4 - Seaborn

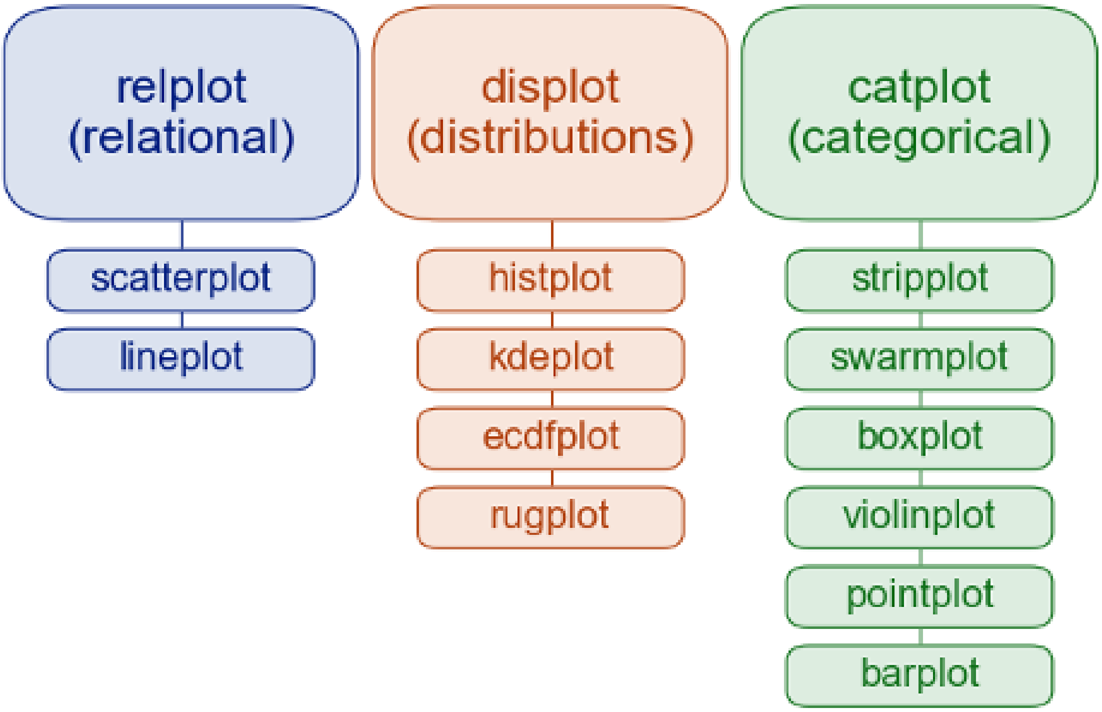
March 22, 2023

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

[1]:

```
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")
```

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

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

```
[3]:    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
```

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

```
[4]:    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
```

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

```
[5]:   species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
     0  Adelie  Torgersen            39.1           18.7              181.0
     1  Adelie  Torgersen            39.5           17.4              186.0
     2  Adelie  Torgersen            40.3           18.0              195.0
     3  Adelie  Torgersen             NaN            NaN                NaN
     4  Adelie  Torgersen            36.7           19.3              193.0

        body_mass_g     sex
     0       3750.0    Male
     1       3800.0  Female
     2       3250.0  Female
     3          NaN     NaN
     4       3450.0  Female
```

[6]:

```
flights.head()
```

[6]:

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

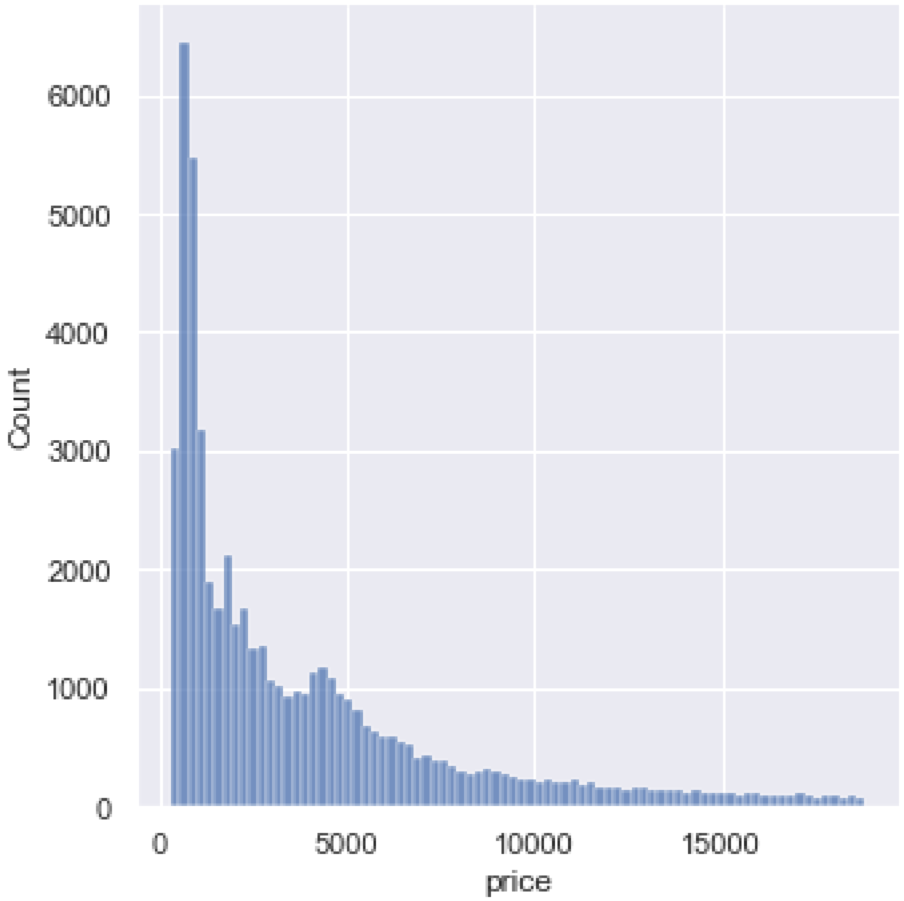
0.4 http://seaborn.pydata.org/tutorial.html

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

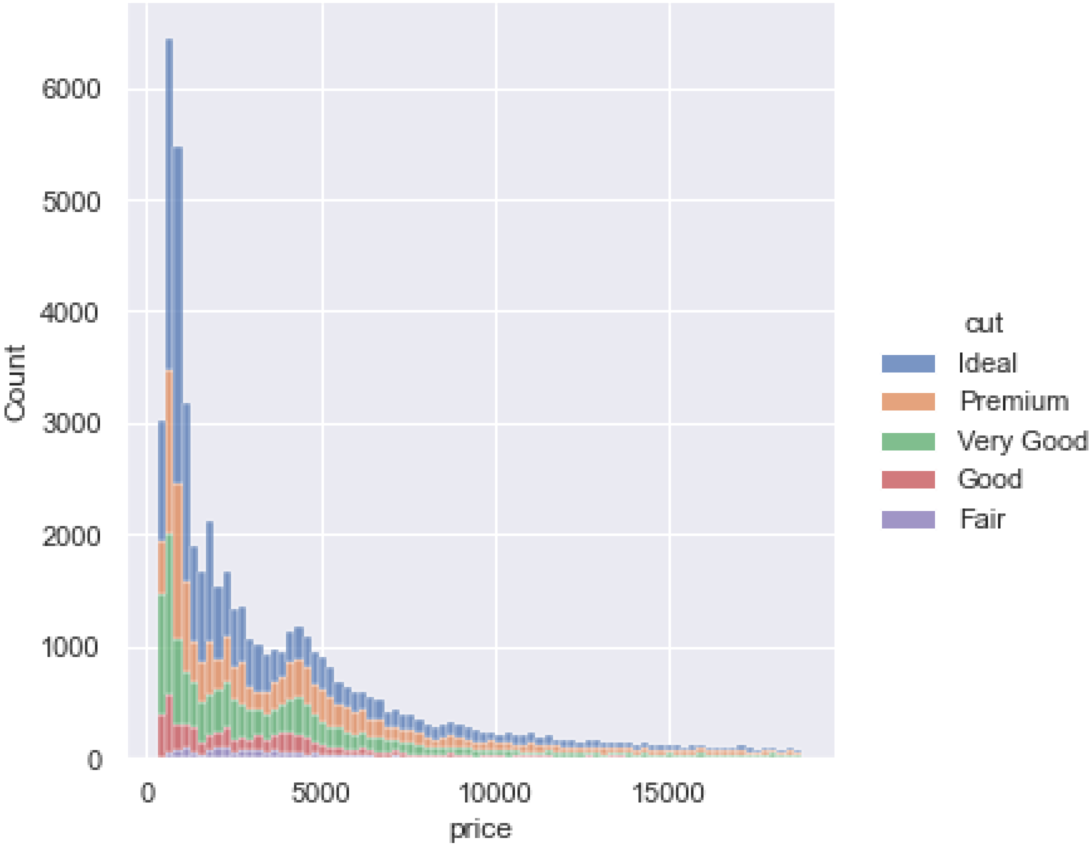
[7]:

```
# 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')
```



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

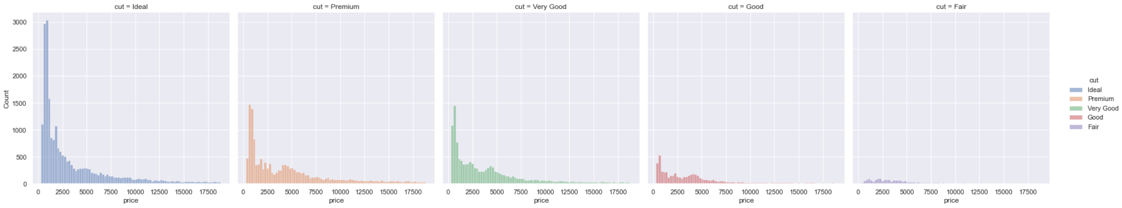
```
[8]: <seaborn.axisgrid.FacetGrid at 0x11333e490>
```



0.6 Using the ‘kind’ kwarg

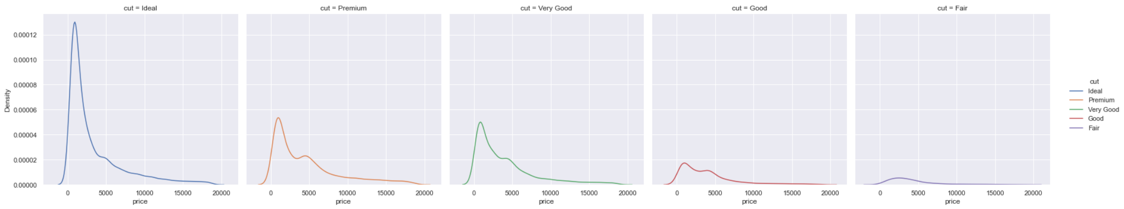
```
[9]: sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'hist')
```

```
[9]: <seaborn.axisgrid.FacetGrid at 0x128c81d30>
```



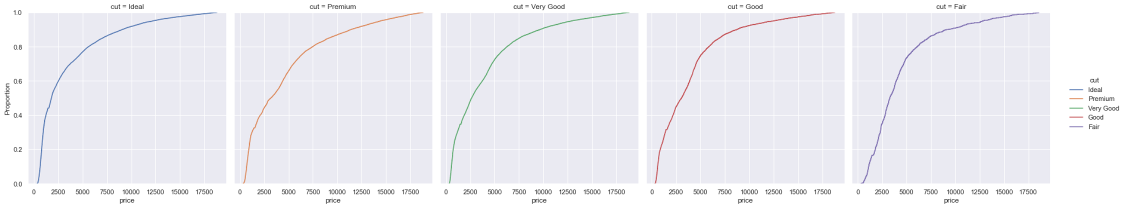
```
[10]: sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'kde')

      # kernel density estimation
```

```
[10]: <seaborn.axisgrid.FacetGrid at 0x129387040>
```



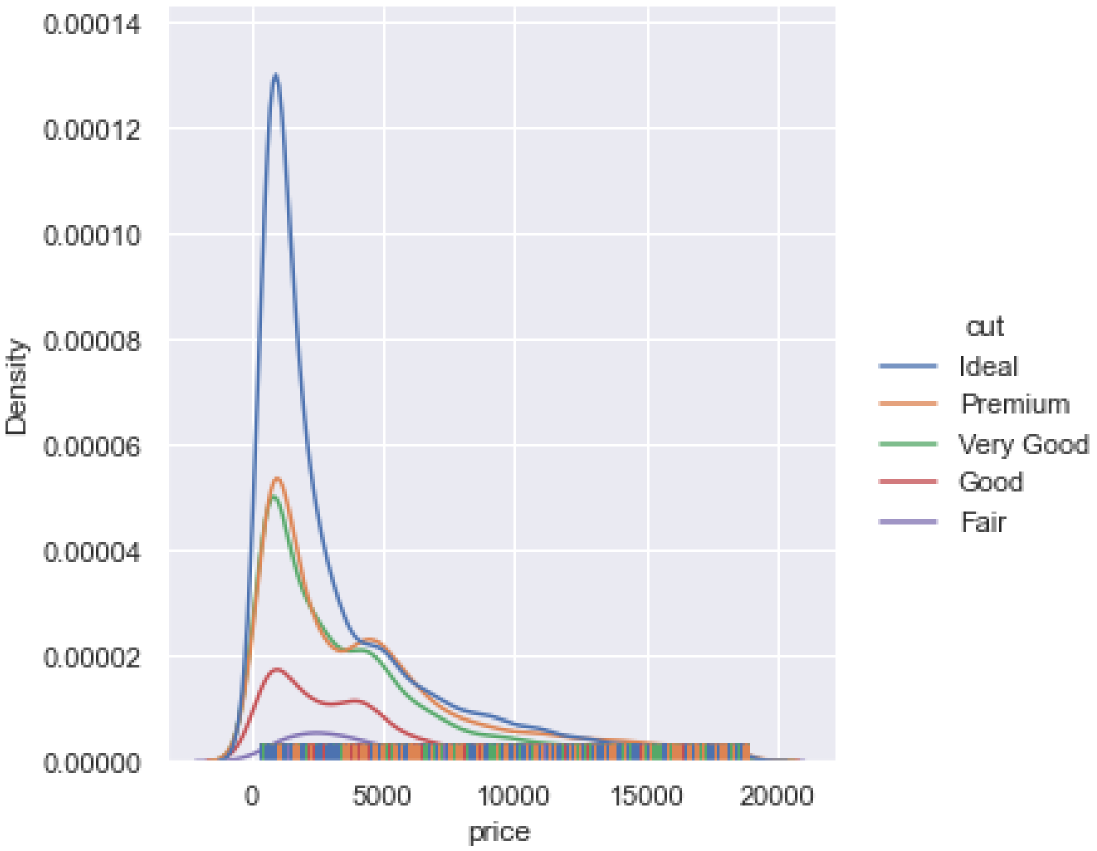
```
[11]: sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'ecdf')

      # empirical cumulative distribution functions
```

```
[11]: <seaborn.axisgrid.FacetGrid at 0x128f3a550>
```



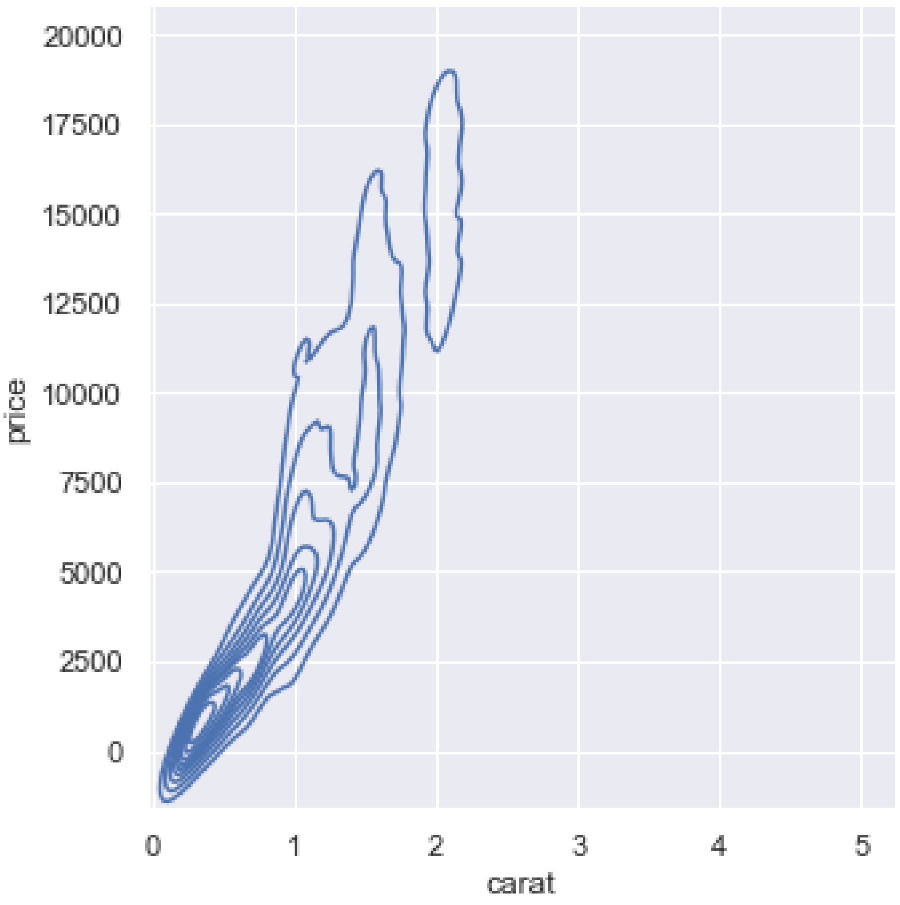
```
[12]: sns.displot(data=df, x="price", hue="cut", kind = 'kde', rug = True)
```

```
[12]: <seaborn.axisgrid.FacetGrid at 0x129cb8d90>
```



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

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

```
[13]: <seaborn.axisgrid.FacetGrid at 0x129ccf1f0>
```



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.

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

[14]:
```
   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
```

[15]:
```
tips.shape
```

[15]:
```
(244, 7)
```

[16]:
```
# Place scatterplot here
```

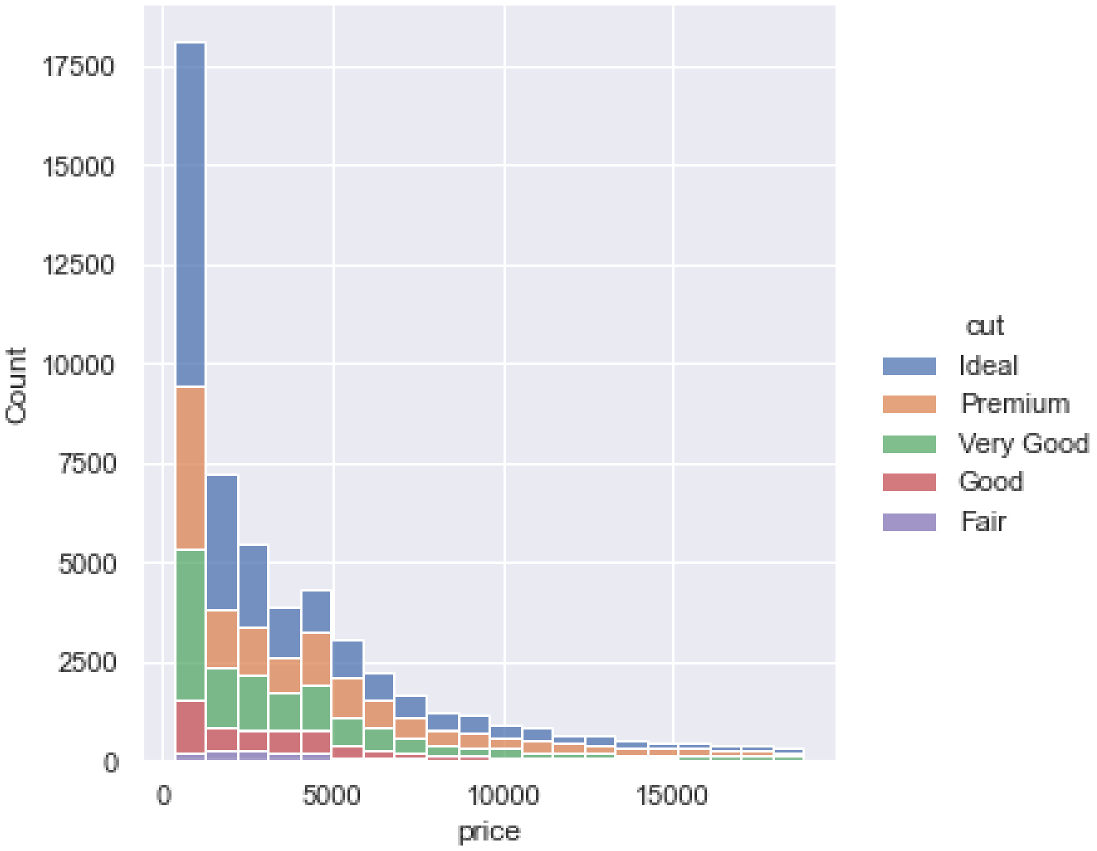
[ ]:
```

```

[17]:
```
# Place line chart here


#line aggregates data to the mean.  If each data point is required, use the␣
 ↪index
#sns.relplot(x=tips.index, y="tip", data=tips, kind = 'line', marker = 'h');
```

0.8 What's the difference?

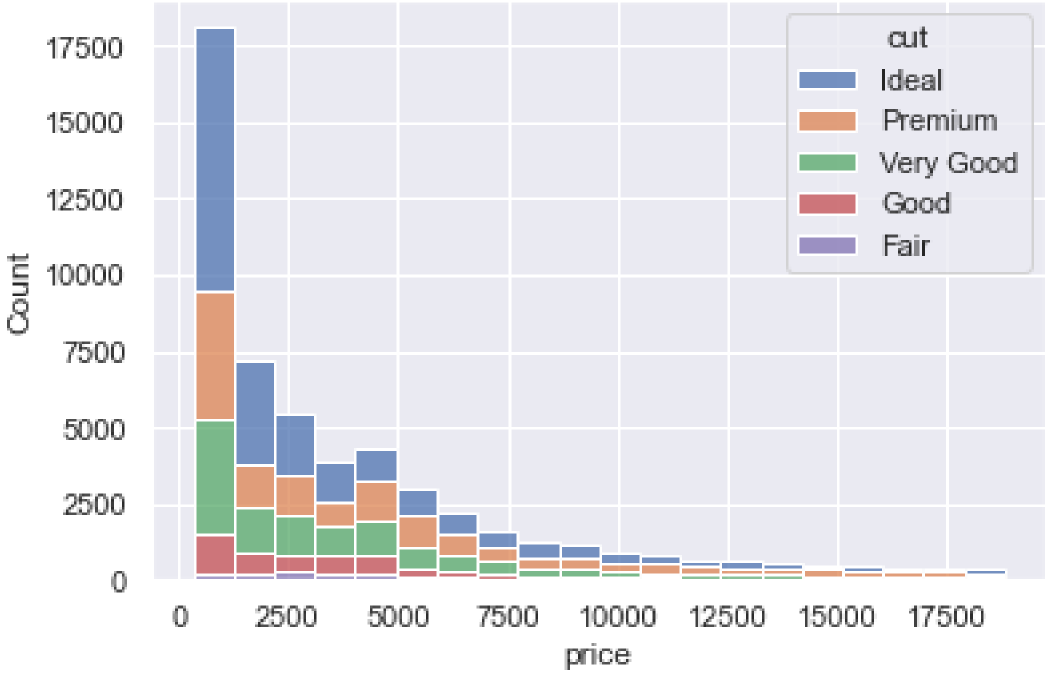
[18]:
```
sns.displot(data=df, x="price", hue="cut", multiple="stack", kind = 'hist',␣
 ↪bins = 20)
```

[18]:
```
<seaborn.axisgrid.FacetGrid at 0x129cc0af0>
```



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

```
[19]: <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

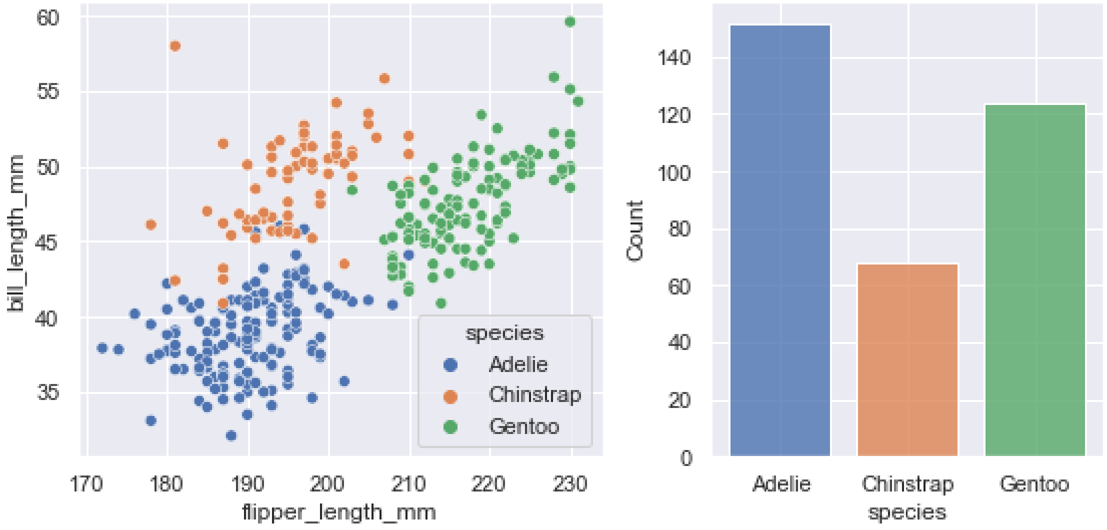
```
[20]: penguins.head()
```

```
[20]:   species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
      0  Adelie  Torgersen            39.1           18.7              181.0
      1  Adelie  Torgersen            39.5           17.4              186.0
      2  Adelie  Torgersen            40.3           18.0              195.0
      3  Adelie  Torgersen             NaN            NaN                NaN
      4  Adelie  Torgersen            36.7           19.3              193.0

         body_mass_g     sex
      0       3750.0    Male
```

```
1       3800.0  Female
2       3250.0  Female
3          NaN     NaN
4       3450.0  Female
```

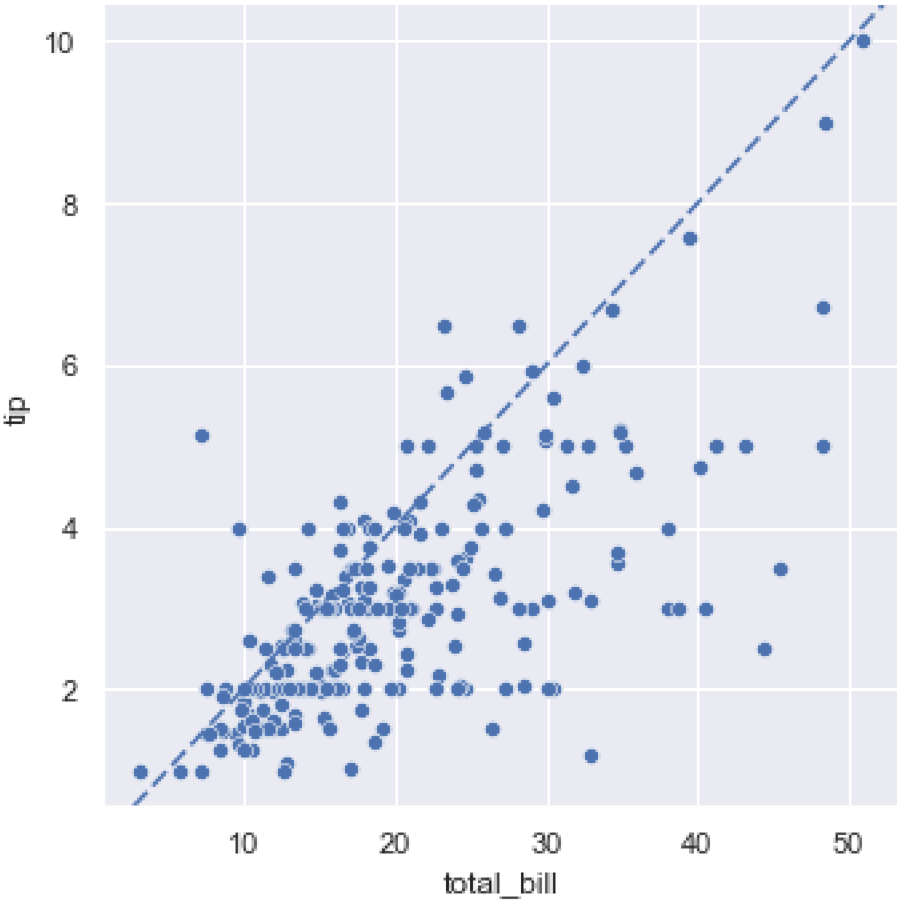
[21]:

```
# 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.
```



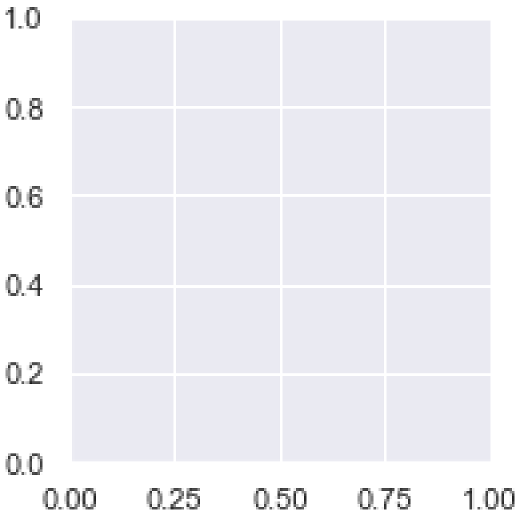
[22]:
```
# Example taken from Seaborn documentation
# Use tips dataset

g = sns.relplot(data=tips, x="total_bill", y="tip")
g.ax.axline(xy1=(10, 2), slope=.2, color="b", dashes=(5, 2))
```

[22]:
```
<matplotlib.lines._AxLine at 0x12bda6370>
```

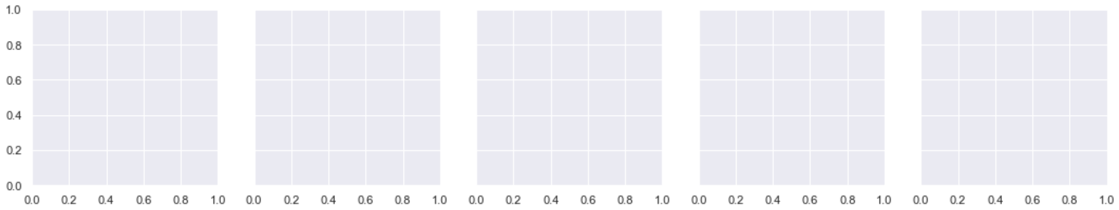


0.9 Facet Grid

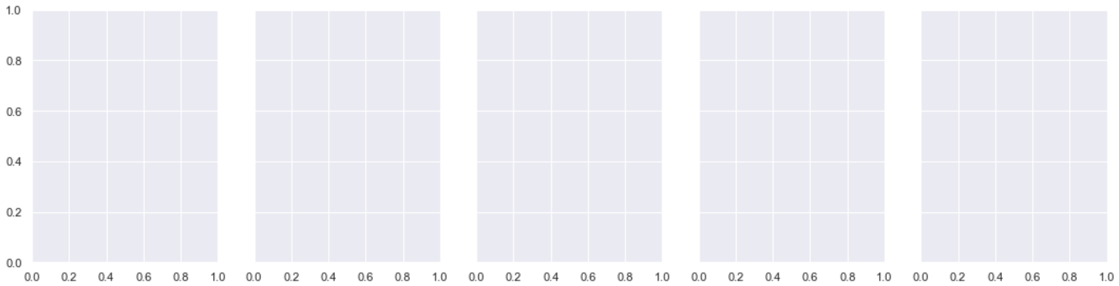
[23]:
```
p = sns.FacetGrid(df)
```



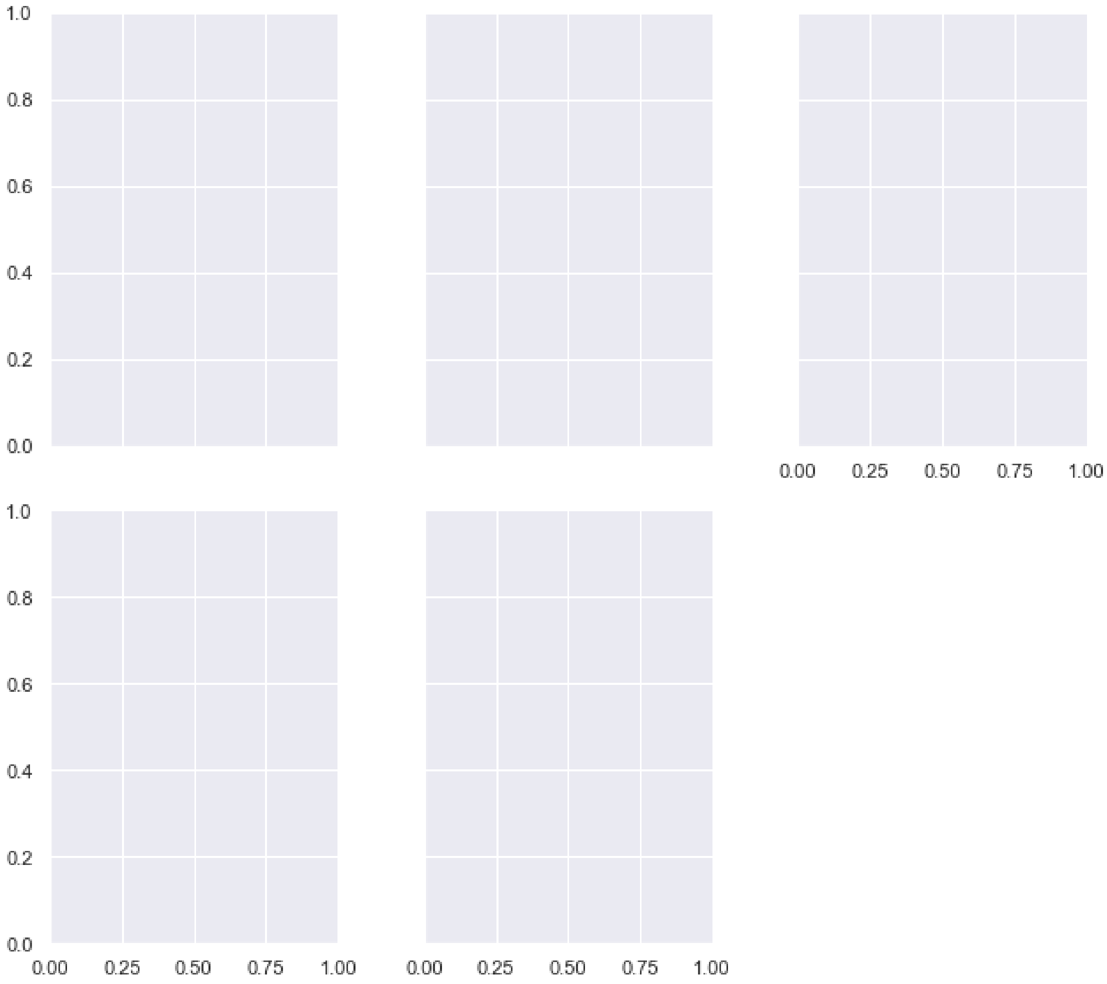
[24]:
```
p = sns.FacetGrid(df, col = 'cut')

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



[25]:
```
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.
```

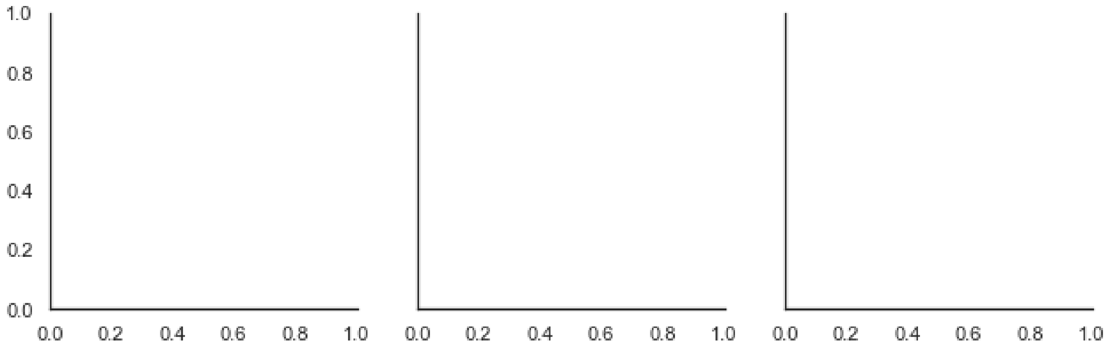


[26]:
```
p = sns.FacetGrid(df, col = 'cut', height = 4, aspect = 0.75, col_wrap = 3)
# Aspect ratio of each facet, so that aspect * height gives the width of each␣
↪facet.
```



[27]:
```
sns.set_style('white')
penguins = sns.load_dataset("penguins")
```

[28]:
```
p = sns.FacetGrid(penguins, col='island');
```



[29]:

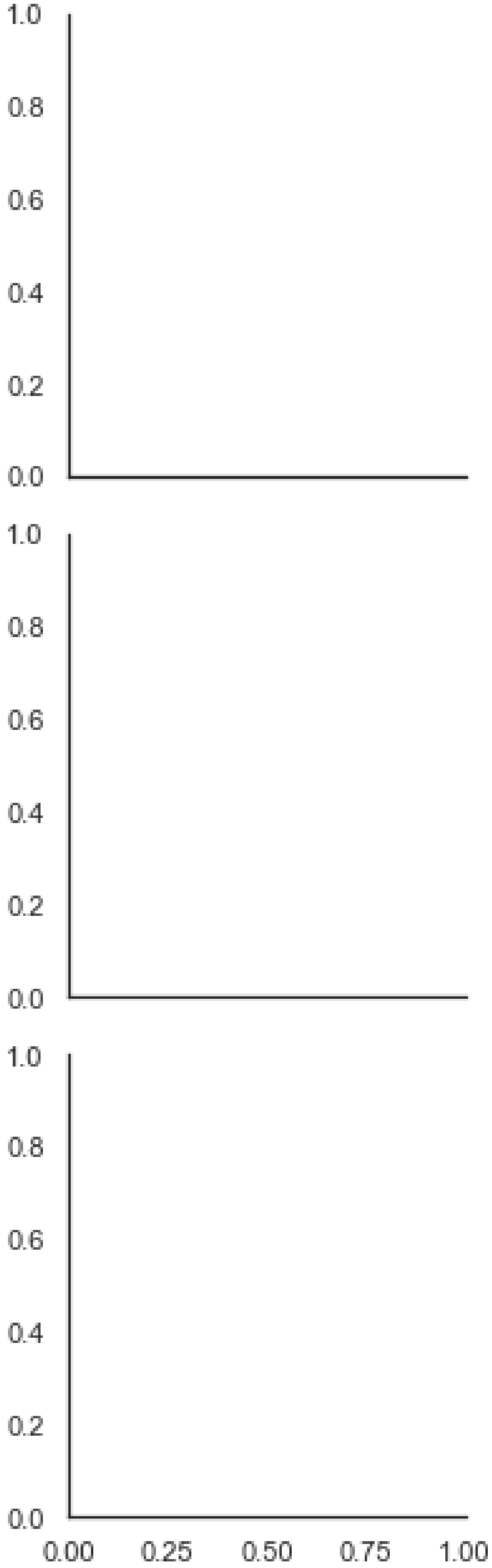
```
type(p)
```

[29]:

```
seaborn.axisgrid.FacetGrid
```

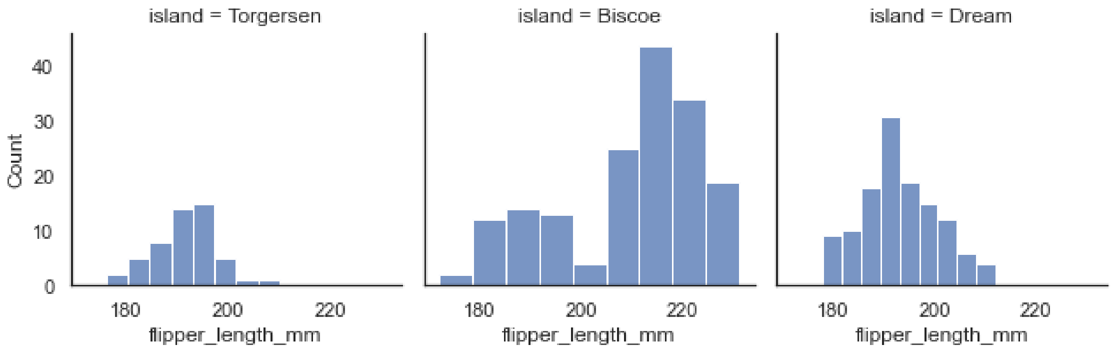
[30]:

```
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

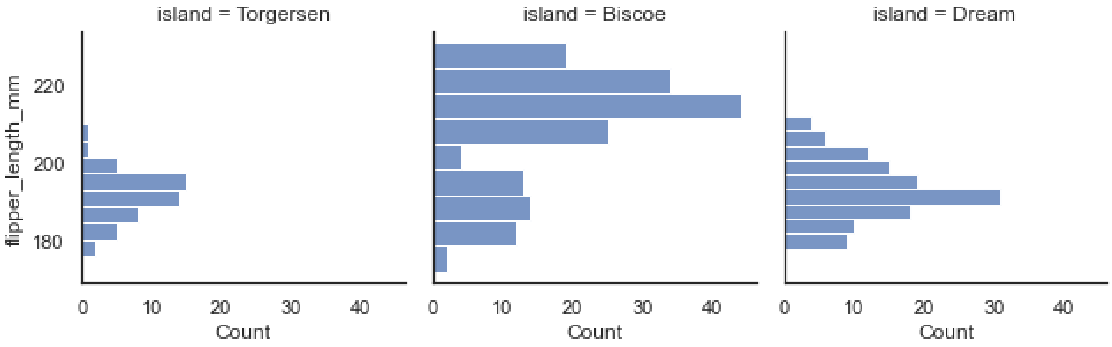
0.9.1 .map()

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



0.9.2 .map_dataframe()

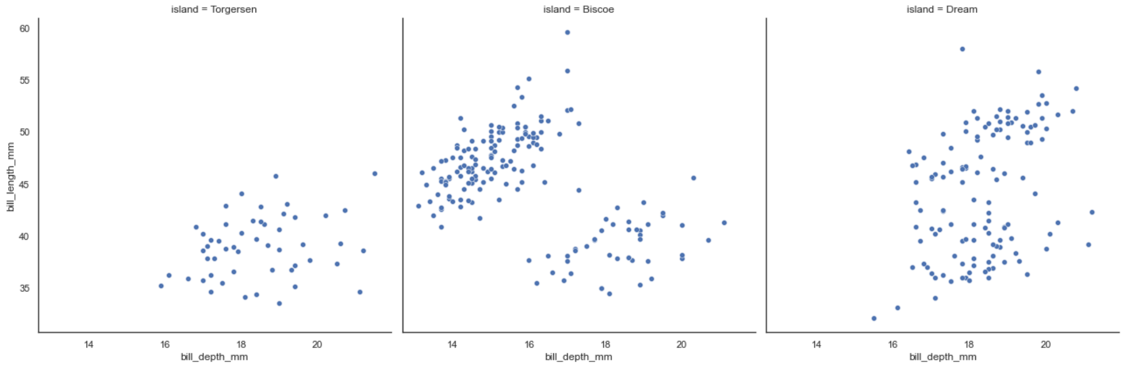
```
[32]: p = sns.FacetGrid(penguins, col='island')
      p.map_dataframe(sns.histplot, y='flipper_length_mm');
```



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

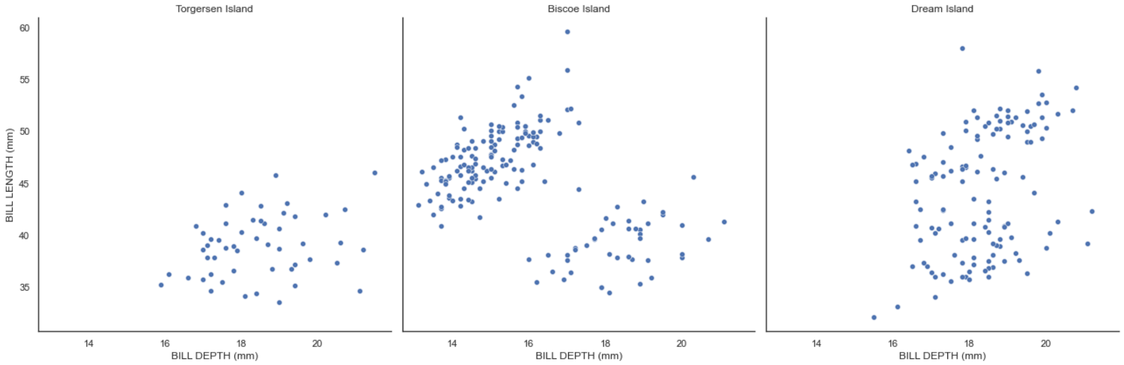
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm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

[34]:
```
p = sns.FacetGrid(penguins, col='island', height = 6, aspect =1)
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm');
```



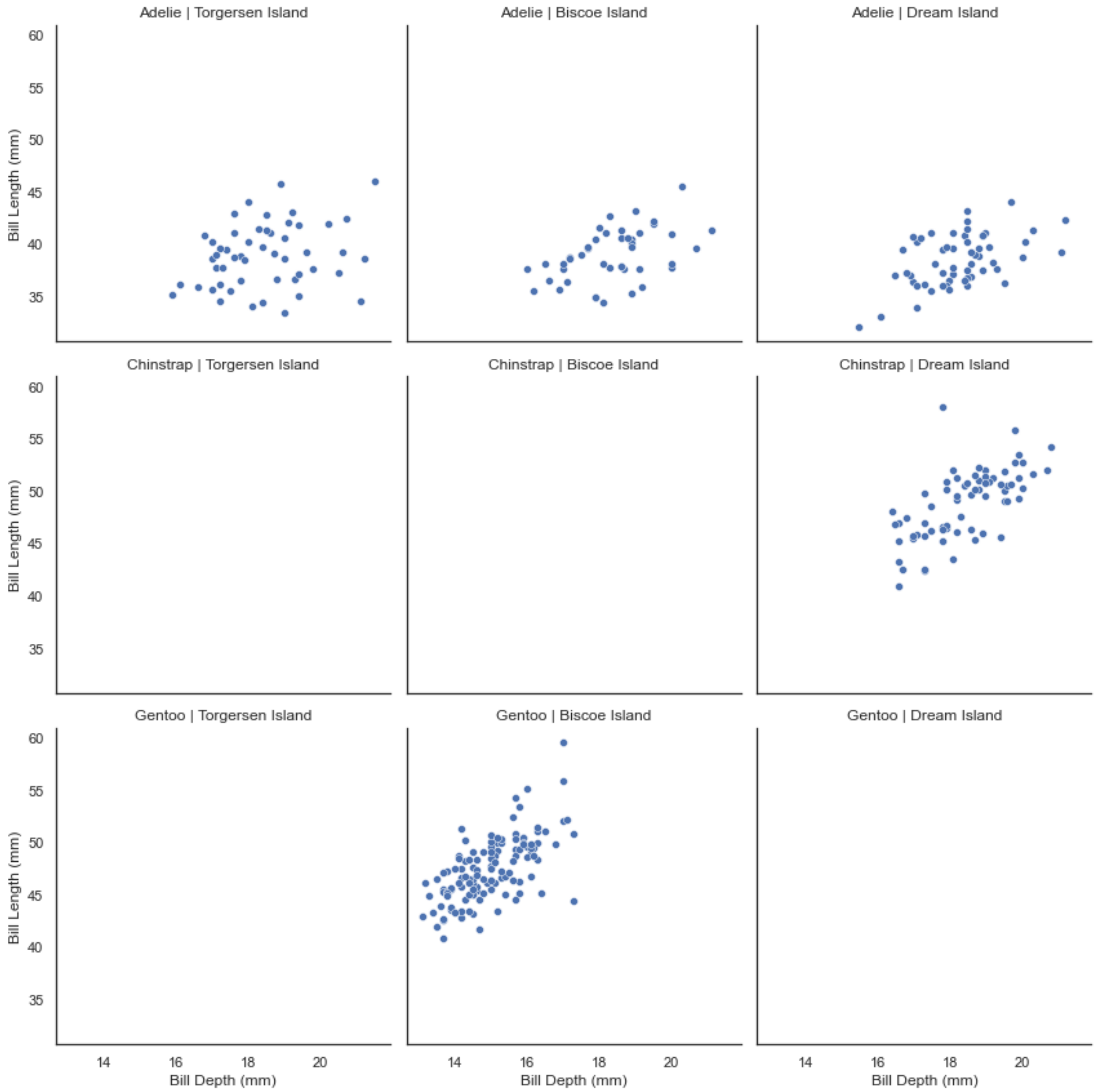
0.9.3 .set_axis_labels(), .set_titles(), sharey, ylim

[35]:
```
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
```



[36]:

```python
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');
```

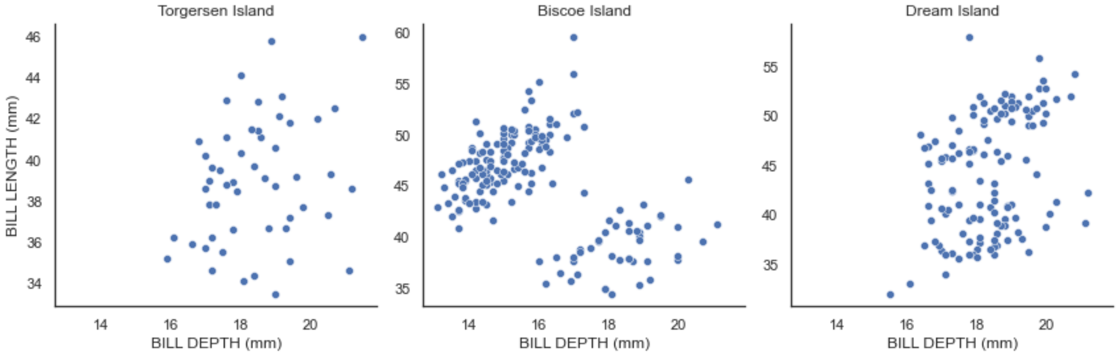


- sharey > False: the y-axis will not be shared and each plot will get its own y-axis.
- ylim > Sets a specified range for all y-axes shown

[37]:

```python
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)');
```

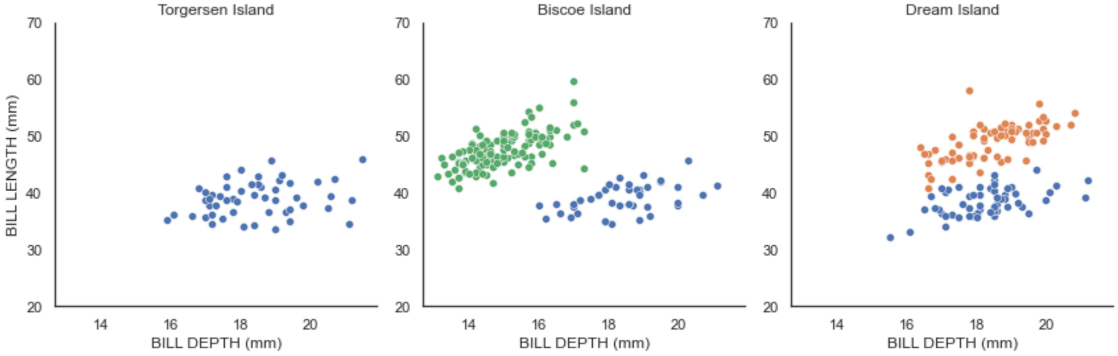
```
p.set_titles(col_template='{col_name} Island');
```



0.9.4 hue & pallette

[38]:

```
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');
```

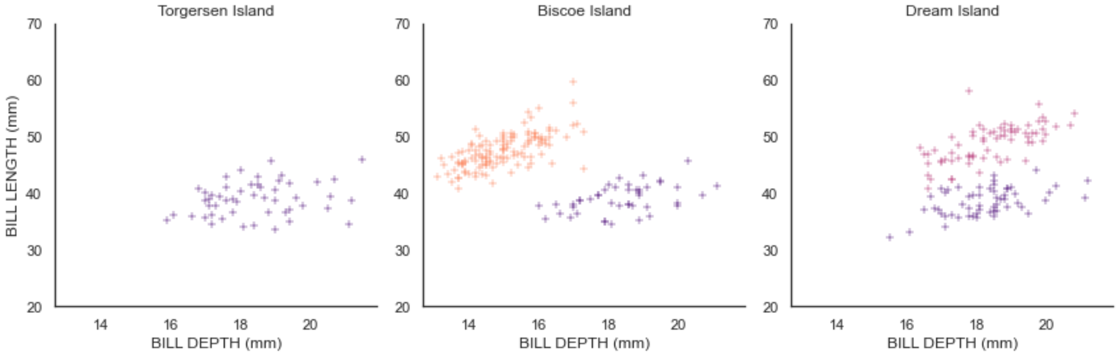


Note: If hue is placed inside the scatterplot

[39]:

```
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');
```

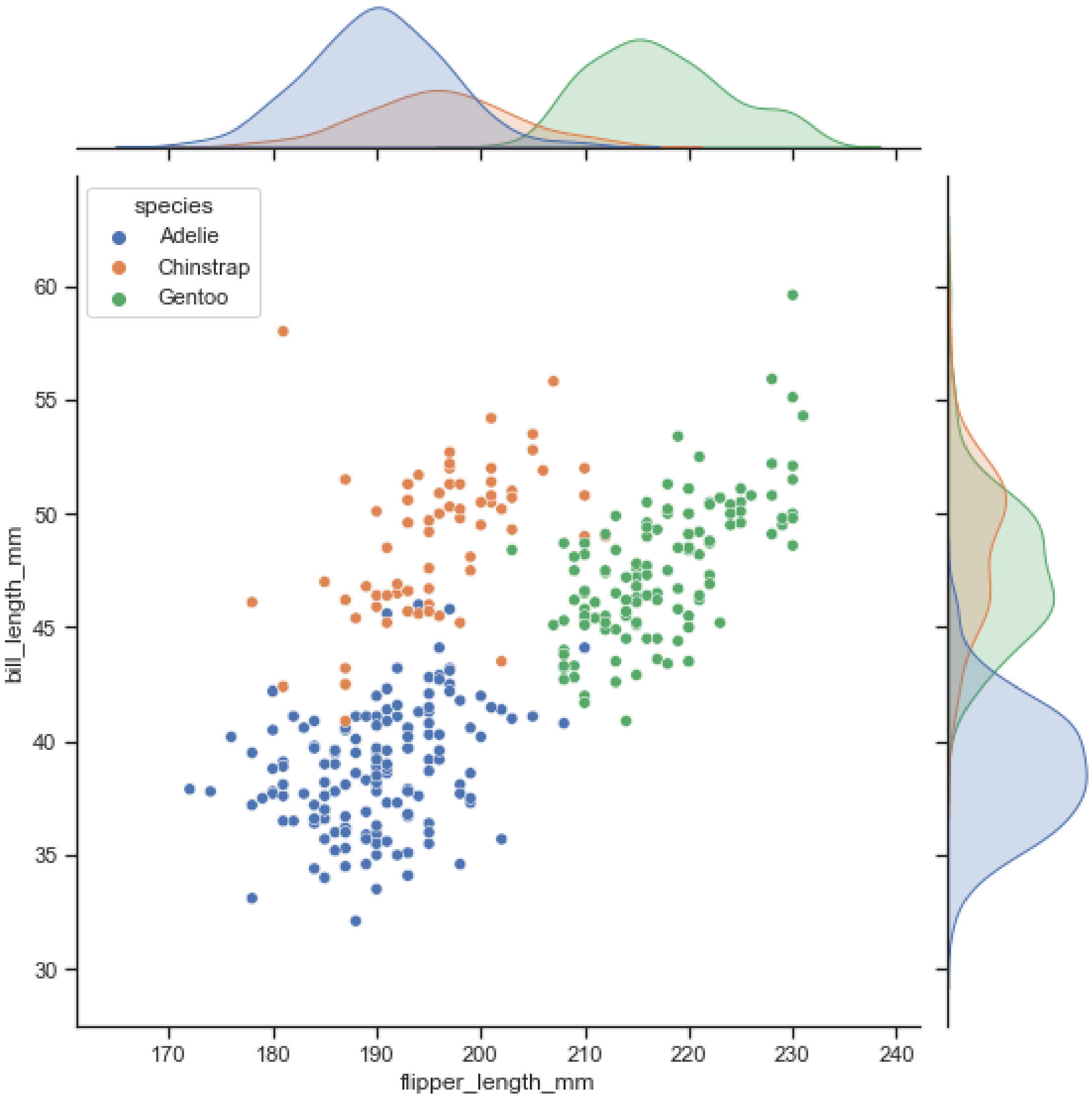


0.10 Multiple Views

0.10.1 Jointplot

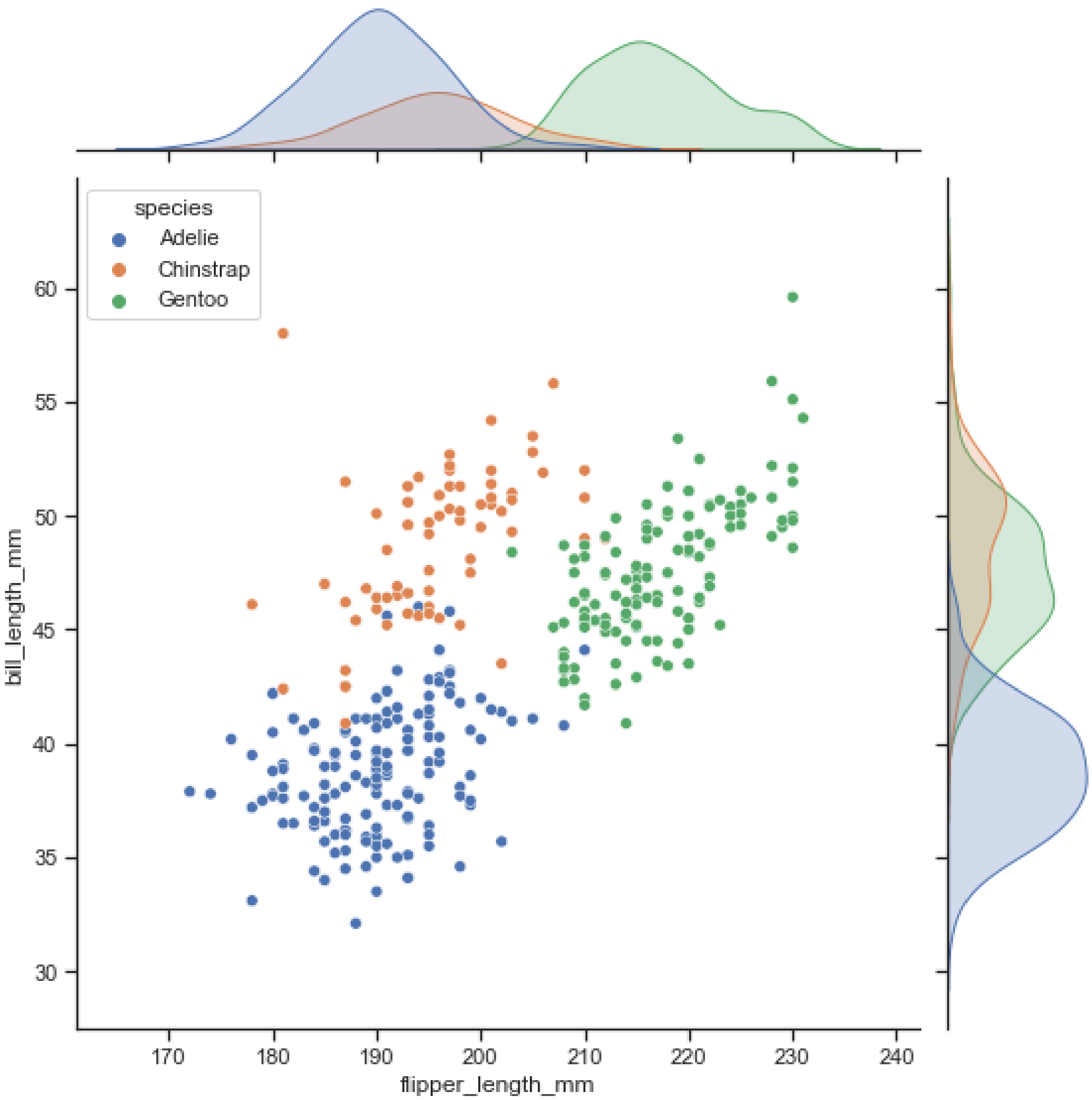
[40]:
```
sns.set_style("ticks")
sns.jointplot(data = penguins, x="flipper_length_mm", y="bill_length_mm", hue="species", height = 8 )
```

[40]:
```
<seaborn.axisgrid.JointGrid at 0x12c71a0d0>
```



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

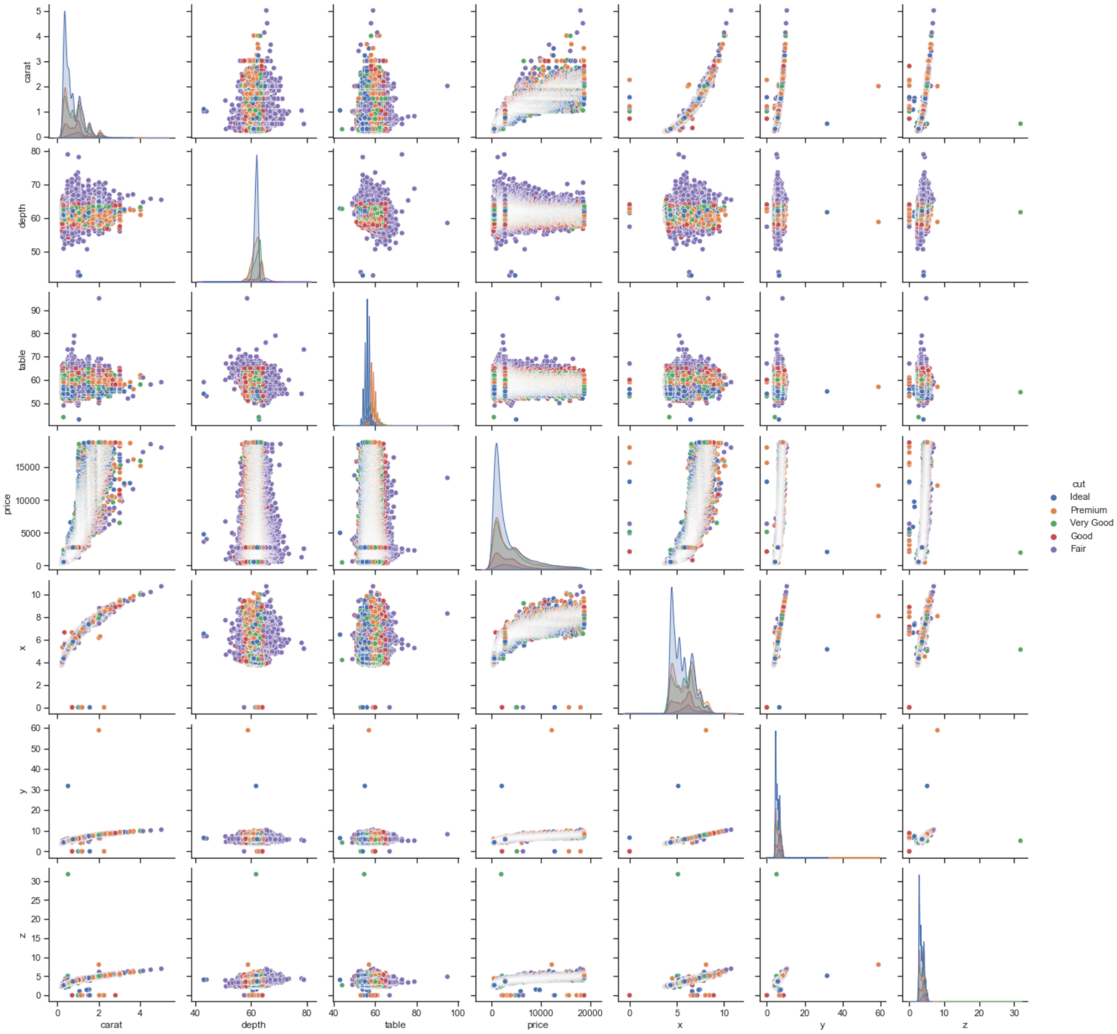
```
[41]: <seaborn.axisgrid.JointGrid at 0x12c919070>
```



0.10.2 Pairplot

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

```
[42]: <seaborn.axisgrid.PairGrid at 0x12ca33af0>
```

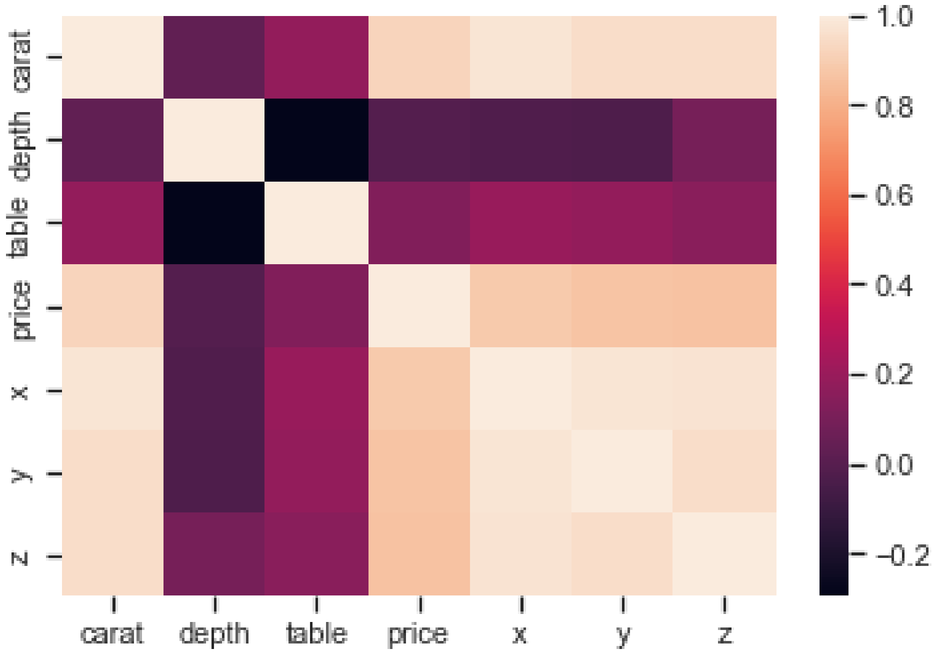


[43]:
```
xyz = df.corr()
xyz
```

[43]:
```
          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
```

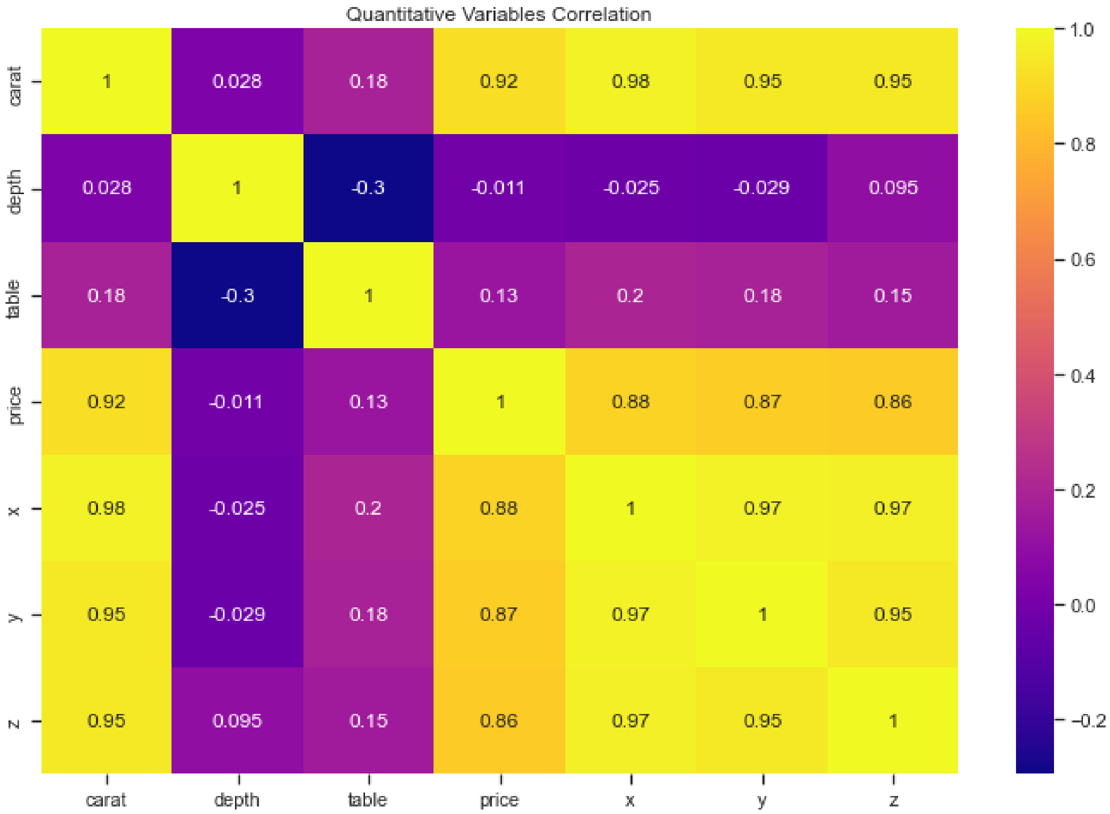
[44]:
```
sns.heatmap(xyz, annot=False)
```

[44]:
```
<AxesSubplot:>
```

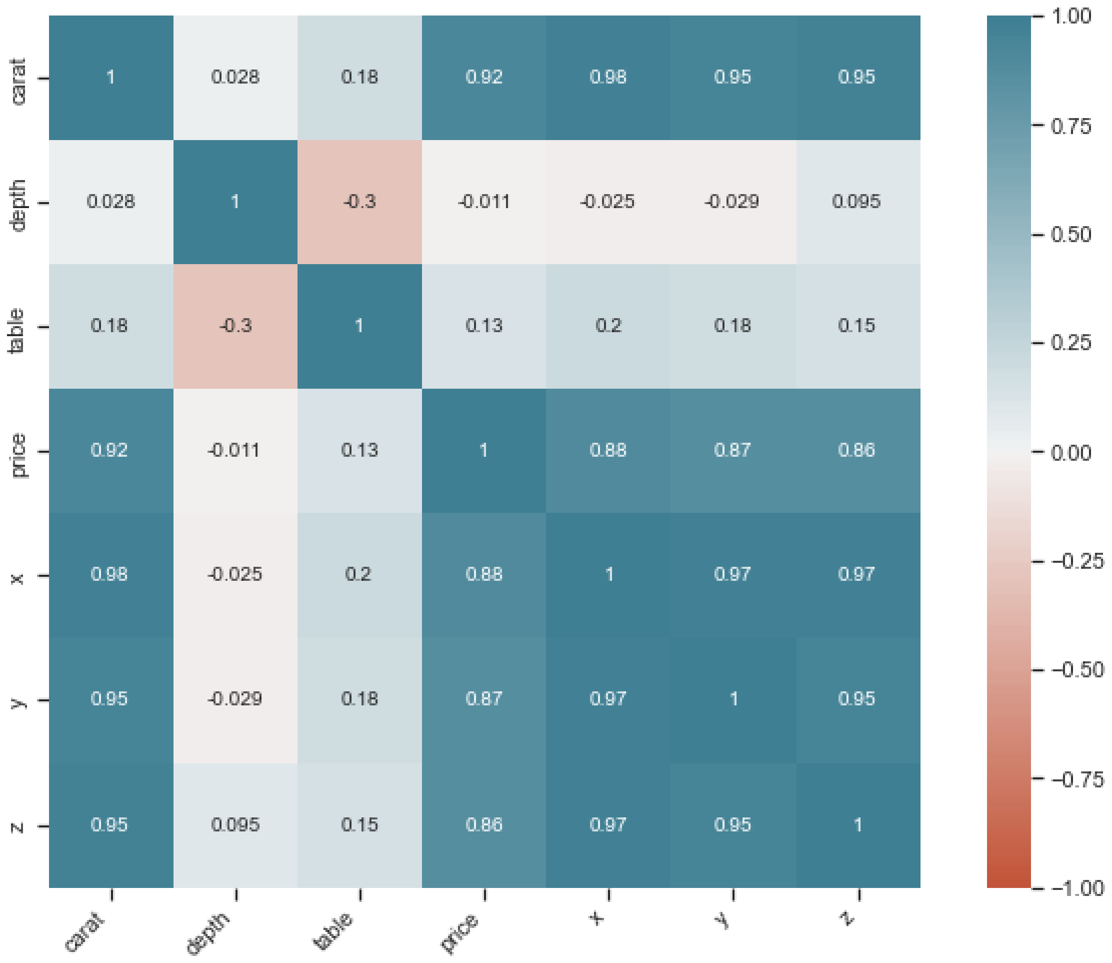


```
[45]: # Calculate correlations
      corr = df.corr()
      plt.figure(figsize=(12,8))
      plt.title('Quantitative Variables Correlation')

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

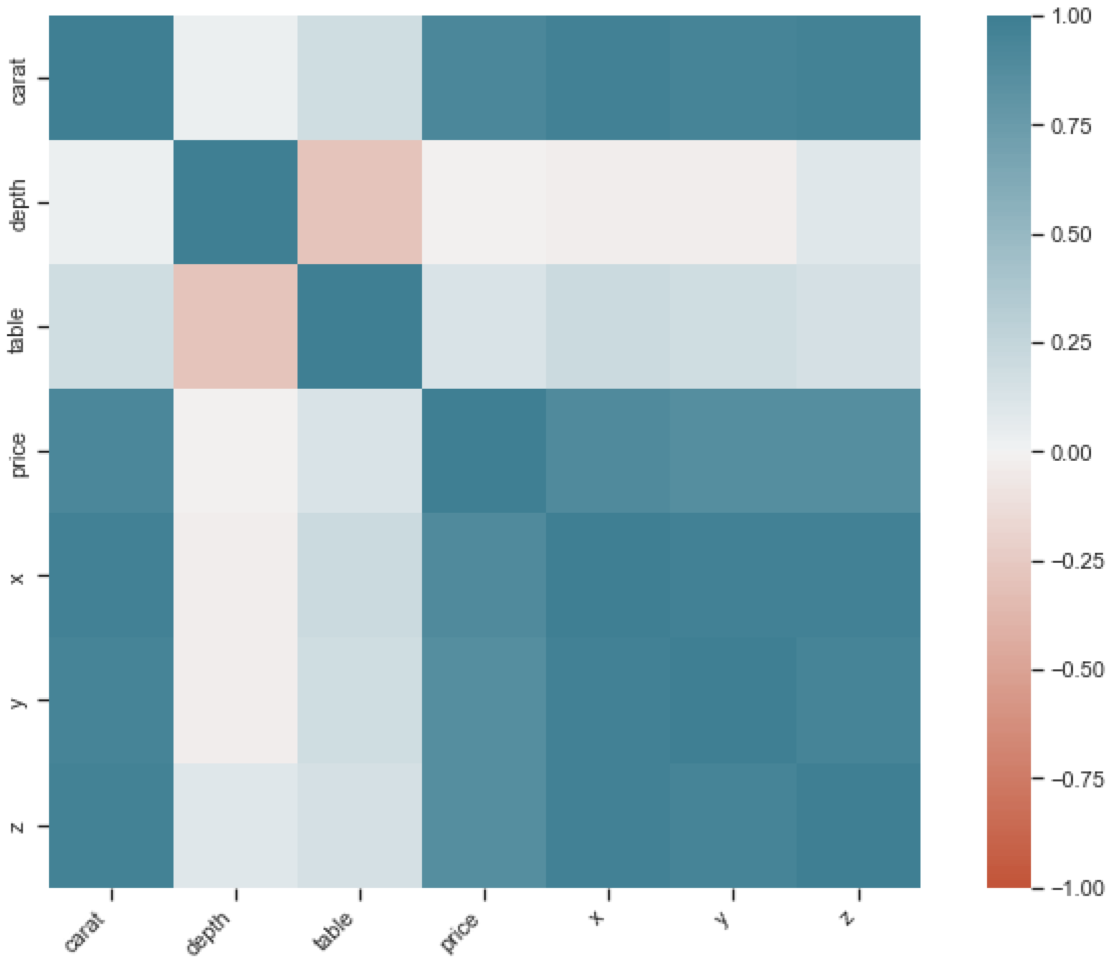
```
[45]: <AxesSubplot:title={'center':'Quantitative Variables Correlation'}>
```



```
[46]: 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'
      );
```



```
[47]: 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.

[48]:
```
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
```

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

```
[49]:     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
```

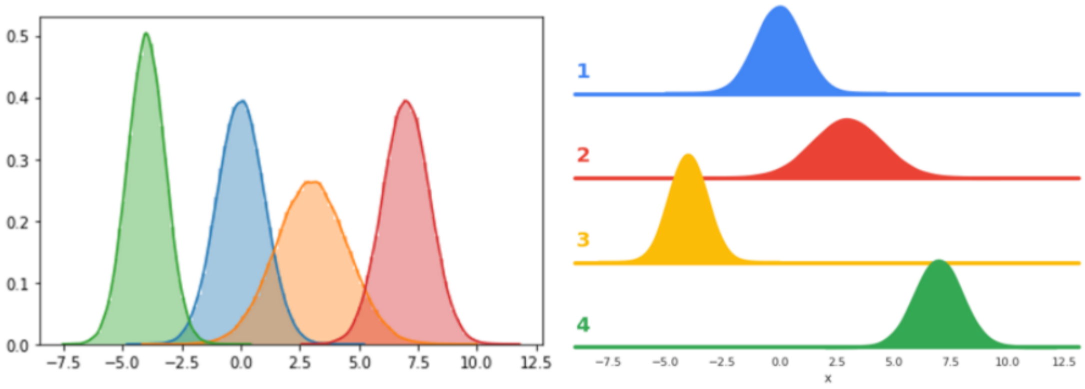
```
[50]: flights.shape
```

```
[50]: (144, 3)
```

```
[51]: # Place solution here
```

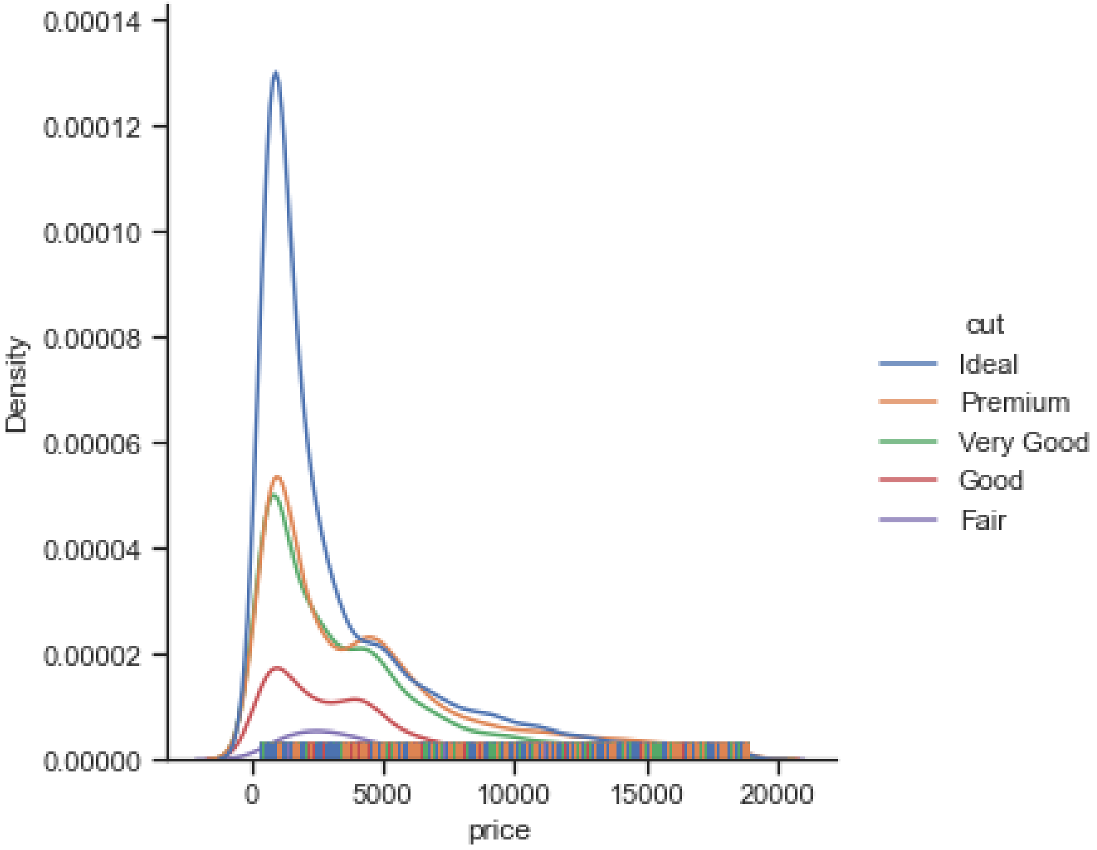
0.12 Seaborn Exercise 3 - 15 minutes

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.



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

```
[52]: <seaborn.axisgrid.FacetGrid at 0x12db30790>
```



```
[53]: # 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
```