

Introduction to Data Visualization with Matplotlib

Tuesday, 19 May 2020 12:34 AM

Introduction to Matplotlib

Data visualization

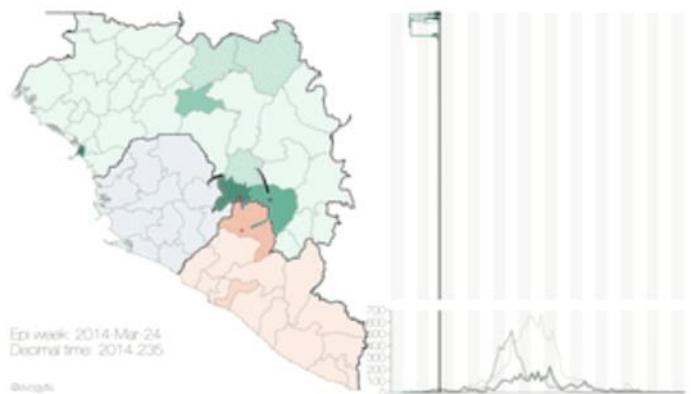
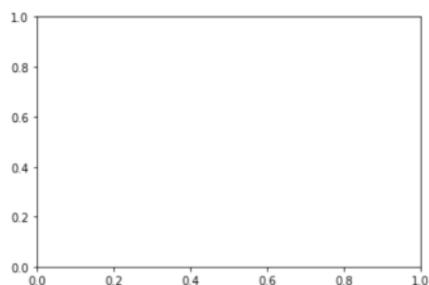


Image credit: [Gytis Dudas](#) and [Andrew Rambaut](#)

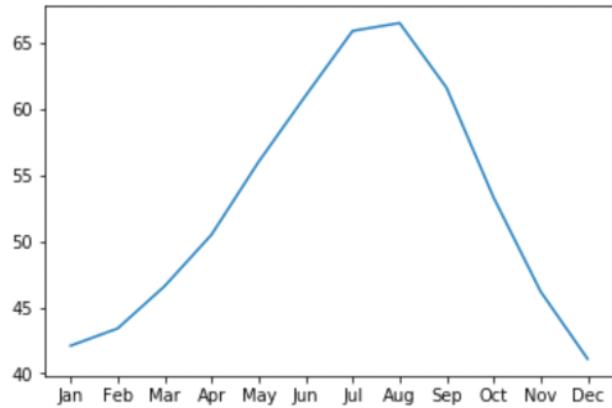
Introducing the pyplot interface

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
plt.show()
```



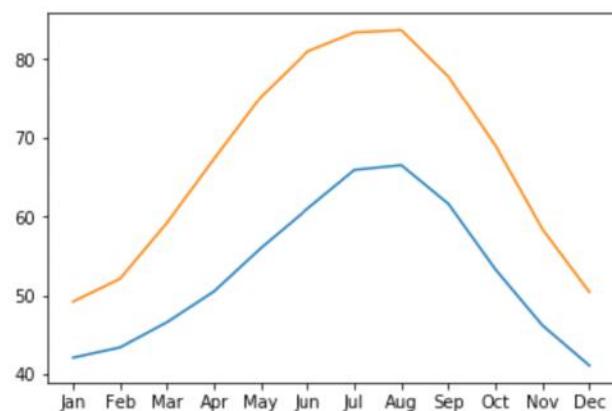
Adding data to axes

```
ax.plot(seattle_weather[ "MONTH" ], seattle_weather[ "MLY-TAVG-NORMAL" ])
plt.show()
```



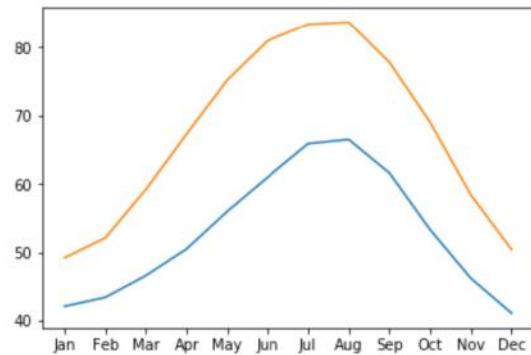
Adding more data

```
ax.plot(austin_weather[ "MONTH" ], austin_weather[ "MLY-TAVG-NORMAL" ])
plt.show()
```



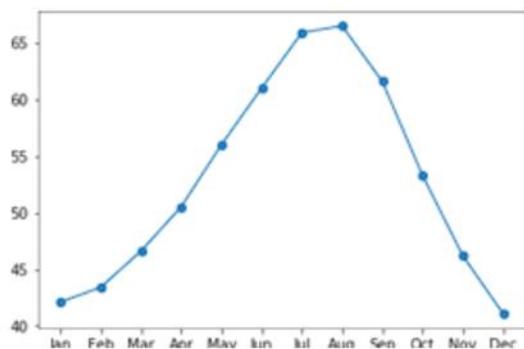
Putting it all together

```
fig, ax = plt.subplots()
ax.plot(seattle_weather[ "MONTH" ], seattle_weather[ "MLY-TAVG-NORMAL" ])
ax.plot(austin_weather[ "MONTH" ], austin_weather[ "MLY-TAVG-NORMAL" ])
plt.show()
```



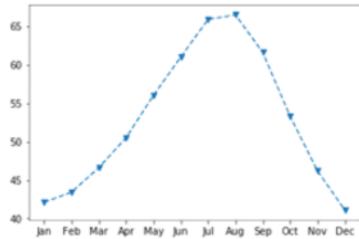
Adding markers

```
ax.plot(seattle_weather[ "MONTH" ],
        seattle_weather[ "MLY-PRCP-NORMAL" ],
        marker="o")
plt.show()
```



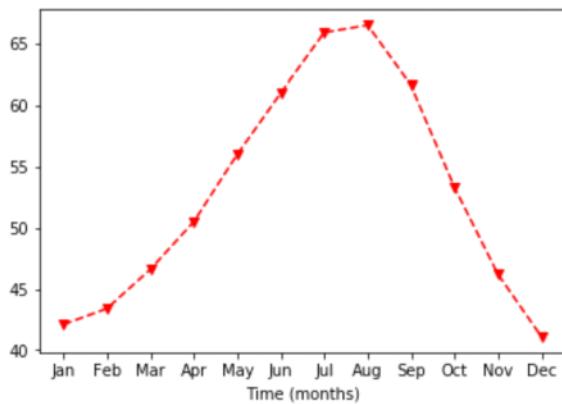
Setting the linestyle

```
fig, ax = plt.subplots()  
ax.plot(seattle_weather[ "MONTH" ],  
        seattle_weather[ "MLY-TAVG-NORMAL" ],  
        marker="v", linestyle="--")  
plt.show()
```



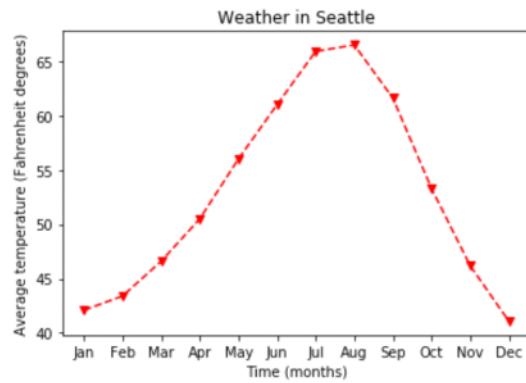
Customizing the axes labels

```
ax.set_xlabel("Time (months)")  
plt.show()
```



Adding a title

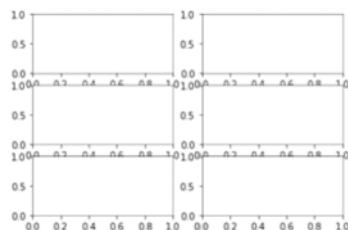
```
ax.set_title("Weather in Seattle")
plt.show()
```



Small multiples with plt.subplots

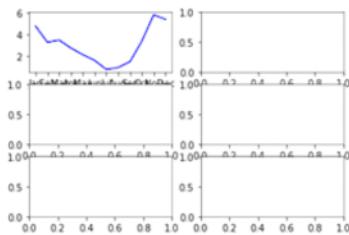
```
fig, ax = plt.subplots()
```

```
fig, ax = plt.subplots(3, 2)
plt.show()
```



Adding data to subplots

```
ax.shape  
(3, 2)  
  
ax[0, 0].plot(seattle_weather[ "MONTH" ],  
              seattle_weather[ "MLY-PRCP-NORMAL" ],  
              color='b')  
plt.show()
```

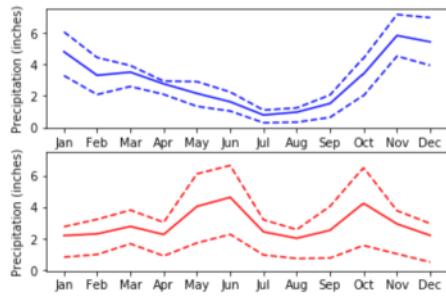


Subplots with data

```
fig, ax = plt.subplots(2, 1)  
ax[0].plot(seattle_weather[ "MONTH" ], seattle_weather[ "MLY-PRCP-NORMAL" ],  
           color='b')  
ax[0].plot(seattle_weather[ "MONTH" ], seattle_weather[ "MLY-PRCP-25PCTL" ],  
           linestyle='--', color='b')  
ax[0].plot(seattle_weather[ "MONTH" ], seattle_weather[ "MLY-PRCP-75PCTL" ],  
           linestyle='--', color='b')  
ax[1].plot(austin_weather[ "MONTH" ], austin_weather[ "MLY-PRCP-NORMAL" ],  
           color='r')  
ax[1].plot(austin_weather[ "MONTH" ], austin_weather[ "MLY-PRCP-25PCTL" ],  
           linestyle='--', color='r')  
ax[1].plot(austin_weather[ "MONTH" ], austin_weather[ "MLY-PRCP-75PCTL" ],  
           linestyle='--', color='r')  
ax[0].set_ylabel("Precipitation (inches)")  
ax[1].set_ylabel("Precipitation (inches)")  
ax[1].set_xlabel("Time (months)")  
plt.show()
```

Sharing the y-axis range

```
fig, ax = plt.subplots(2, 1, sharey=True)
```



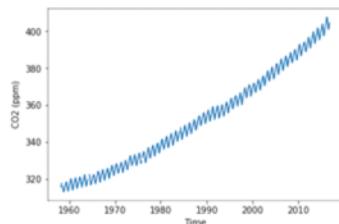
To make sure that all the subplots have the same range of y-axis values, we

Plotting time-series

Plotting time-series data

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()

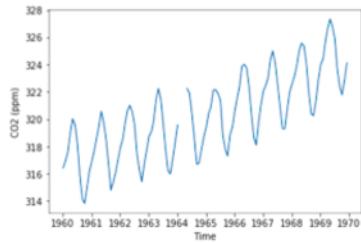
ax.plot(climate_change.index, climate_change['co2'])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)')
plt.show()
```



Zooming in on a decade

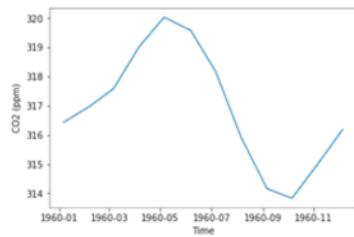
```
sixties = climate_change["1960-01-01":"1969-12-31"]
```

```
fig, ax = plt.subplots()
ax.plot(sixties.index, sixties['co2'])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)')
plt.show()
```



Zooming in on one year

```
sixty_nine = climate_change["1969-01-01":"1969-12-31"]
fig, ax = plt.subplots()
ax.plot(sixty_nine.index, sixty_nine['co2'])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)')
plt.show()
```



Plotting two time-series together

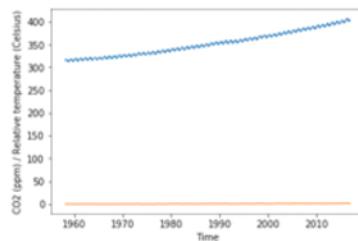
```
import pandas as pd
climate_change = pd.read_csv('climate_change.csv',
                             parse_dates=[ "date" ],
                             index_col="date")
```

```
climate_change
```

```
      co2  relative_temp
date
1958-03-06  315.71      0.10
1958-04-06  317.45      0.01
1958-07-06  315.86      0.06
...
2016-11-06  403.55      0.93
2016-12-06  404.45      0.81
```

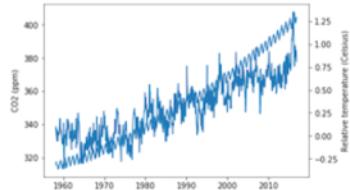
Plotting two time-series together

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change[ "co2"])
ax.plot(climate_change.index, climate_change[ "relative_temp"])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm) / Relative temperature')
plt.show()
```



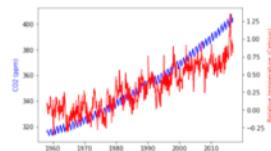
Using twin axes

```
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)')
ax2 = ax.twinx()
ax2.plot(climate_change.index, climate_change["relative_temp"])
ax2.set_ylabel('Relative temperature (Celsius)')
plt.show()
```



Separating variables by color

```
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"], color='blue')
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)', color='blue')
ax2 = ax.twinx()
ax2.plot(climate_change.index, climate_change["relative_temp"],
          color='red')
ax2.set_ylabel('Relative temperature (Celsius)', color='red')
plt.show()
```

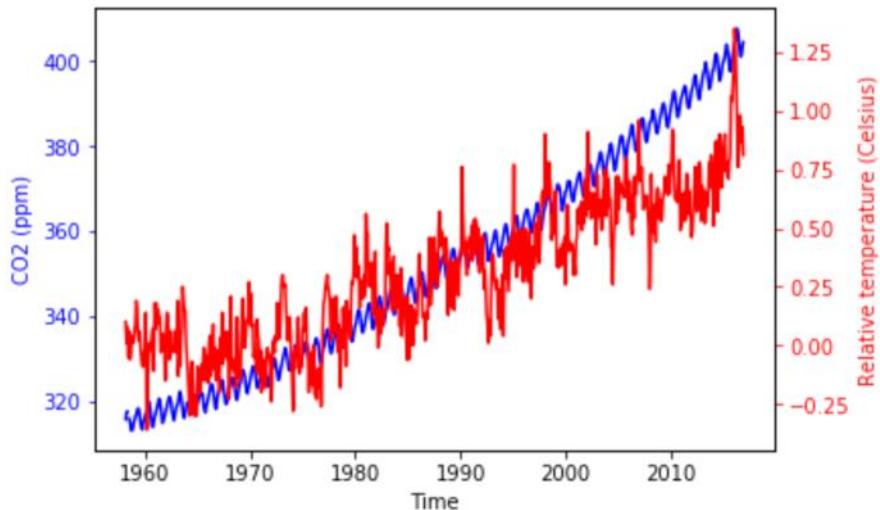


Coloring the ticks

```
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"],
         color='blue')
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)', color='blue')
ax.tick_params('y', colors='blue')

ax2 = ax.twinx()
ax2.plot(climate_change.index,
          climate_change["relative_temp"],
          color='red')
ax2.set_ylabel('Relative temperature (Celsius)',
               color='red')
ax2.tick_params('y', colors='red')
plt.show()
```

Coloring the ticks

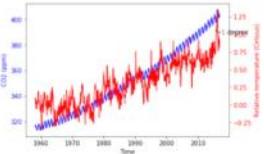


A function that plots time-series

```
def plot_timeseries(axes, x, y, color, xlabel, ylabel):
    axes.plot(x, y, color=color)
    axes.set_xlabel(xlabel)
    axes.set_ylabel(ylabel, color=color)
    axes.tick_params('y', colors=color)
```

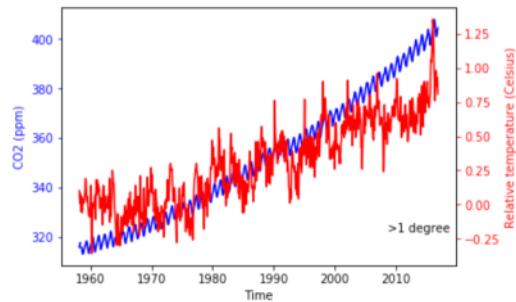
Annotation

```
fig, ax = plt.subplots()
plot_timeseries(ax, climate_change.index, climate_change['co2'],
                 'blue', 'Time', 'CO2 (ppm)')
ax2 = ax.twinx()
plot_timeseries(ax2, climate_change.index,
                climate_change['relative_temp'],
                'red', 'Time', 'Relative temperature (Celsius)')
ax2.annotate(">1 degree", xy=[pd.Timestamp("2015-10-06"), 1])
plt.show()
```



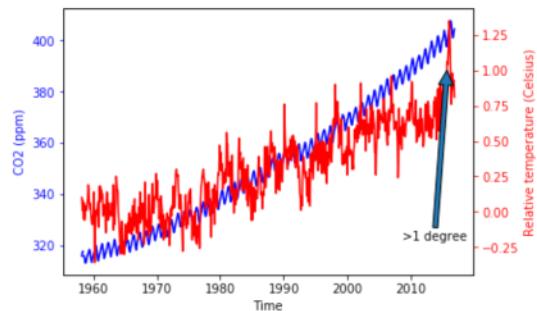
Positioning the text

```
ax2.annotate(" >1 degree",
             xy=(pd.Timestamp('2015-10-06'), 1),
             xytext=(pd.Timestamp('2008-10-06'), -0.2))
```



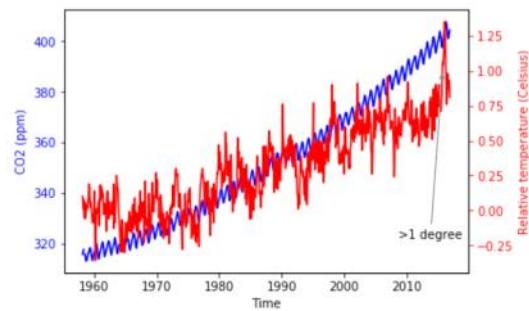
Adding arrows to annotation

```
ax2.annotate(" >1 degree",
             xy=(pd.Timestamp('2015-10-06'), 1),
             xytext=(pd.Timestamp('2008-10-06'), -0.2),
             arrowprops={})
```



Customizing arrow properties

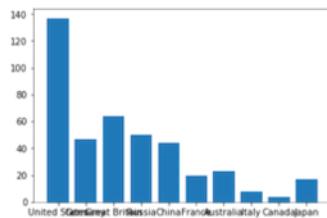
```
ax2.annotate(" >1 degree",
             xy=(pd.Timestamp('2015-10-06'), 1),
             xytext=(pd.Timestamp('2008-10-06'), -0.2),
             arrowprops={"arrowstyle": "->", "color": "gray"})
```



Quantitative comparison and statistical visualization

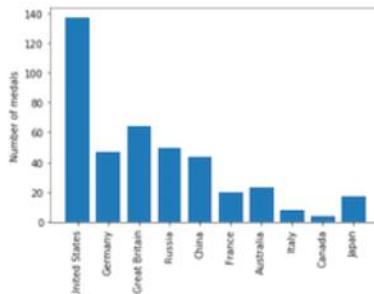
Olympic medals: visualizing the data

```
medals = pd.read_csv('medals_by_country_2016.csv', index_col=0)
fig, ax = plt.subplots()
ax.bar(medals.index, medals["Gold"])
plt.show()
```



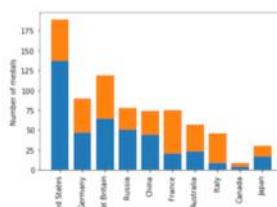
Interlude: rotate the tick labels

```
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()
```



Olympic medals: visualizing the other medals

```
fig, ax = plt.subplots()
ax.bar(medals.index, medals["Gold"])
ax.bar(medals.index, medals["Silver"], bottom=medals["Gold"])
ax.set_xticklabels(medals.index, rotation=90)
ax.set_ylabel("Number of medals")
plt.show()
```

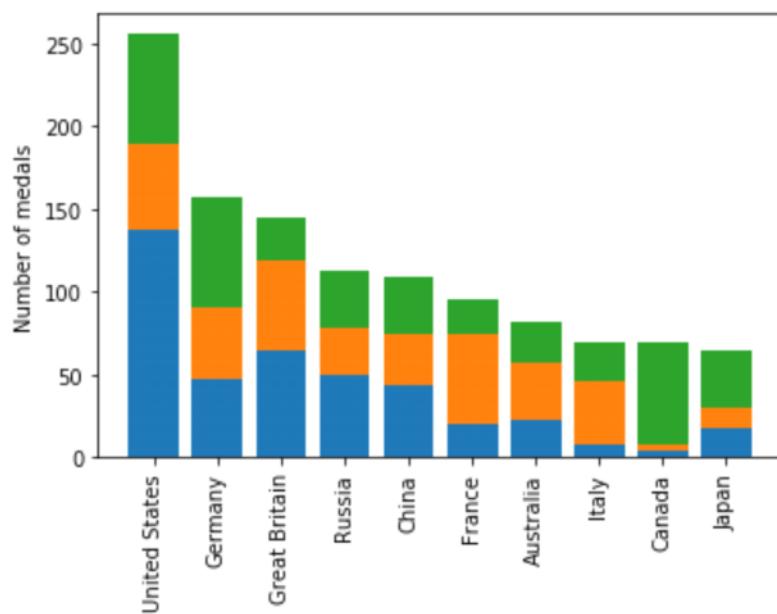


Olympic medals: visualizing all three

```
fig, ax = plt.subplots
ax.bar(medals.index, medals["Gold"])

ax.bar(medals.index, medals["Silver"], bottom=medals["Gold"])
ax.bar(medals.index, medals["Bronze"],
       bottom=medals["Gold"] + medals["Silver"])
ax.set_xticklabels(medals.index, rotation=90)
ax.set_ylabel("Number of medals")
plt.show()
```

Stacked bar chart

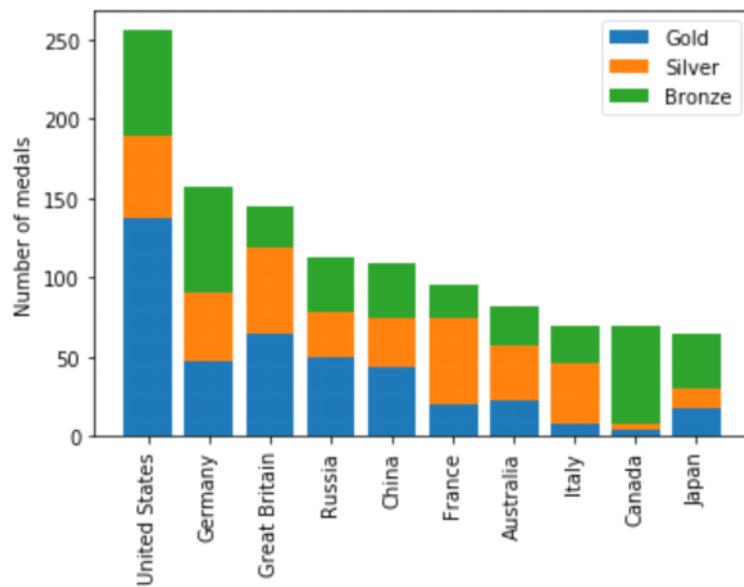


Adding a legend

```
fig, ax = plt.subplots
ax.bar(medals.index, medals["Gold"], label="Gold")
ax.bar(medals.index, medals["Silver"], bottom=medals["Gold"],
       label="Silver")
ax.bar(medals.index, medals["Bronze"],
       bottom=medals["Gold"] + medals["Silver"],
       label="Bronze")

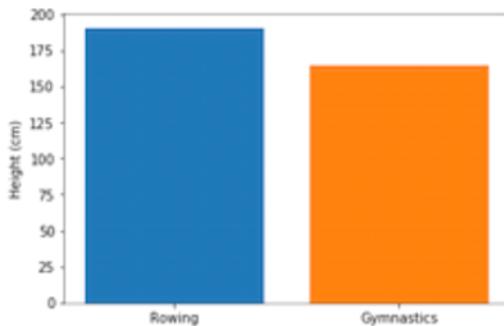
ax.set_xticklabels(medals.index, rotation=90)
ax.set_ylabel("Number of medals")
ax.legend()
plt.show()
```

Stacked bar chart with legend



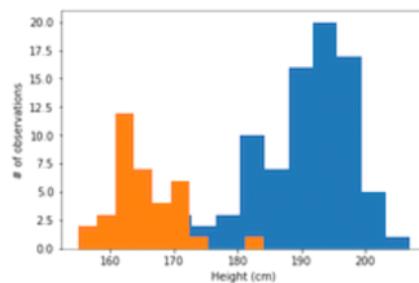
A bar chart again

```
fig, ax = plt.subplots()  
ax.bar("Rowing", mens_rowing["Height"].mean())  
ax.bar("Gymnastics", mens_gymnastics["Height"].mean())  
ax.set_ylabel("Height (cm)")  
plt.show()
```



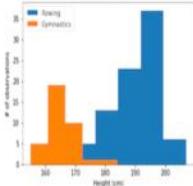
Introducing histograms

```
fig, ax = plt.subplots()  
ax.hist(mens_rowing["Height"])  
ax.hist(mens_gymnastic["Height"])  
ax.set_xlabel("Height (cm)")  
ax.set_ylabel("# of observations")  
plt.show()
```



Customizing histograms: setting the number of bins

```
ax.hist(mens_rowing["Height"], label="Rowing", bins=5)
ax.hist(mens_gymnastic["Height"], label="Gymnastics", bins=5)
ax.set_xlabel("Height (cm)")
ax.set_ylabel("# of observations")
ax.legend()
plt.show()
```

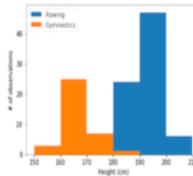


Customizing histograms: setting bin boundaries

```
ax.hist(mens_rowing["Height"], label="Rowing",
        bins=[150, 160, 170, 180, 190, 200, 210])

ax.hist(mens_gymnastic["Height"], label="Gymnastics",
        bins=[150, 160, 170, 180, 190, 200, 210])

ax.set_xlabel("Height (cm)")
ax.set_ylabel("# of observations")
ax.legend()
plt.show()
```



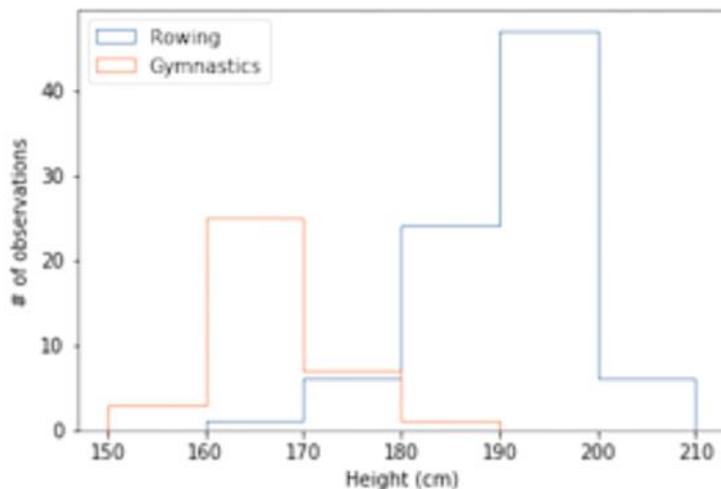
Customizing histograms: transparency

```
ax.hist(mens_rowing["Height"], label="Rowing",
        bins=[150, 160, 170, 180, 190, 200, 210],
        histtype="step")

ax.hist(mens_gymnastic["Height"], label="Gymnastics",
        bins=[150, 160, 170, 180, 190, 200, 210],
        histtype="step")

ax.set_xlabel("Height (cm)")
ax.set_ylabel("# of observations")
ax.legend()
plt.show()
```

Histogram with a histtype of step



Adding error bars to bar charts

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

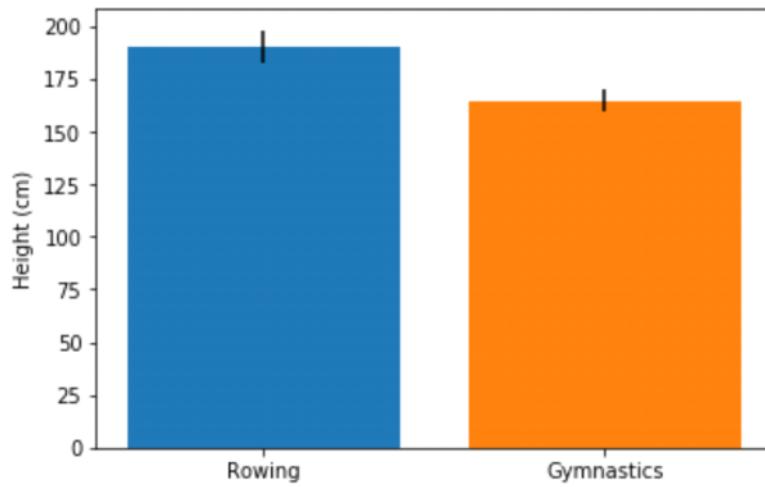
ax.bar("Rowing",
       mens_rowing["Height"].mean(),
       yerr=mens_rowing["Height"].std())

ax.bar("Gymnastics",
       mens_gymnastics["Height"].mean(),
       yerr=mens_gymnastics["Height"].std())

ax.set_ylabel("Height (cm)")

plt.show()
```

Error bars in a bar chart



Adding error bars to plots

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

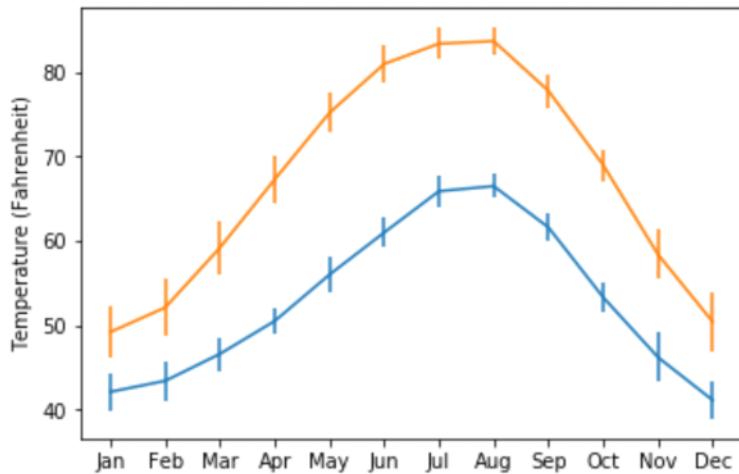
ax.errorbar(seattle_weather["MONTH"],
            seattle_weather["MLY-TAVG-NORMAL"],
            yerr=seattle_weather["MLY-TAVG-STDDEV"])

ax.errorbar(austin_weather["MONTH"],
            austin_weather["MLY-TAVG-NORMAL"],
            yerr=austin_weather["MLY-TAVG-STDDEV"])

ax.set_ylabel("Temperature (Fahrenheit)")

plt.show()
```

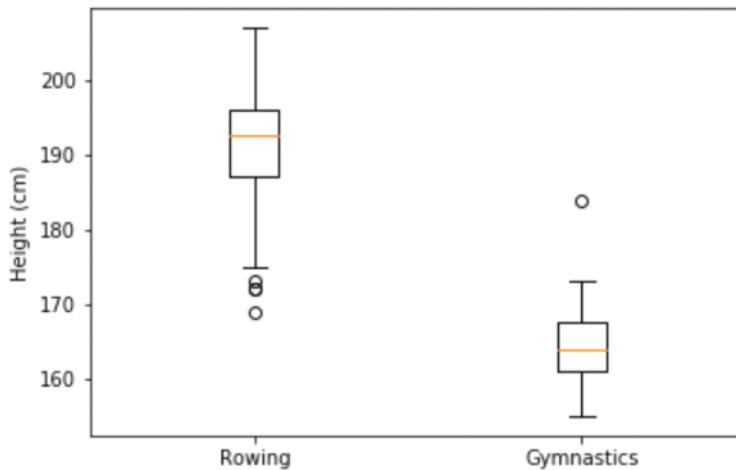
Error bars in plots



Adding boxplots

```
fig, ax = plt.subplots()  
  
ax.boxplot([mens_rowing[ "Height" ],  
            mens_gymnastics[ "Height" ]])  
  
ax.set_xticklabels(["Rowing", "Gymnastics"])  
ax.set_ylabel("Height (cm)")  
  
plt.show()
```

Interpreting boxplots



The red line indicates the median height.

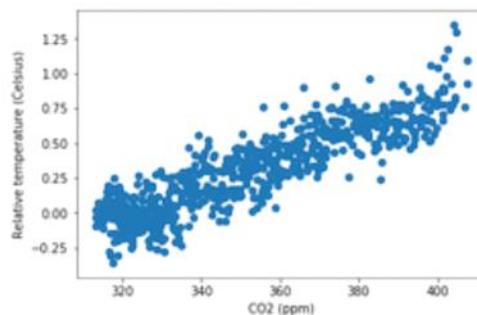
The edges of the box portion at the center indicate the inter-quartile range of the data, between the 25th and the 75th percentiles.

The whiskers at the ends of the thin bars indicate one and a half times

the size of the inter-quartile range beyond the 75th and 25th percentiles.

Introducing scatter plots

```
fig, ax = plt.subplots()
ax.scatter(climate_change["co2"], climate_change["relative_temp"])
ax.set_xlabel("CO2 (ppm)")
ax.set_ylabel("Relative temperature (Celsius)")
plt.show()
```



Customizing scatter plots

```
eighties = climate_change[ "1980-01-01":"1989-12-31"]
nineties = climate_change[ "1990-01-01":"1999-12-31"]

fig, ax = plt.subplots()

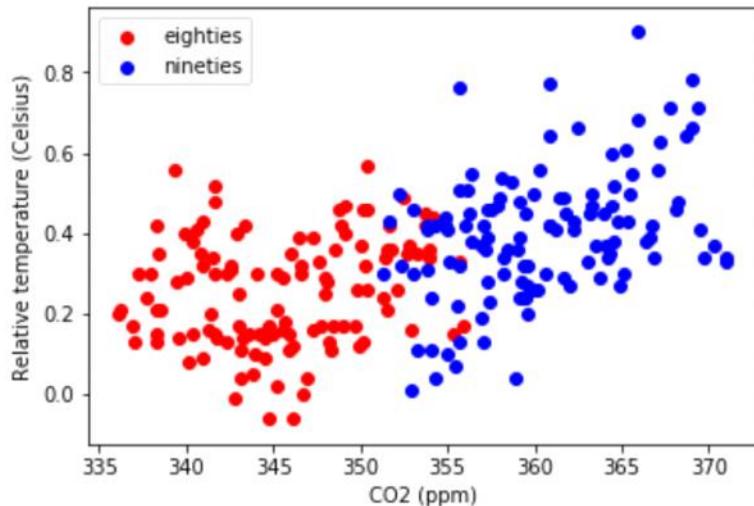
ax.scatter(eighties[ "co2"], eighties[ "relative_temp"],
           color="red", label="eighties")
ax.scatter(nineties[ "co2"], nineties[ "relative_temp"],
           color="blue", label="nineties")

ax.legend()

ax.set_xlabel("CO2 (ppm)")
ax.set_ylabel("Relative temperature (Celsius)")

plt.show()
```

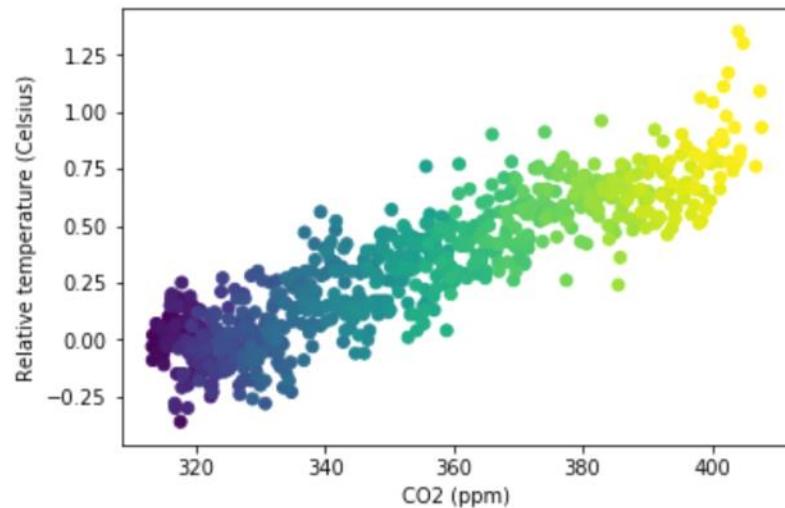
Encoding a comparison by color



Encoding a third variable by color

```
fig, ax = plt.subplots()
ax.scatter(climate_change[ "co2"], climate_change[ "relative_temp"],
           c=climate_change.index)
ax.set_xlabel("CO2 (ppm)")
ax.set_ylabel("Relative temperature (Celsius)")
plt.show()
```

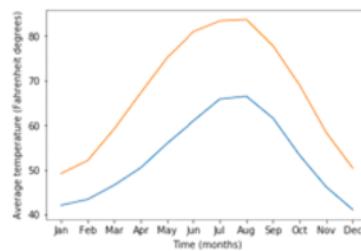
Encoding time in color



Sharing Visualizations with Others

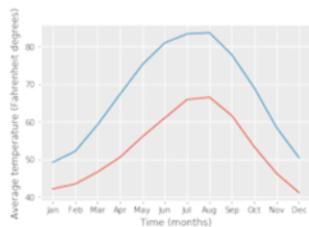
Changing plot style

```
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

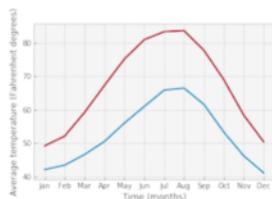
```
plt.style.use("default")
```

The available styles

https://matplotlib.org/gallery/style_sheets/style_sheets_reference.html

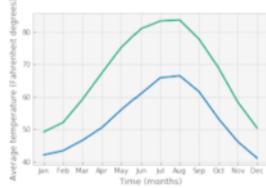
The "bmh" style

```
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

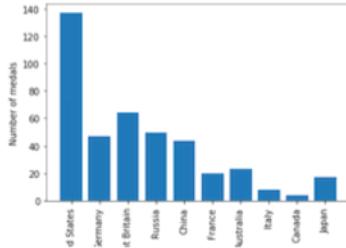
- Dark backgrounds are usually less visible
- If color is important, consider choosing colorblind-friendly options
 - "seaborn-colorblind" or "tableau-colorblind10"
- If you think that someone will want to print your figure, use less ink
- If it will be printed in black-and-white, use the "grayscale"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

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")
```

```
ls
```

```
gold_medals.png
```

Different file formats

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

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

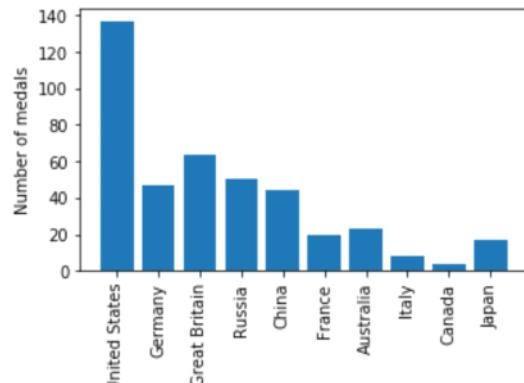
```
fig.savefig("gold_medals.svg")
```

Resolution

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

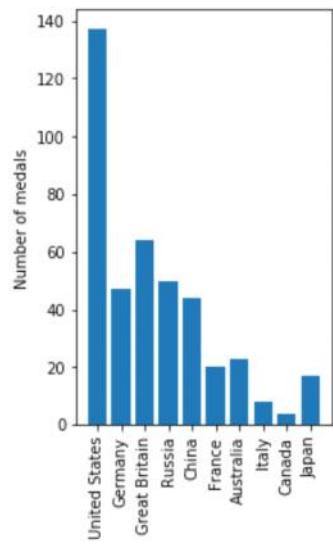
Size

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



Another aspect ratio

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



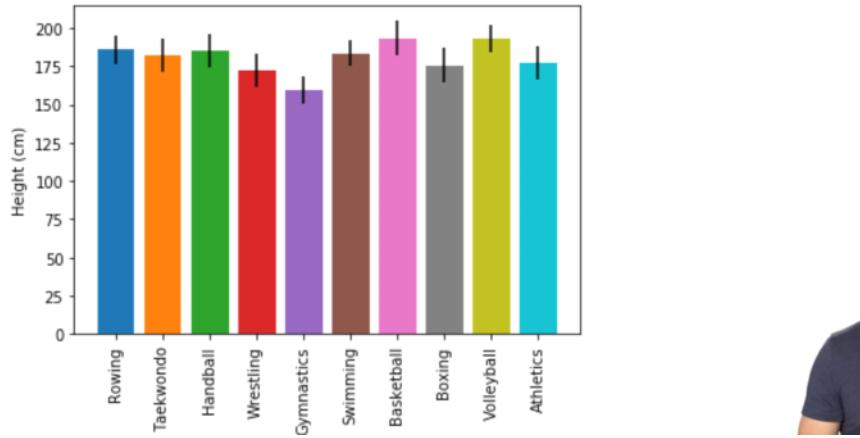
Getting unique values of a column

```
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]  
    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()
```

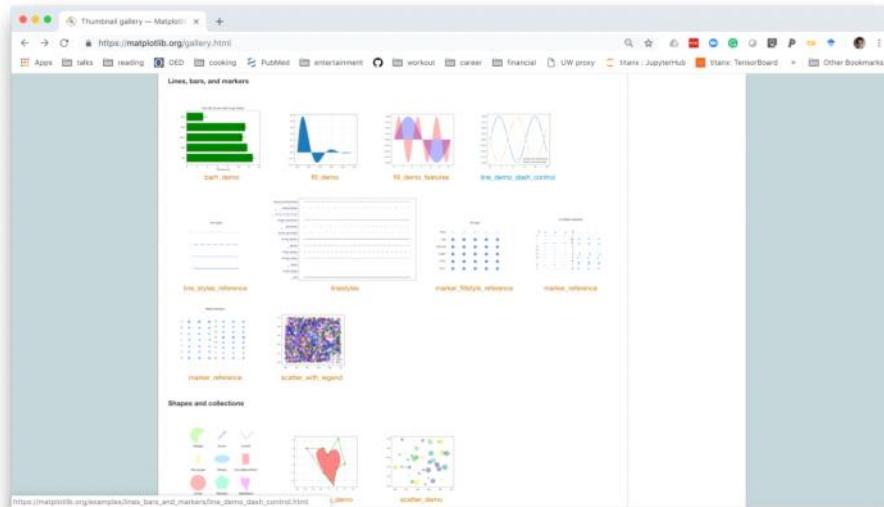
Figure derived automatically from the data



The Matplotlib gallery

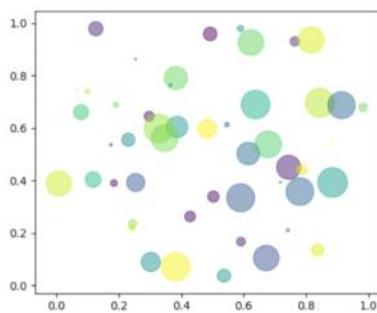
<https://matplotlib.org/gallery.html>

Gallery of examples



Example page with code

[shapes_and_collections example code: scatter_demo.py](#)
(Source code, png, pdf)

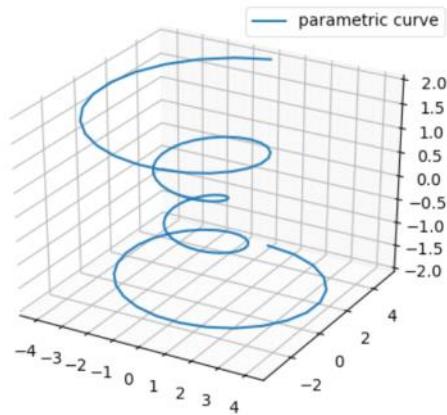


```
"""
Simple demo of a scatter plot.
"""

import numpy as np
import matplotlib.pyplot as plt

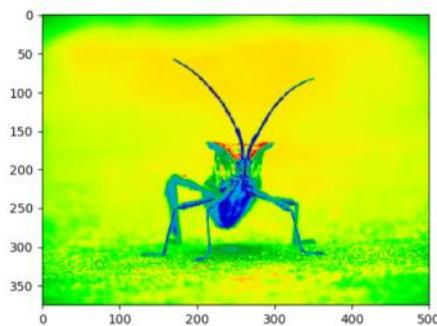
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2  # 0 to 15 point radii
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

Plotting data in 3D



https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

Visualizing images with pseudo-color



https://matplotlib.org/users/image_tutorial.html

Animations

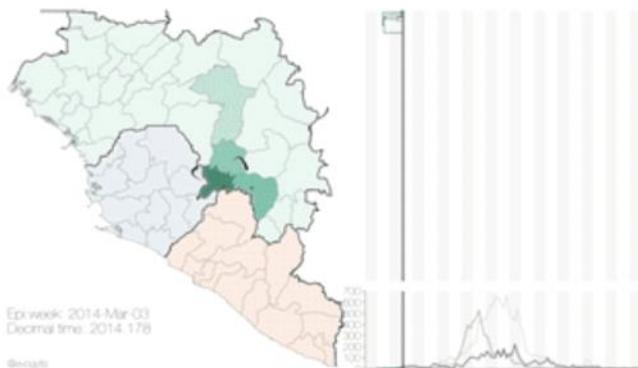


Image credit: [Gytis Dudas](#) and [Andrew Rambaut](#)

https://matplotlib.org/api/animation_api.html

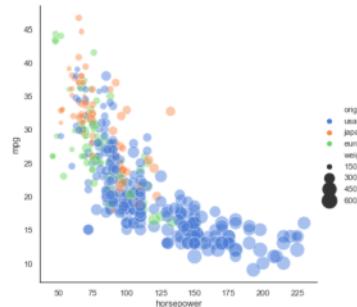
Using Matplotlib for geospatial data



<https://scitools.org.uk/cartopy/docs/latest/>

Pandas + Matplotlib = Seaborn

```
seaborn.relplot(x="horsepower", y="mpg", hue="origin", size="weight",
                 sizes=(40, 400), alpha=.5, palette="muted",
                 height=6, data=mpg)
```



Seaborn example gallery

<https://seaborn.pydata.org/examples/index.html>