Lecture 6-3

Visual exploration with Seaborn

Week 6 Friday

Miles Chen, PhD

References:

- https://seaborn.pydata.org/tutorial/function_overview.html
- https://seaborn.pydata.org/generated/seaborn.displot.html
- https://seaborn.pydata.org/api.html

In [1]:

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

Seaborn is for visual exploration

The primary purpose of seaborn is to make plots and visualize data.

You can use seaborn occassionally to fit a model (e.g. linear model or logistic regression model) to your data. But keep in mind that these are simply for visual exploration. You cannot 'extract' the model (e.g. regression coefficients) from Seaborn

In [2]:

```python
penguins = sns.load_dataset("penguins")
```

In [3]:

```python
penguins.head(40)
```

Out[3]:

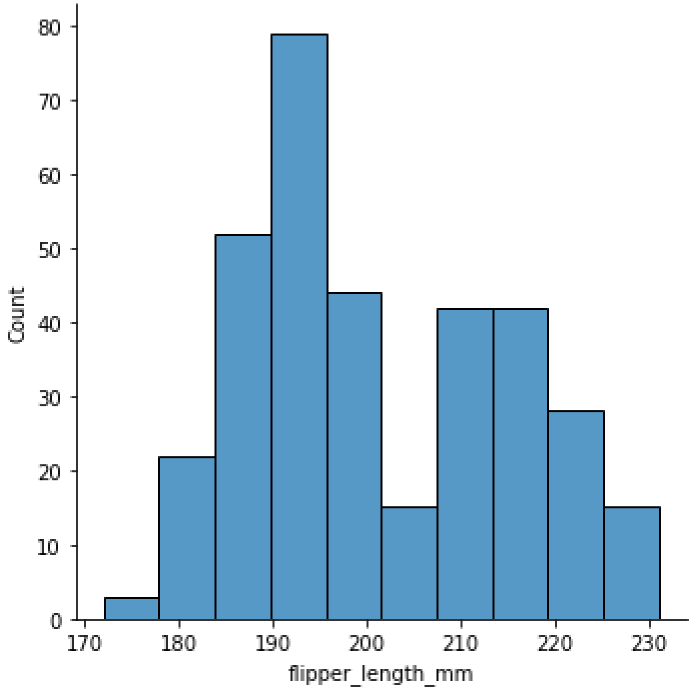
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	Male
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	Female
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	Male
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN
10	Adelie	Torgersen	37.8	17.1	186.0	3300.0	NaN
11	Adelie	Torgersen	37.8	17.3	180.0	3700.0	NaN
12	Adelie	Torgersen	41.1	17.6	182.0	3200.0	Female
13	Adelie	Torgersen	38.6	21.2	191.0	3800.0	Male
14	Adelie	Torgersen	34.6	21.1	198.0	4400.0	Male
15	Adelie	Torgersen	36.6	17.8	185.0	3700.0	Female
16	Adelie	Torgersen	38.7	19.0	195.0	3450.0	Female
17	Adelie	Torgersen	42.5	20.7	197.0	4500.0	Male
18	Adelie	Torgersen	34.4	18.4	184.0	3325.0	Female
19	Adelie	Torgersen	46.0	21.5	194.0	4200.0	Male
20	Adelie	Biscoe	37.8	18.3	174.0	3400.0	Female
21	Adelie	Biscoe	37.7	18.7	180.0	3600.0	Male
22	Adelie	Biscoe	35.9	19.2	189.0	3800.0	Female
23	Adelie	Biscoe	38.2	18.1	185.0	3950.0	Male
24	Adelie	Biscoe	38.8	17.2	180.0	3800.0	Male
25	Adelie	Biscoe	35.3	18.9	187.0	3800.0	Female
26	Adelie	Biscoe	40.6	18.6	183.0	3550.0	Male

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
27	Adelie	Biscoe	40.5	17.9	187.0	3200.0	Female
28	Adelie	Biscoe	37.9	18.6	172.0	3150.0	Female
29	Adelie	Biscoe	40.5	18.9	180.0	3950.0	Male
30	Adelie	Dream	39.5	16.7	178.0	3250.0	Female
31	Adelie	Dream	37.2	18.1	178.0	3900.0	Male
32	Adelie	Dream	39.5	17.8	188.0	3300.0	Female
33	Adelie	Dream	40.9	18.9	184.0	3900.0	Male
34	Adelie	Dream	36.4	17.0	195.0	3325.0	Female
35	Adelie	Dream	39.2	21.1	196.0	4150.0	Male
36	Adelie	Dream	38.8	20.0	190.0	3950.0	Male
37	Adelie	Dream	42.2	18.5	180.0	3550.0	Female
38	Adelie	Dream	37.6	19.3	181.0	3300.0	Female
39	Adelie	Dream	39.8	19.1	184.0	4650.0	Male

Univariate exploration

In [4]:
```
sns.displot(data = penguins, x = "flipper_length_mm")
# specify the dataframe and which variable to plot
```

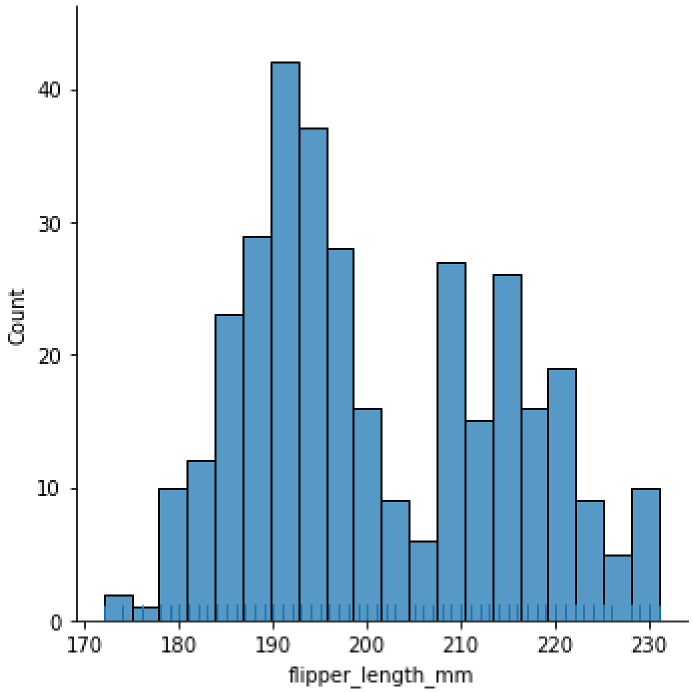
Out[4]: <seaborn.axisgrid.FacetGrid at 0x24a11df8dc8>



In [5]:

```
sns.displot(data = penguins, x = "flipper_length_mm", bins = 20, rug=True)
# use bins to specify bins
# use rug to add a rug plot
```

Out[5]: <seaborn.axisgrid.FacetGrid at 0x24a15096948>

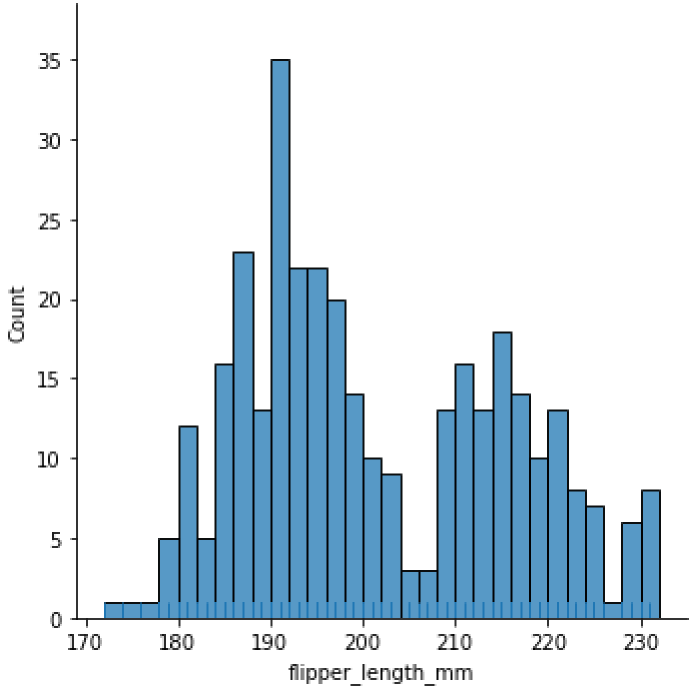


In [6]:

```
sns.displot(data = penguins, x = "flipper_length_mm", binwidth = 2, rug=True)
# specify binwidth
```

Out[6]:

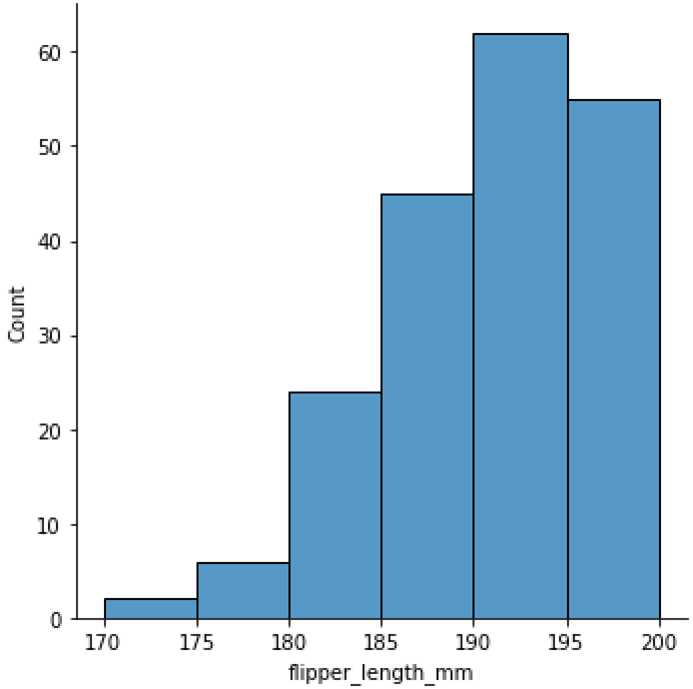
```
<seaborn.axisgrid.FacetGrid at 0x24a15219d48>
```



In [7]:

```
sns.displot(data = penguins, x = "flipper_length_mm", bins = [170, 175, 180, 185, 190, 195, 200])
# custom breakpoints
```

Out[7]: <seaborn.axisgrid.FacetGrid at 0x24a1530b788>

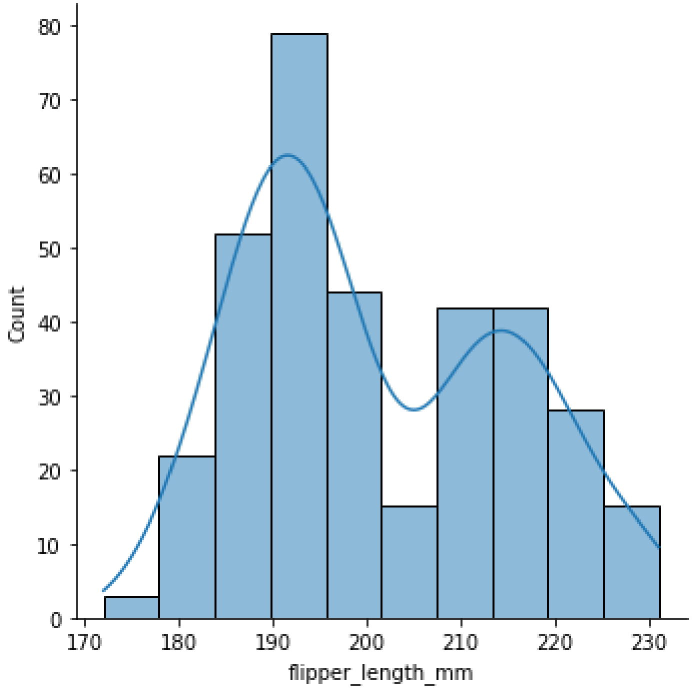


In [8]:

```python
sns.displot(data = penguins, x = "flipper_length_mm", kde = True)
# you can add a kernel density estimate curve
```

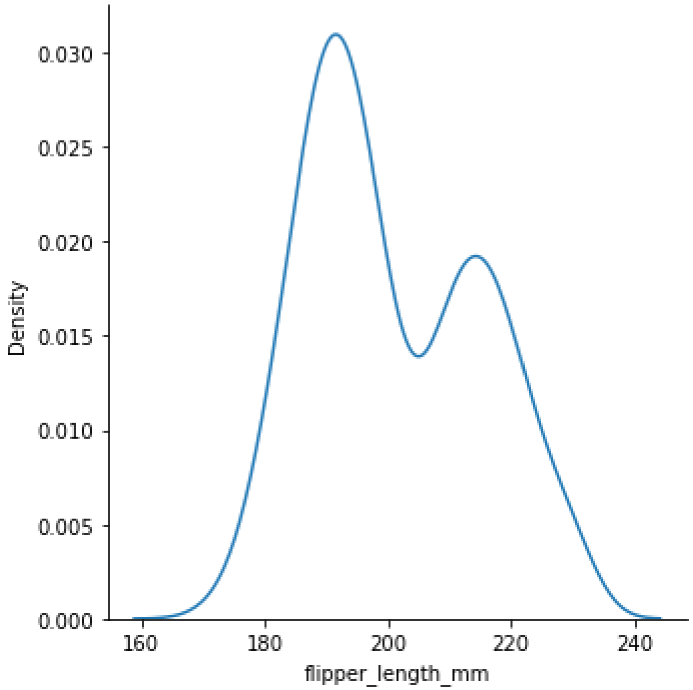
Out[8]:

```
<seaborn.axisgrid.FacetGrid at 0x24a1531fa88>
```



In [9]:
```
sns.displot(data = penguins, x = "flipper_length_mm", kind = "kde")
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x24a1543ca48>
```

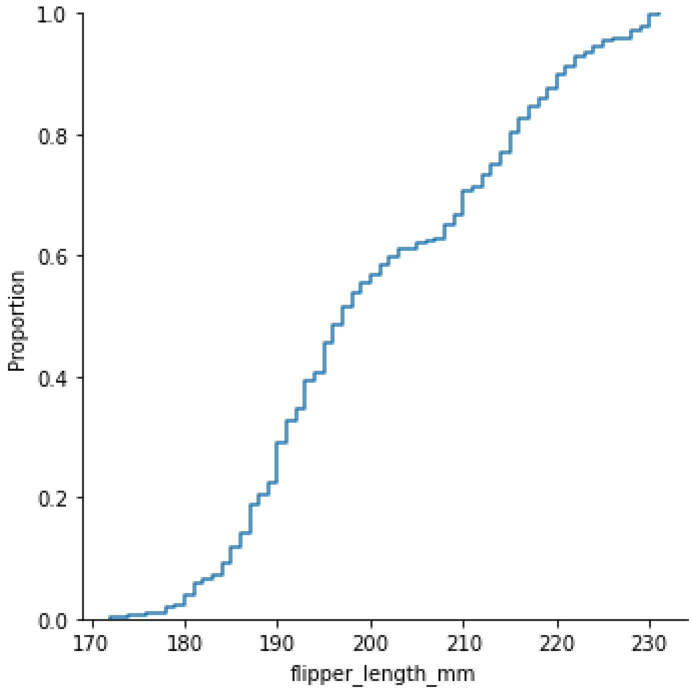


In [10]:

```
sns.displot(data = penguins, x = "flipper_length_mm", kind = "ecdf")
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x24a154c0448>
```



bivariate and multivariate plots

In [11]:

```
penguins.head(20)
```

Out[11]:

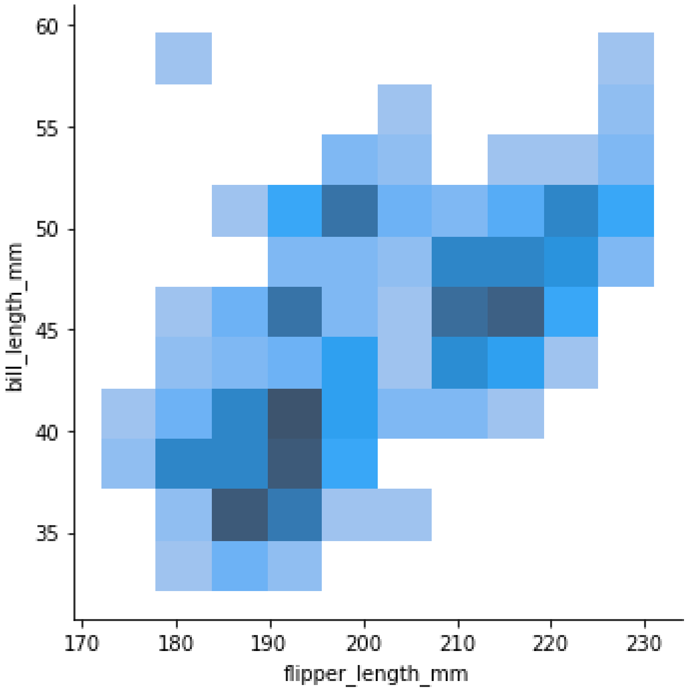
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	Male
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	Female
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	Male
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN
10	Adelie	Torgersen	37.8	17.1	186.0	3300.0	NaN
11	Adelie	Torgersen	37.8	17.3	180.0	3700.0	NaN
12	Adelie	Torgersen	41.1	17.6	182.0	3200.0	Female
13	Adelie	Torgersen	38.6	21.2	191.0	3800.0	Male
14	Adelie	Torgersen	34.6	21.1	198.0	4400.0	Male
15	Adelie	Torgersen	36.6	17.8	185.0	3700.0	Female
16	Adelie	Torgersen	38.7	19.0	195.0	3450.0	Female
17	Adelie	Torgersen	42.5	20.7	197.0	4500.0	Male
18	Adelie	Torgersen	34.4	18.4	184.0	3325.0	Female
19	Adelie	Torgersen	46.0	21.5	194.0	4200.0	Male

In [12]:

```
sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm")
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x24a15410e88>
```

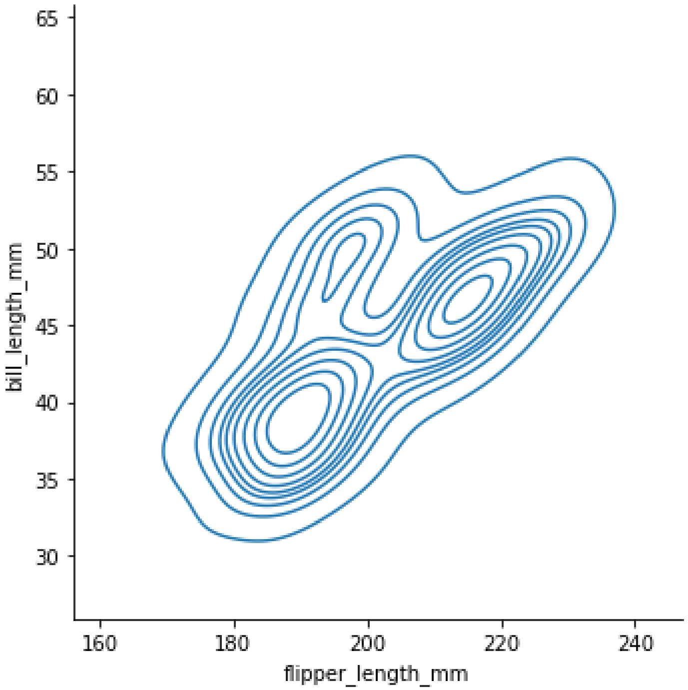


In [13]:

```
sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm", kind="kde")
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x24a154b0308>
```

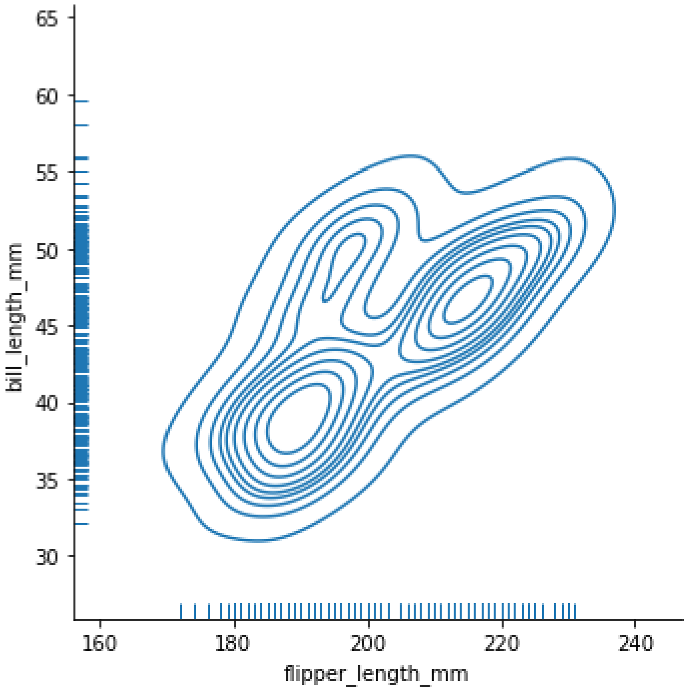


In [14]:

```
sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm", kind="kde", rug=True)
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x24a16cf8c48>
```

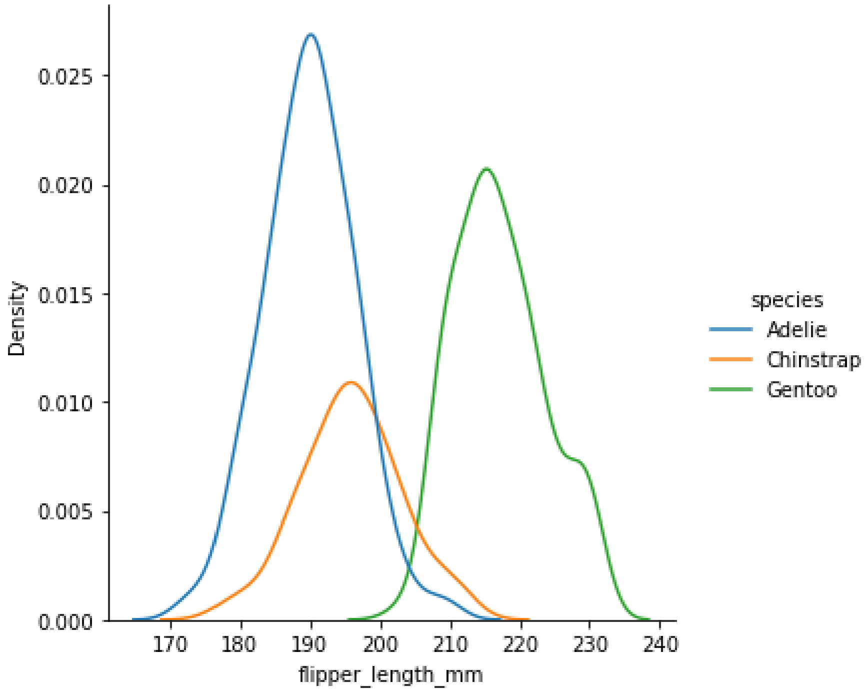


In [15]:

```
sns.displot(data=penguins, x="flipper_length_mm", hue="species", kind="kde")
```

Out[15]:

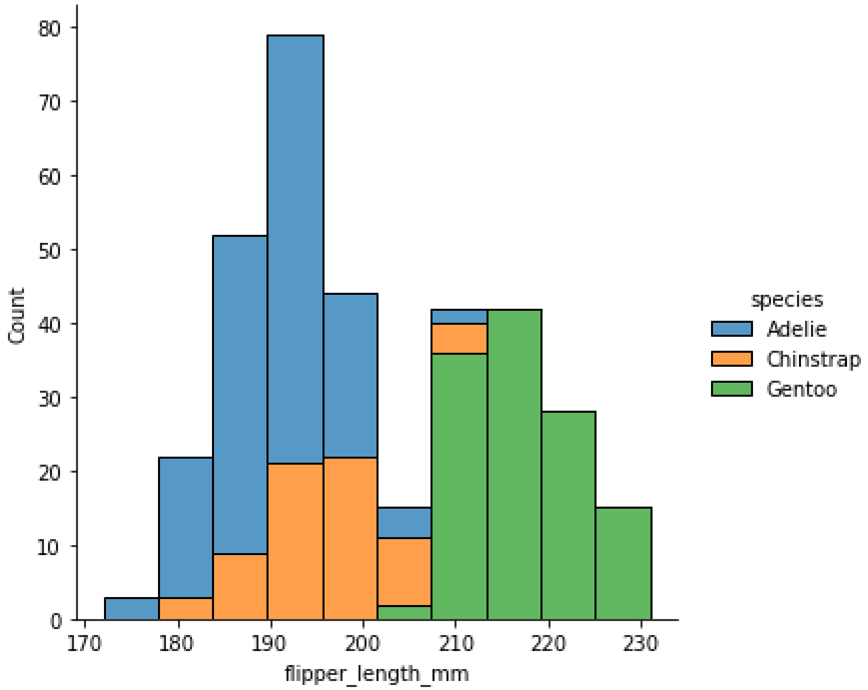
```
<seaborn.axisgrid.FacetGrid at 0x24a16ced988>
```



In [16]:

```
sns.displot(data=penguins, x="flipper_length_mm", hue="species", multiple="stack")
```

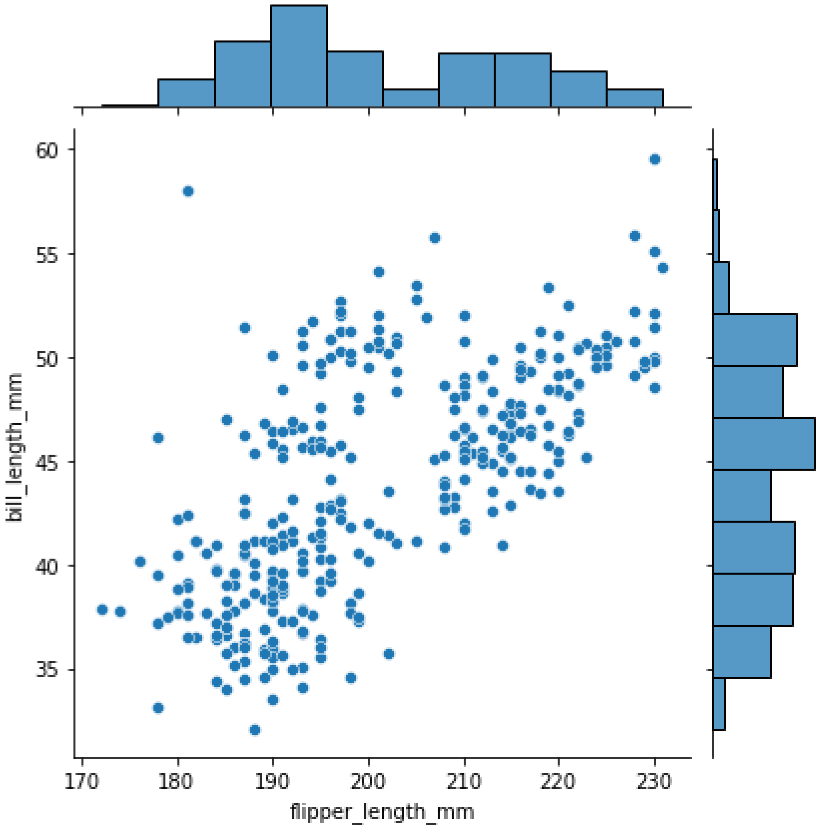
Out[16]: <seaborn.axisgrid.FacetGrid at 0x24a16e88b48>



In [17]:

```
sns.jointplot(data=penguins, x="flipper_length_mm", y="bill_length_mm")
```

Out[17]: <seaborn.axisgrid.JointGrid at 0x24a17083c48>

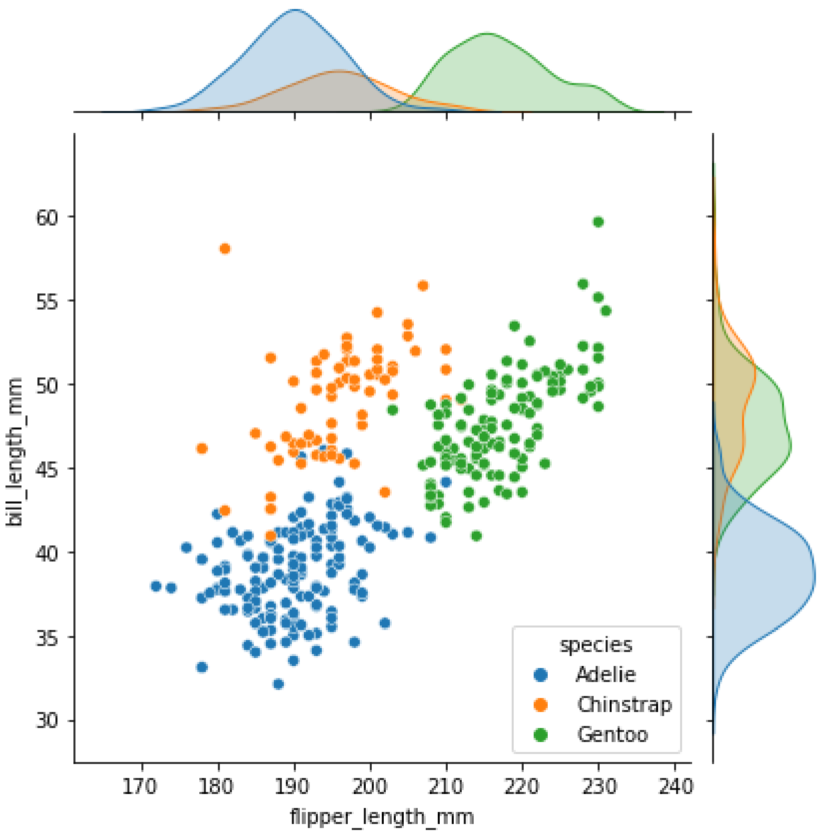


In [18]:

```
sns.jointplot(data=penguins, x="flipper_length_mm", y="bill_length_mm", hue = "species")
```

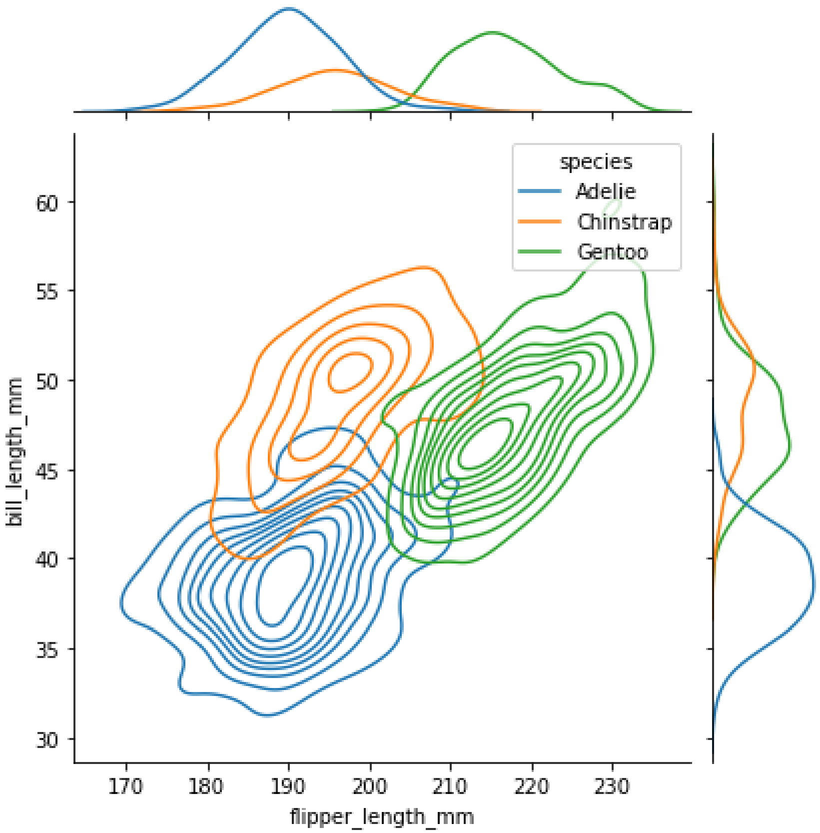
Out[18]:

```
<seaborn.axisgrid.JointGrid at 0x24a170256c8>
```



In [19]:
```
sns.jointplot(data=penguins, x="flipper_length_mm", y="bill_length_mm", hue = "species", kind = "
```

Out[19]: <seaborn.axisgrid.JointGrid at 0x24a17328148>



Pair-wise plots for multiple numeric data variables

In [ ]:

```
# good ol' iris data
iris = sns.load_dataset("iris")
sns.pairplot(iris)
```

In [ ]:

```
sns.pairplot(iris, hue="species")
```

Univariate plots separated by category

In [ ]:

```
tips = sns.load_dataset("tips")
tips.head(10)
# tips data, contains numeric vars: total_bill, tip, size
# categorical vars: sex, smoker, day, time
```

In [ ]:

```
tips.info()
```

In [ ]:

```
sns.stripplot(x="day", y="total_bill", data=tips, jitter = False)
# like a dotplot, but the dots are plotted on top of each other
```

In [ ]:

```
sns.stripplot(x="day", y="total_bill", data=tips, jitter=True)
# adding jitter allows us to see where points were overlapping
```

In [ ]:

```
sns.swarmplot(x="day", y="total_bill", data=tips)
# swarmplots are like symmetric dotplots
```

In [ ]:

```
sns.swarmplot(x="day", y="total_bill", hue="sex", data=tips)
# you can change the color of the point based on another categorical variable
# seaborn automatically adds a legend
```

In [ ]:

```
sns.swarmplot(x="sex", y="total_bill", hue="day", data=tips)
# this plot is harder on the eyes, but contains the same info as above
# the data is separated based on sex and colored based on the day
# but the colors aren't helping me
```

In [ ]:

```
sns.boxplot(x="day", y="total_bill", data=tips) # boxplots
```

In [ ]:

```
sns.boxplot(y="day", x="total_bill", data=tips) # boxplots
```

In [ ]:

```
sns.boxplot(x="day", y="total_bill", hue="sex", data=tips) # boxplots
```

In [ ]:

```
sns.violinplot(x="day", y="total_bill", hue="sex", data=tips, split=True)
# i'm not a huge fan of these plots
```

facet grids to make several plots

In [ ]:

```
# specify the column or row to create a grid based on that variable
sns.displot(data=penguins, x="flipper_length_mm", hue="species", row="sex", kind="kde")
```

In [ ]:

```
sns.displot(data=penguins, x="flipper_length_mm", hue="species", row="sex", col = "island", kind=
```

In [ ]:

```
g = sns.FacetGrid(tips, col="sex", margin_titles=True)  # define the facet grid
# the facet grid will subset the data based on the categorical variable you provide it
g.map(sns.histplot, data = tips, x = "total_bill", bins=10)
# you then 'map' a plot command (e.g. sns.distplot, or plt.hist) to the facet grid
# be sure to pass the appropriate arguments
```

In [ ]:

```
g = sns.FacetGrid(tips, row = 'day', margin_titles=True)
g.map(plt.hist, "total_bill")
```

In [ ]:

```
g = sns.FacetGrid(tips, row = 'time', col='sex', margin_titles=True) # you can even have a 2D fac
g.map(plt.hist, "total_bill", density = True, bins=10)
```

bar charts (count plots) for data that is only categorical

In [ ]:

```
sns.countplot(x="sex",data=tips)
```

In [ ]:

```
sns.countplot(x="sex",hue = 'time', data=tips)
```

fitting basic statistical models

In [ ]:

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

In [ ]:

```
sns.lmplot(x="total_bill", y="tip", data=tips, lowess=True)
```

In [ ]:

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

In [ ]:

```
sns.lmplot(x = "total_bill", y = "tip", hue = "smoker", data=tips)
```

In [ ]:

```

```