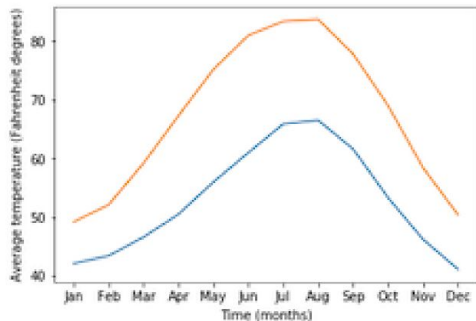


**Preparing your
figures to share with
others**

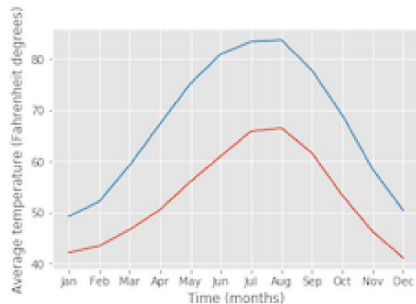
Changing plot style

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])
ax.set_xlabel("Time (months)")
ax.set_ylabel("Average temperature (Fahrenheit degrees)")
plt.show()
```



Choosing a style

```
plt.style.use("ggplot")  
fig, ax = plt.subplots()  
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])  
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])  
ax.set_xlabel("Time (months)")  
ax.set_ylabel("Average temperature (Fahrenheit degrees)")  
plt.show()
```



Back to the default

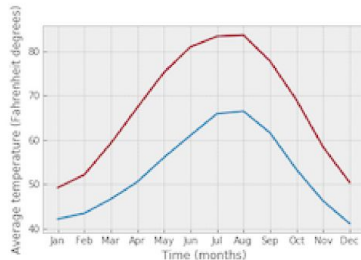
```
plt.style.use("default")
```

The available styles

[https://matplotlib.org/gallery/style_sheets/style_sheets_reference.h](https://matplotlib.org/gallery/style_sheets/style_sheets_reference.html)

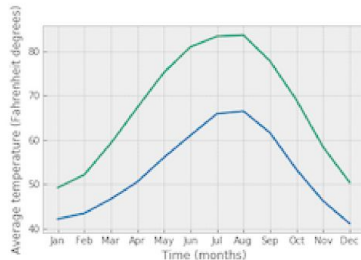
The "bmh" style

```
plt.style.use("bmh")  
fig, ax = plt.subplots()  
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])  
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])  
ax.set_xlabel("Time (months)")  
ax.set_ylabel("Average temperature (Fahrenheit degrees)")  
plt.show()
```



Seaborn styles

```
plt.style.use("seaborn-colorblind")  
fig, ax = plt.subplots()  
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])  
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])  
ax.set_xlabel("Time (months)")  
ax.set_ylabel("Average temperature (Fahrenheit degrees)")  
plt.show()
```



Guidelines for choosing plotting style

Guidelines for choosing plotting style

- Dark backgrounds are usually less visible

Guidelines for choosing plotting style

- Dark backgrounds are usually less visible
- If color is important, consider choosing colorblind-friendly options

Guidelines for choosing plotting style

- Dark backgrounds are usually less visible
- If color is important, consider choosing colorblind-friendly options
 - "seaborn-colorblind" or "tableau-colorblind10"

Guidelines for choosing plotting style

- Dark backgrounds are usually less visible
- If color is important, consider choosing colorblind-friendly options
 - "seaborn-colorblind" or "tableau-colorblind10"
- If you think that someone will want to print your figure, use less ink

Guidelines for choosing plotting style

- Dark backgrounds are usually less visible
- If color is important, consider choosing colorblind-friendly options
 - "seaborn-colorblind" or "tableau-colorblind10"
- If you think that someone will want to print your figure, use less ink
- If it will be printed in black-and-white, use the "grayscale" style

**Practice choosing
the right style for
you!**

**Sharing your
visualizations with
others**

A figure to share

```
fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"])
ax.set_xticklabels(medals.index, rotation=90)
ax.set_ylabel("Number of medals")

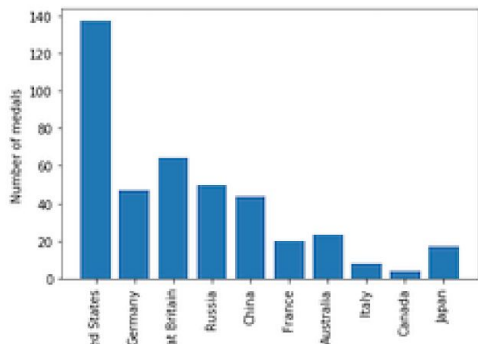
plt.show()
```


A figure to share

```
fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"])
ax.set_xticklabels(medals.index, rotation=90)
ax.set_ylabel("Number of medals")

plt.show()
```



Saving the figure to file

```
fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"])
ax.set_xticklabels(medals.index, rotation=90)
ax.set_ylabel("Number of medals")

fig.savefig("gold_medals.png")
```

Saving the figure to file

```
fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"])
ax.set_xticklabels(medals.index, rotation=90)
ax.set_ylabel("Number of medals")

fig.savefig("gold_medals.png")
```

```
ls
```

```
gold_medals.png
```

Different file formats

```
fig.savefig("gold_medals.jpg")
```

Different file formats

```
fig.savefig("gold_medals.jpg")
```

```
fig.savefig("gold_medals.jpg", quality=50)
```

Different file formats

```
fig.savefig("gold_medals.jpg")
```

```
fig.savefig("gold_medals.jpg", quality=50)
```

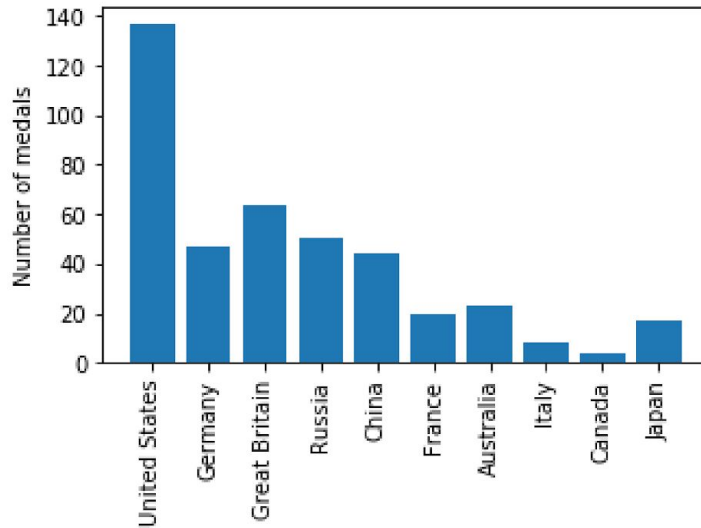
```
fig.savefig("gold_medals.svg")
```

Resolution

```
fig.savefig("gold_medals.png", dpi=300)
```

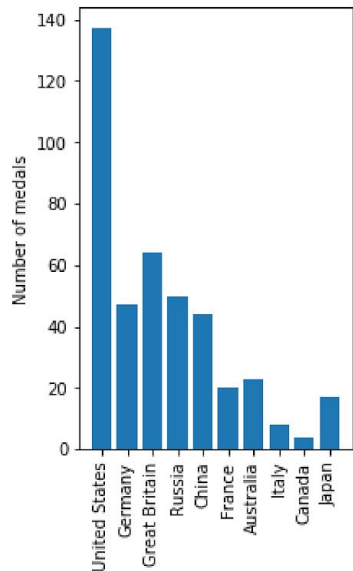
Size

```
fig.set_size_inches([5, 3])
```



Another aspect ratio

```
fig.set_size_inches([3, 5])
```



**Practice saving your
visualizations!**

Automating figures from data

Why automate?

- Ease and speed
- Flexibility
- Robustness
- Reproducibility

How many different kinds of data?

```
summer_2016_medals["Sport"]
```

How many different kinds of data?

```
summer_2016_medals["Sport"]
```

```
ID
62      Rowing
65      Taekwondo
73      Handball
...
134759   Handball
135132   Volleyball
135205   Boxing
Name: Sport, Length: 976, dtype: object
```

Getting unique values of a column

```
sports = summer_2016_medals["Sport"].unique()
```

Getting unique values of a column

```
sports = summer_2016_medals["Sport"].unique()
print(sports)

['Rowing' 'Taekwondo' 'Handball' 'Wrestling'
 'Gymnastics' 'Swimming' 'Basketball' 'Boxing'
 'Volleyball' 'Athletics']
```


Bar-chart of heights for all sports

```
fig, ax = plt.subplots()

for sport in sports:
    sport_df = summer_2016_medals[summer_2016_medals["Sport"] == sport]
```

Bar-chart of heights for all sports

```
fig, ax = plt.subplots()

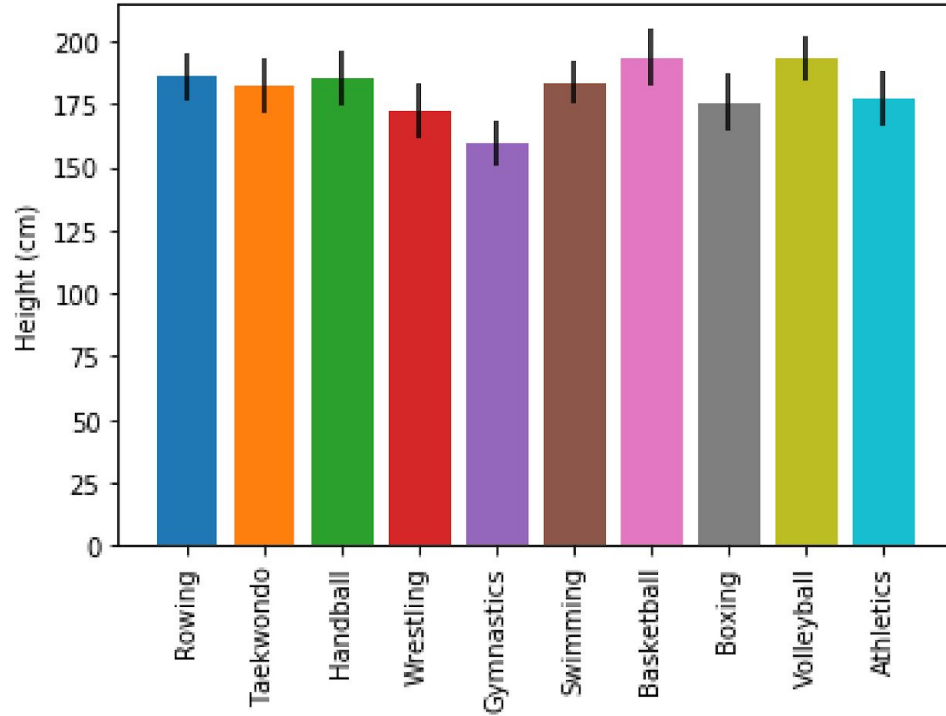
for sport in sports:
    sport_df = summer_2016_medals[summer_2016_medals["Sport"] == sport]
    ax.bar(sport, sport_df["Height"].mean(),
           yerr=sport_df["Height"].std())
```

Bar-chart of heights for all sports

```
fig, ax = plt.subplots()

for sport in sports:
    sport_df = summer_2016_medals[summer_2016_medals["Sport"] == sport]
    ax.bar(sport, sport_df["Height"].mean(),
           yerr=sport_df["Height"].std())
ax.set_ylabel("Height (cm)")
ax.set_xticklabels(sports, rotation=90)
plt.show()
```

Figure derived automatically from the data



**Practice automating
visualizations!**