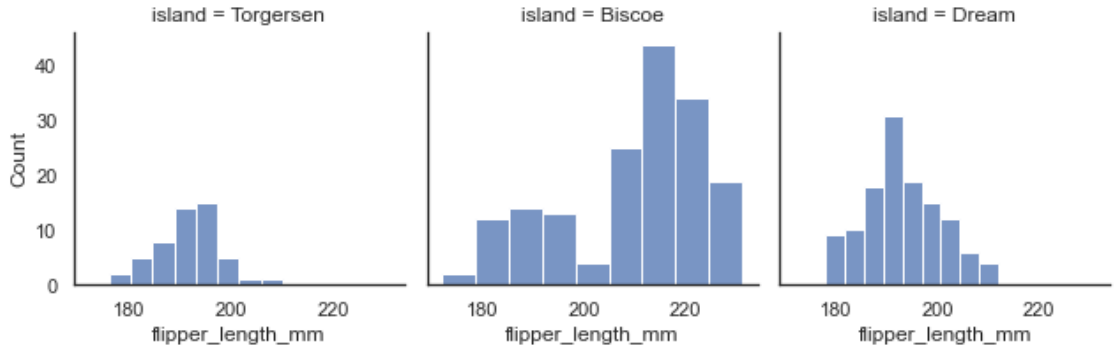
[37]:
```
seaborn.axisgrid.FacetGrid
```

[38]:
```
p = sns.FacetGrid(penguins, row='island');
```


1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00

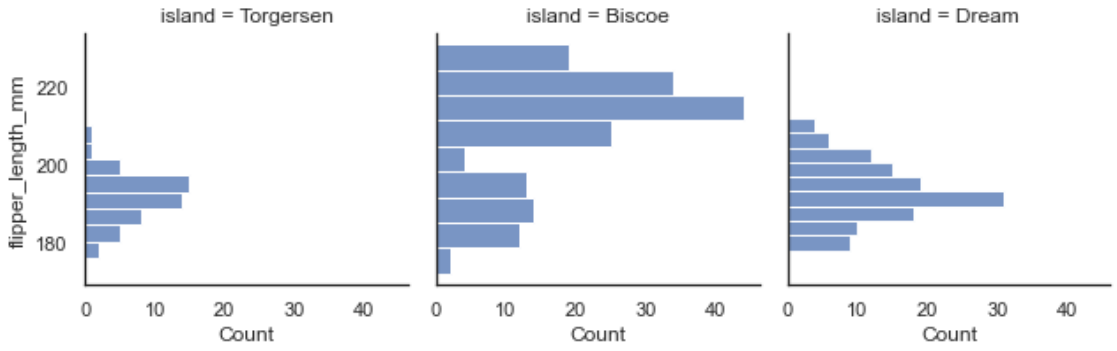
0.9.1 .map()

[39]:
```
p = sns.FacetGrid(penguins, col='island')
p.map(sns.histplot, "flipper_length_mm");
```



0.9.2 .map_dataframe()

[40]:
```
p = sns.FacetGrid(penguins, col='island')
p.map_dataframe(sns.histplot, y='flipper_length_mm');
```

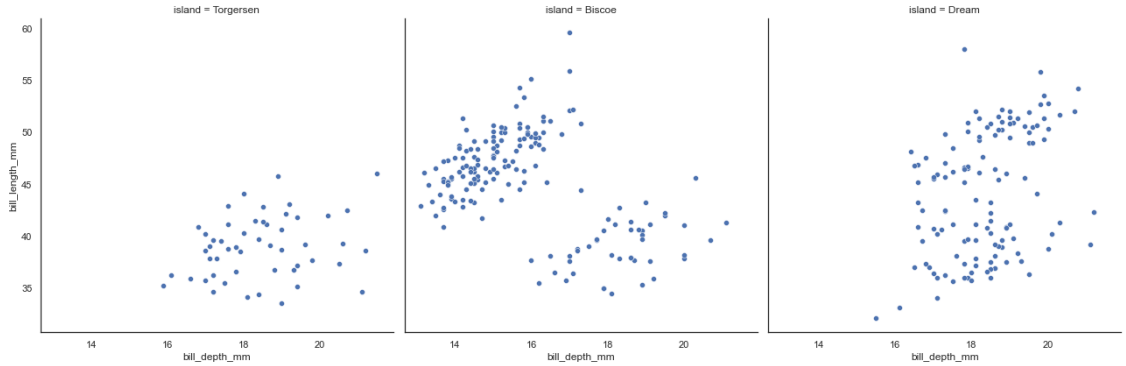


[41]:
```
penguins.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
```

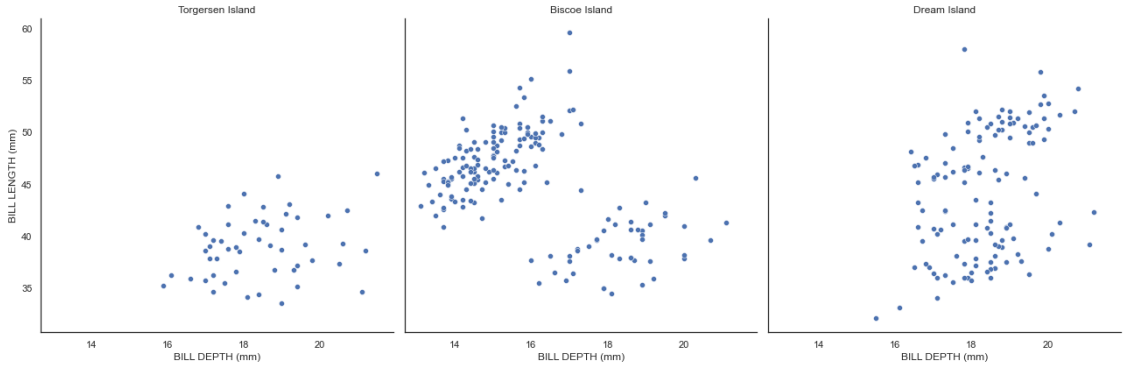
```
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

[42]:
```
p = sns.FacetGrid(penguins, col='island', height = 6, aspect =1)
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm');
```

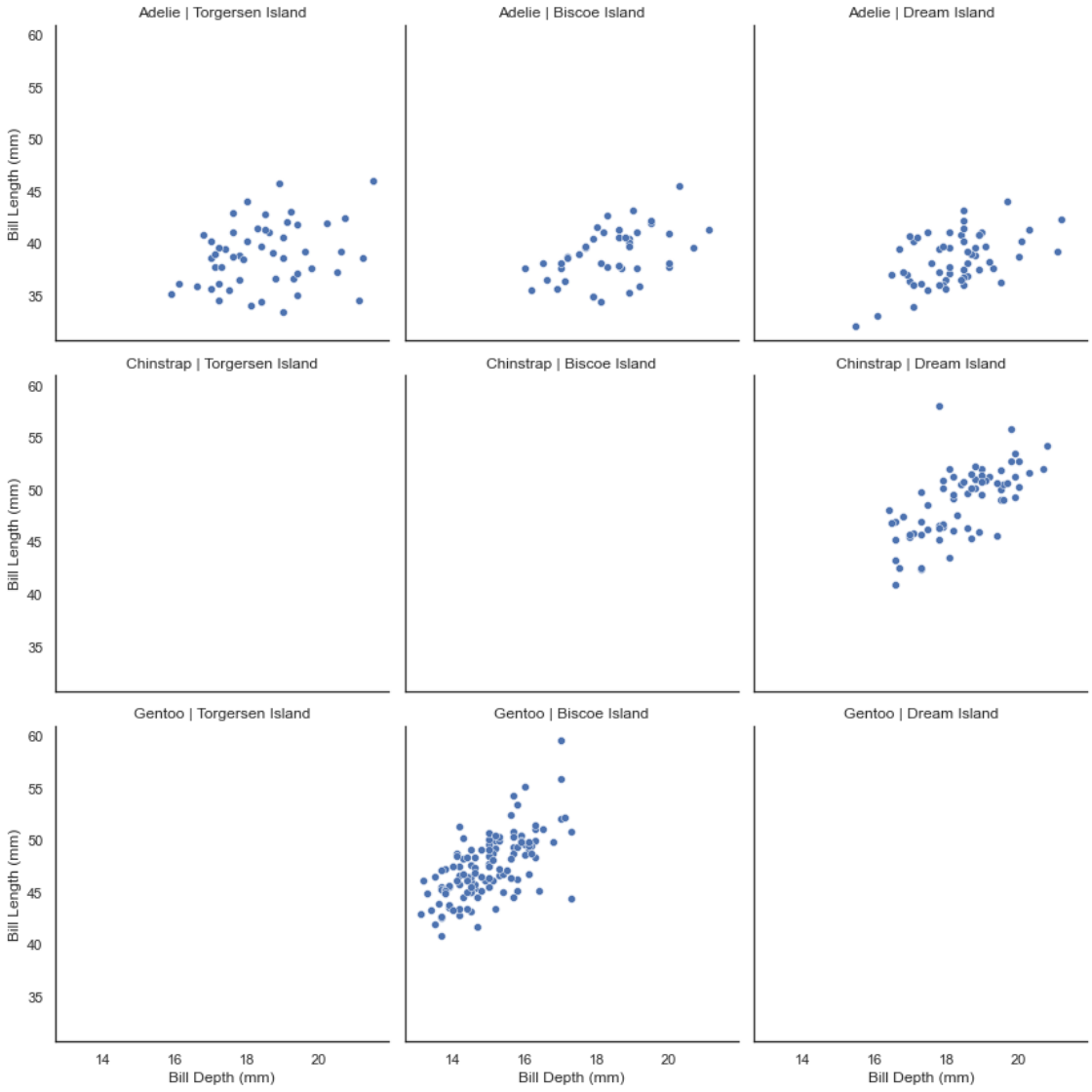


0.9.3 .set_axis_labels(), .set_titles(), sharey, ylim

[43]:
```
p = sns.FacetGrid(penguins, col='island', height = 6, aspect =1)
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm');
p.set_axis_labels('BILL DEPTH (mm)', 'BILL LENGTH (mm)'); # if the LABELS needs␣
 ↪to be changed
p.set_titles(col_template='{col_name} Island'); # if the TITLE needs to be␣
 ↪changed
```



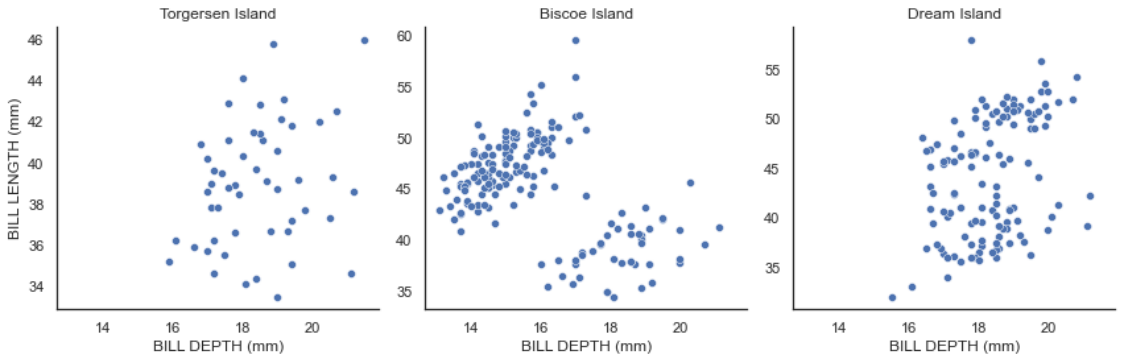
[44]:
```
p = sns.FacetGrid(penguins, col='island', row='species', height = 4, aspect =1)
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm')
p.set_axis_labels('Bill Depth (mm)', 'Bill Length (mm)')
p.set_titles(row_template='{row_name}', col_template='{col_name} Island');
```



- sharey > False: the y-axis will not be shared and each plot will get its own y-axis.
- ylim > Sets a specified range for all y-axes shown

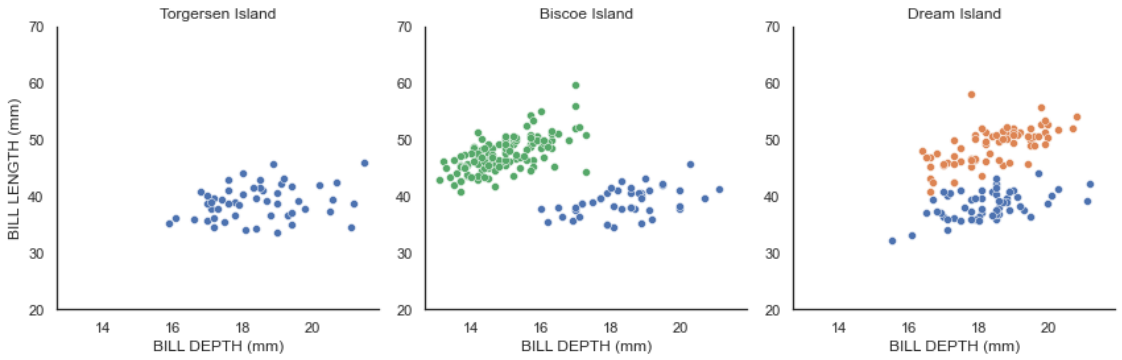
[45]:
```
p = sns.FacetGrid(penguins, col='island', height = 4, aspect =1, sharey=False)
#p = sns.FacetGrid(penguins, col='island', height = 4, aspect =1, sharey=False,␣
↪ylim=(20, 70))
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm');
p.set_axis_labels('BILL DEPTH (mm)', 'BILL LENGTH (mm)');
```

```
p.set_titles(col_template='{col_name} Island');
```



0.9.4 hue & pallette

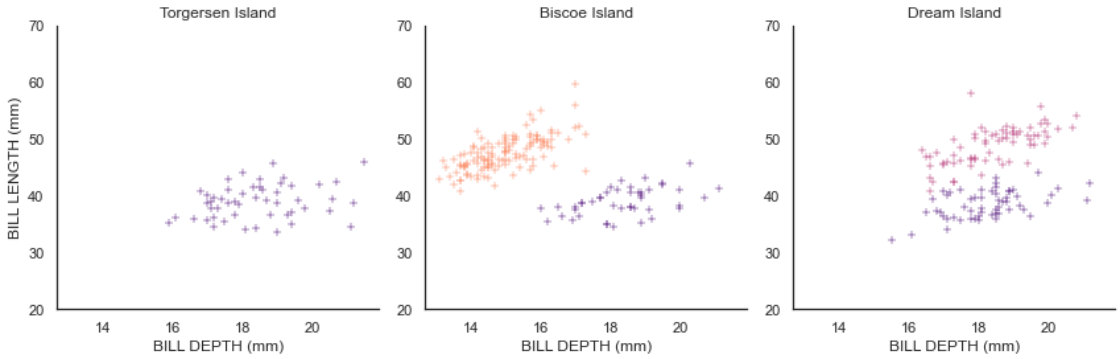
[46]:
```
p = sns.FacetGrid(penguins, col='island', height = 4, aspect =1, sharey=False,␣
↪ylim=(20, 70), hue = 'species')
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm');
p.set_axis_labels('BILL DEPTH (mm)', 'BILL LENGTH (mm)');
p.set_titles(col_template='{col_name} Island');
```



Note: If hue is placed inside the scatterplot

[47]:
```
p = sns.FacetGrid(penguins,
                  col='island',
                  height = 4,
                  aspect =1,
                  sharey=False,
                  ylim=(20, 70),
                  hue = 'species',
                  palette = 'magma')
```

```
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm', marker␣
↪= '+');
p.set_axis_labels('BILL DEPTH (mm)', 'BILL LENGTH (mm)');
p.set_titles(col_template='{col_name} Island');
```

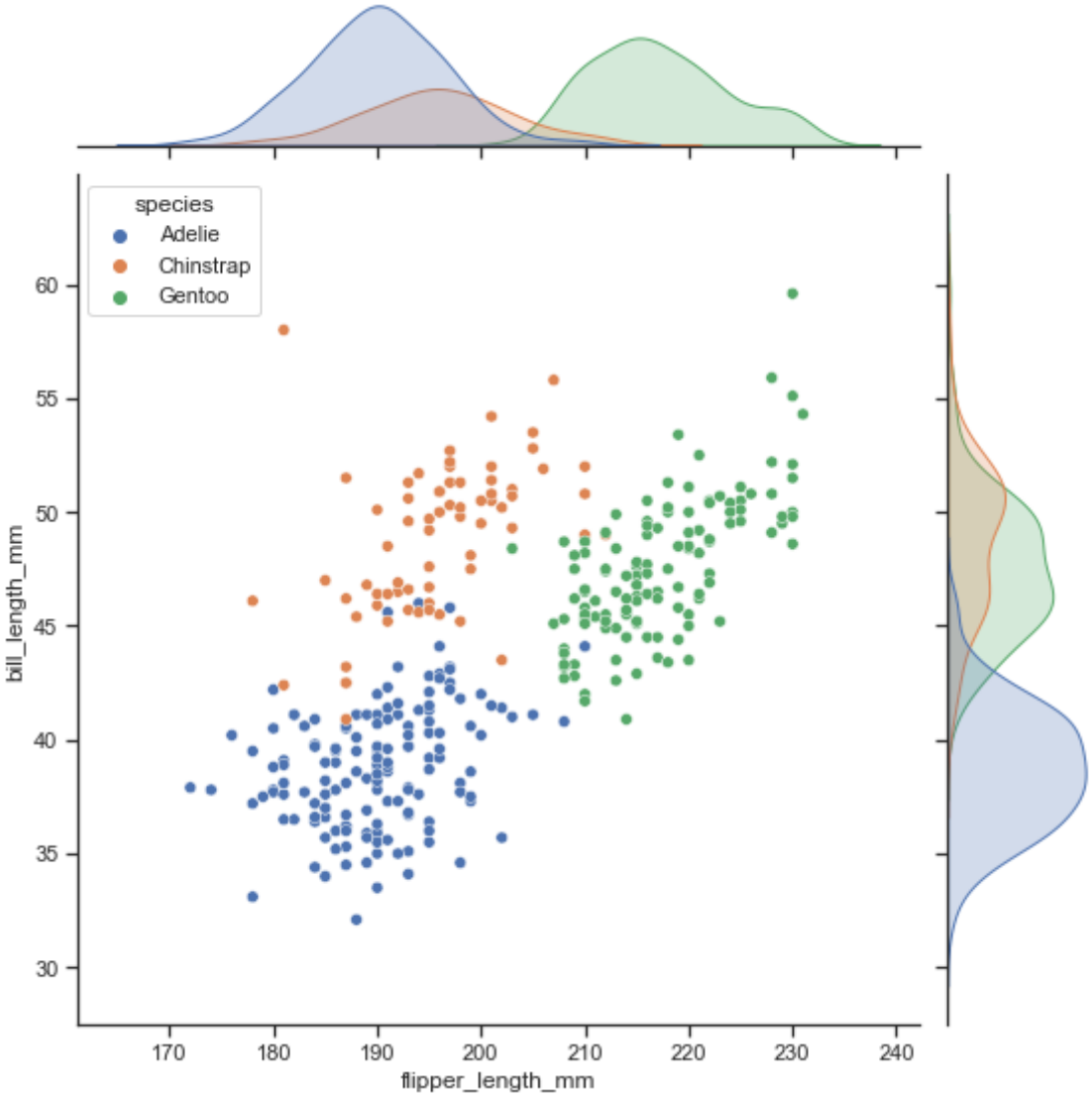


0.10 Multiple Views

0.10.1 Jointplot

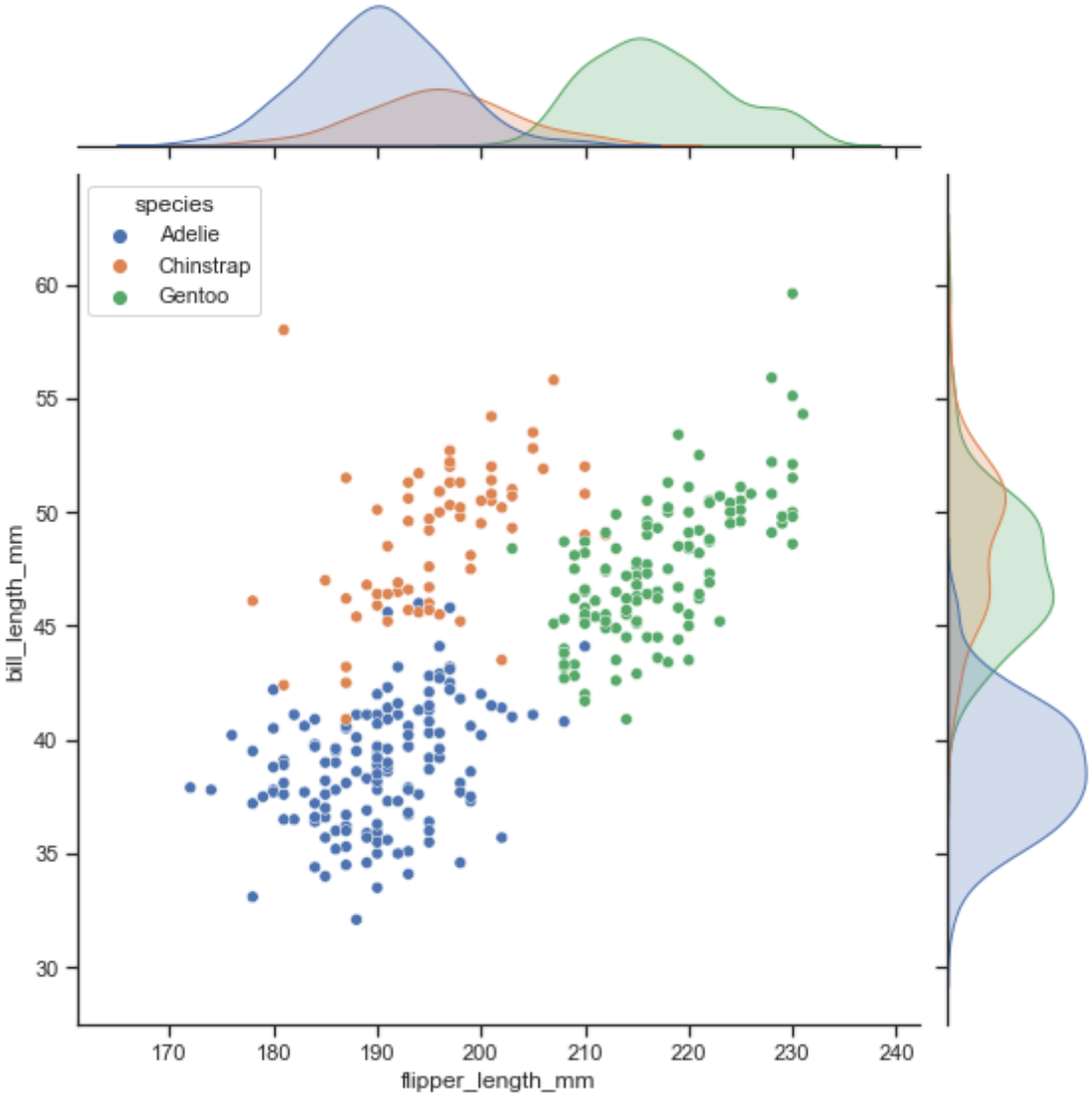
[48]:
```
sns.set_style("ticks")
sns.jointplot(data = penguins, x="flipper_length_mm", y="bill_length_mm",␣
↪hue="species", height = 8 )
```

[48]: <seaborn.axisgrid.JointGrid at 0x7ff9388db370>



[49]:
```
sns.set_style("ticks")
sns.jointplot(data = penguins, x="flipper_length_mm", y="bill_length_mm",␣
↪hue="species", height = 8 )
```

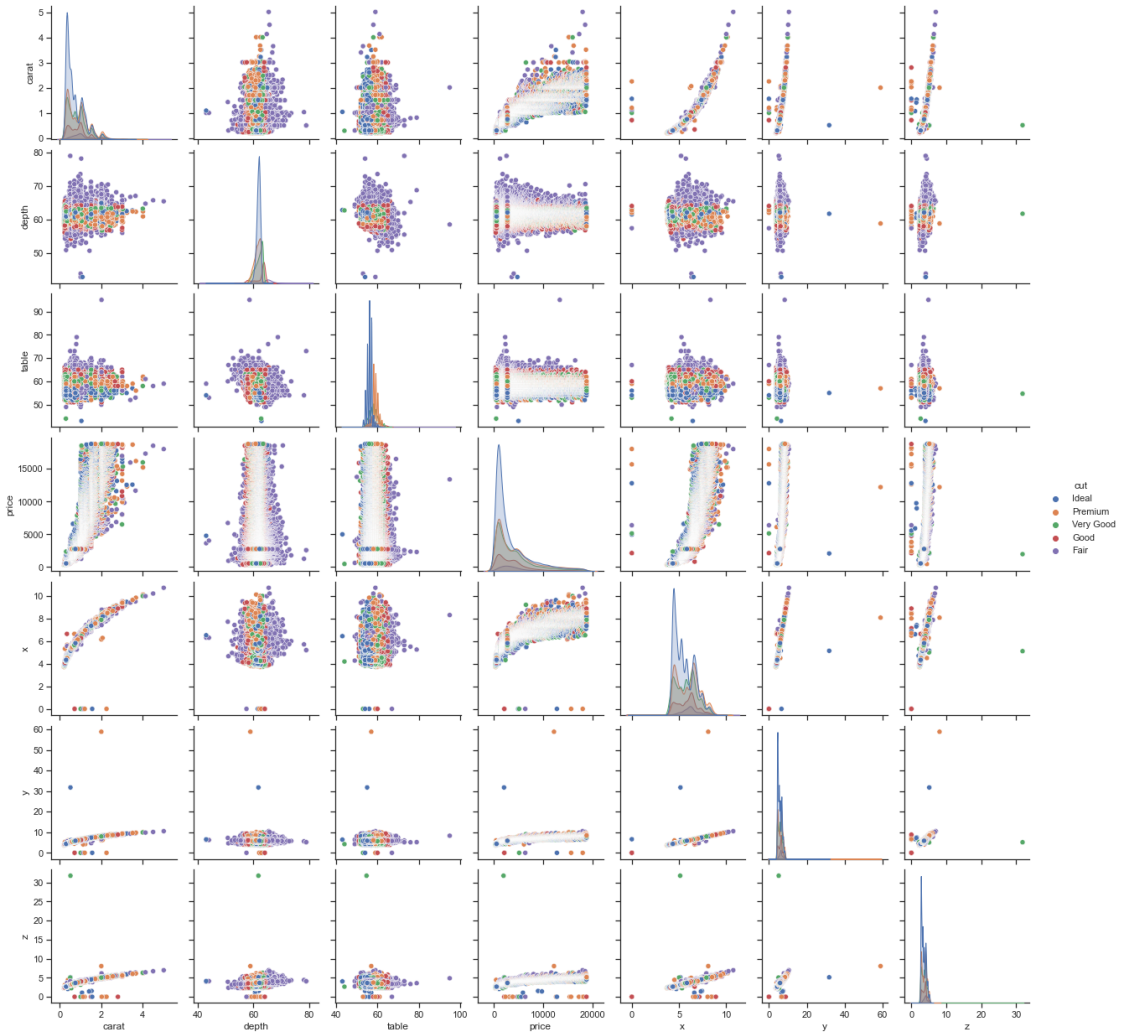
[49]: <seaborn.axisgrid.JointGrid at 0x7ff938e69970>



0.10.2 Pairplot

[50]:
```
sns.pairplot(data = df, hue = 'cut')
```

[50]:
```
<seaborn.axisgrid.PairGrid at 0x7ff94c3fa910>
```



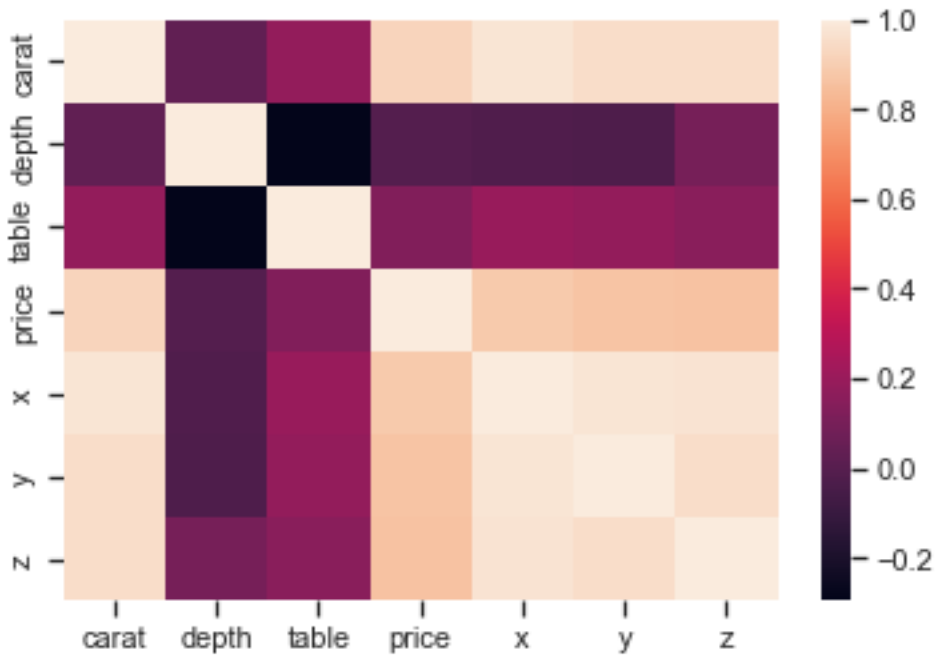
[51]:
```
xyz = df.corr()
xyz
```

[51]:

	carat	depth	table	price	x	y	z
carat	1.000000	0.028224	0.181618	0.921591	0.975094	0.951722	0.953387
depth	0.028224	1.000000	-0.295779	-0.010647	-0.025289	-0.029341	0.094924
table	0.181618	-0.295779	1.000000	0.127134	0.195344	0.183760	0.150929
price	0.921591	-0.010647	0.127134	1.000000	0.884435	0.865421	0.861249
x	0.975094	-0.025289	0.195344	0.884435	1.000000	0.974701	0.970772
y	0.951722	-0.029341	0.183760	0.865421	0.974701	1.000000	0.952006
z	0.953387	0.094924	0.150929	0.861249	0.970772	0.952006	1.000000

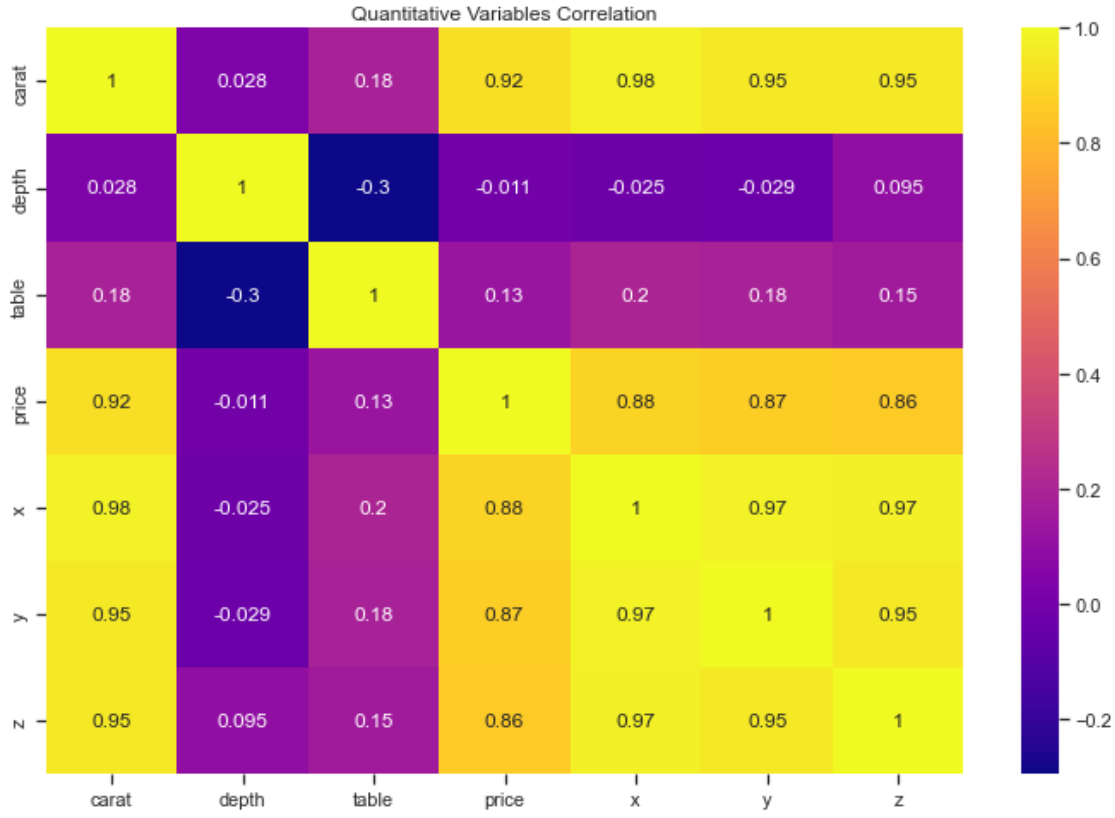
[52]:
```
sns.heatmap(xyz, annot=False)
```

[52]: <AxesSubplot:>

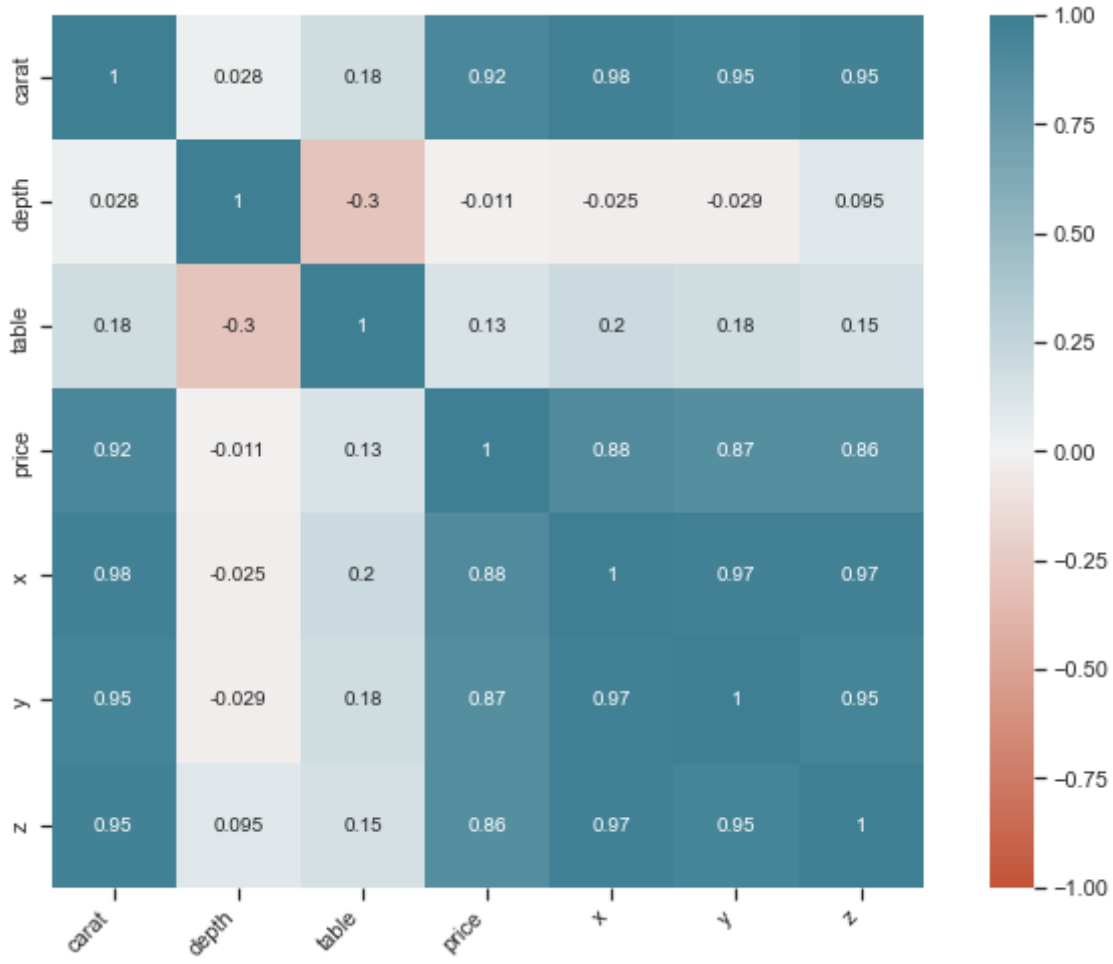


[53]:
```
# Calculate correlations
corr = df.corr()
plt.figure(figsize=(12,8))
plt.title('Quantitative Variables Correlation')

# Heatmap
sns.heatmap(corr,cmap='plasma',annot=True)
```

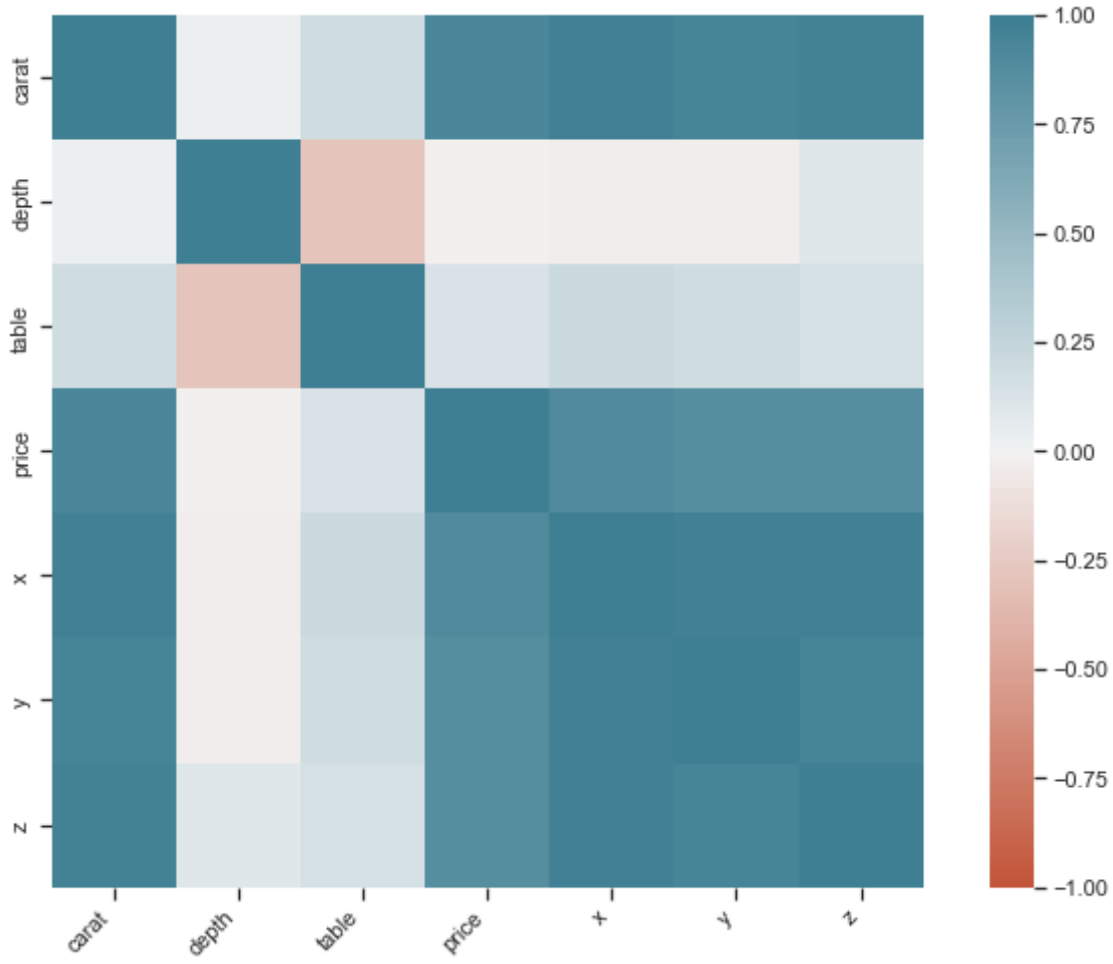
[53]: <AxesSubplot:title={'center':'Quantitative Variables Correlation'}>



[54]:
```
plt.figure(figsize=(12,8))
corr = df.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True,
    annot=True, annot_kws={"size":10}
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```



[55]:
```
plt.figure(figsize=(12,8))
corr = df.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True,
    annot=False, annot_kws={"size":20}
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```



0.11 Seaborn Exercise 2 - 10 minutes

Using the flights info, create a visualization that plots - for each month - the number of passengers by year.
There should be one plot per month.

[56]:
```
flights.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   year        144 non-null    int64
 1   month       144 non-null    category
 2   passengers  144 non-null    int64
dtypes: category(1), int64(2)
memory usage: 2.9 KB
```

[57]:
```
flights.head(20)
```

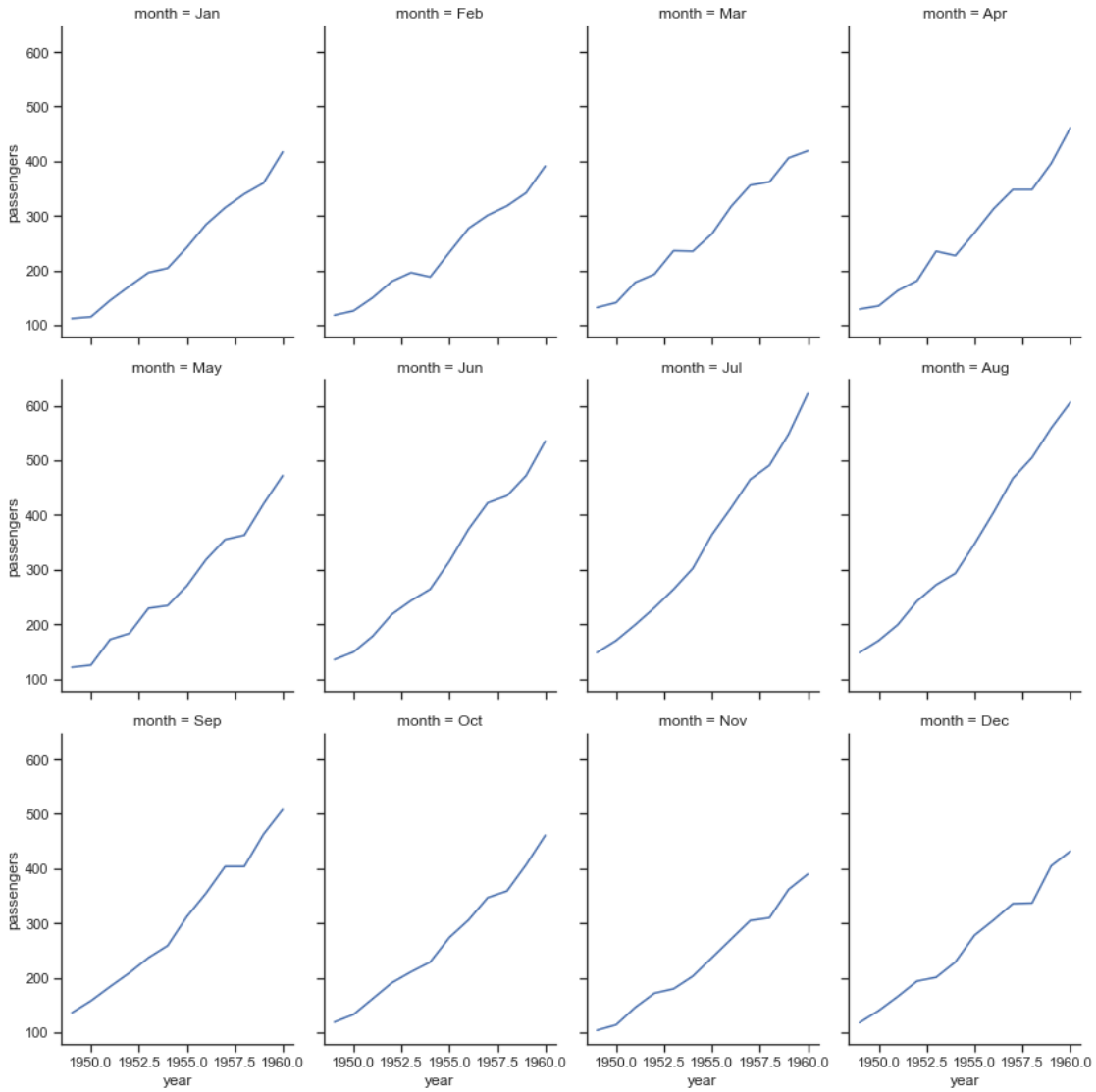
[57]:

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
5	1949	Jun	135
6	1949	Jul	148
7	1949	Aug	148
8	1949	Sep	136
9	1949	Oct	119
10	1949	Nov	104
11	1949	Dec	118
12	1950	Jan	115
13	1950	Feb	126
14	1950	Mar	141
15	1950	Apr	135
16	1950	May	125
17	1950	Jun	149
18	1950	Jul	170
19	1950	Aug	170

[58]:
```
flights.shape
```

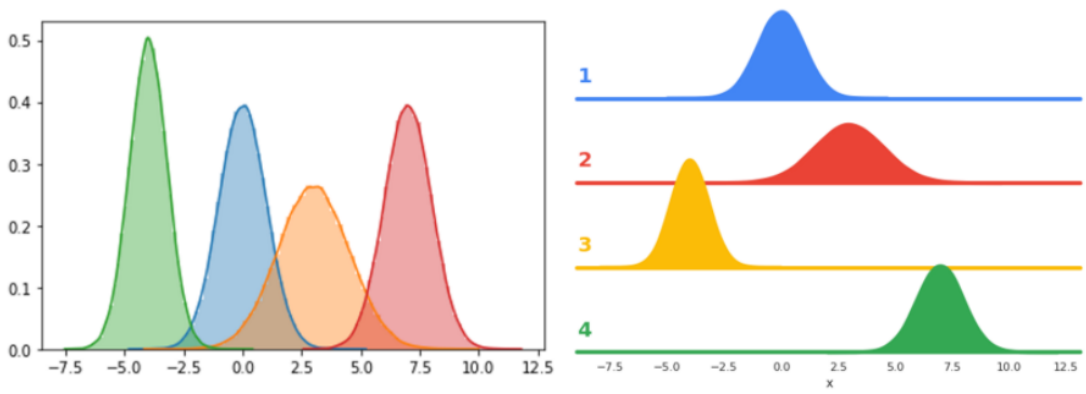
[58]:
```
(144, 3)
```

[59]:
```
# Place solution here
p = sns.FacetGrid(flights, col = 'month', height = 4, aspect = 0.75, col_wrap =␣
↪4)
p.map_dataframe(sns.lineplot, x='year', y='passengers');
```



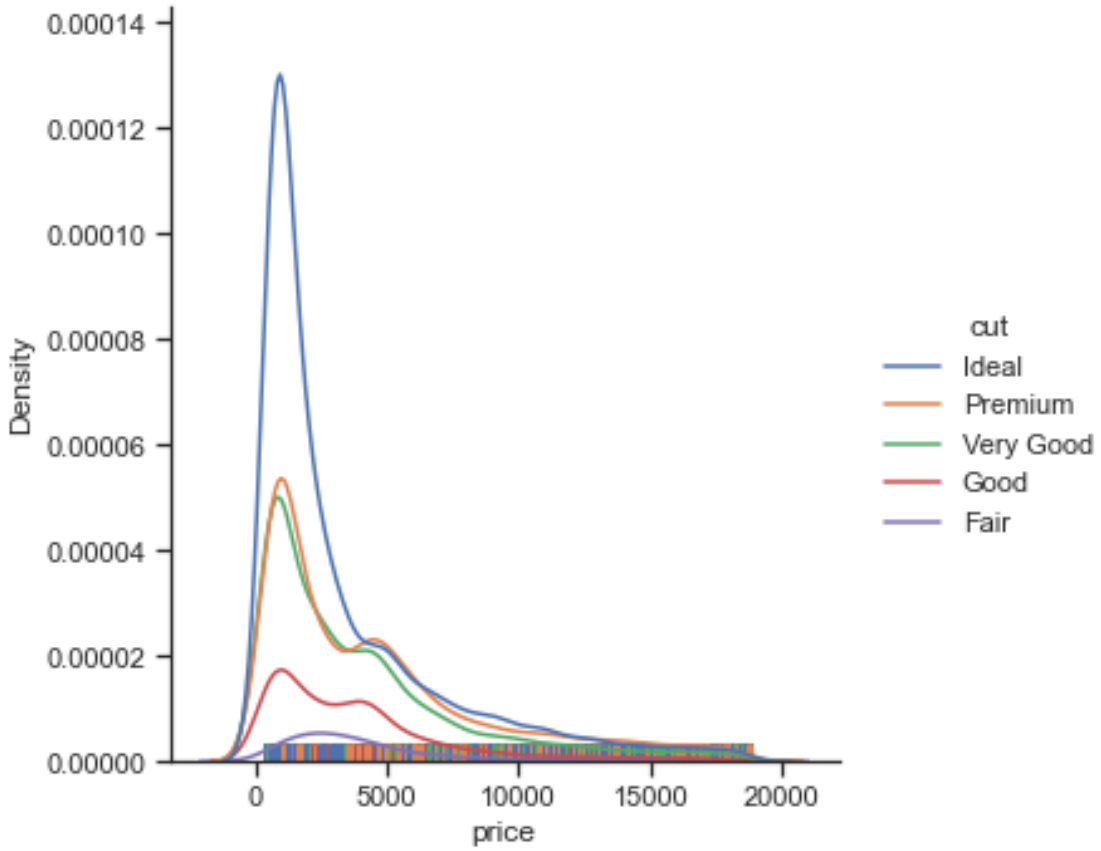
0.12 Seaborn Exercise 3 - 15 minutes

The distplot below is quick 'one-liner' plot. Take a little more time to create an axes for each cut and the axes are one above the other.



[60]:
```
sns.displot(data=df, x="price", hue="cut", kind = 'kde', rug = True)
```

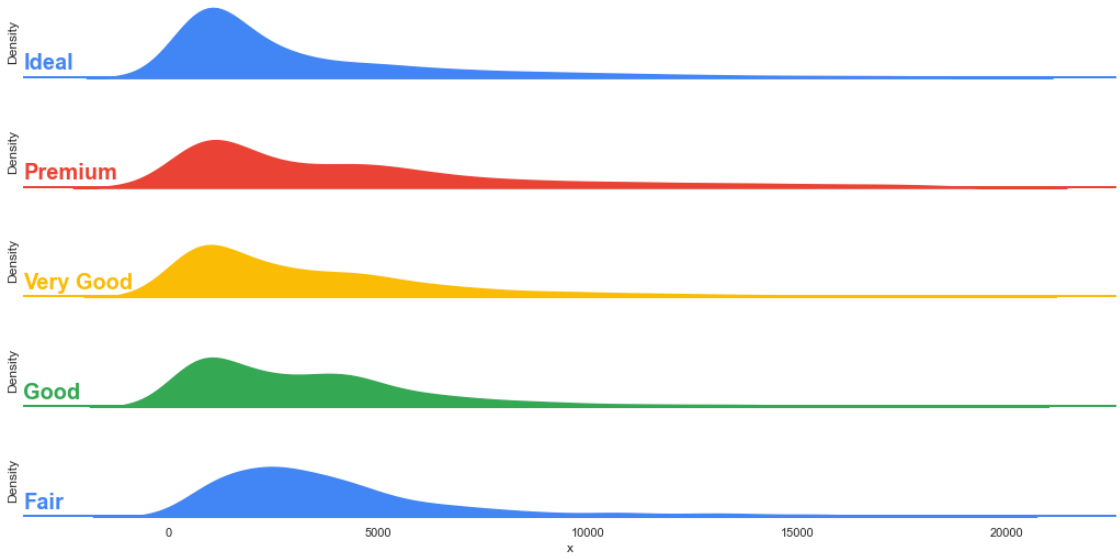
[60]: <seaborn.axisgrid.FacetGrid at 0x7ff9258ff130>



[61]:
```
# Place Exercise 3 solution here.
```

```
# How do we get 5 separate plots?  How do we get each on a row?

# https://towardsdatascience.com/
 ↪sorry-but-sns-distplot-just-isnt-good-enough-this-is-though-ef2ddbf28078
df = sns.load_dataset("diamonds")

sns.set_style('white')
g = sns.FacetGrid(df, #the dataframe to pull from
                  row="cut", #define the column for each subplot row to be␣
 ↪differentiated by
                  hue="cut", #define the column for each subplot color to be␣
 ↪differentiated by
                  aspect=10, #aspect * height = width
                  height=1.5, #height of each subplot
                  palette=['#4285F4','#EA4335','#FBBC05','#34A853'] #google␣
 ↪colors
                 )
#shade: True/False, shade area under curve or not
#alpha: transparency, lw: line width, bw: kernel shape specification


#g.map(sns.kdeplot, "price", lw=4, bw_method=0.2)   Same as below but no fill
g.map(sns.kdeplot, "price", shade=True, alpha=1, lw=1.5, bw_method=0.2)
g.map(plt.axhline, y=0, lw=4)

def label(x, color, label):
    ax = plt.gca() #get the axes of the current object
    ax.text(0, .2, #location of text
            label, #text label
            fontweight="bold", color=color, size=20, #text attributes
            ha="left", va="center", #alignment specifications
            transform=ax.transAxes) #specify axes of transformation

g.map(label, "x") #the function counts as a plotting object!


g.set_titles("") #set title to blank
g.set(yticks=[]) #set y ticks to blank
g.despine(bottom=True, left=True) #remove 'spines'
```

[61]: <seaborn.axisgrid.FacetGrid at 0x7ff9278cd7f0>


Density
Ideal
Density
Premium
Density
Very Good
Density
Good
Density
Fair
0
5000
10000
15000
20000
x