

Data

visualization

**Exploratory
Data
Analysis**

Goals of EDA

- Find problems
- Understand
- Communicate

Plotting libraries

- Pandas
 - d.plot.hist
 - d.plot.scatter
 - ...
- Matplotlib
 - plt.hist
 - plt.scatter
 - ...
- Seaborn
 - sns.histplot
 - sns.scatterplot
 - ...
- Many more: plotly, etc.

Plotting libraries

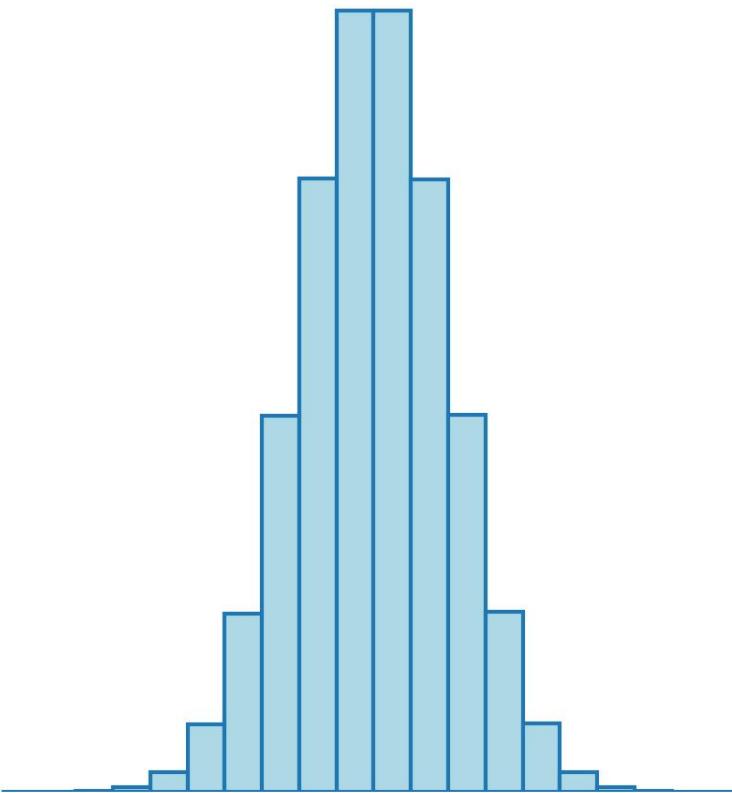
- **Pandas**
`d.plot.hist`
`d.plot.scatter`
...
- **Matplotlib**
`plt.hist`
`plt.scatter`
...
- **Seaborn**
`sns.histplot`
`sns.scatterplot`
...
- **Many more: plotly, etc.**

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

import seaborn as sns
import scipy, sklearn, statsmodels
from collections import Counter
```

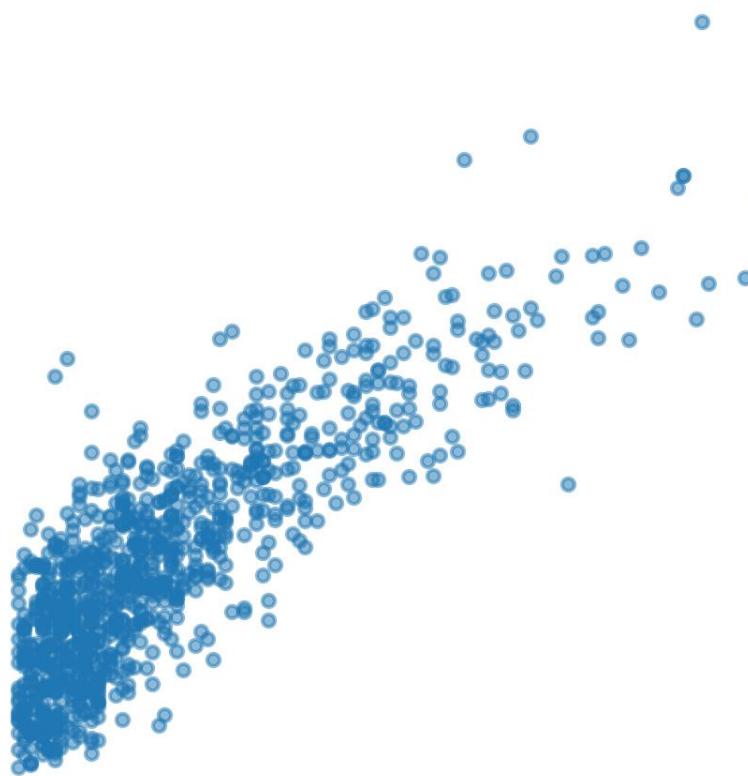
Matplotlib

histogram



`plt.hist(x)`

scatter plot



`plt.scatter(x,y)`

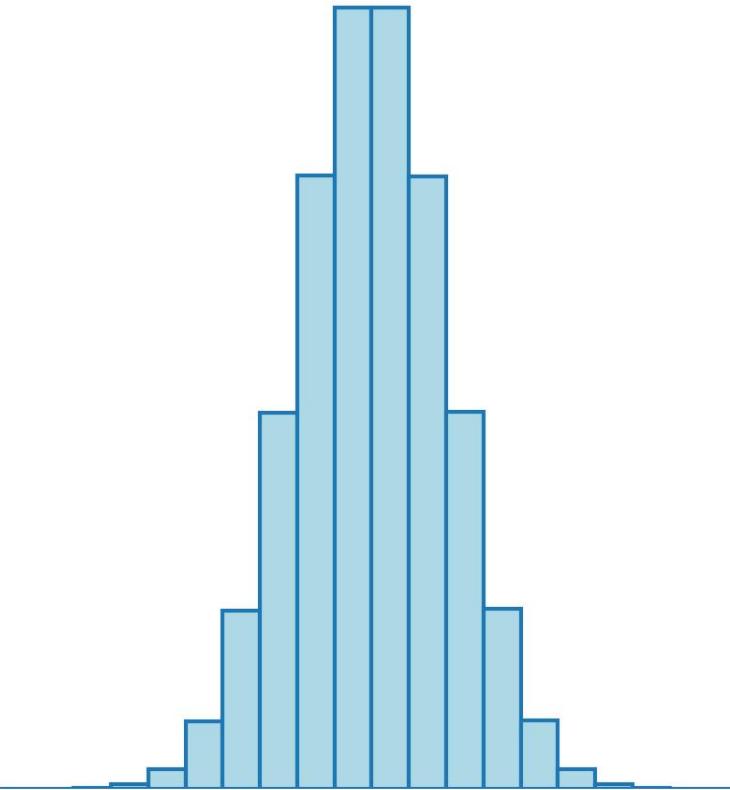
line chart



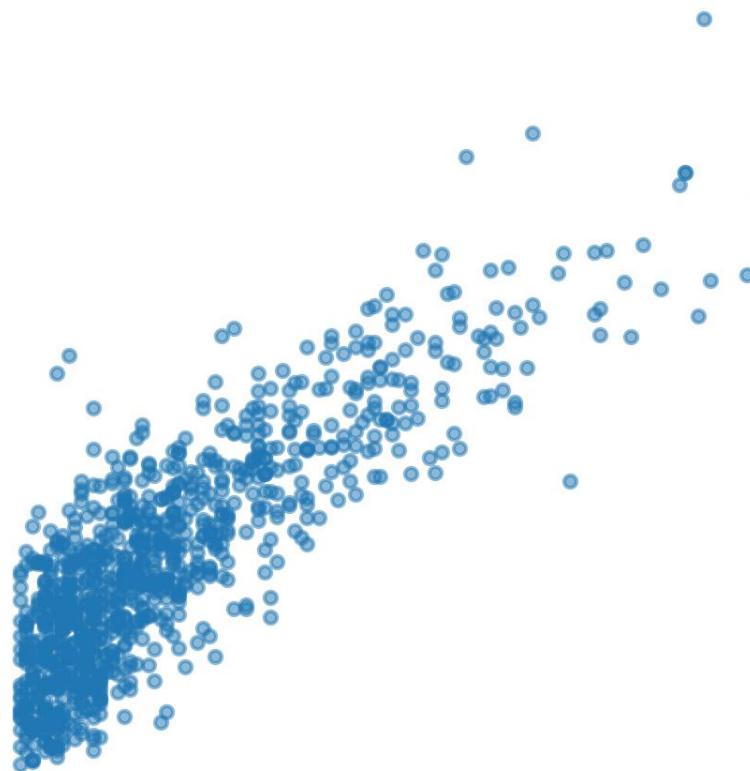
`plt.plot(x,y)`

Pandas

histogram



scatter plot

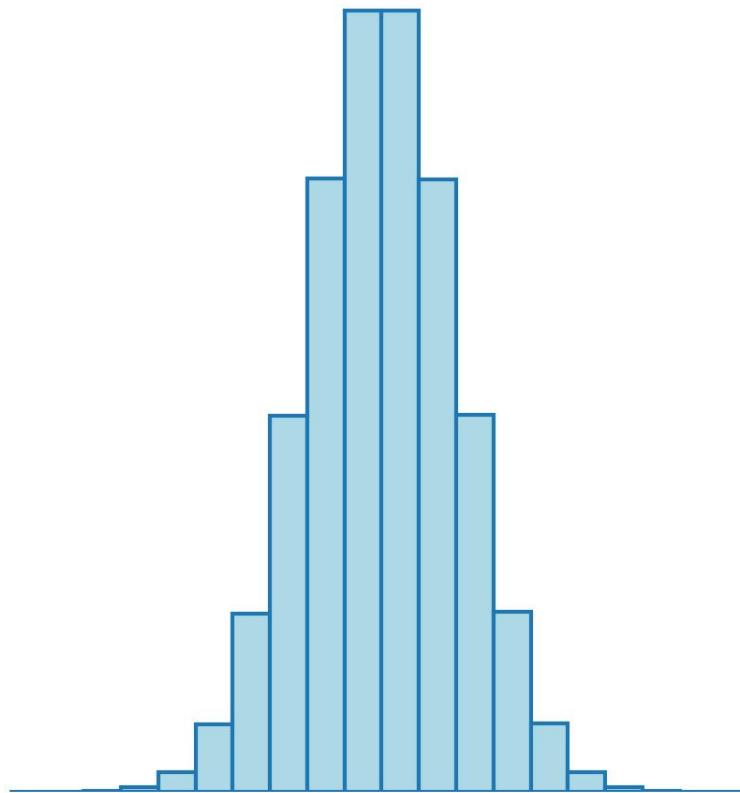


line chart

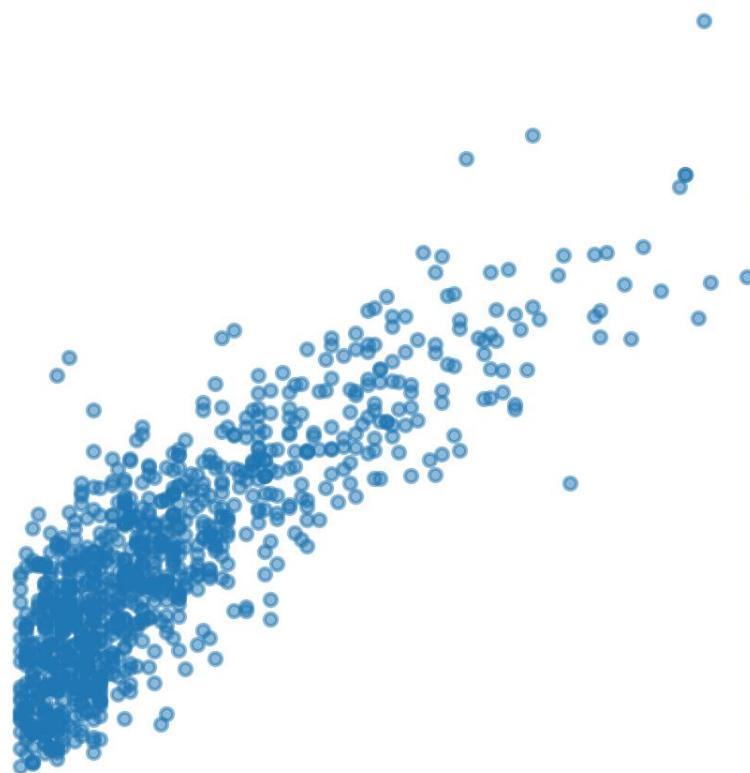


Pandas

histogram



scatter plot



line chart

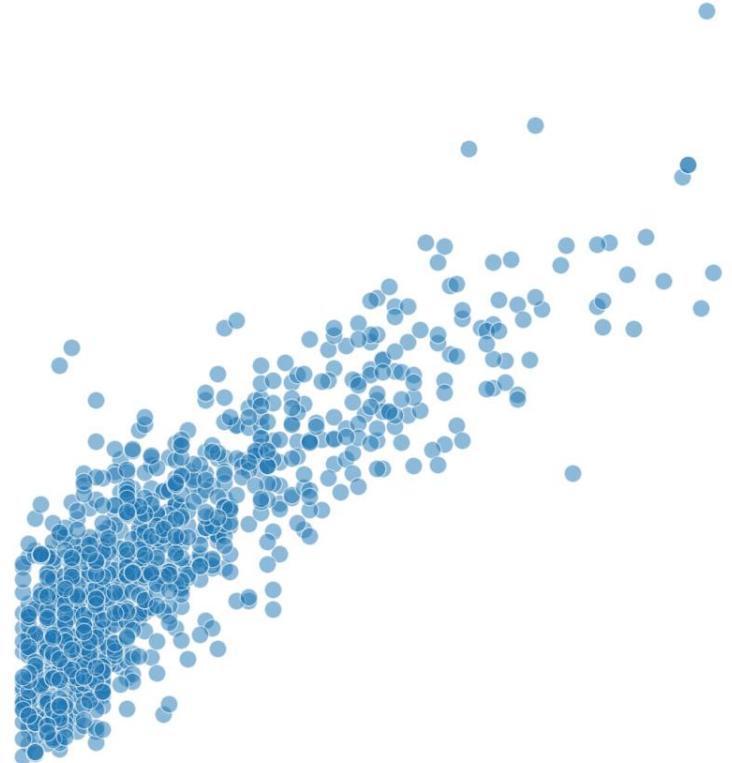


Seaborn

histogram


```
sns.histplot(x)
```

scatter plot



```
sns.scatterplot(d, x="x", y="y")
```

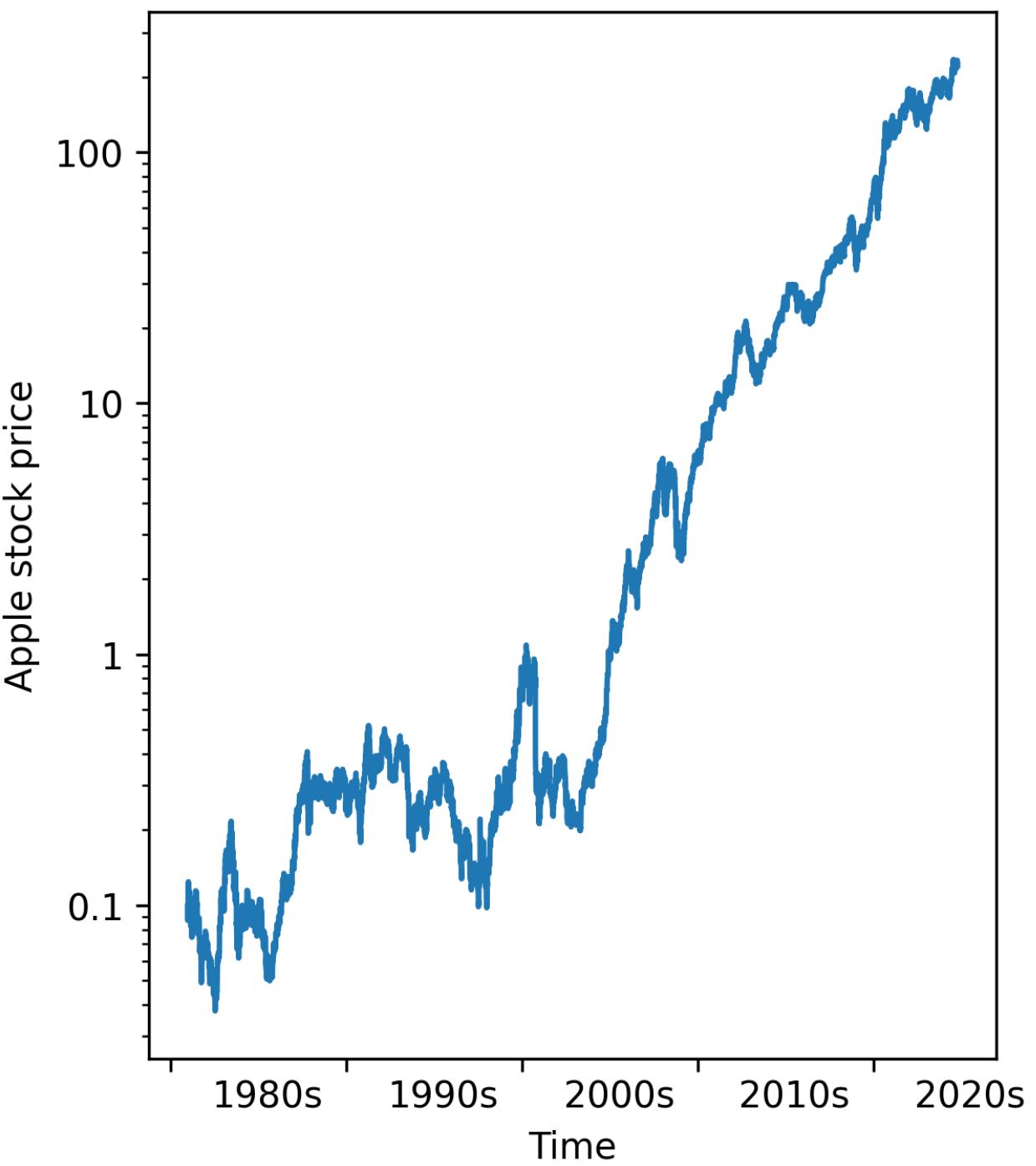
line chart



```
sns.lineplot(x)
```

Line plot

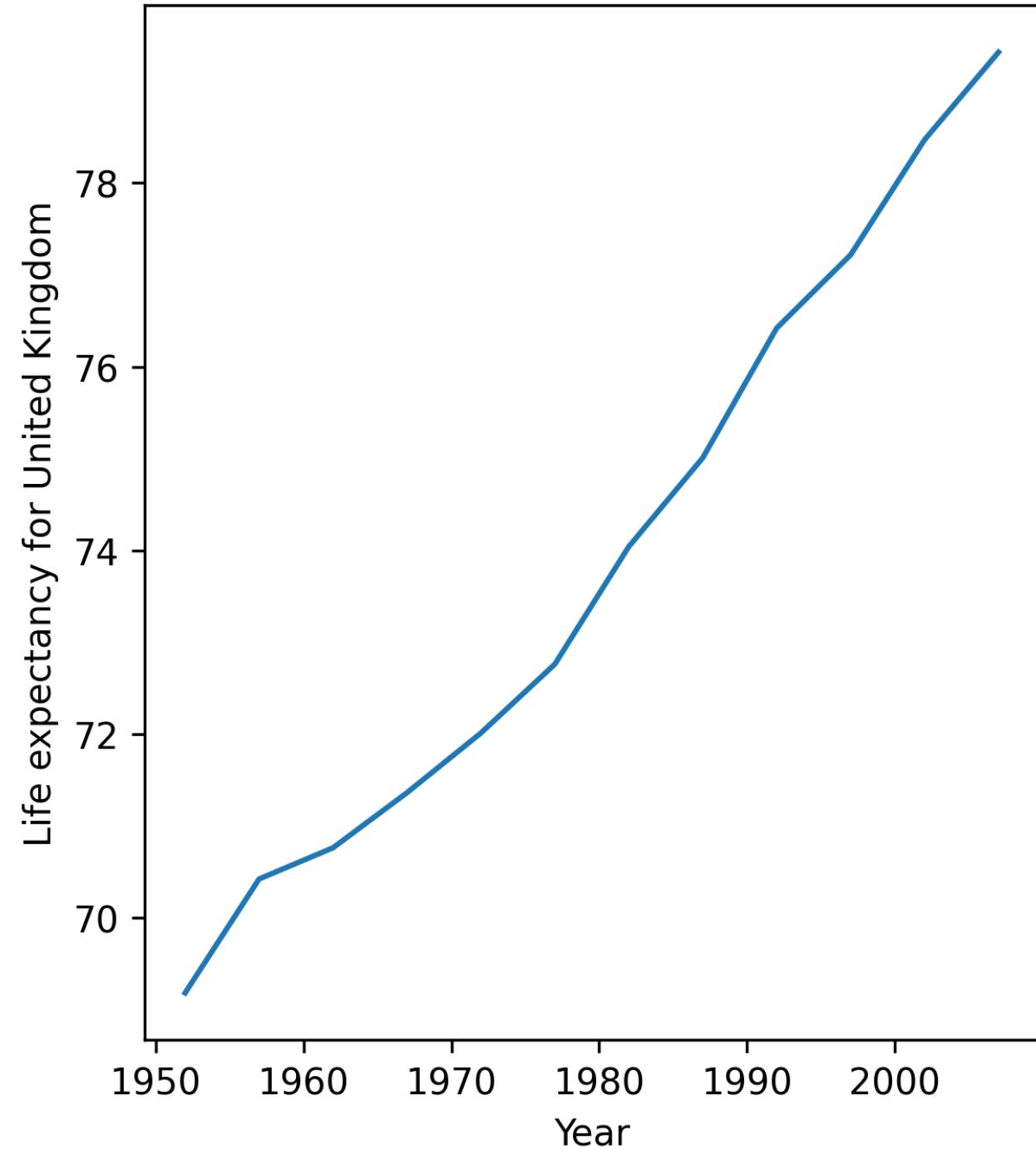
```
d.plot( "x", "y" )
```



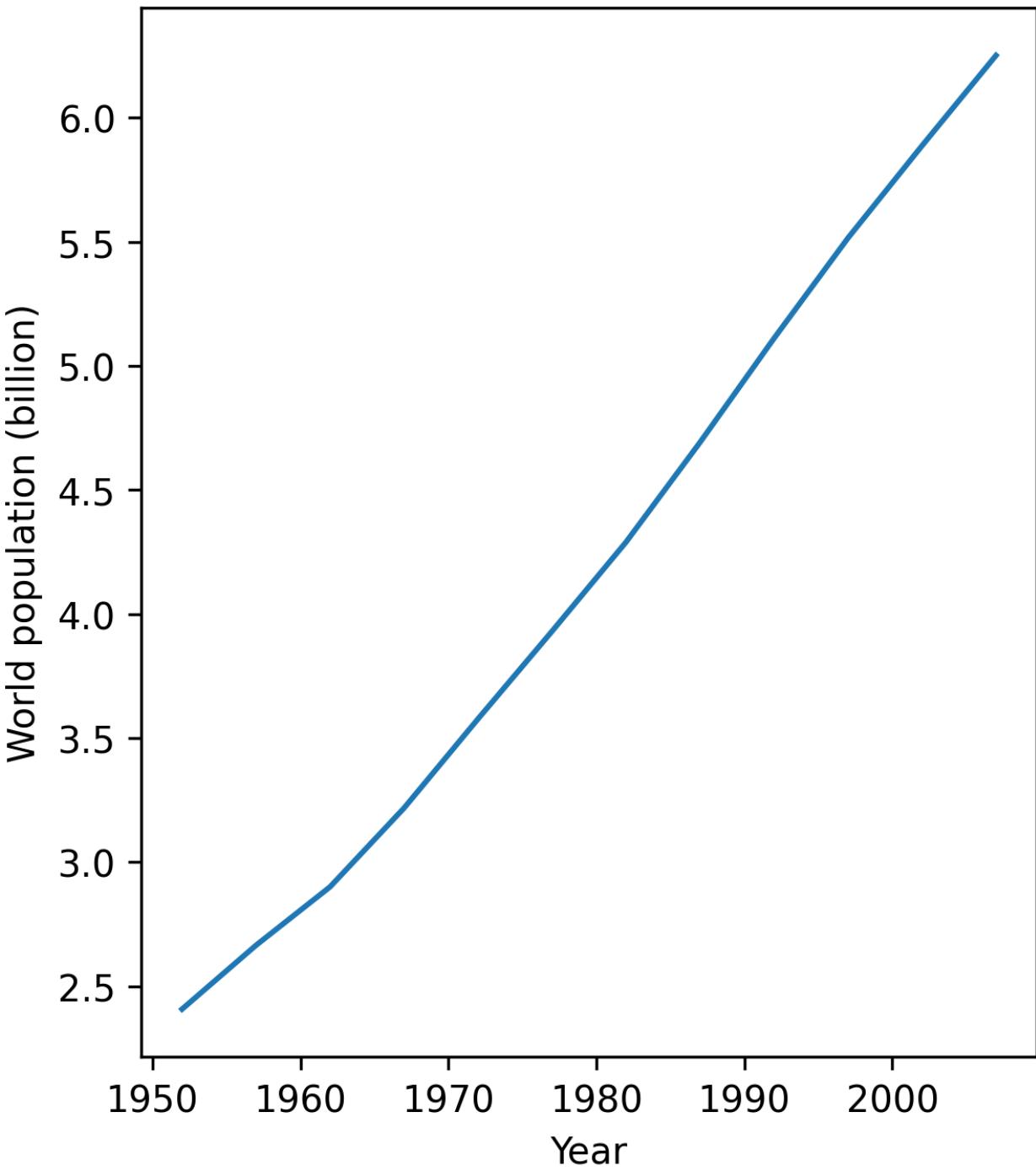
Exercise

- Generate a line plot of the life expectancy, over time, for a country of your choice.
What do you notice?
- Generate a line plot of the world population over time (do it in two steps: first compute the world population, using groupby, then plot it).
What do you notice?

```
# List of all the countries available  
gapminder['country'].unique()  
  
country = 'United Kingdom'  
  
i = gapminder['country'] == country  
gapminder[i].plot.line( 'year', 'lifeExp' )
```

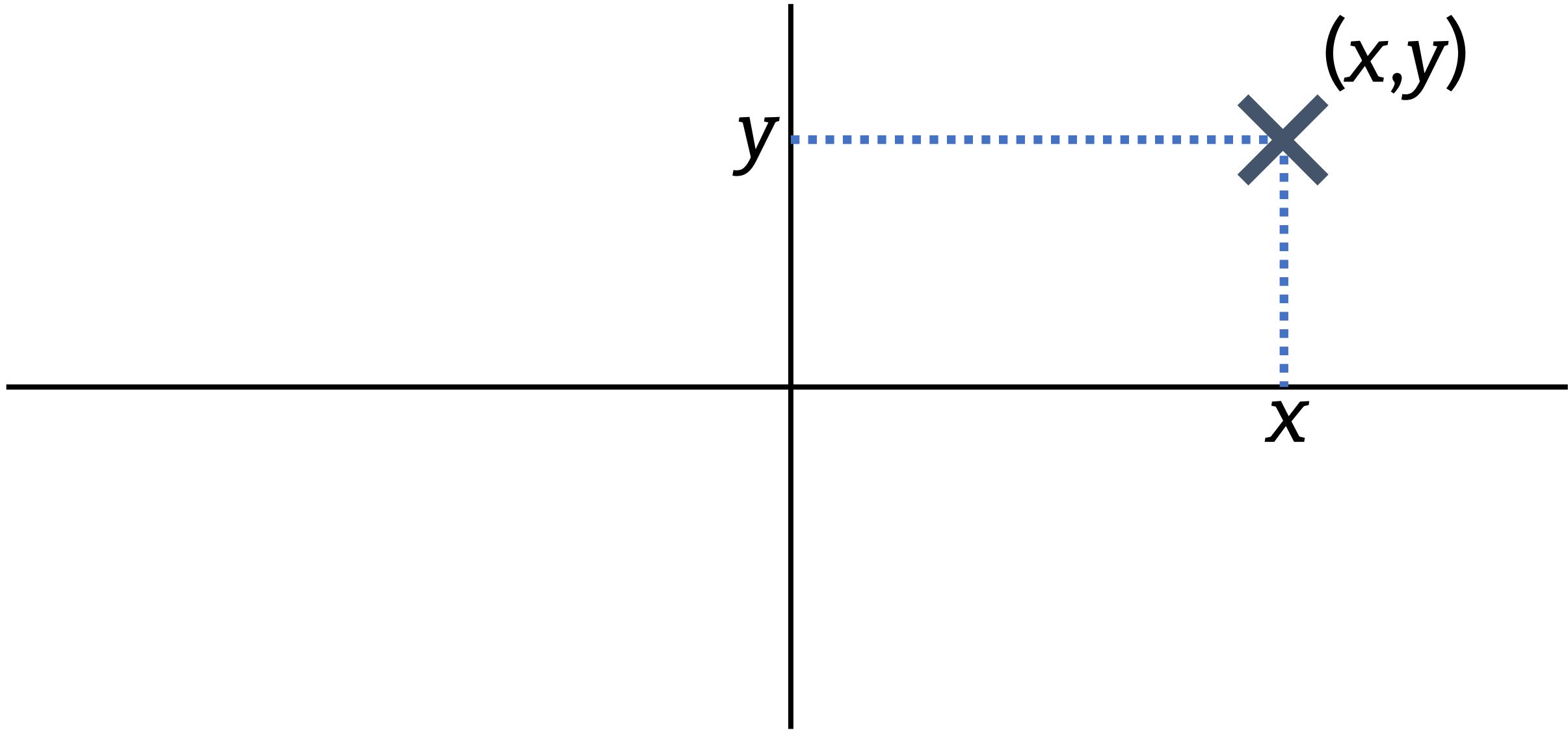


```
population = (
    gapminder.groupby('year')['pop'].sum()
)
population.plot.line()
```

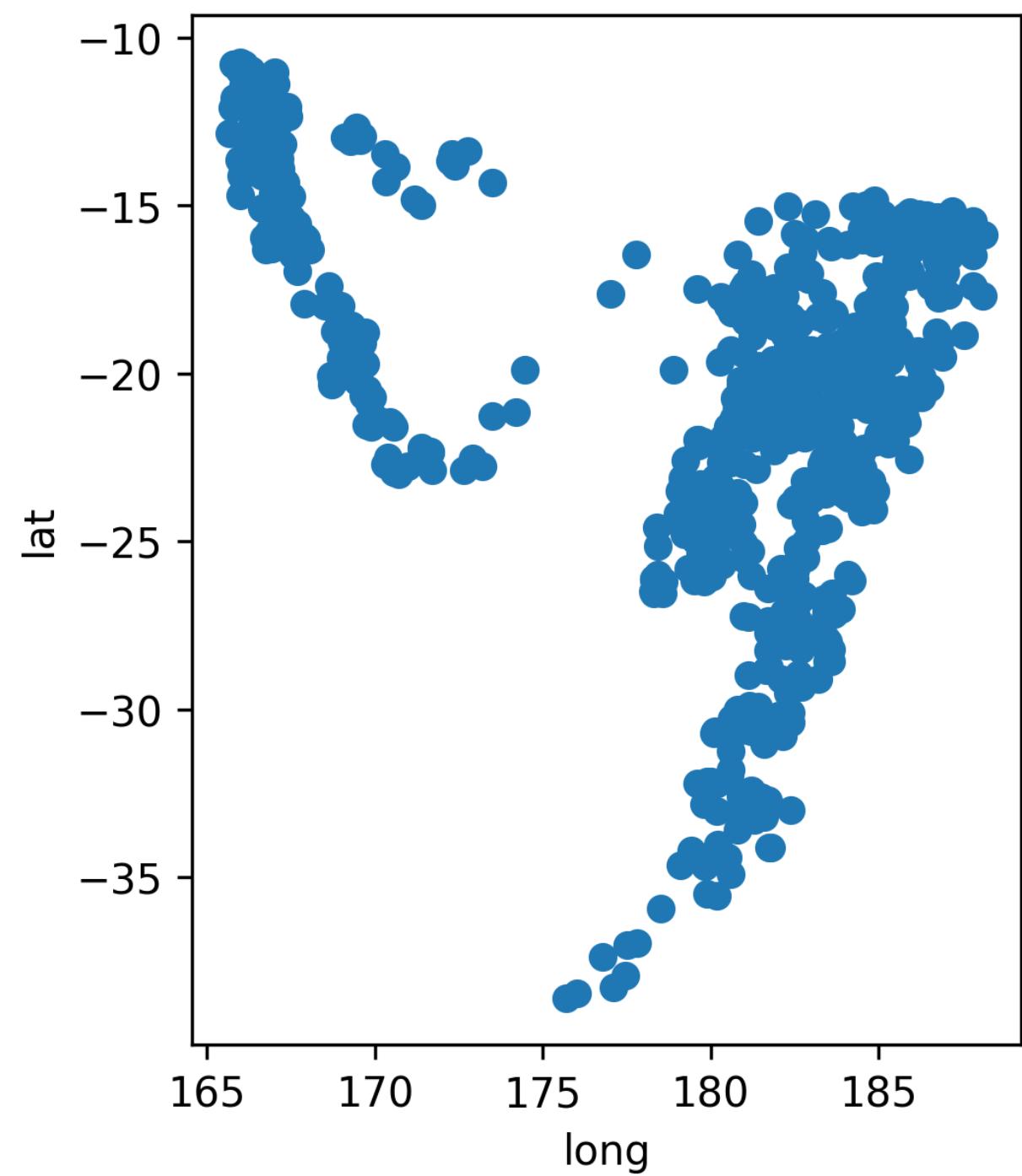


ScatterPlot

Coordinates



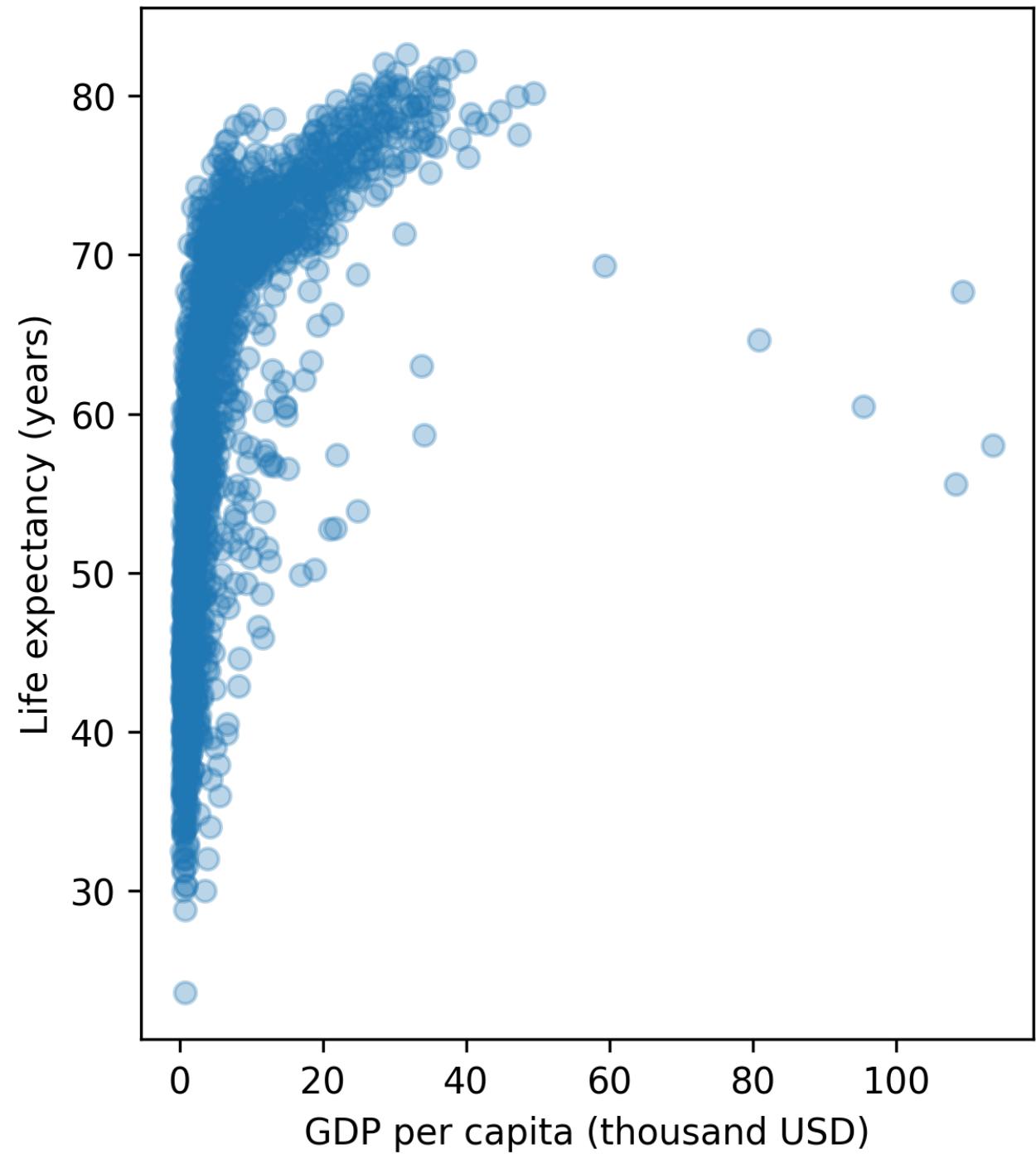
```
d.plot.scatter( "x", "y" )
```



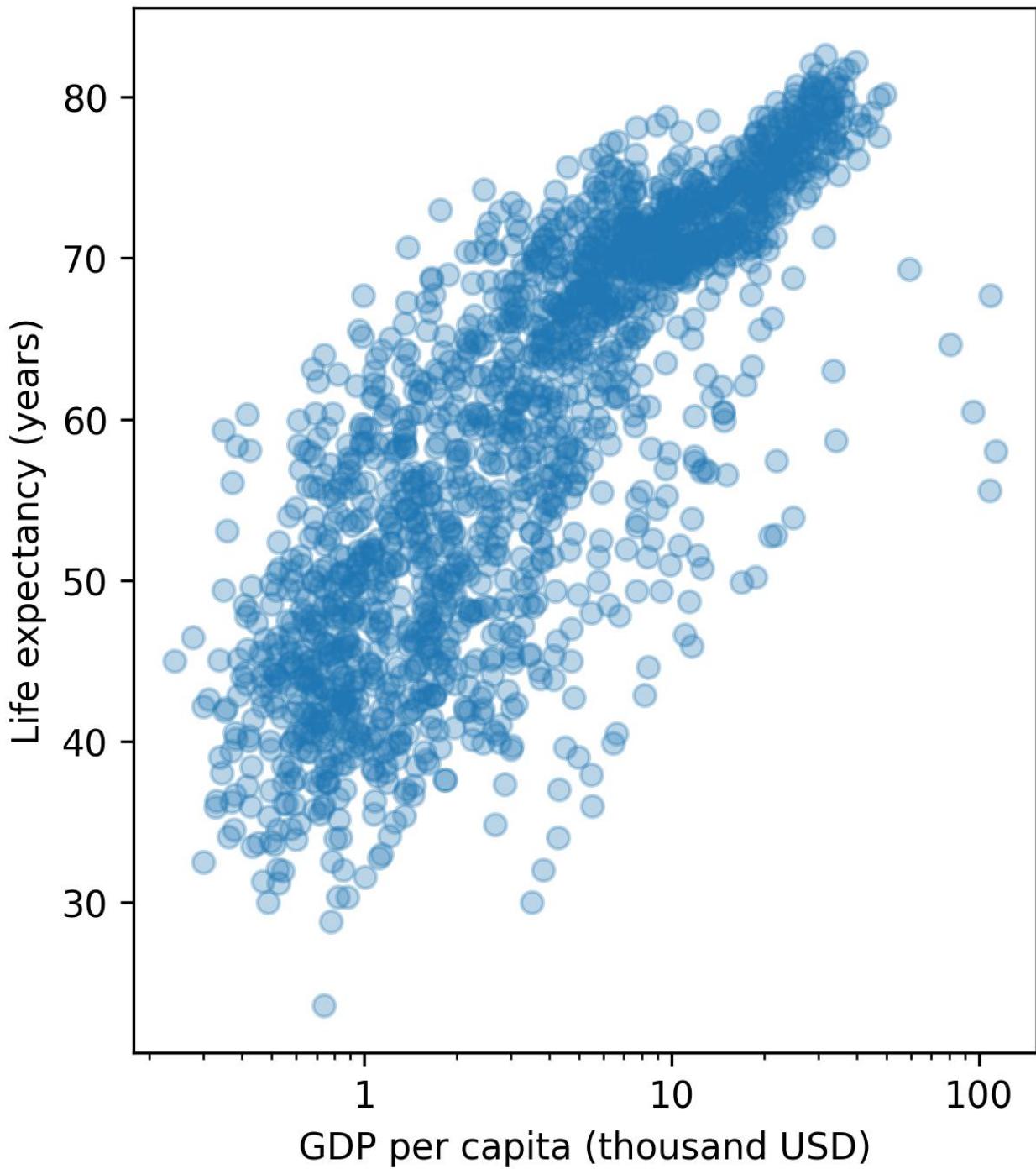
Exercise

- Generate a scatter plot of the life expectancy (vertical axis) versus the GDP per capita (horizontal axis).
- What do you notice?
- What happens if you use `plt.plot` instead of `plt.scatter`?

```
gapminder.plot.scatter(  
  'gdpPerCap',  
  'lifeExp',  
)
```

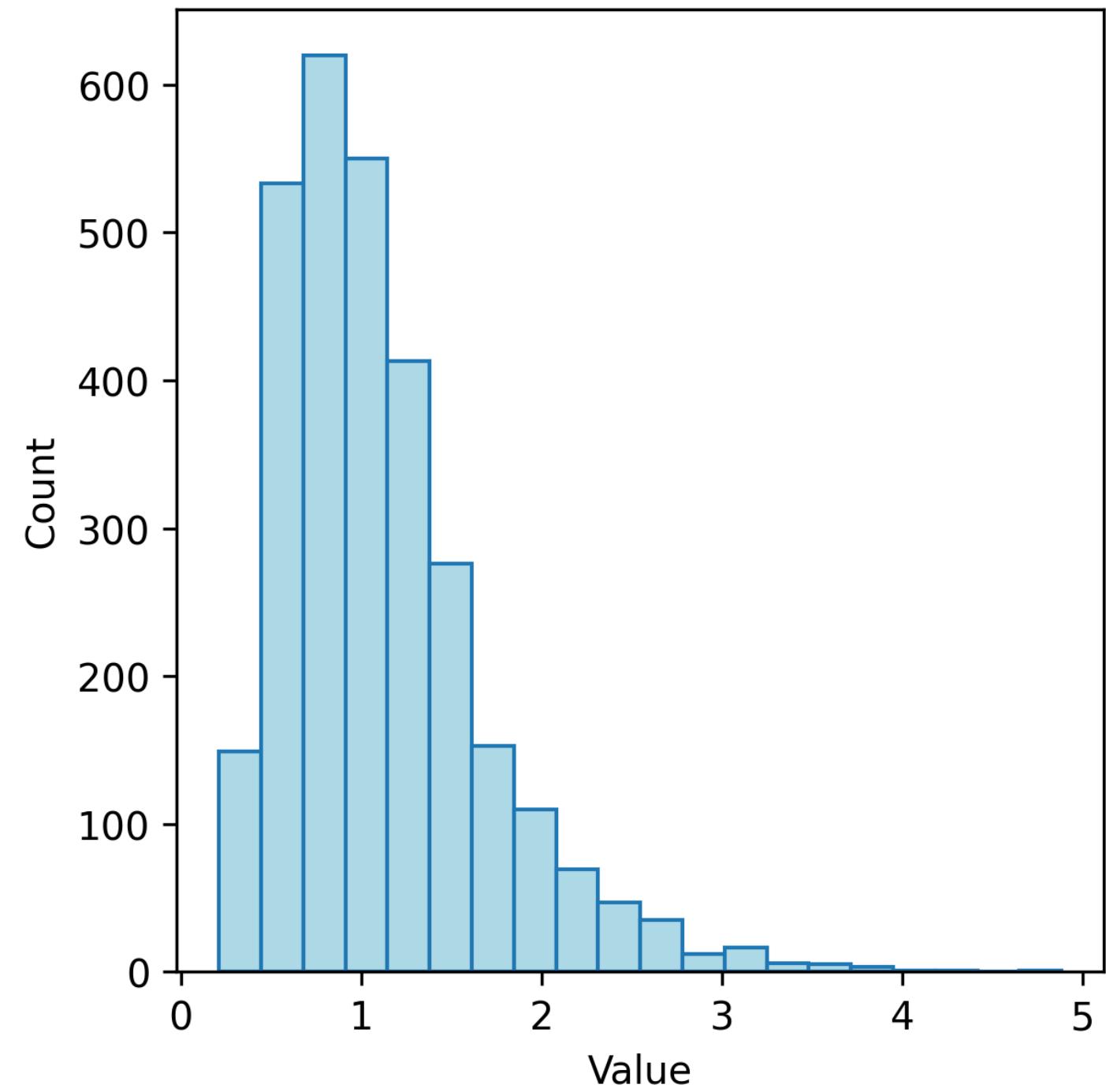


```
gapminder.plot.scatter(  
    'gdpPercap',  
    'lifeExp',  
)  
plt.xscale('log')
```

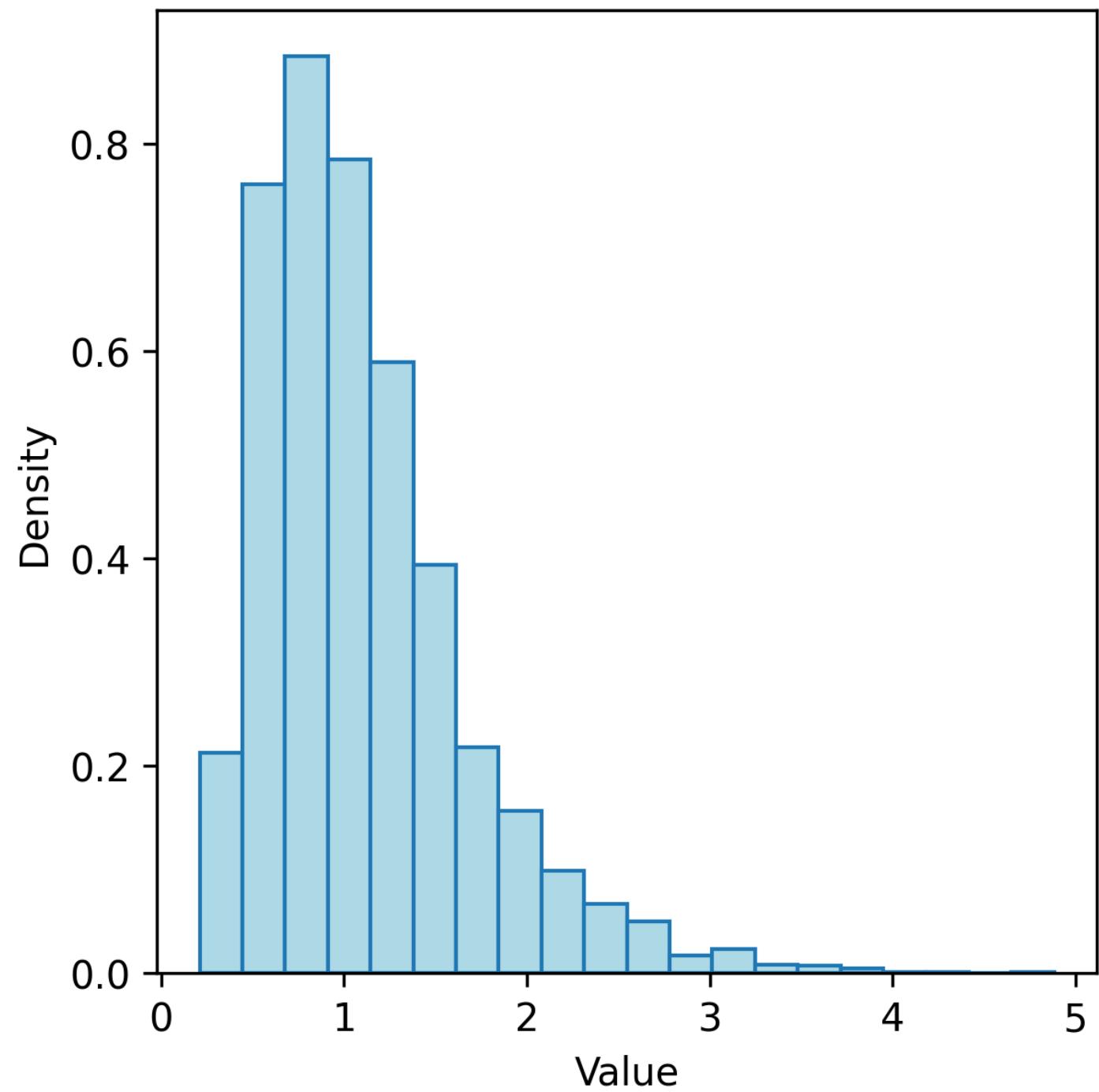


Histogram

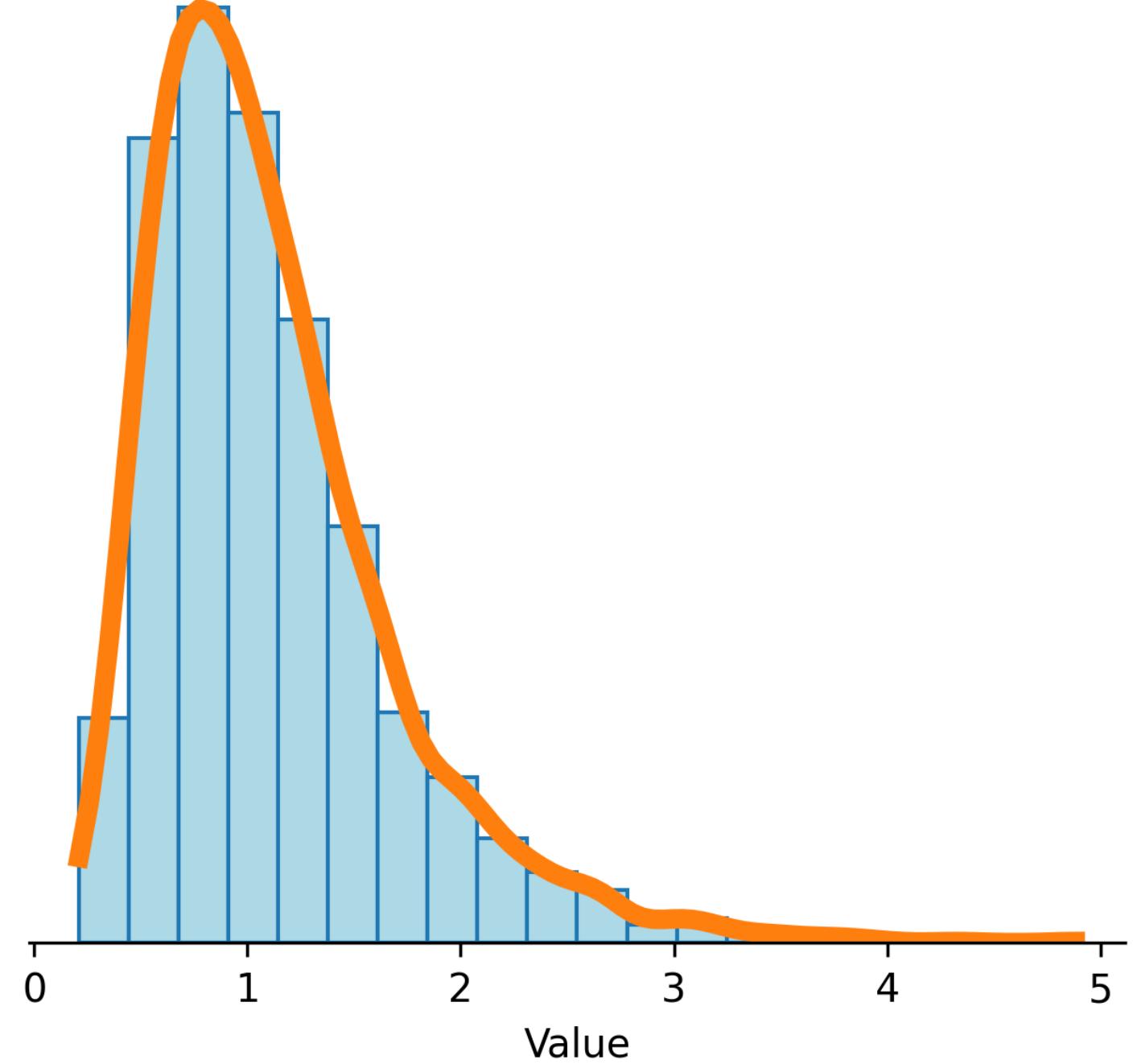
```
x.plot.hist()
```



```
x.plot.hist( density = True )
```



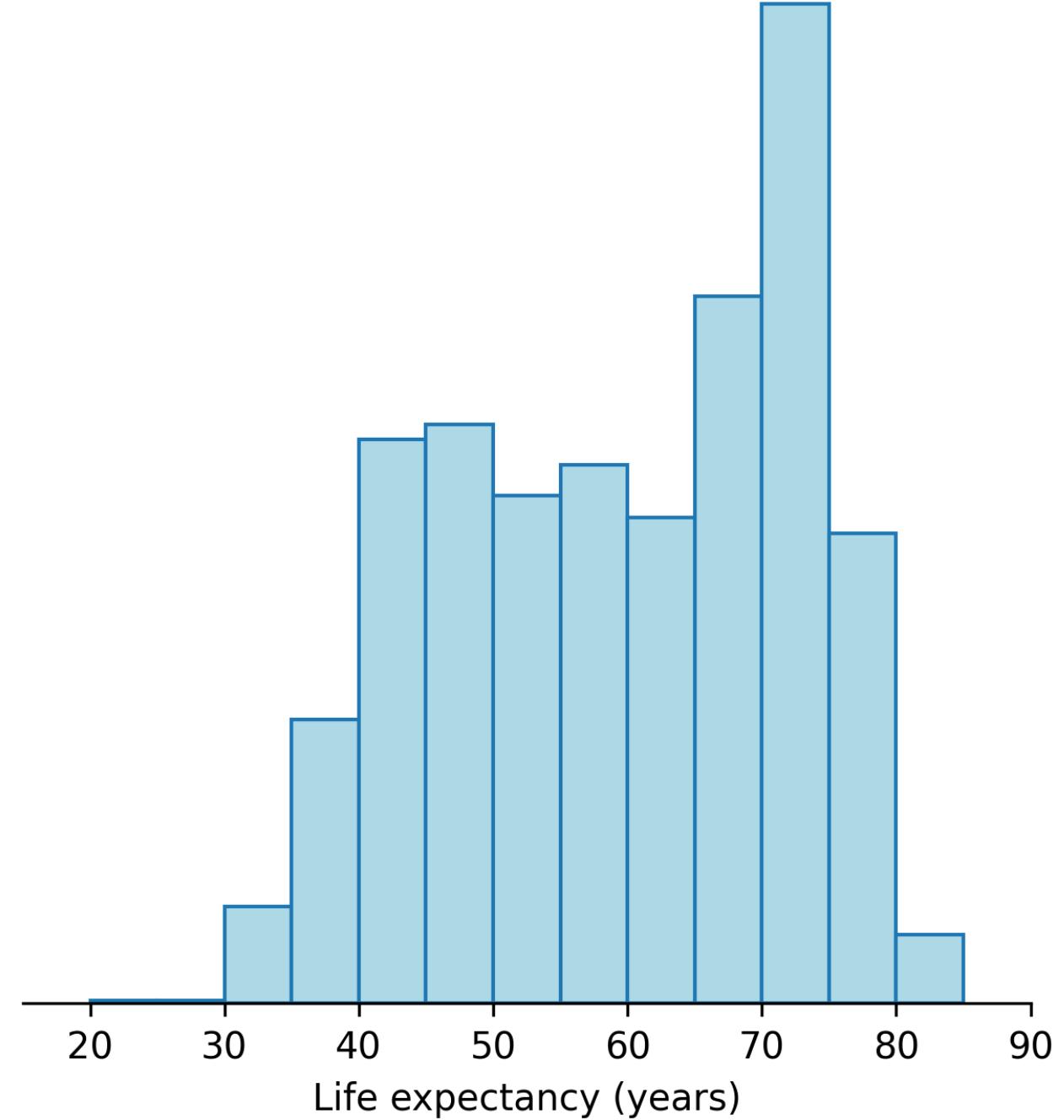
```
x.plot.hist( density = True )  
x.plot.density()
```



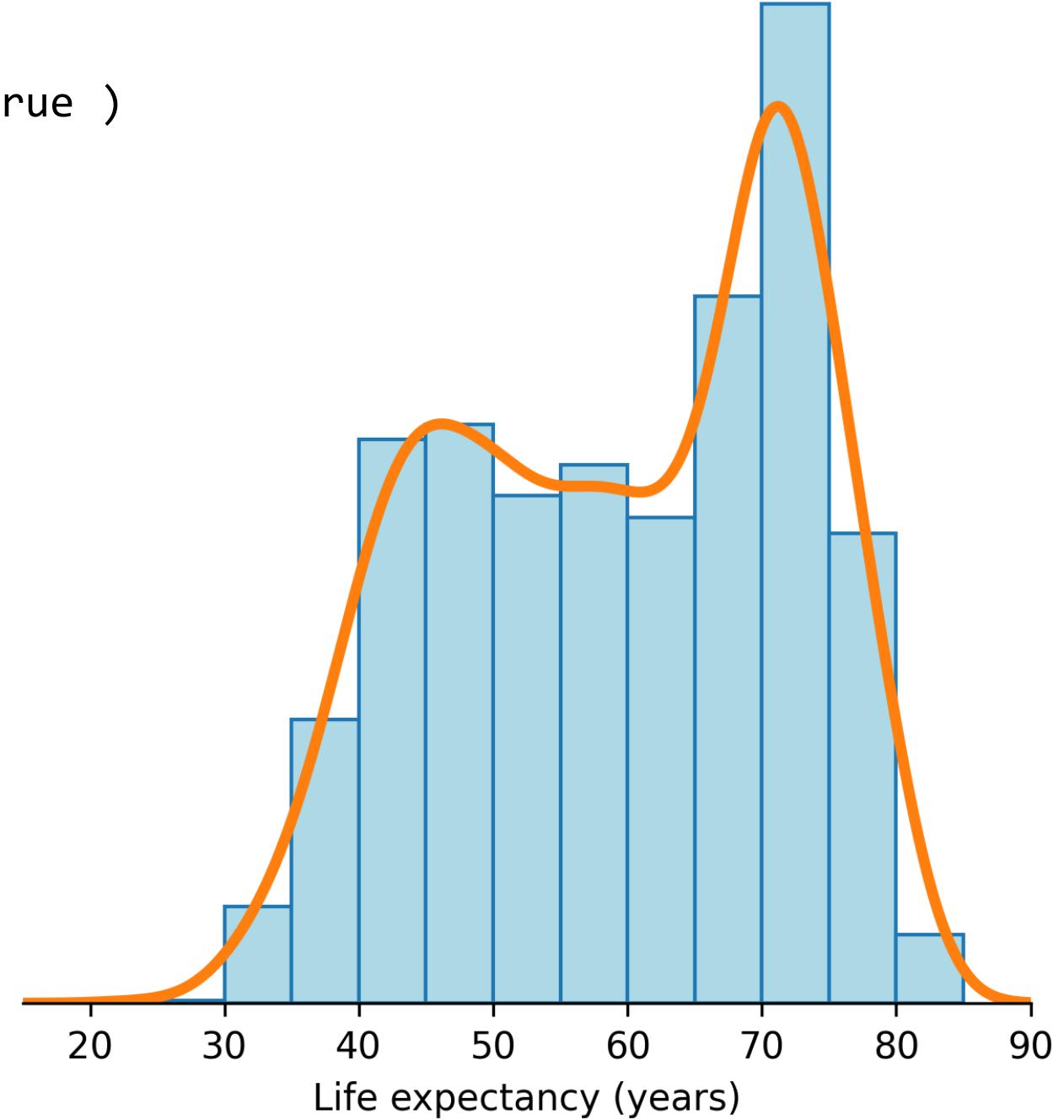
Exercise

- Plot a histogram, and the density of the life expectancy for the whole dataset
- What happens if you omit `density=True`?
- Generate histograms of the life expectancy for the first and last dates of the data. How do they differ?
- Generate a histogram of the population, for the whole dataset. How does it compare with the previous histogram?

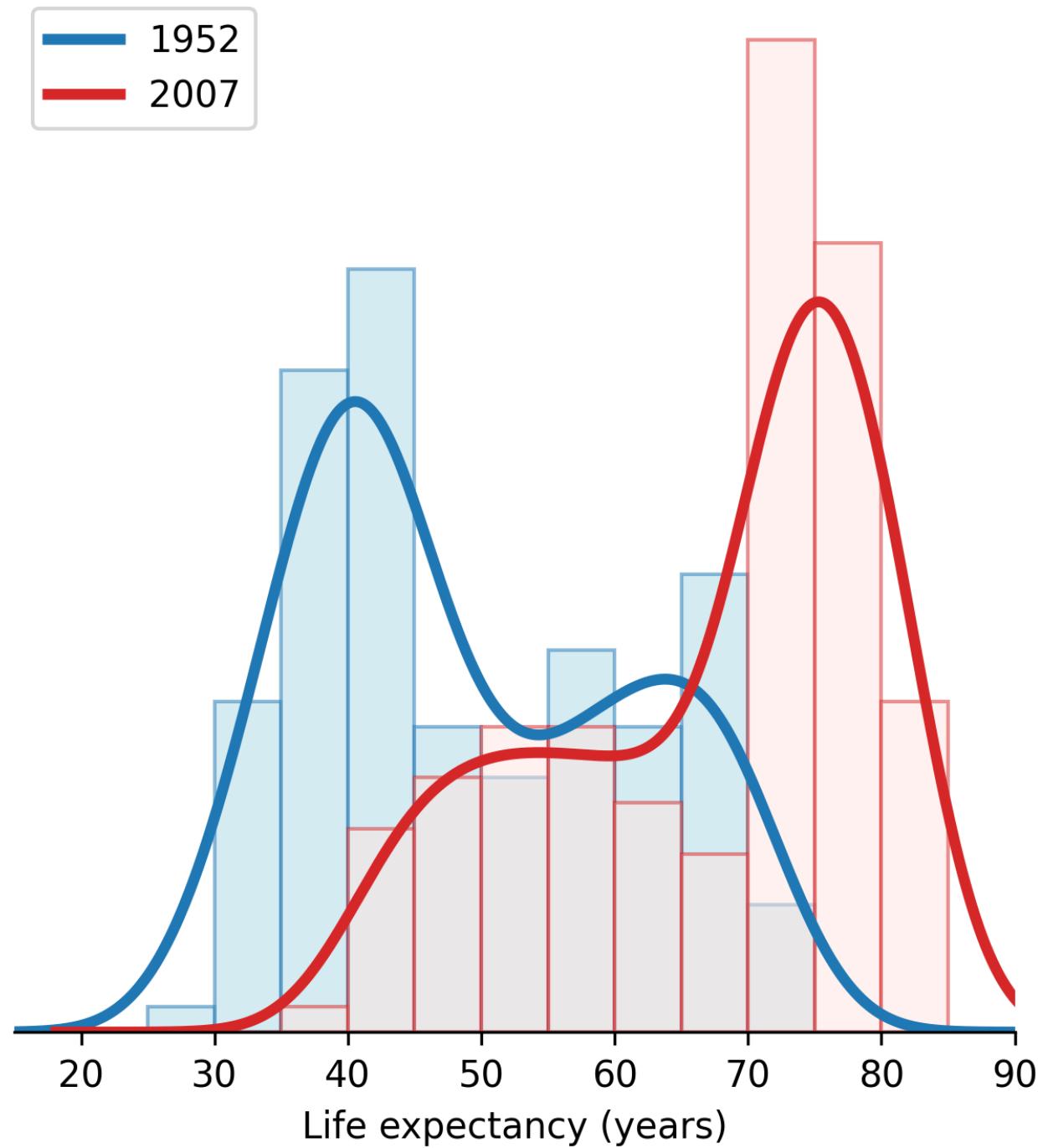
```
gapminder['lifeExp'].plot.hist()
```

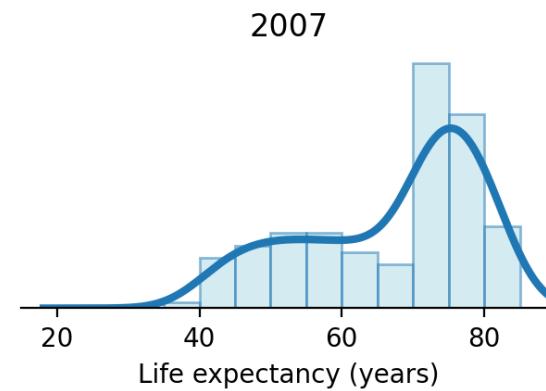
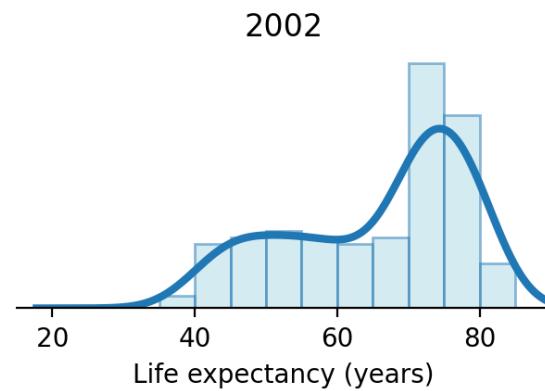
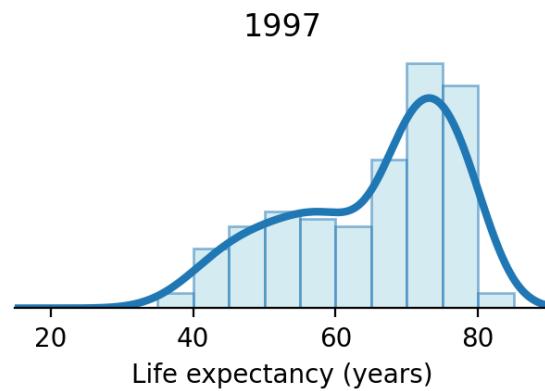
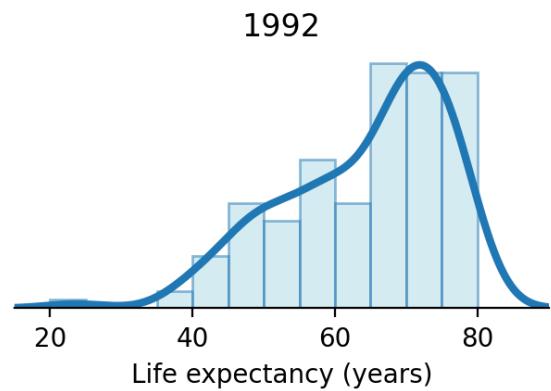
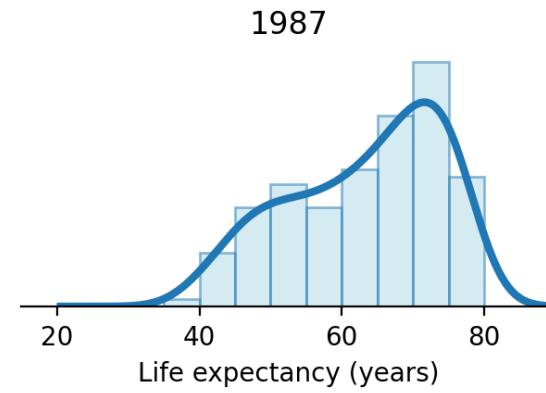
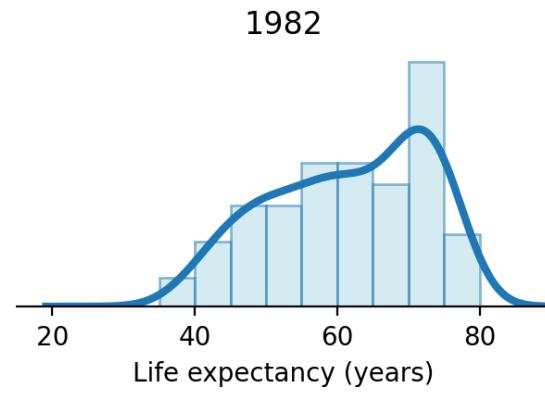
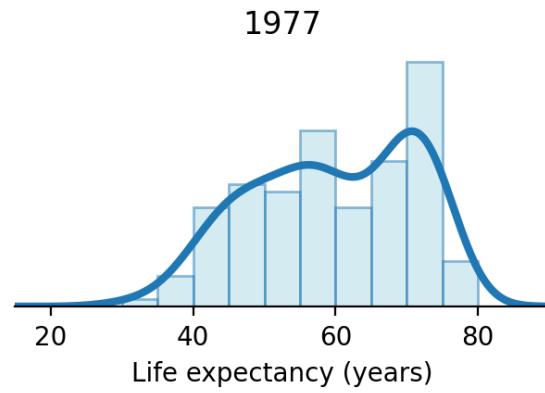
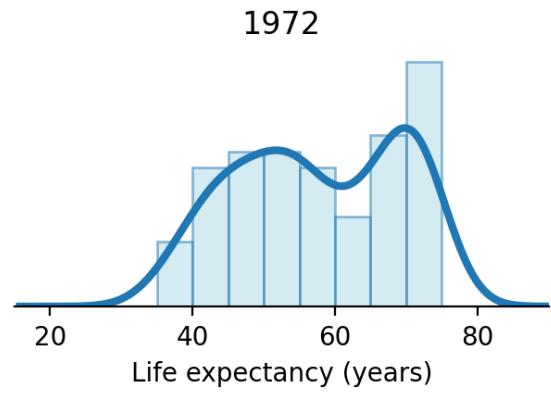
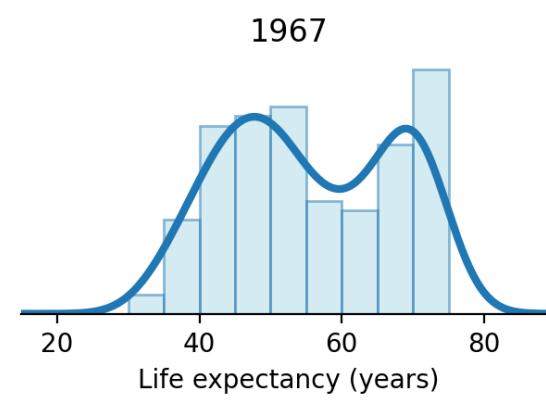
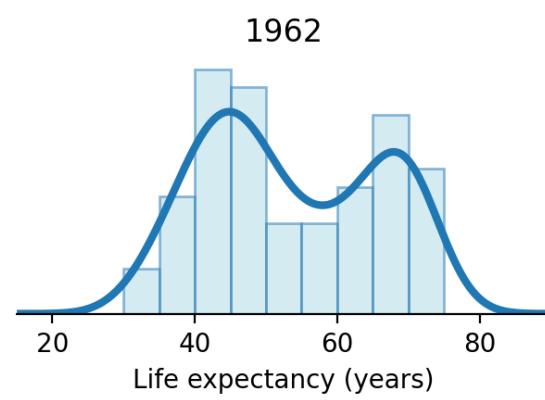
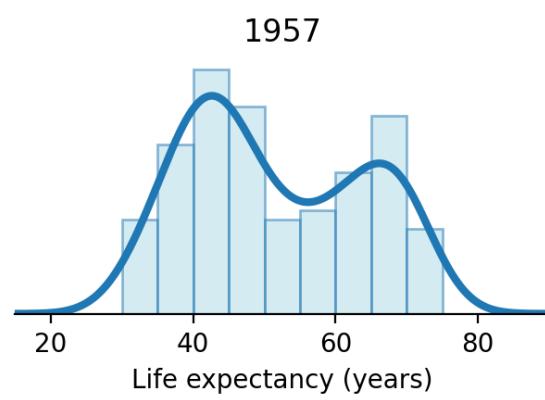
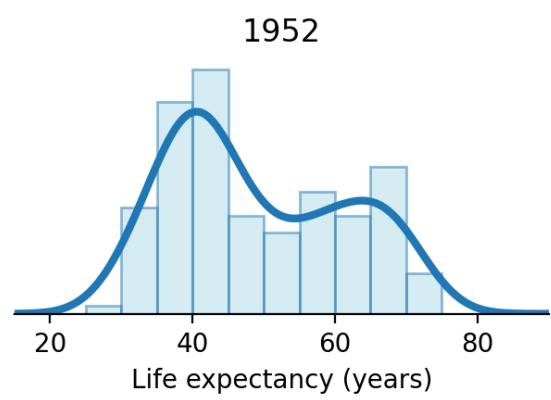


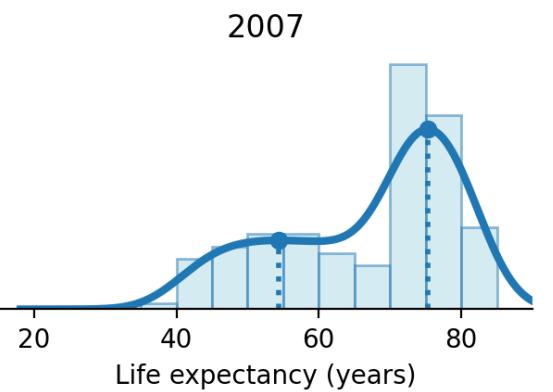
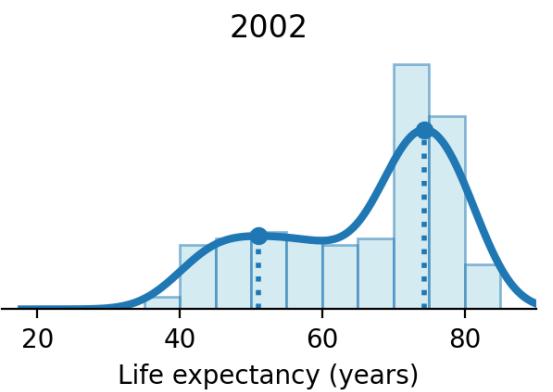
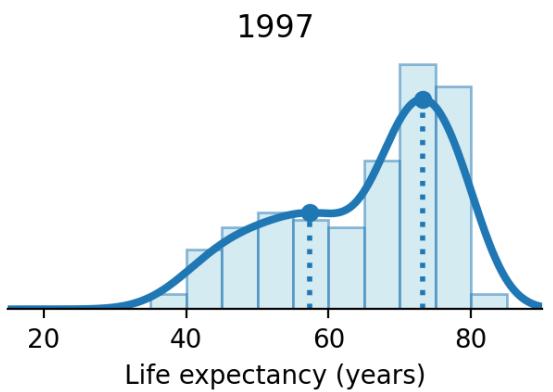
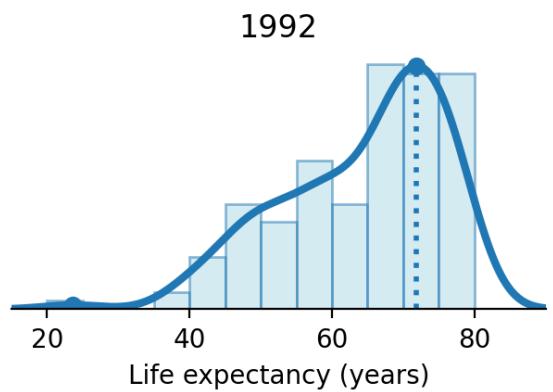
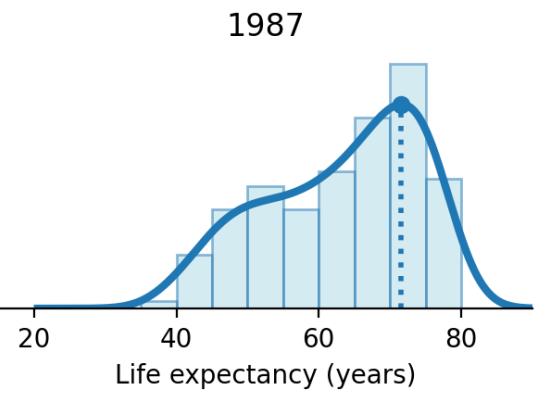
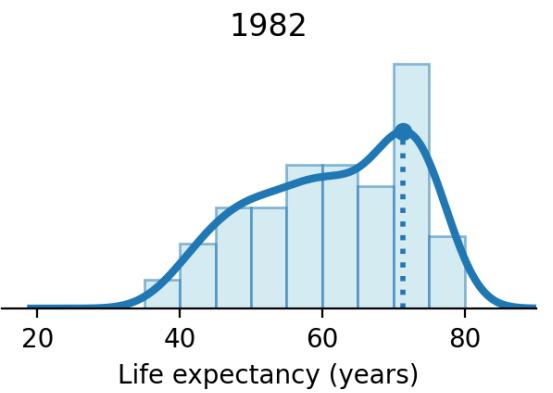
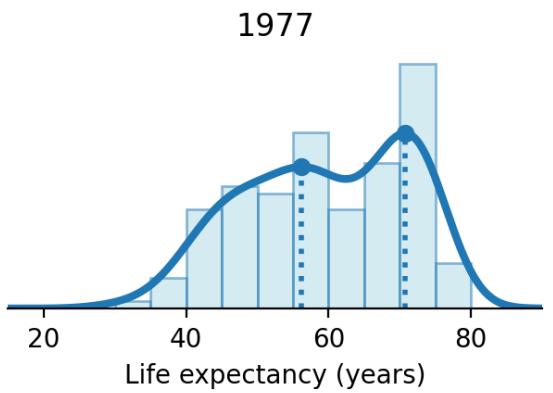
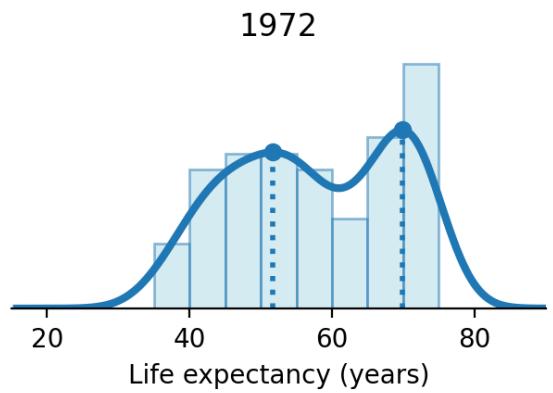
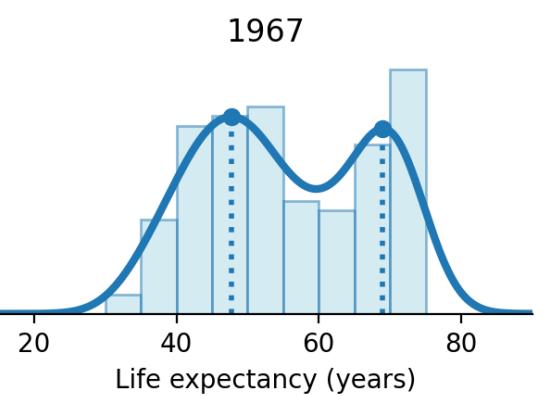
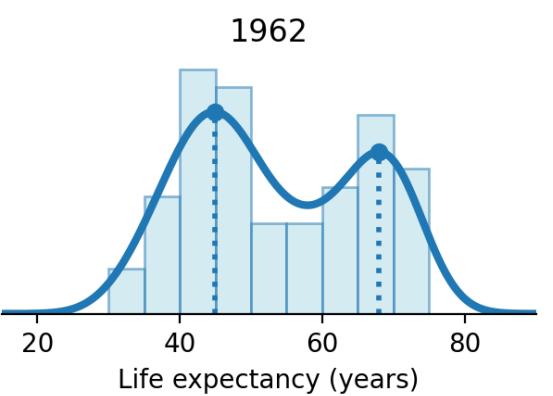
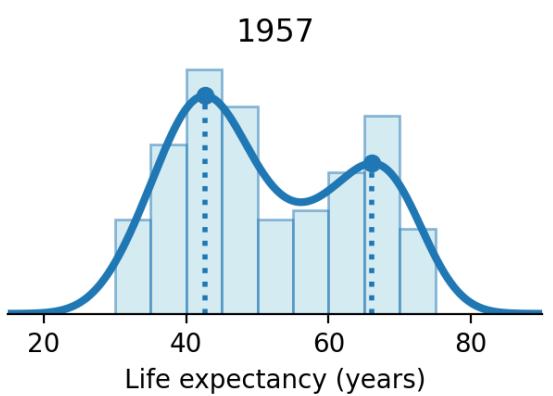
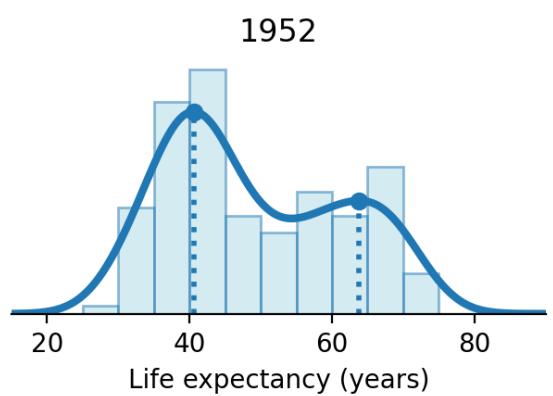
```
gapminder['lifeExp'].plot.hist( density = True )  
gapminder['lifeExp'].plot.density()
```



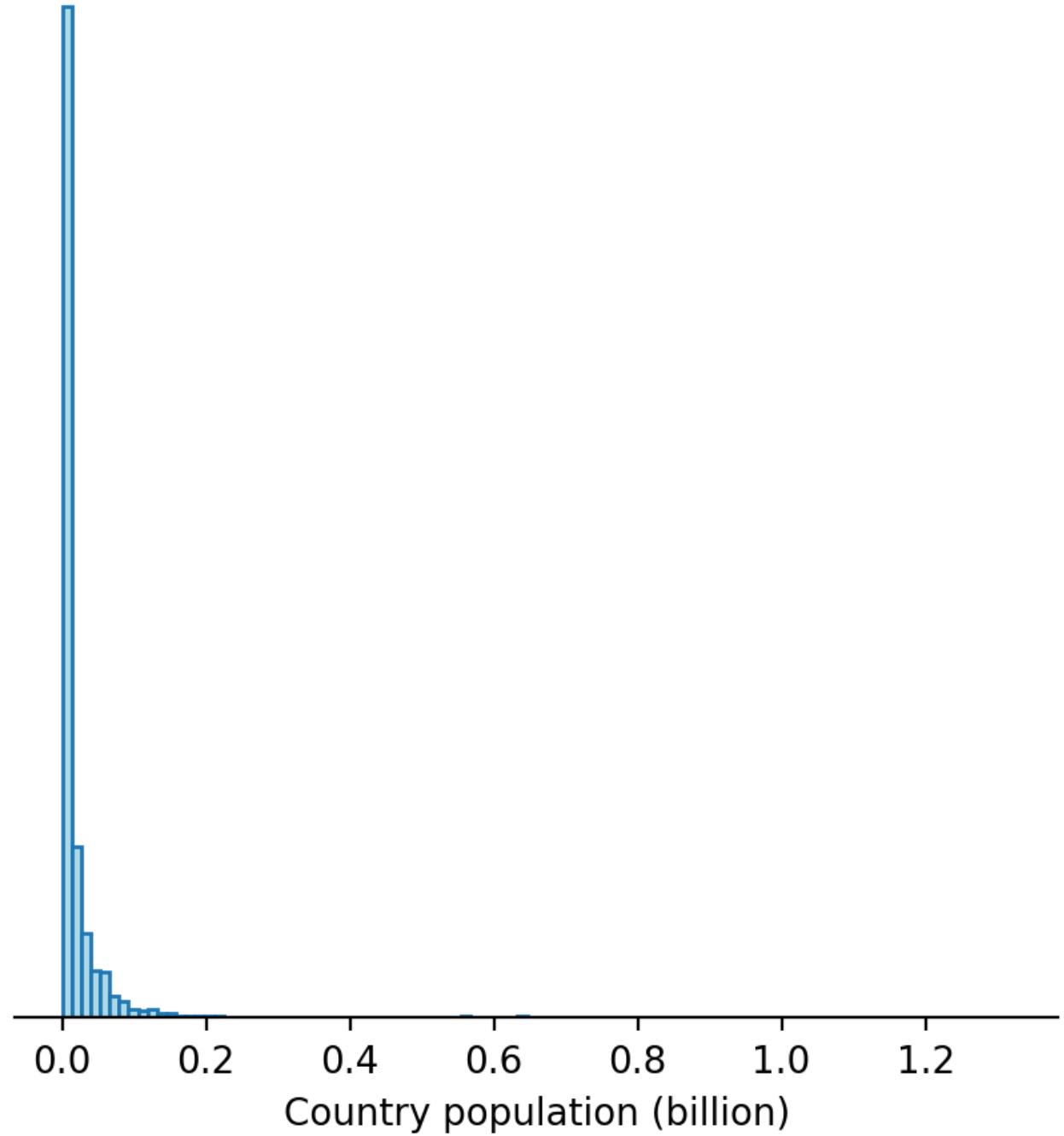
```
first_date = gapminder['year'].min()  
last_date = gapminder['year'].max()  
  
start = gapminder['year'] == first_date  
end = gapminder['year'] == last_date  
  
gapminder[start]['lifeExp'].plot.hist()  
gapminder[end]['lifeExp'].plot.hist()
```





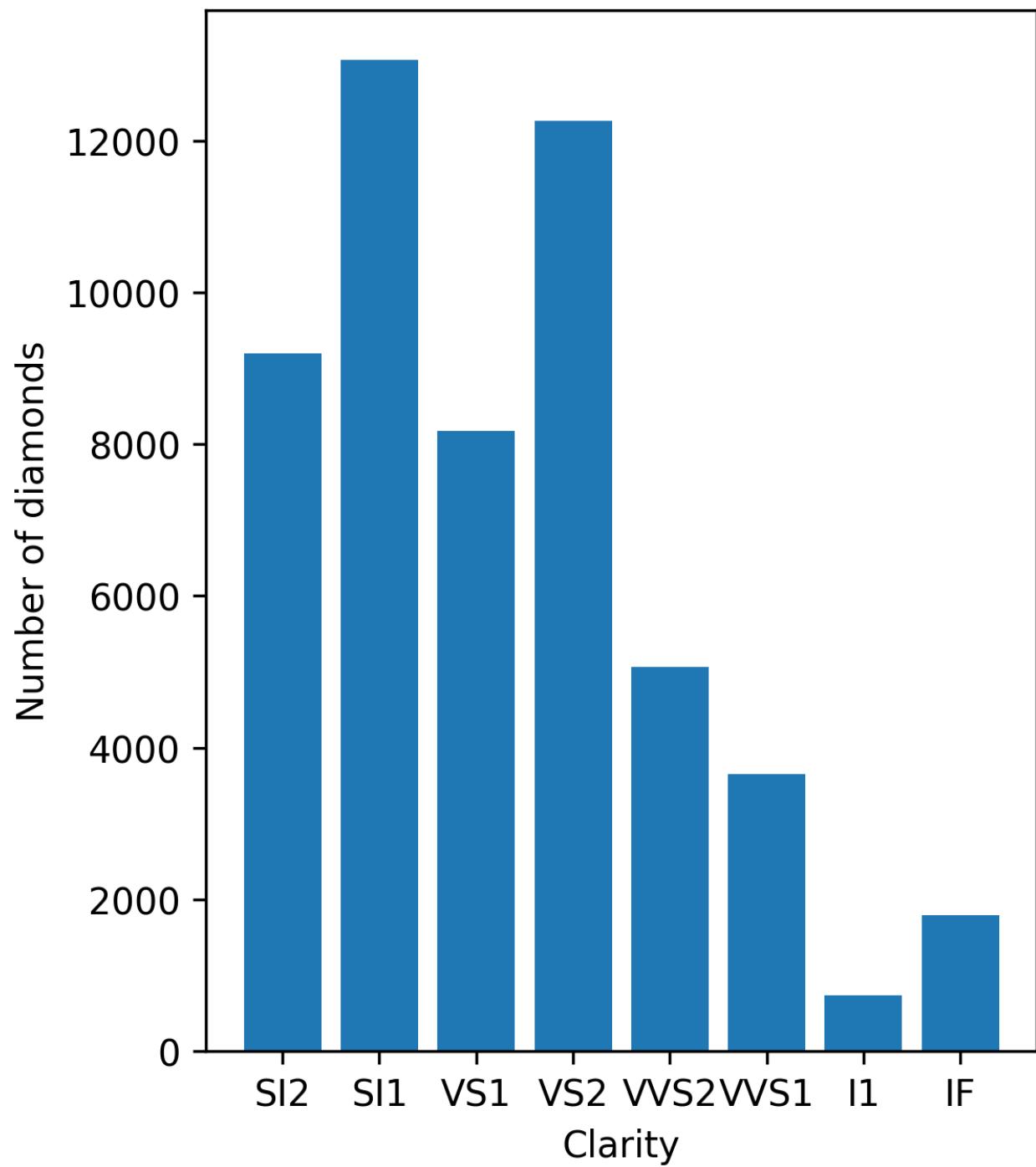


```
gapminder['pop'].plot.hist( bins = 100 )
```

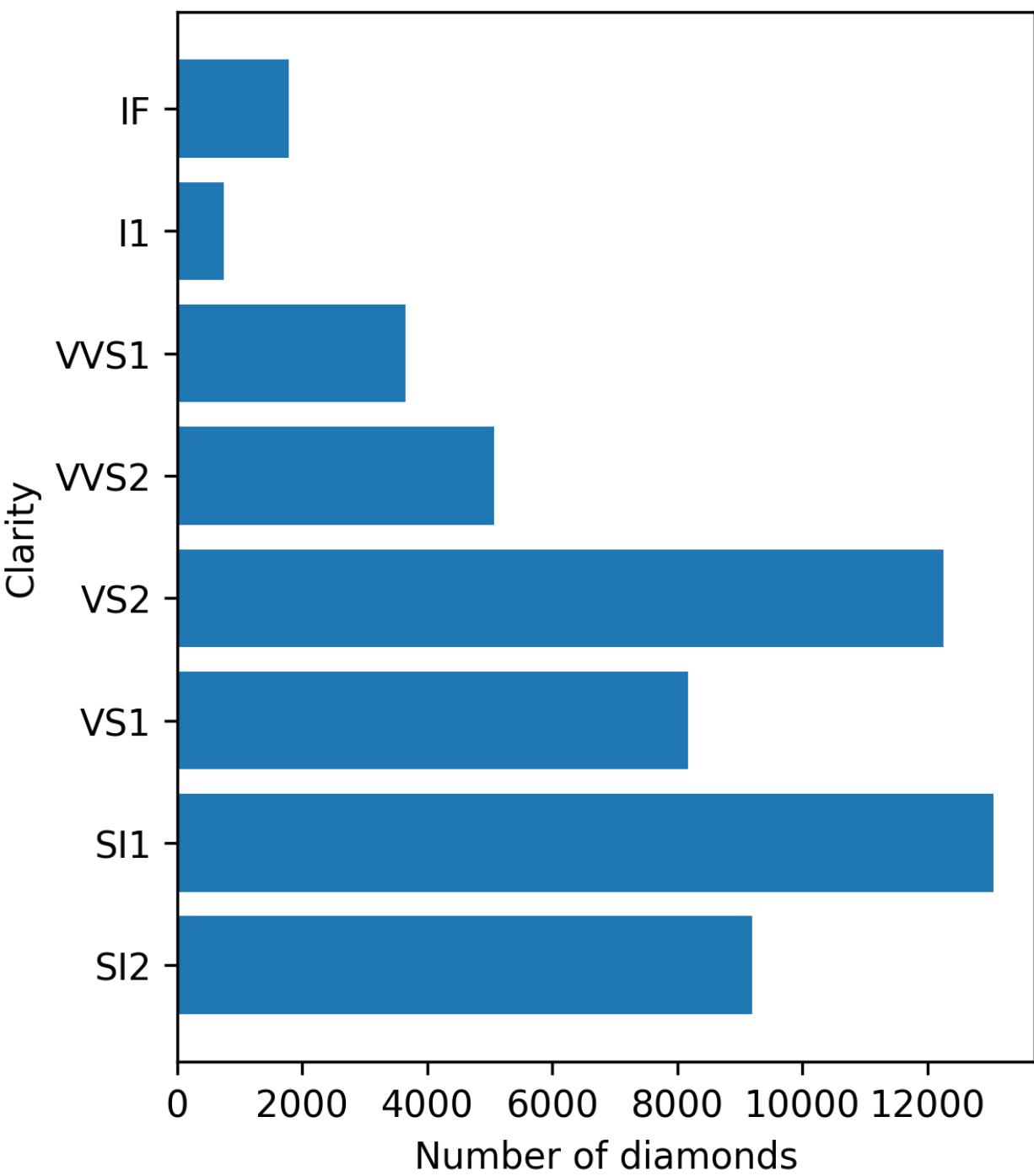


Bar plot

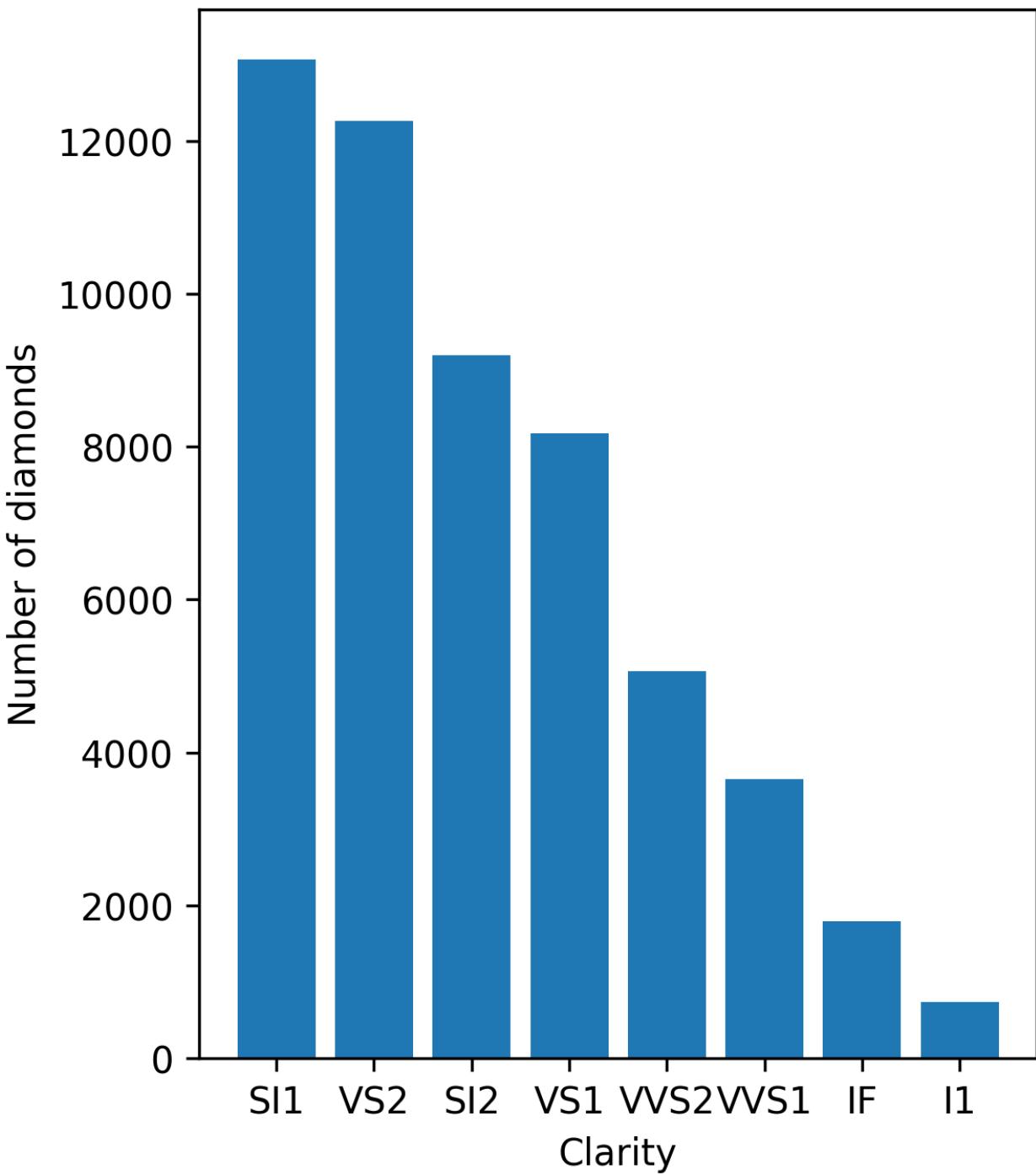
```
d.groupby('clarity').size().plot.bar()
```



```
d.groupby('clarity').size().plot.barh()
```



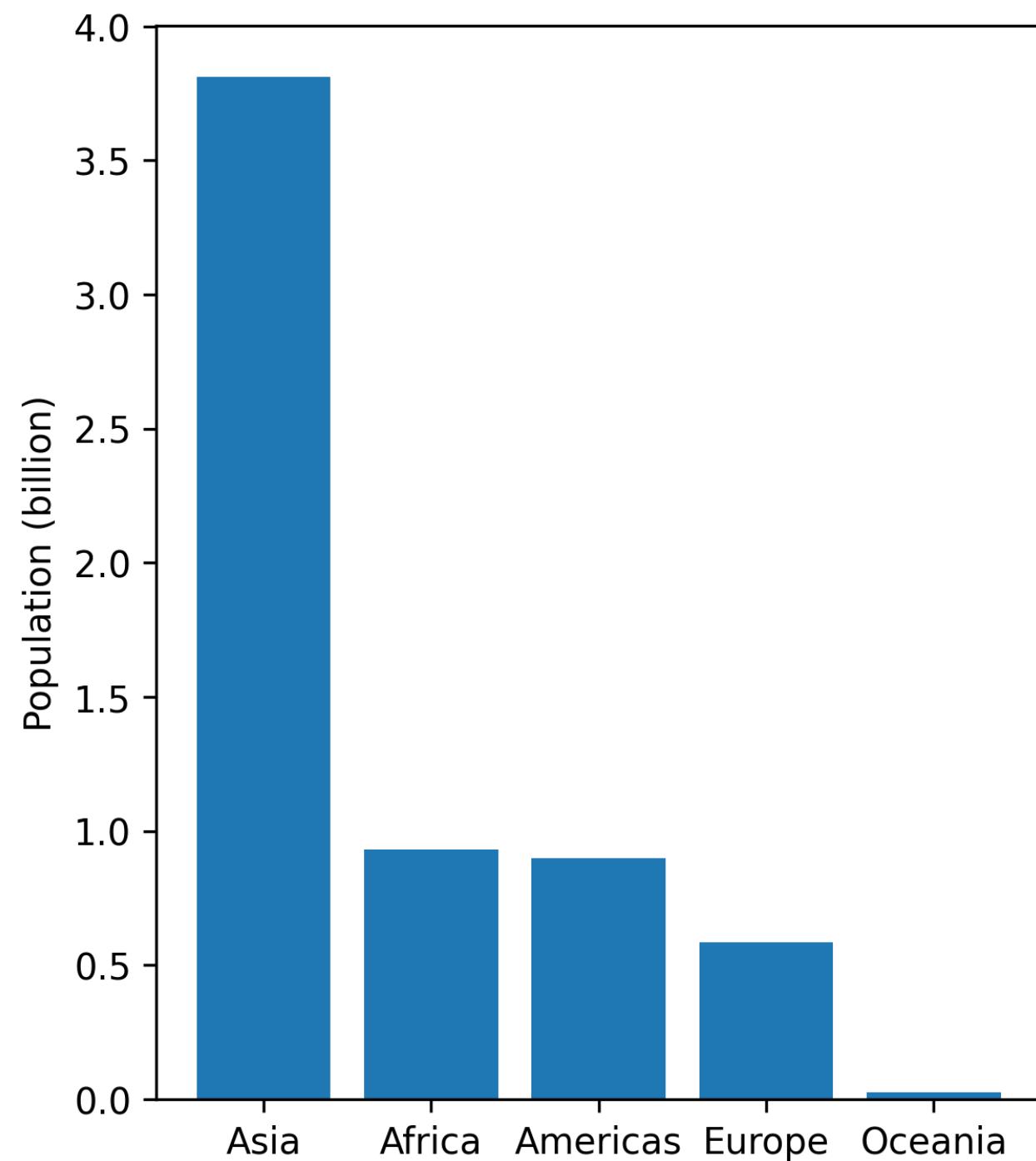
```
(  
    d  
    .groupby('clarity')  
    .size()  
    .sort_values()  
    .plot.barh()  
)
```



Exercise

- Generate a barchart of the population by continent, for the last date available
- Generate a barchart of the population by country, for the last date available
- Generate a barchart of the population by country, for the last date available, but only for the 20 largest countries

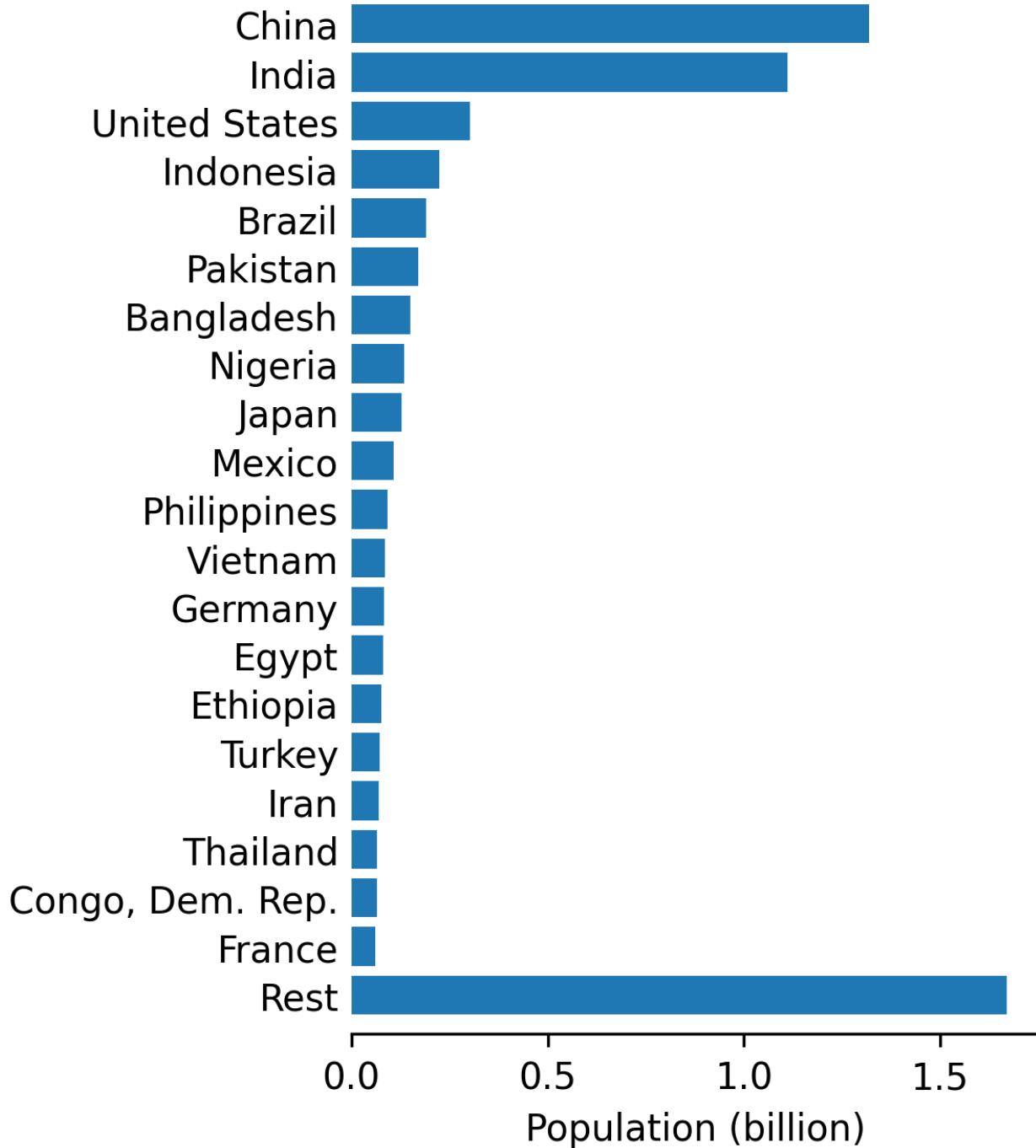
```
last_date = gapminder['year'].max()
end = gapminder['year'] == last_date
population = (
    gapminder[end]
    .groupby('continent')['pop']
    .sum()
    .sort_values( ascending = False )
)
population.plot.bar()
```



```

last_date = gapminder['year'].max()
end = gapminder['year'] == last_date
population = (
    gapminder[end][['country', 'pop']]
    .sort_values('pop', ascending = False)
)
a = 20
population = pd.concat([
    population.iloc[:a],
    pd.DataFrame([
        {
            'country': 'Rest',
            'pop': population.iloc[a:]['pop'].sum()
        }
    ]),
])
population = population.iloc[::-1,:]
population.plot.barh('country', 'pop')

```



Nicer plots

Nicer plots

```
plt.title("Title")
```

```
plt.xlabel("X")
```

```
plt.ylabel("Y")
```

```
plt.scatter(
```

```
    x, y,
```

```
    color = 'tab:red',
```

```
    s = 100,
```

```
    marker = '+',
```

```
    alpha = .5,
```

```
)
```

```
plt.plot(
```

```
    x, y,
```

```
    linestyle = ':',
```

```
    linewidth = 3,
```

```
    label = "USA",
```

```
)
```

```
plt.legend()
```

Nicer plots

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

ax.set_title("Title")
ax.set_xlabel("X")
ax.set_ylabel("Y")

ax.scatter(
    x, y,
    color = 'tab:red',
    s = 100,
    marker = '+',
    alpha = .5,
)

ax.plot(
    x, y,
    linestyle = ':',
    linewidth = 3,
    label = "USA",
)
ax.legend()

plt.show()
```

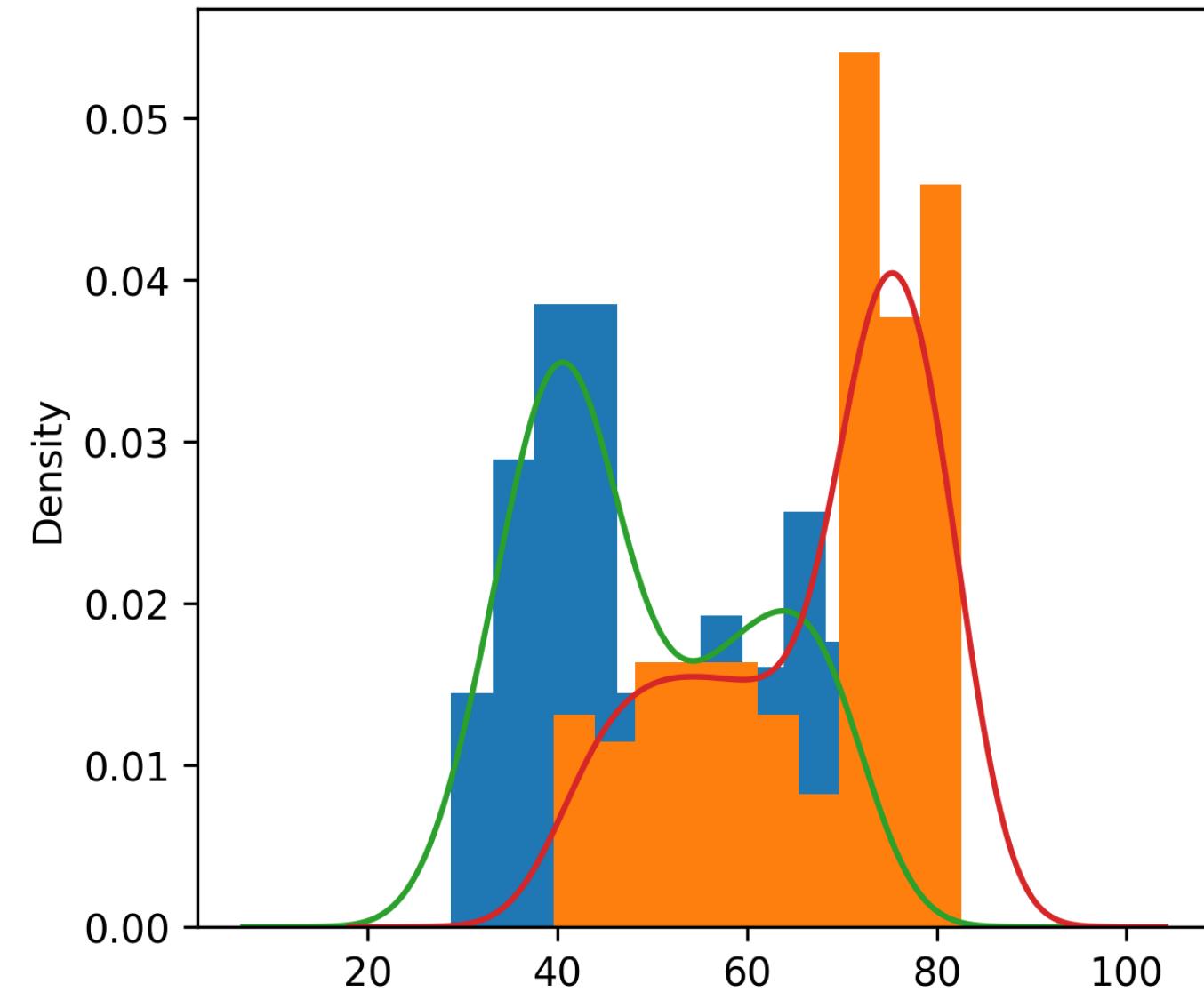
Nicer plots

<https://matplotlib.org/cheatsheets/cheatsheets.pdf>

Exercise

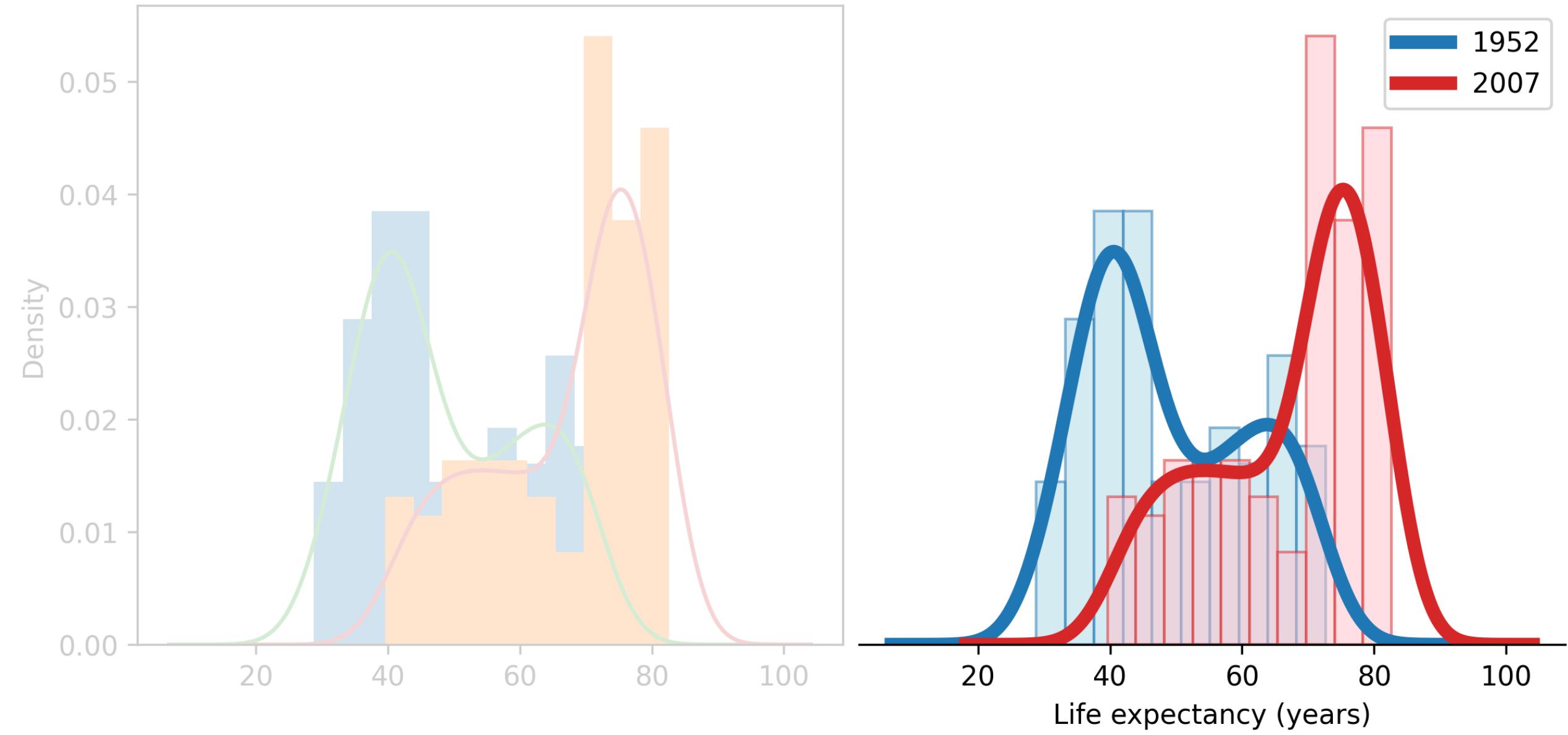
- Pick one of the earlier plots, label the axes, and add a title.
- Change the colour and thickness of the lines.
- Change the size, shape and opacity of the markers
- Add a legend

Before



Before

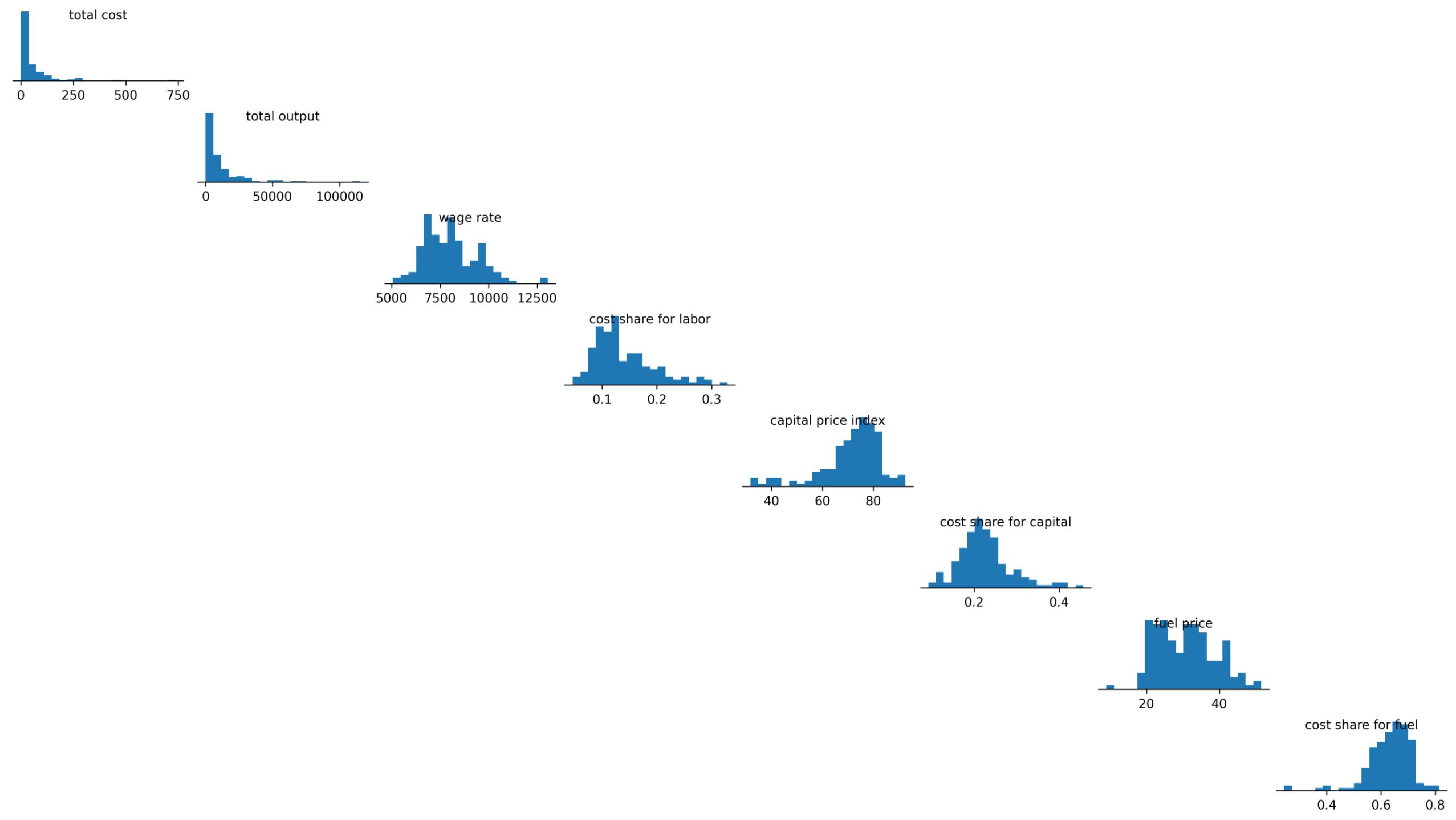
After

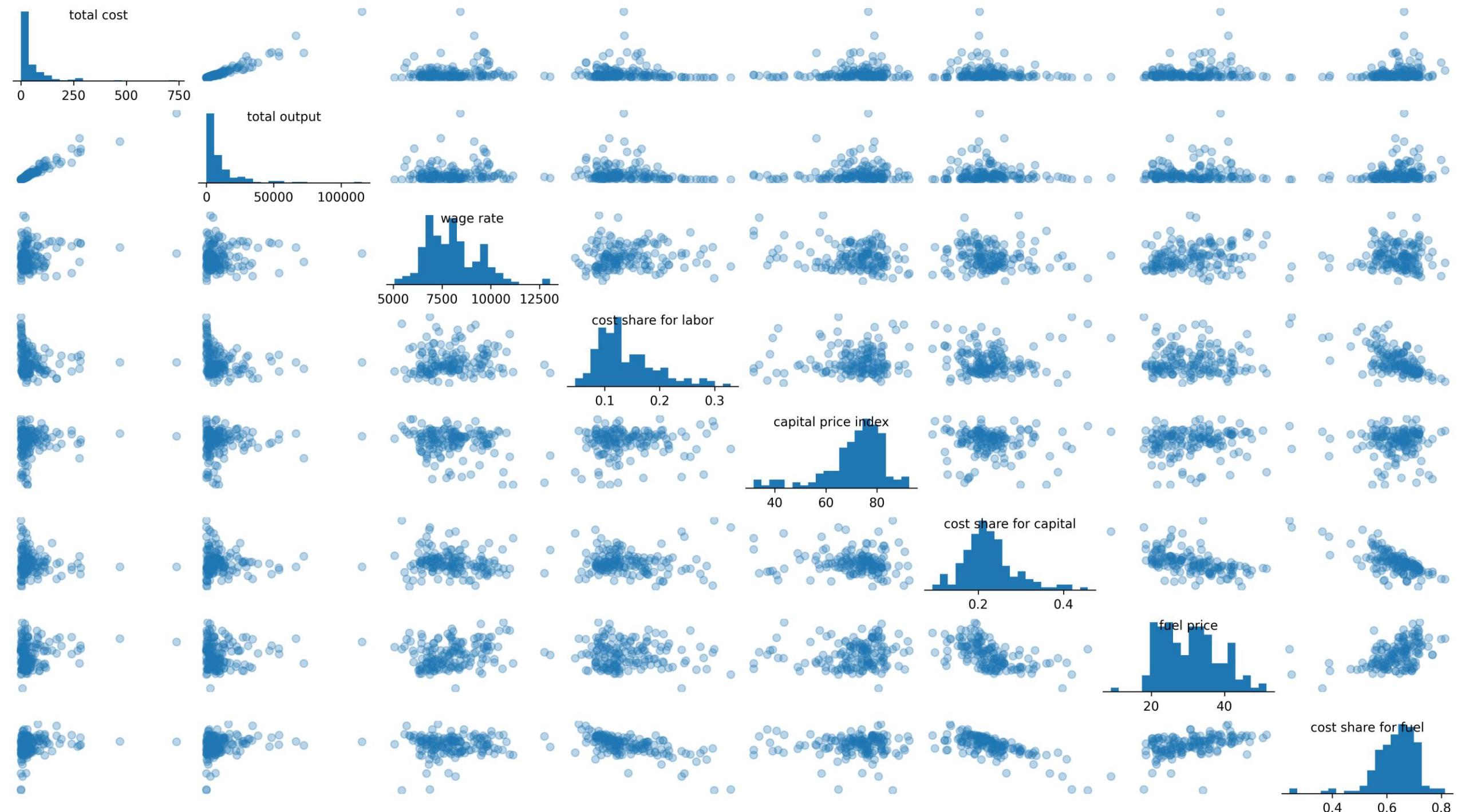


Pair plot

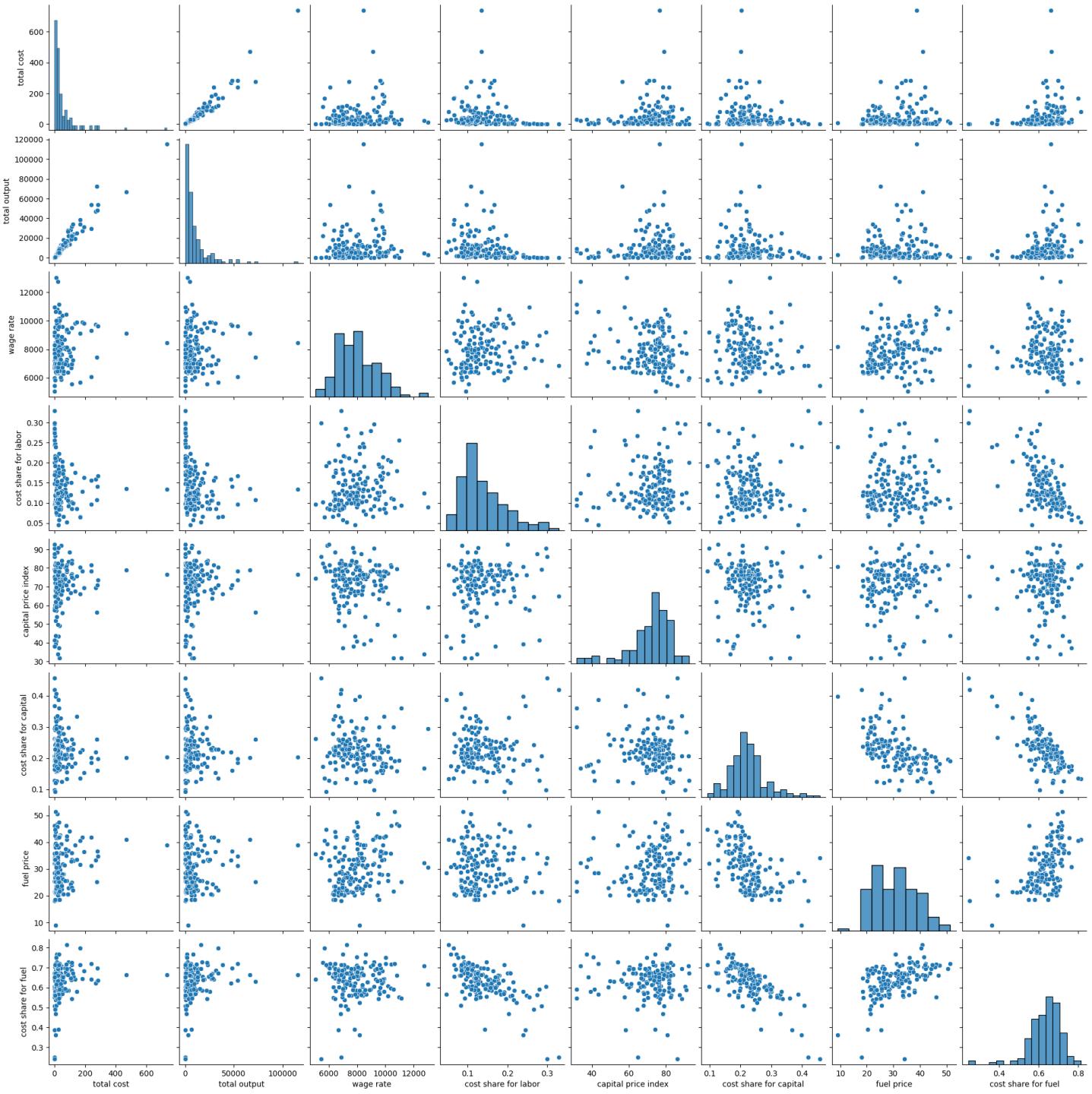
Multivariate Data

cost	q	pl	sl	pk	sk	pf	sf
0.2130	8.0	6869.47	0.3291	64.945	0.4197	18.0000	0.2512
3.0427	869.0	8372.96	0.1030	68.227	0.2913	21.0670	0.6057
9.4059	1412.0	7960.90	0.0891	40.692	0.1567	41.5300	0.7542
0.7606	65.0	8971.89	0.2802	41.243	0.1282	28.5390	0.5916
2.2587	295.0	8218.40	0.1772	71.940	0.1623	39.2000	0.6606





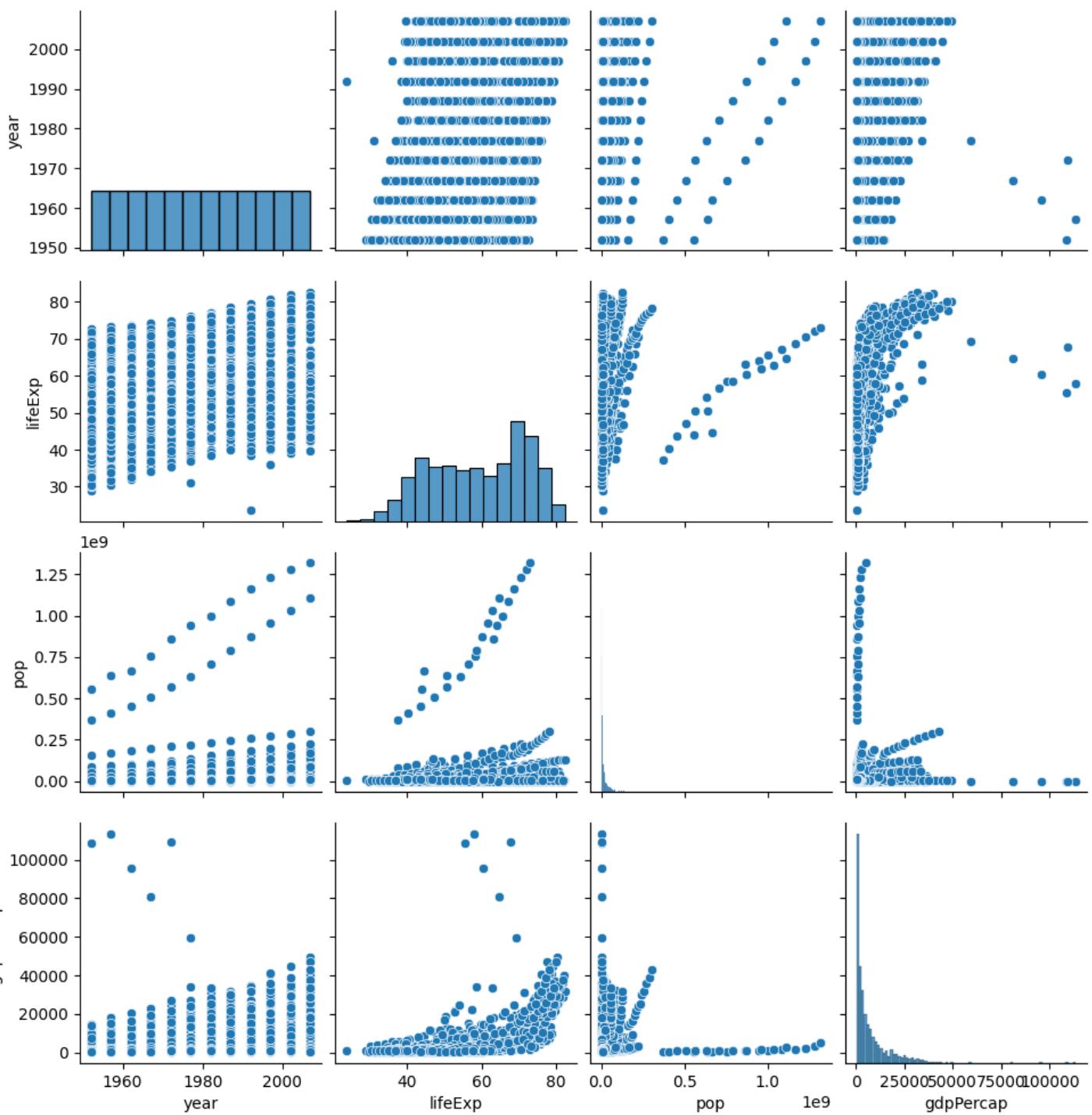
```
sns.pairplot(d)
```

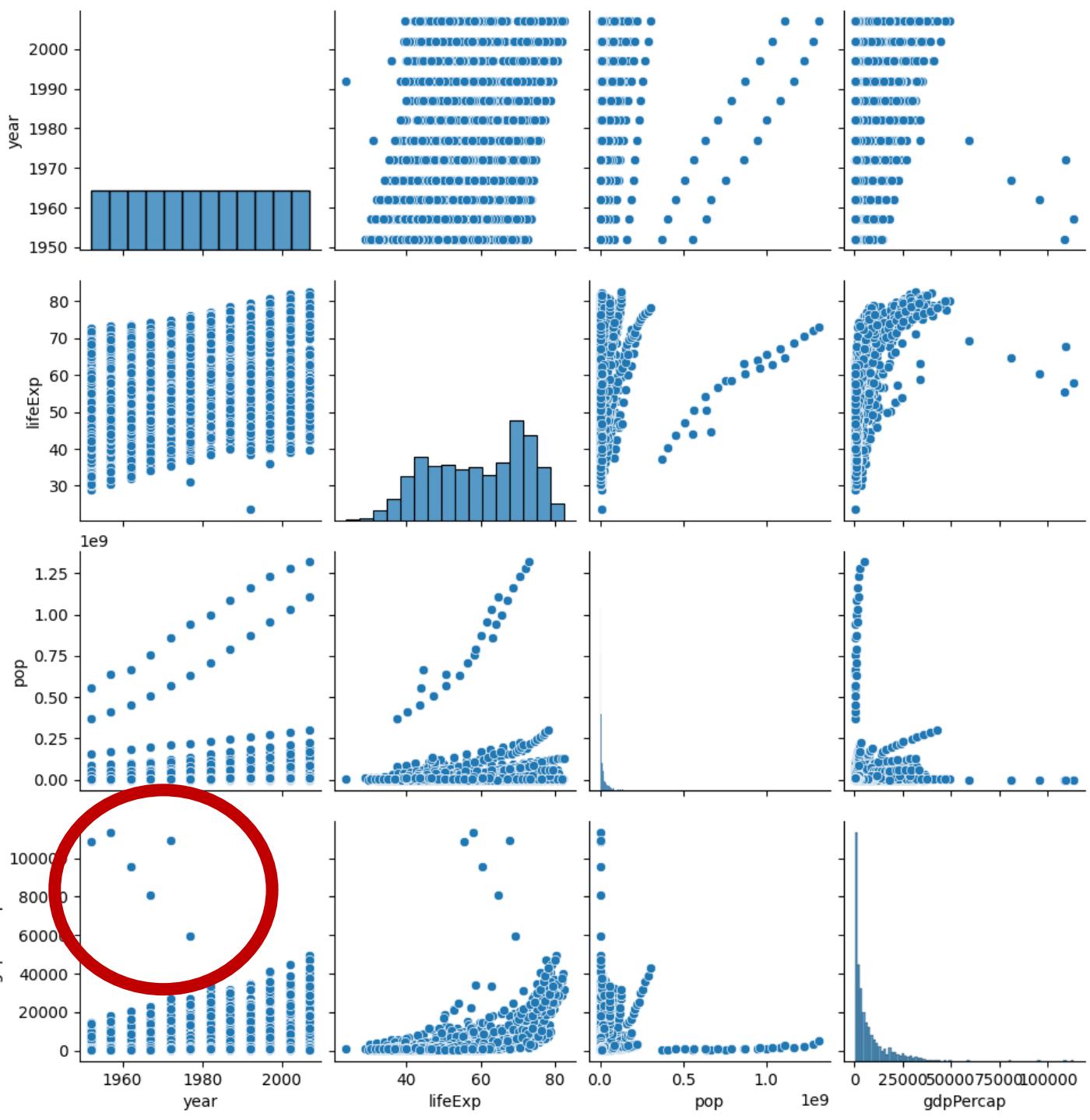


Exercise

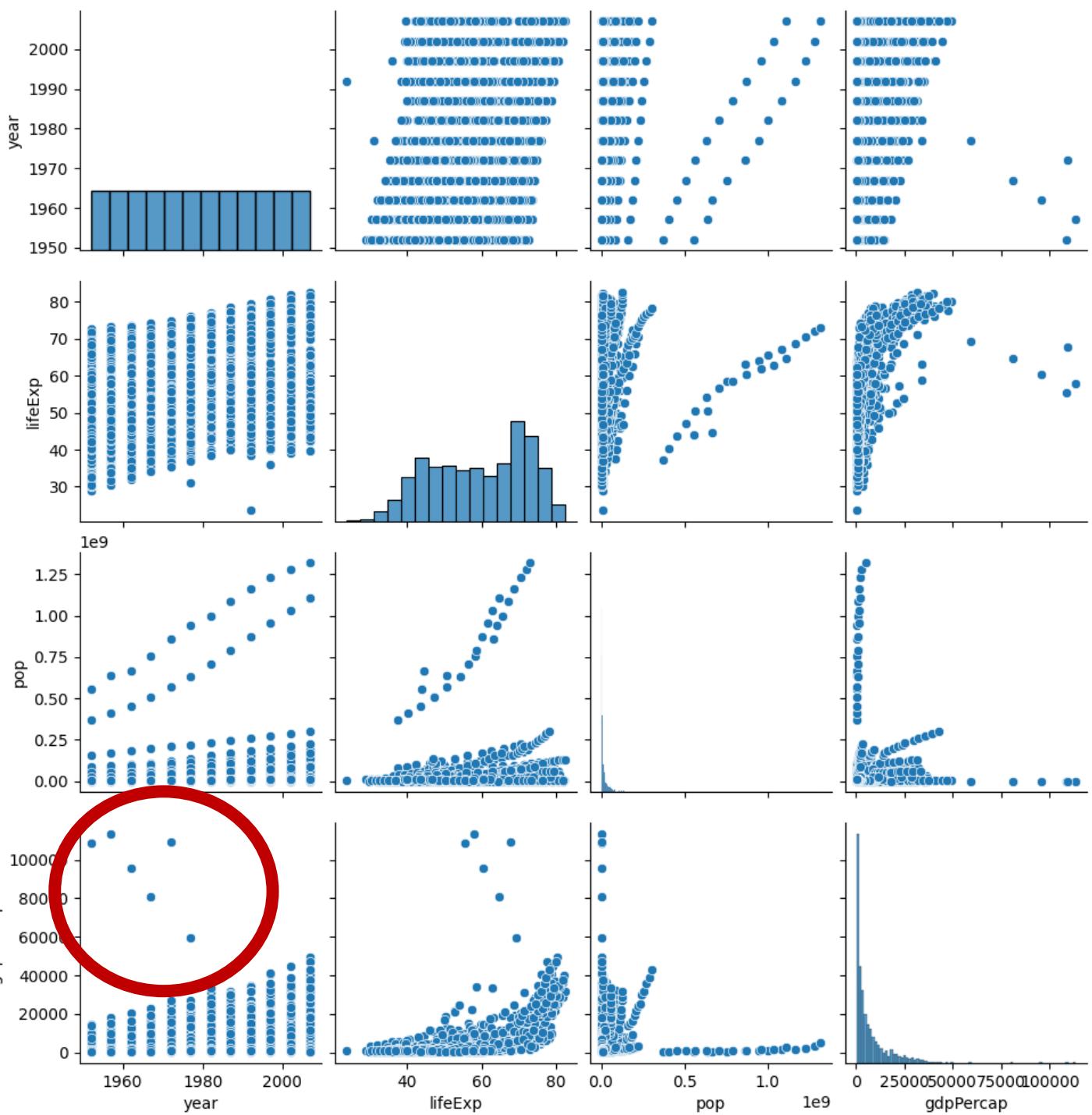
- Generate a pairplot for the gapminder data
- Do you notice anything?

```
sns.pairplot( gapminder )
```

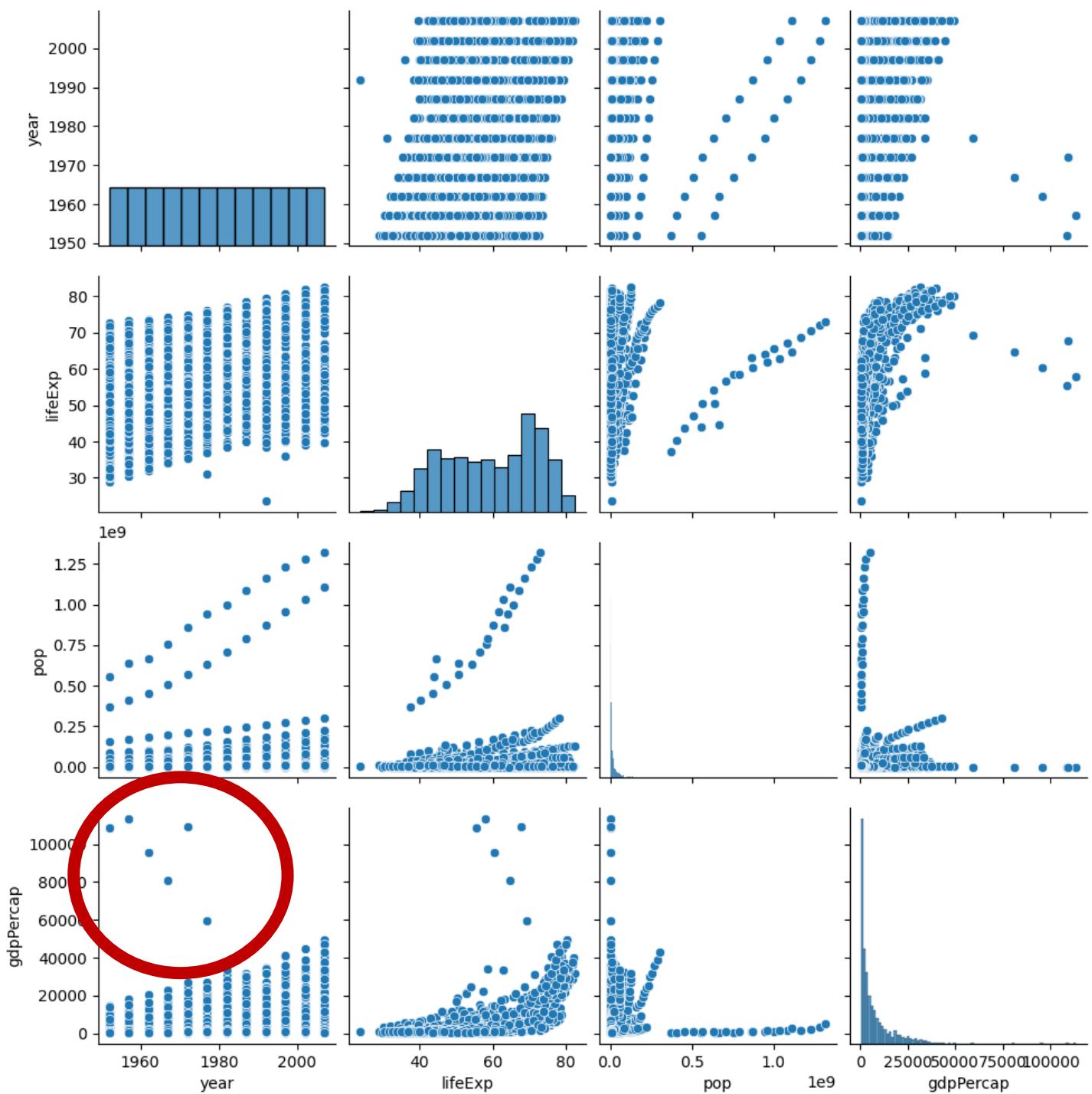
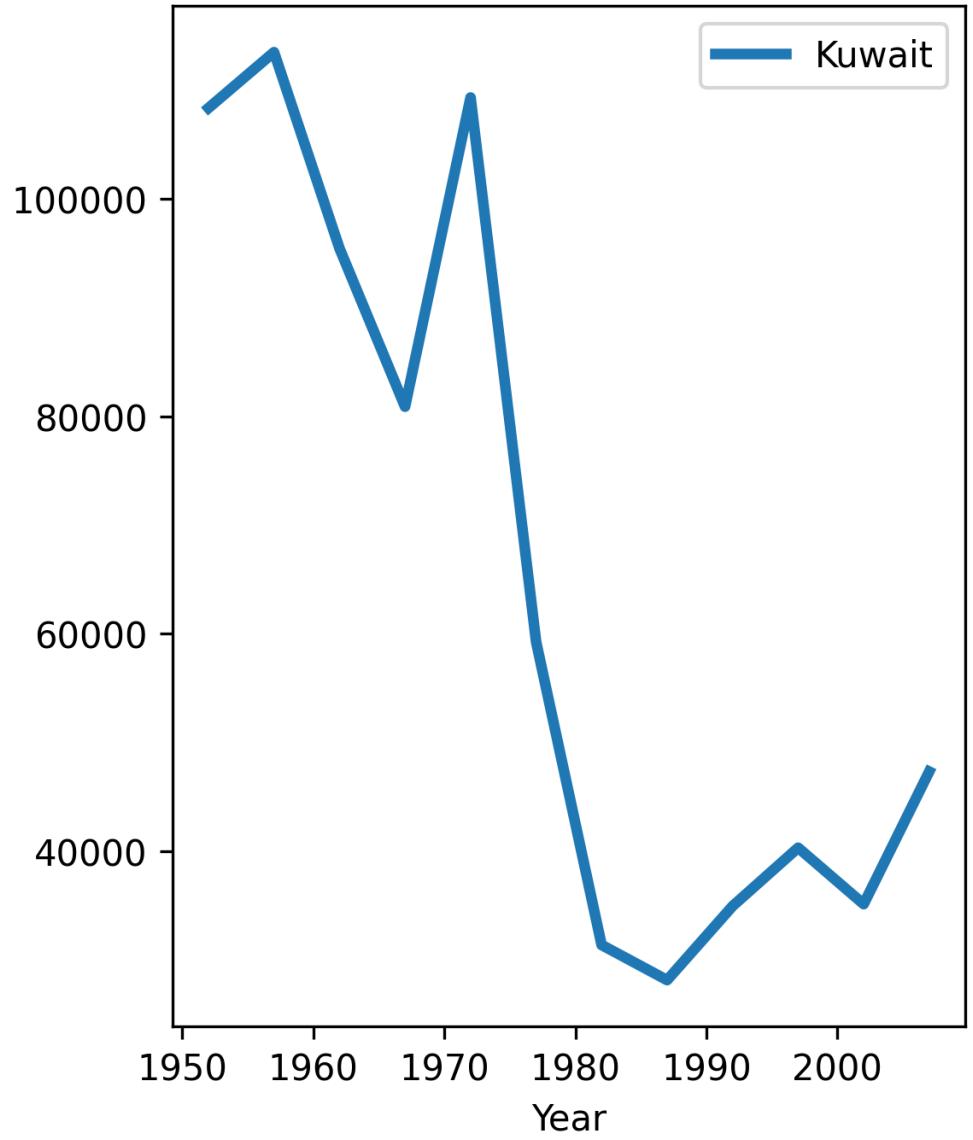


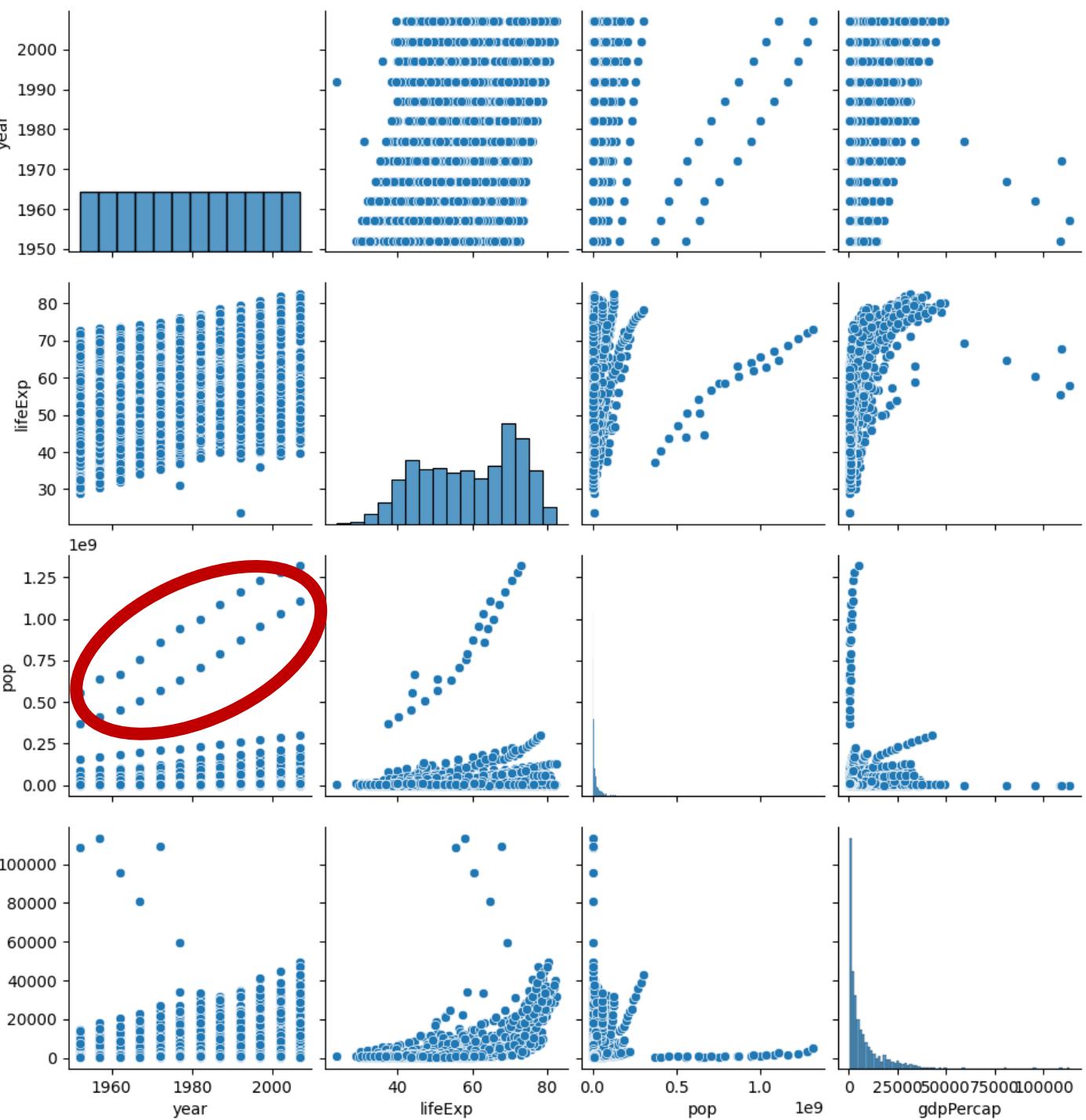


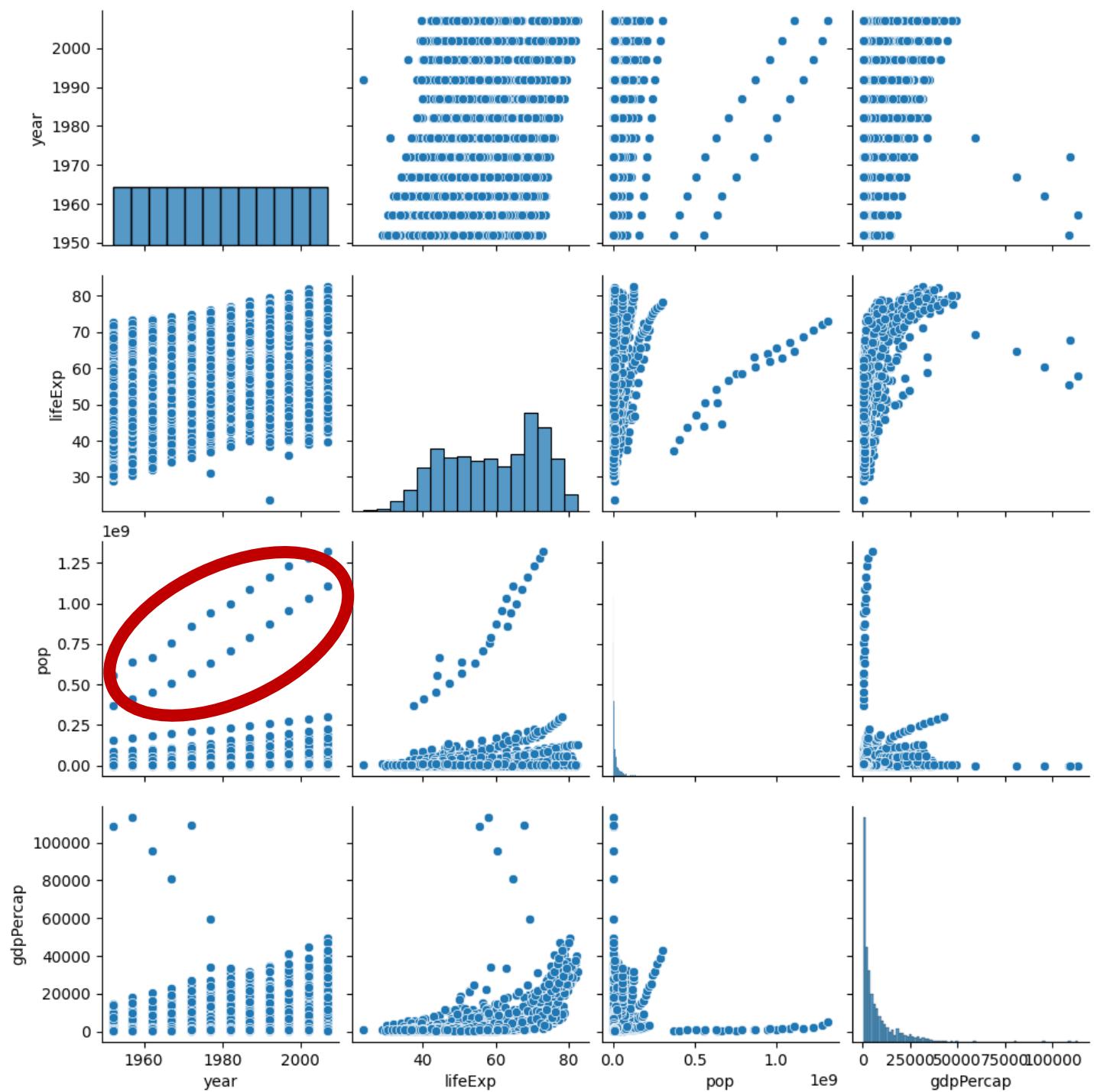
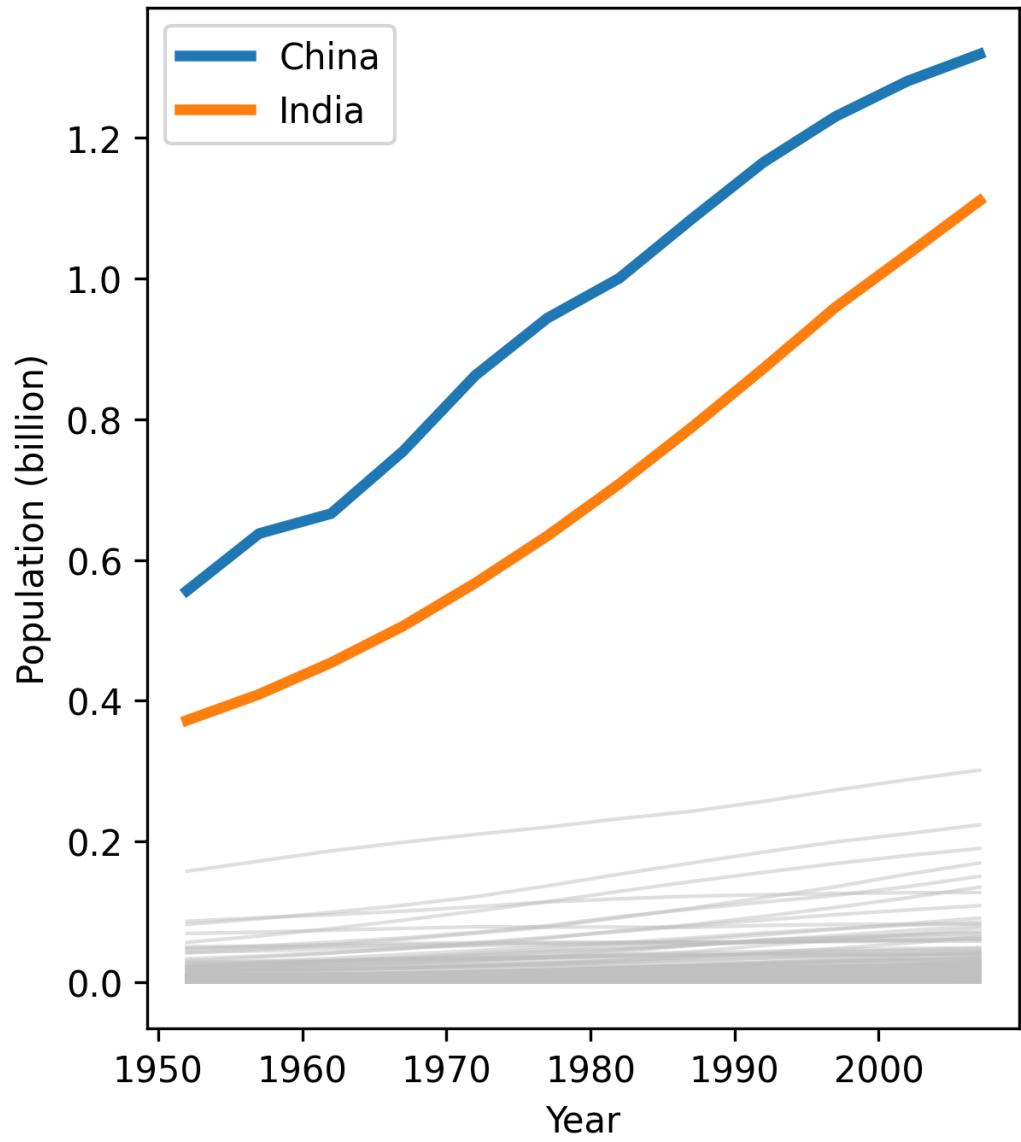
```
gapminder.sort_values(  
    'gdpPercap',  
    ascending = False,  
) .head(20)
```

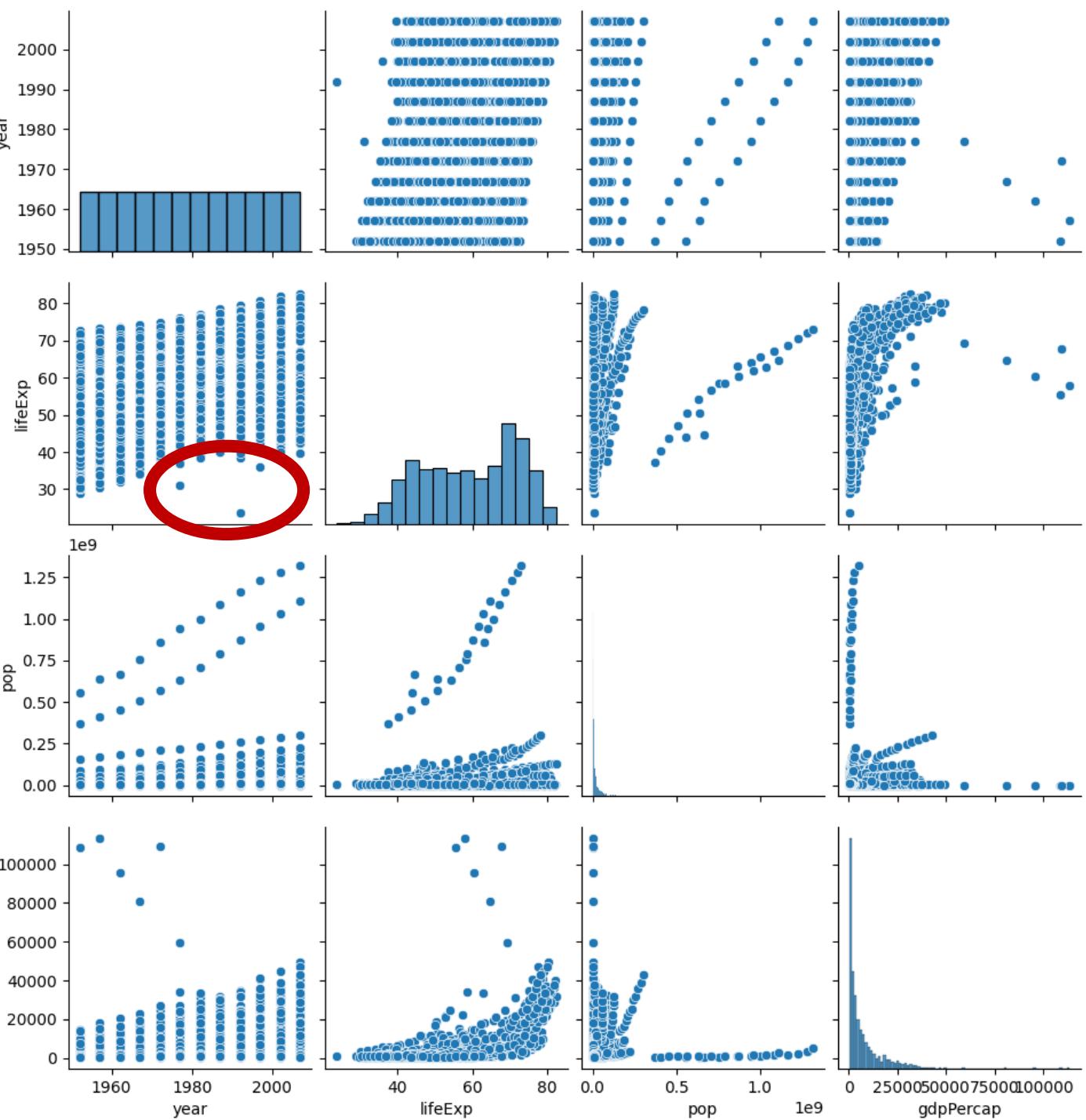


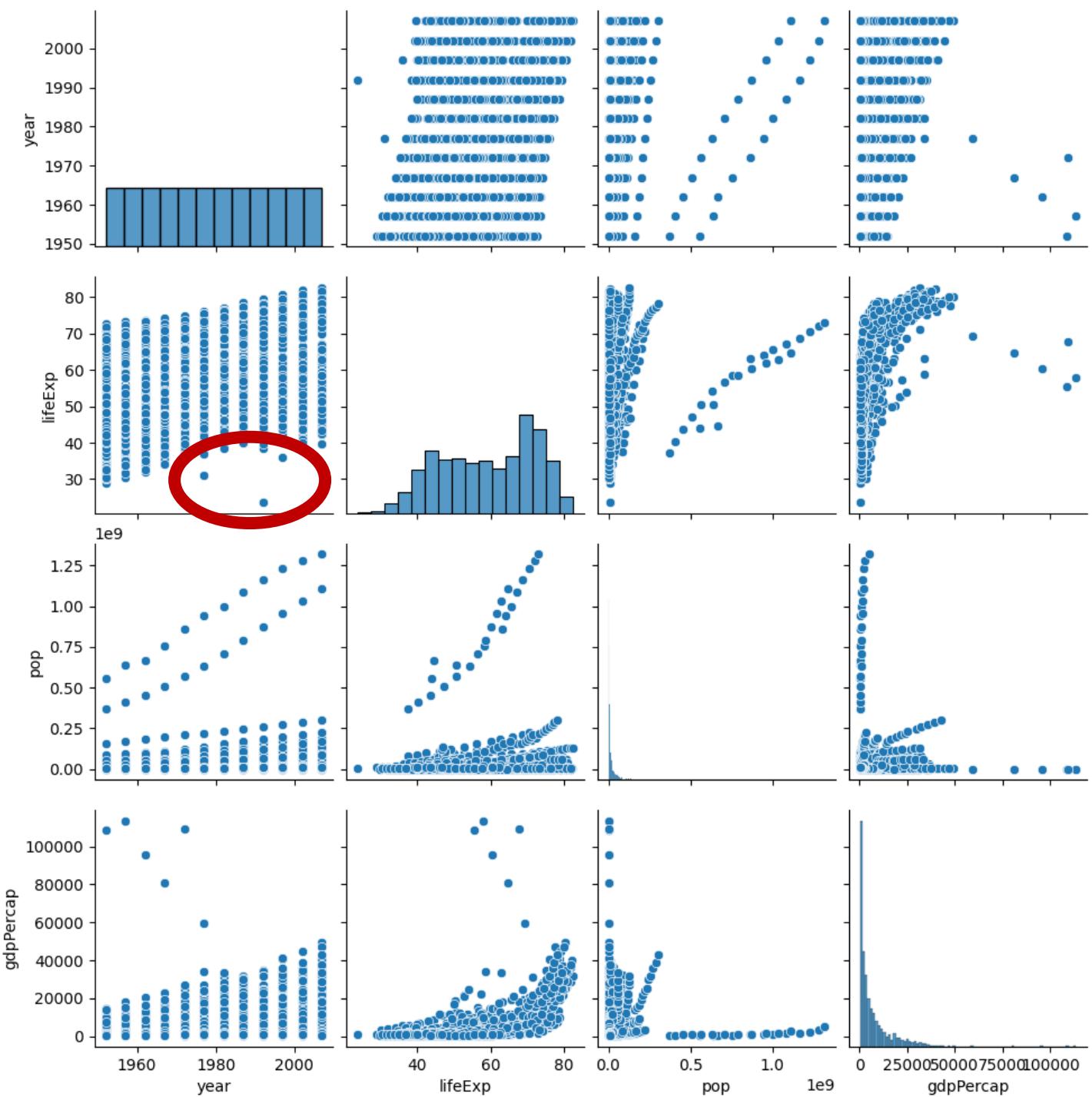
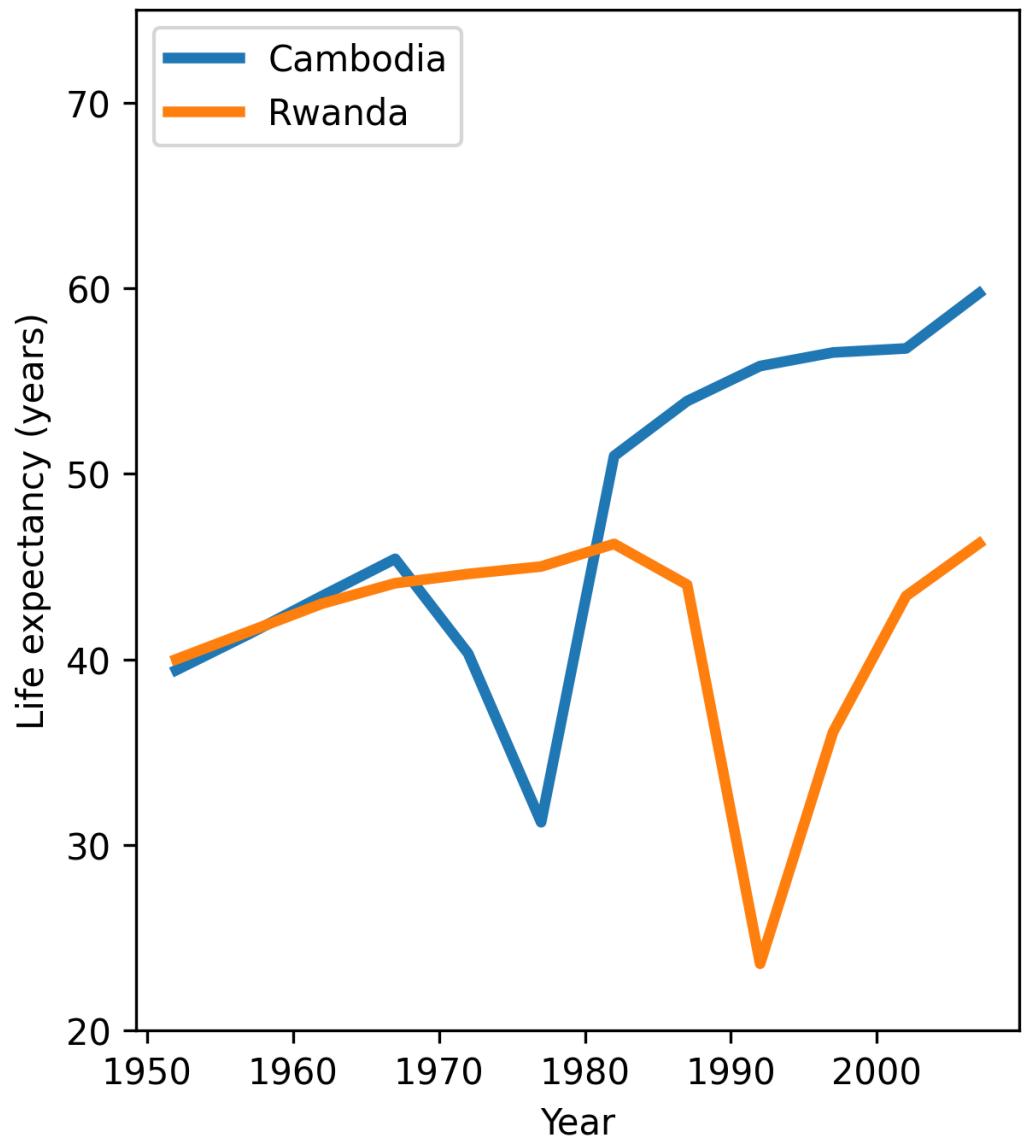
GDP per Capita (USD)

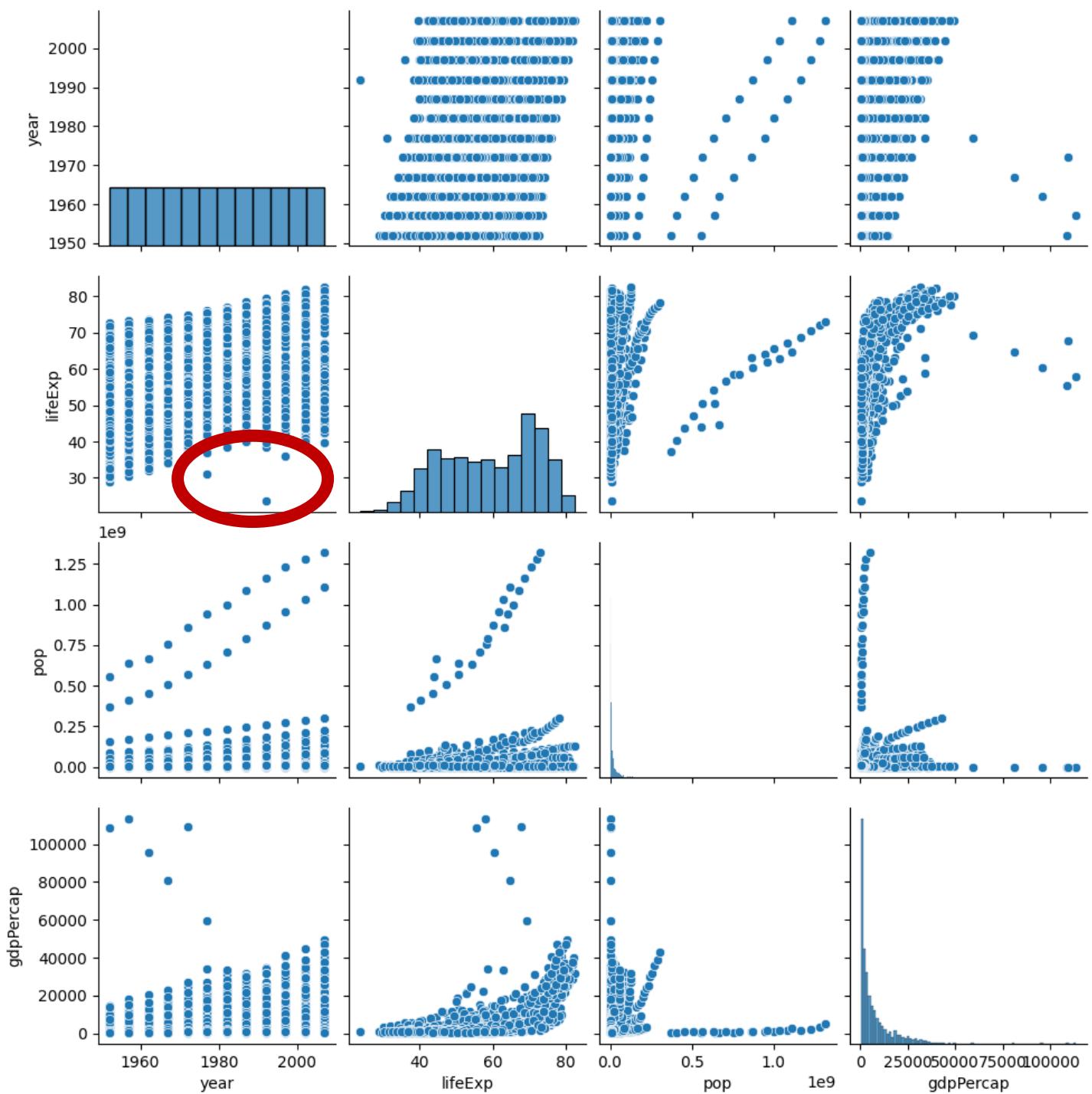
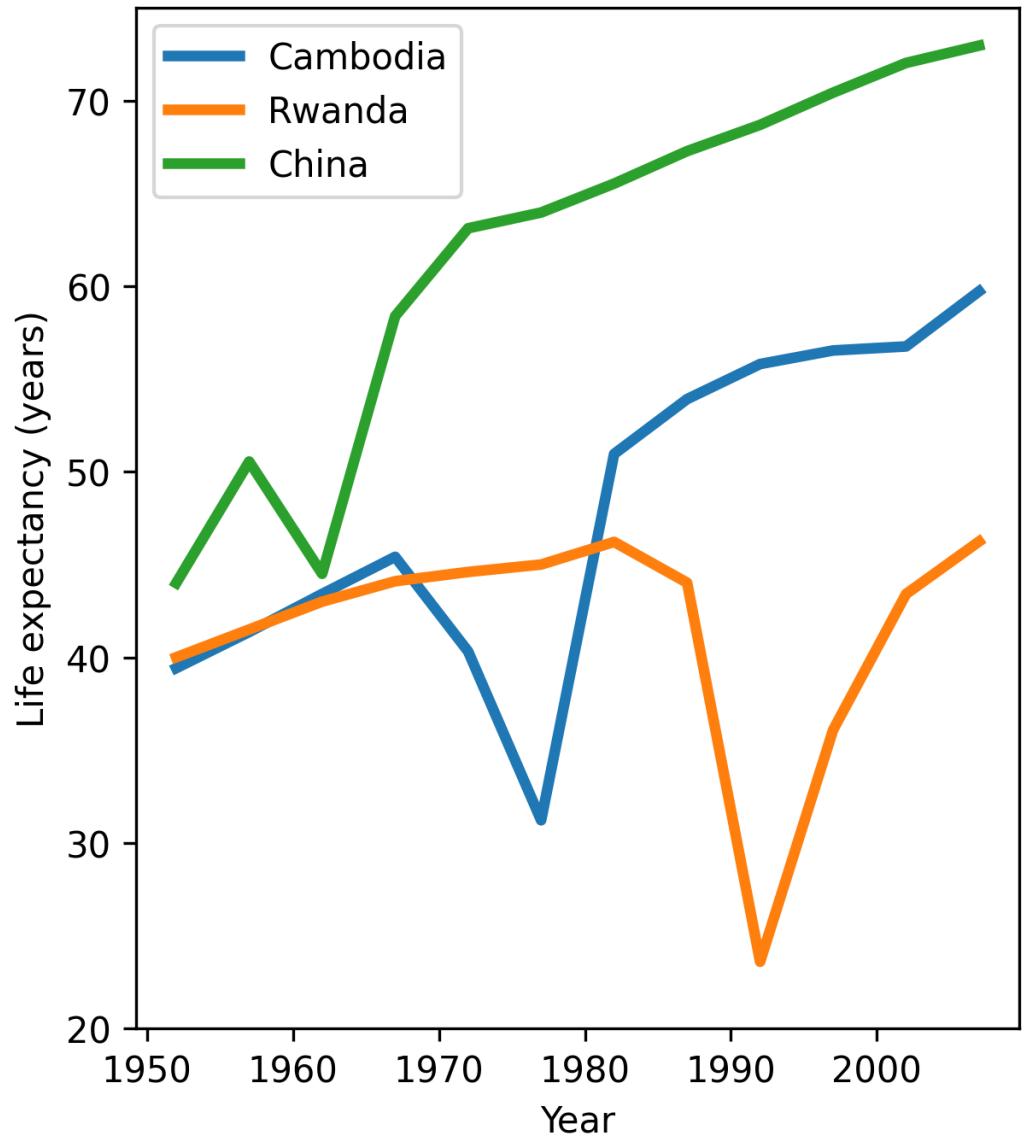








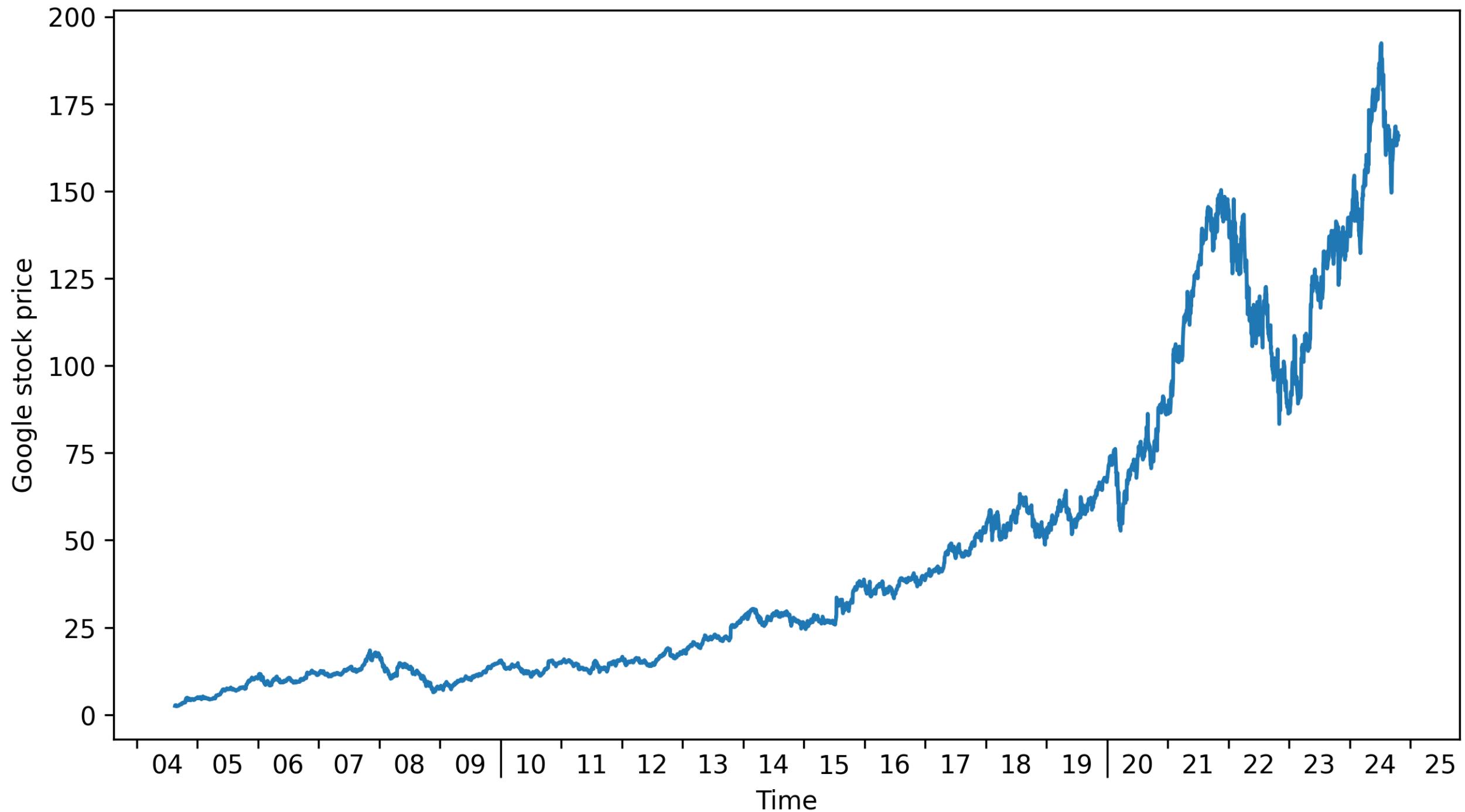


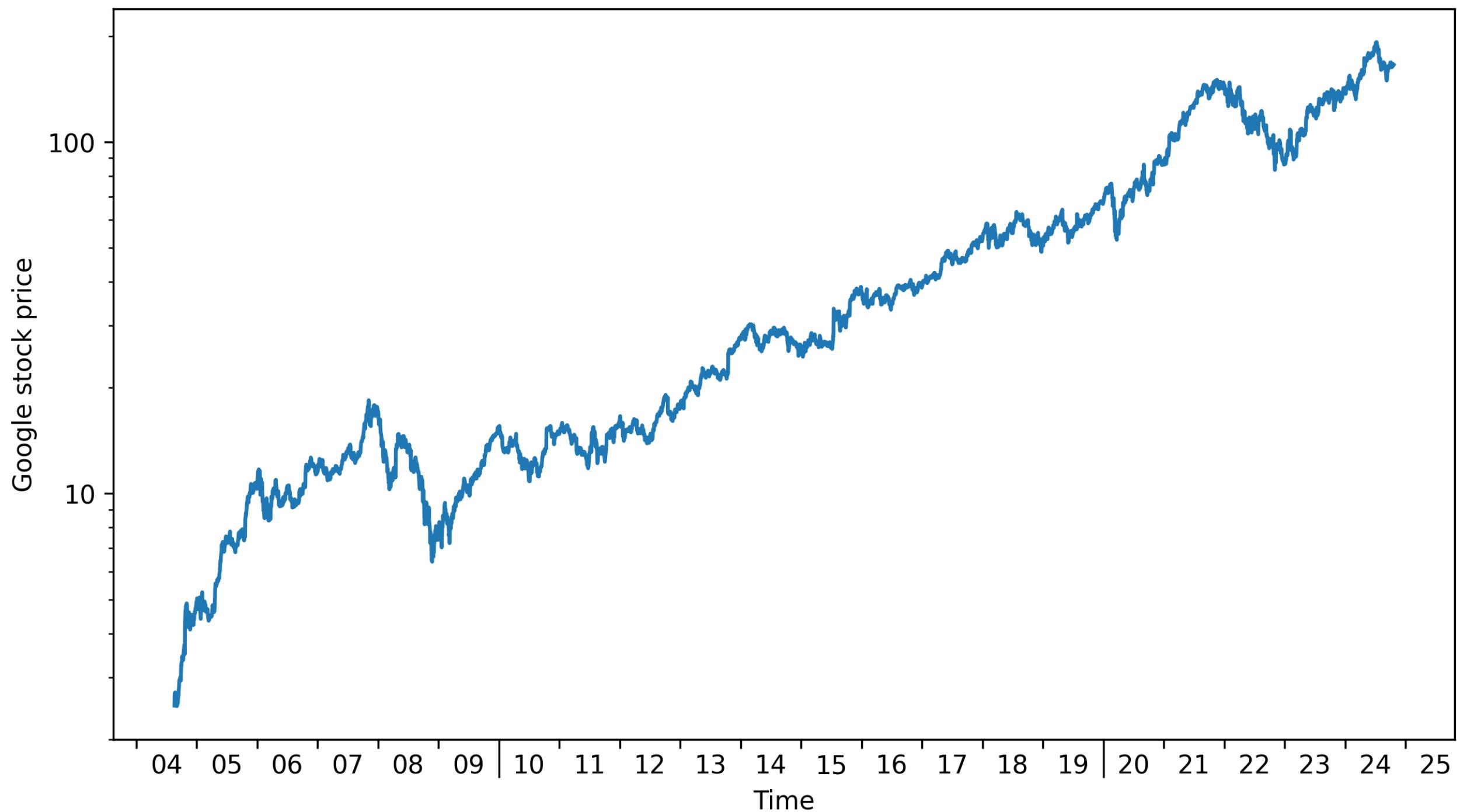


Reading Plots

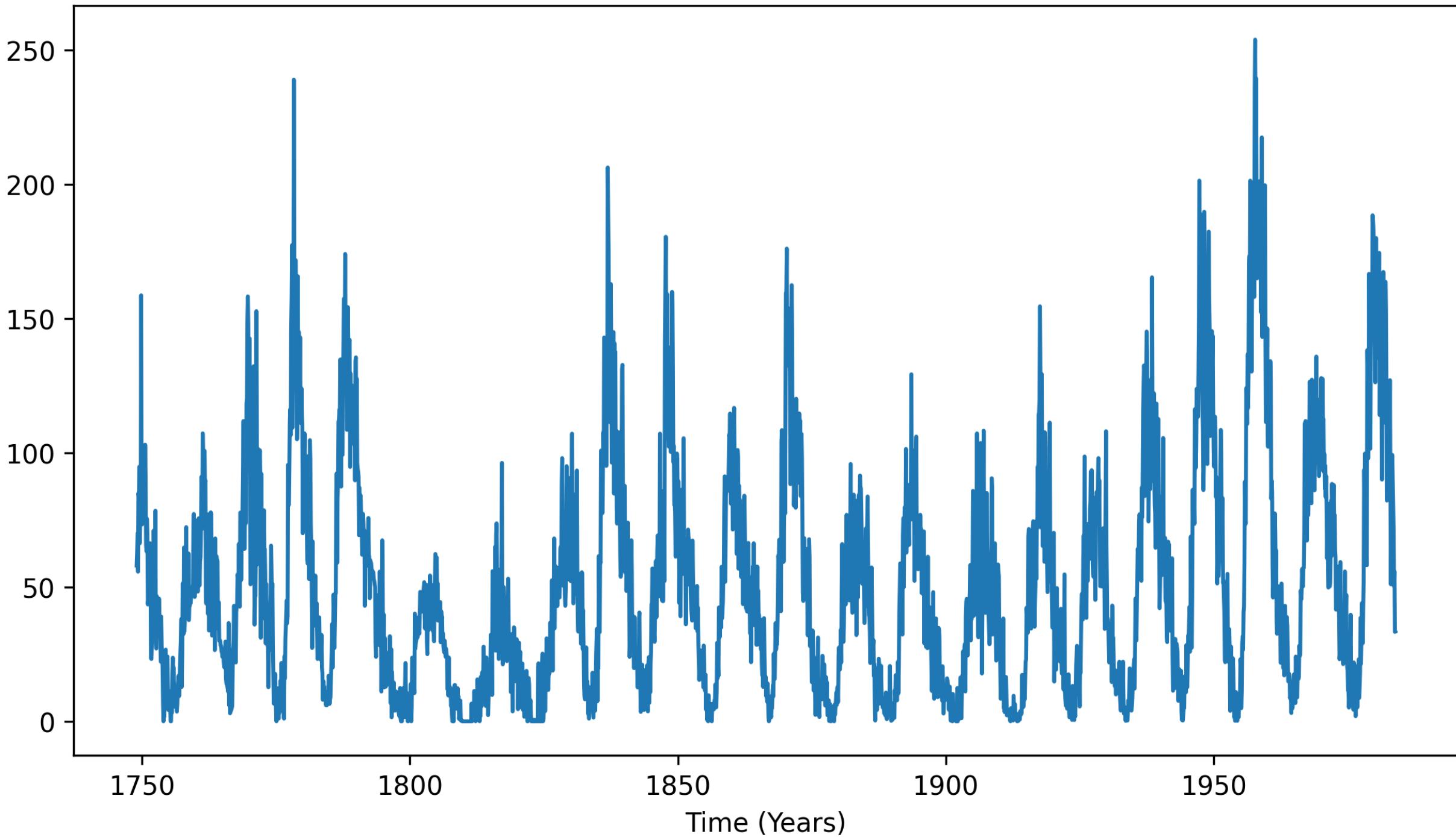
Exercise

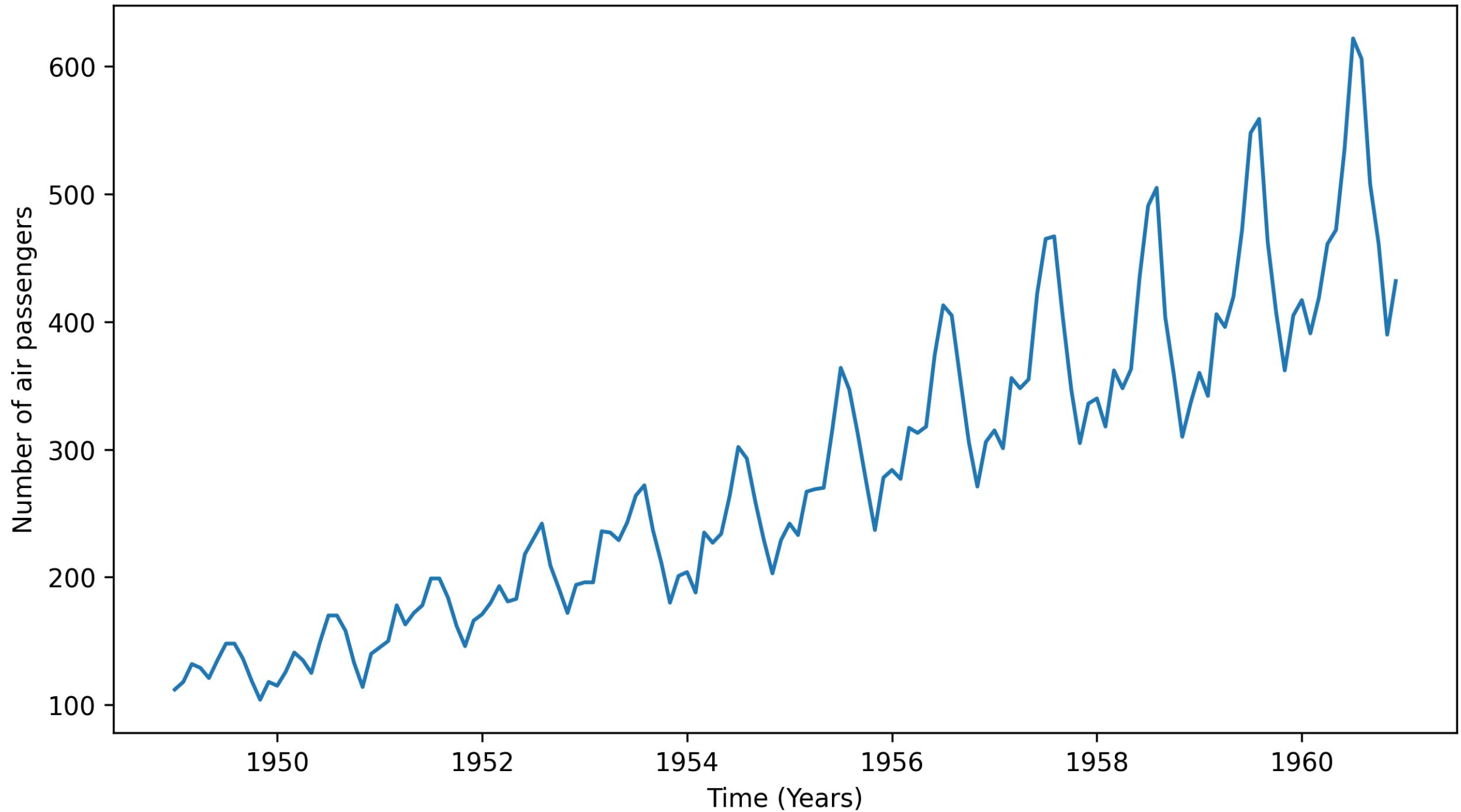
- What can you see in the following plots?

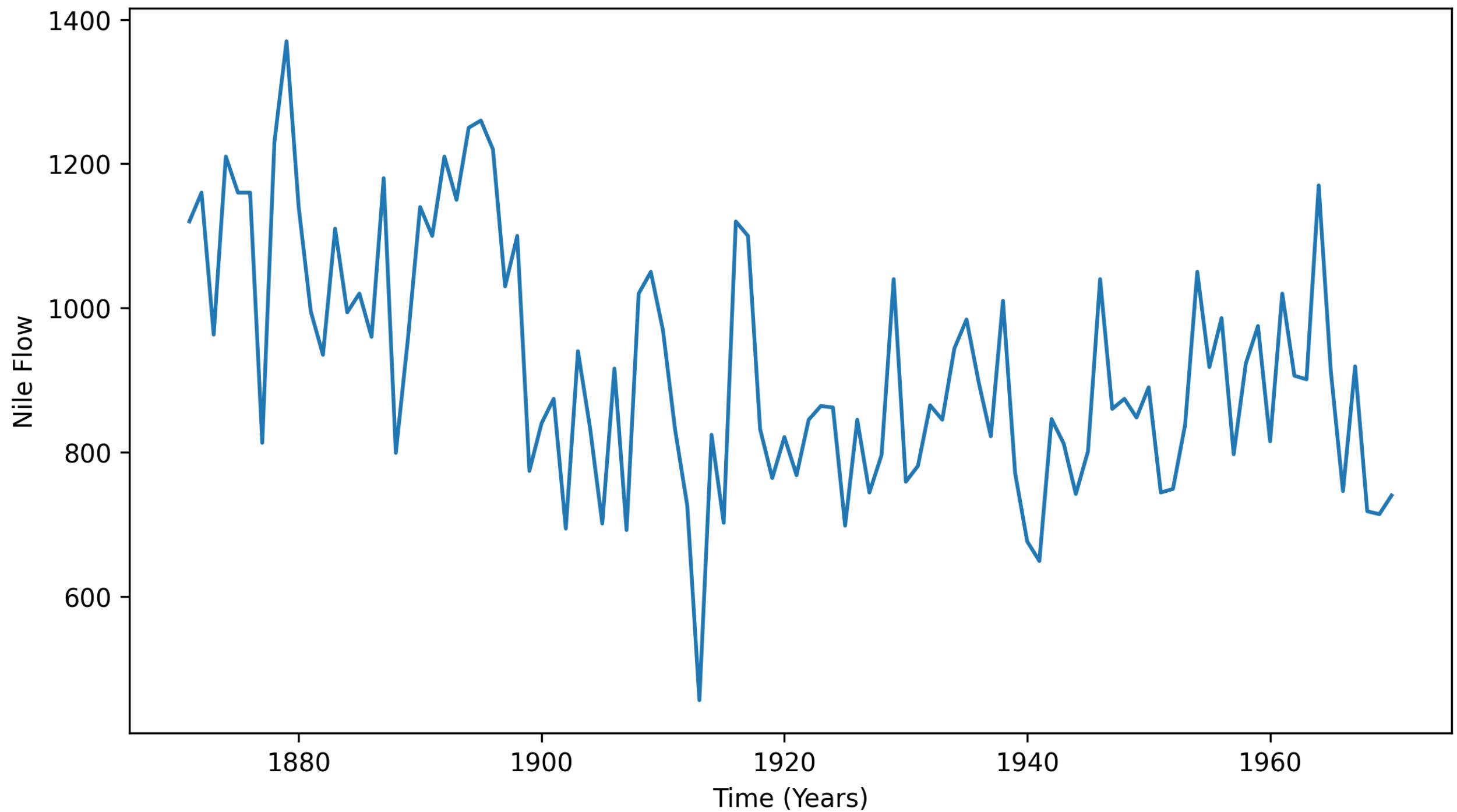


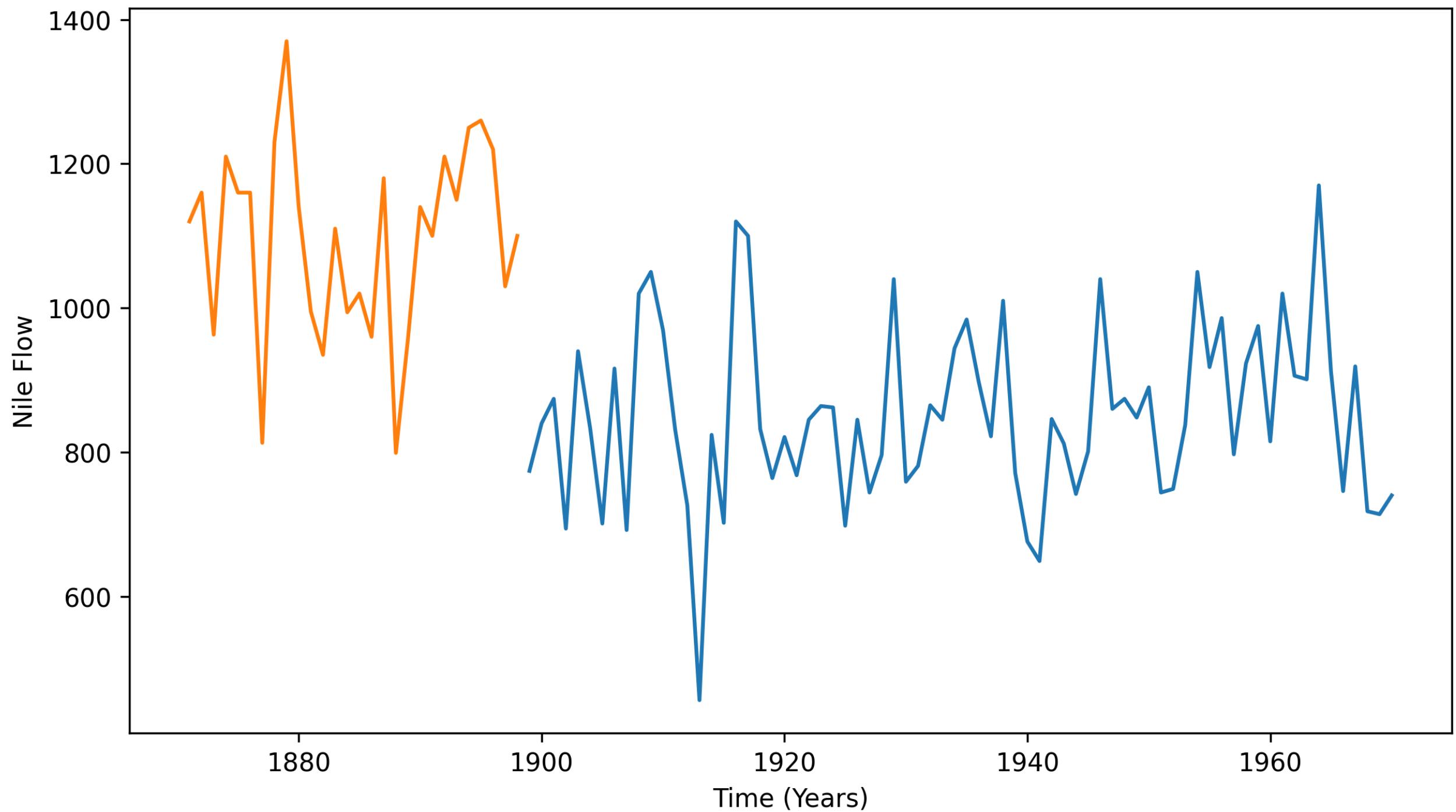


Monthly mean relative sunspot number

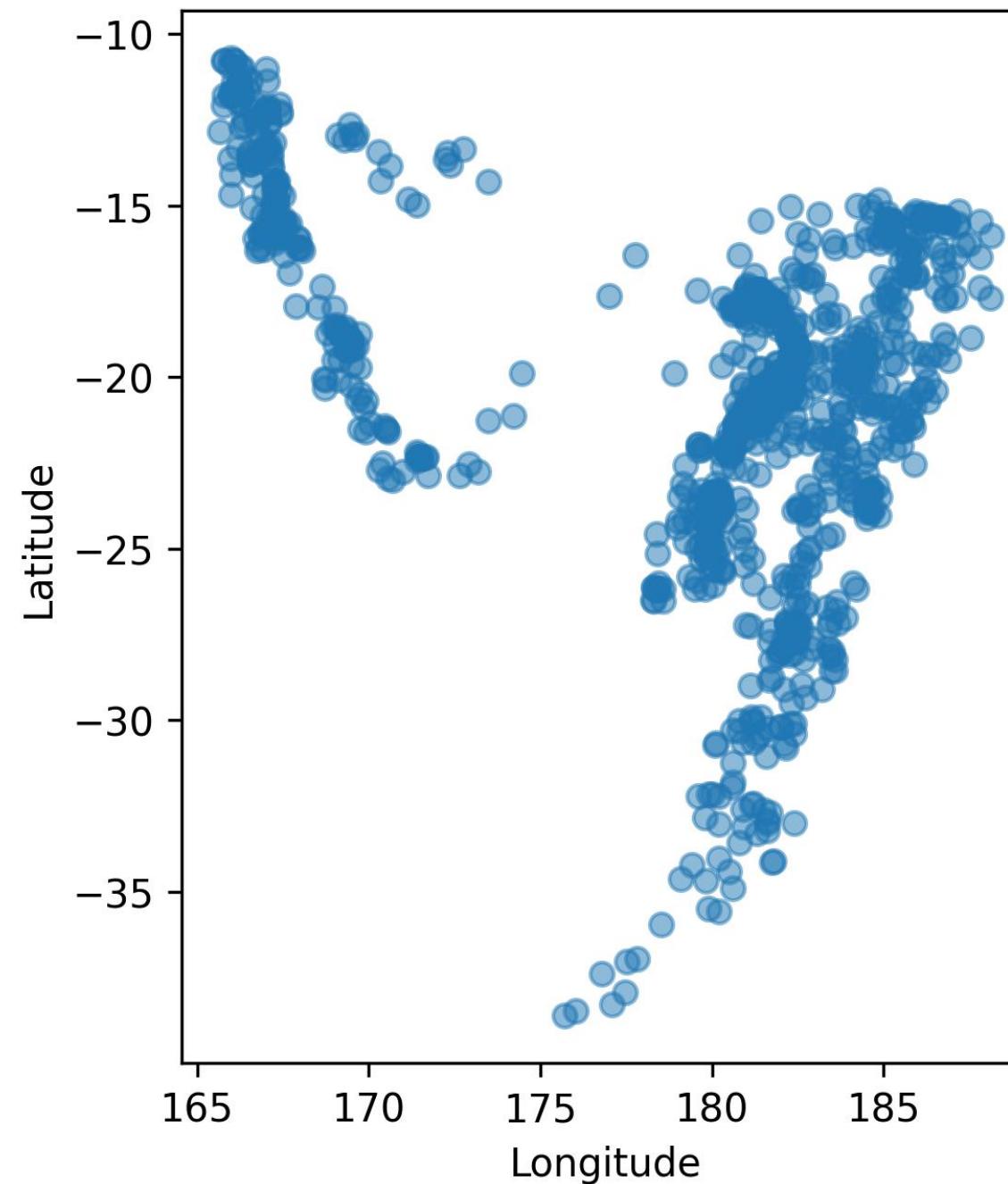




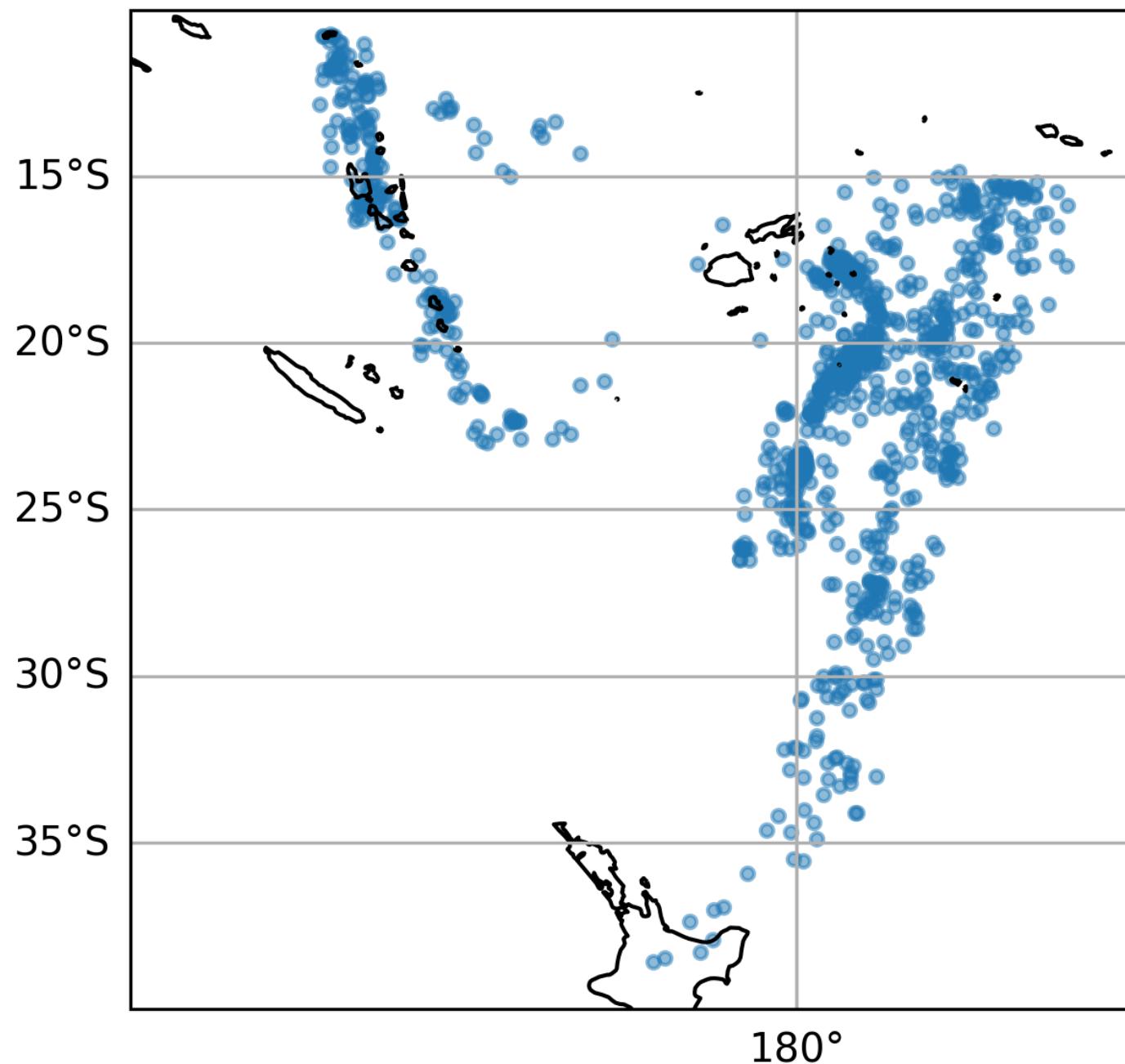




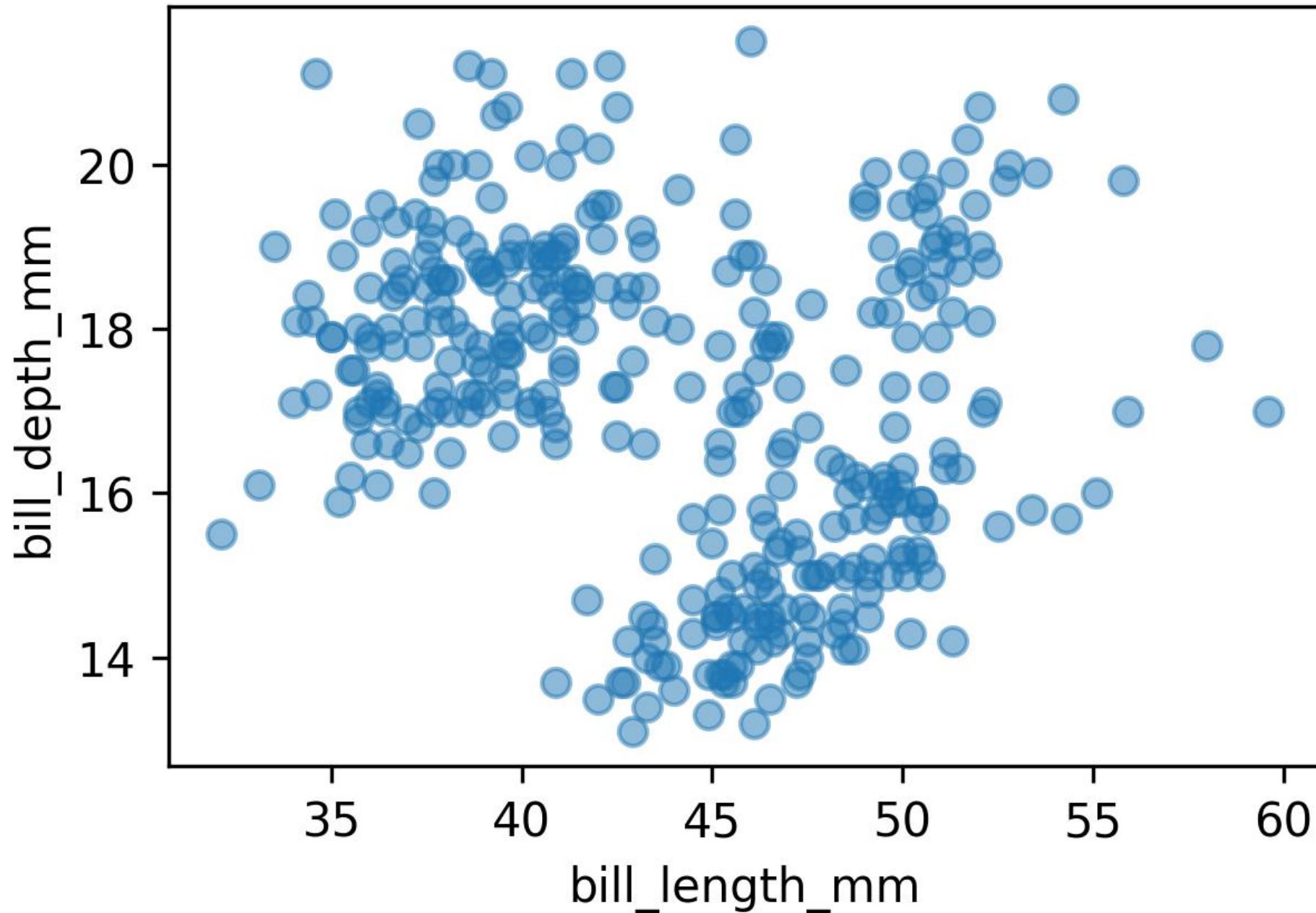
Earthquakes in Fiji



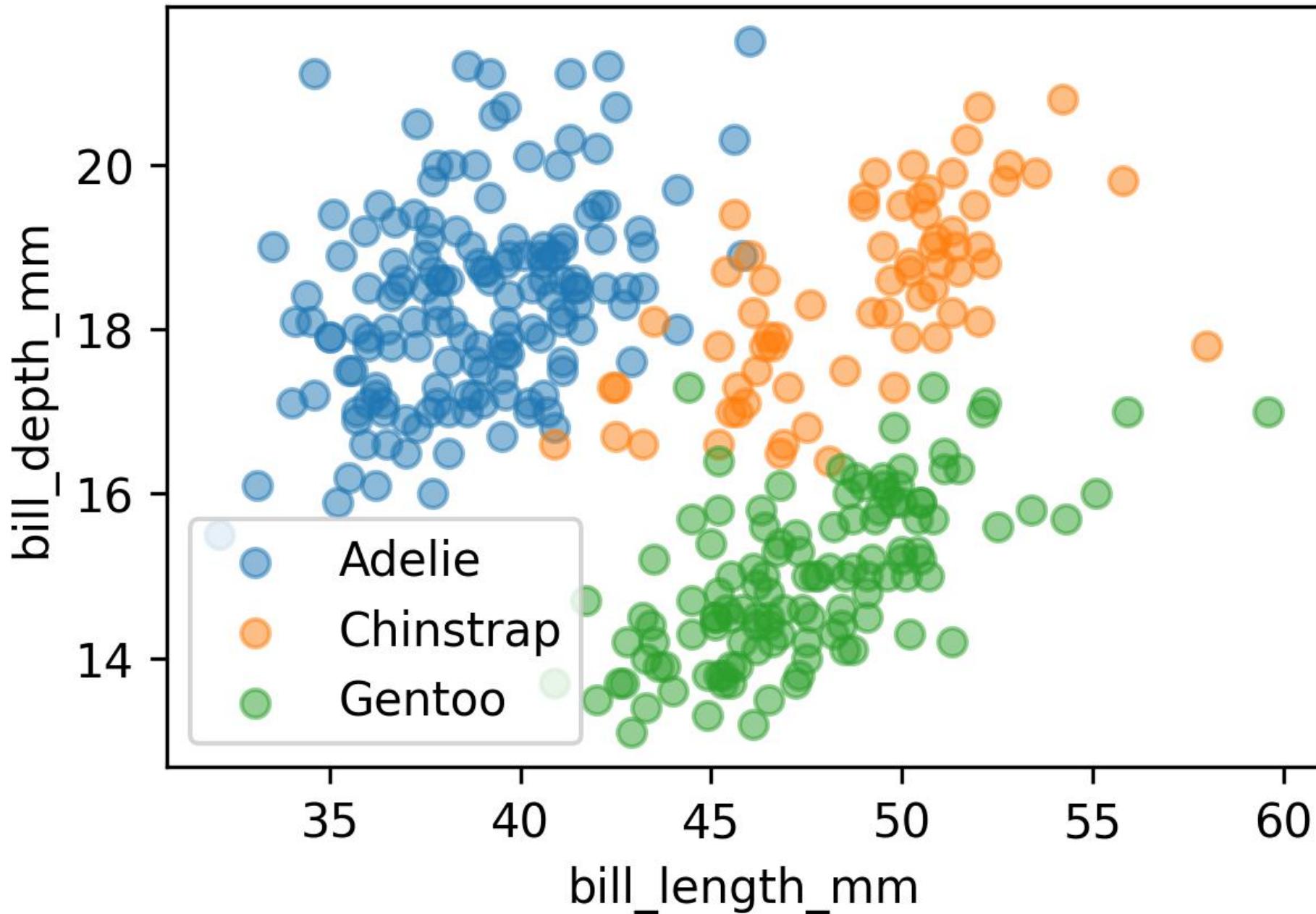
180°



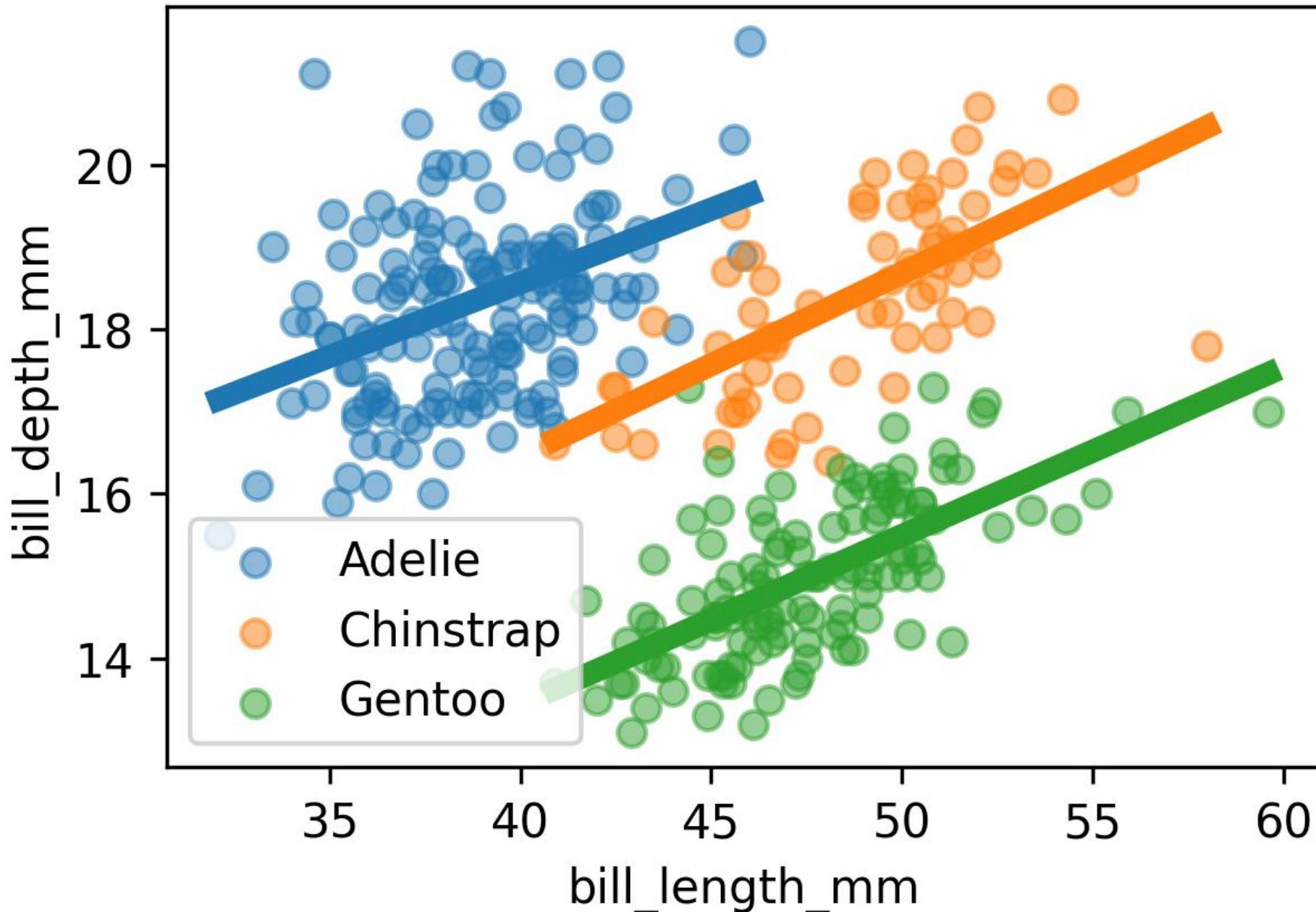
Penguin measurements



Penguin measurements



Penguin measurements



-4

-3

-2

-1

0

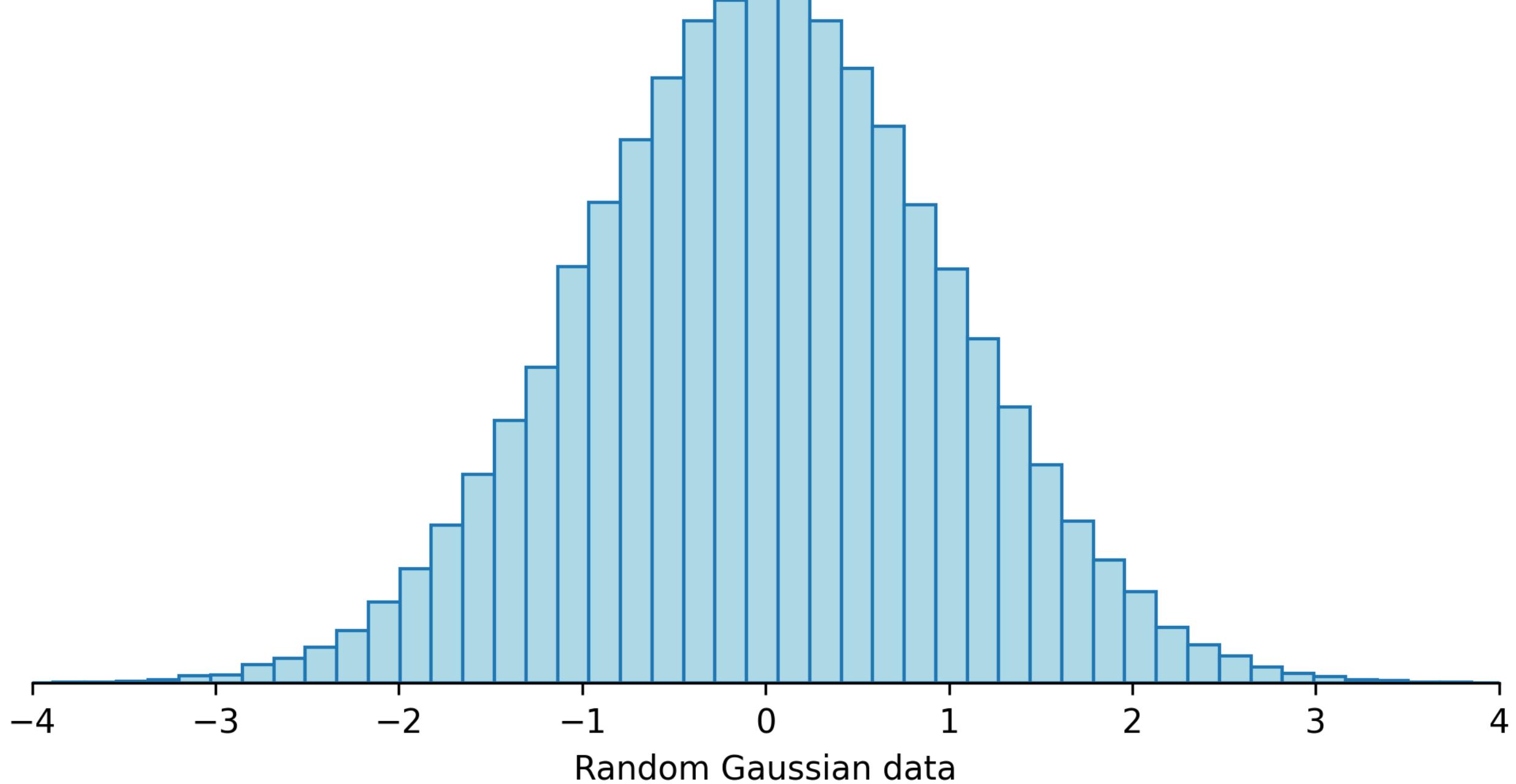
1

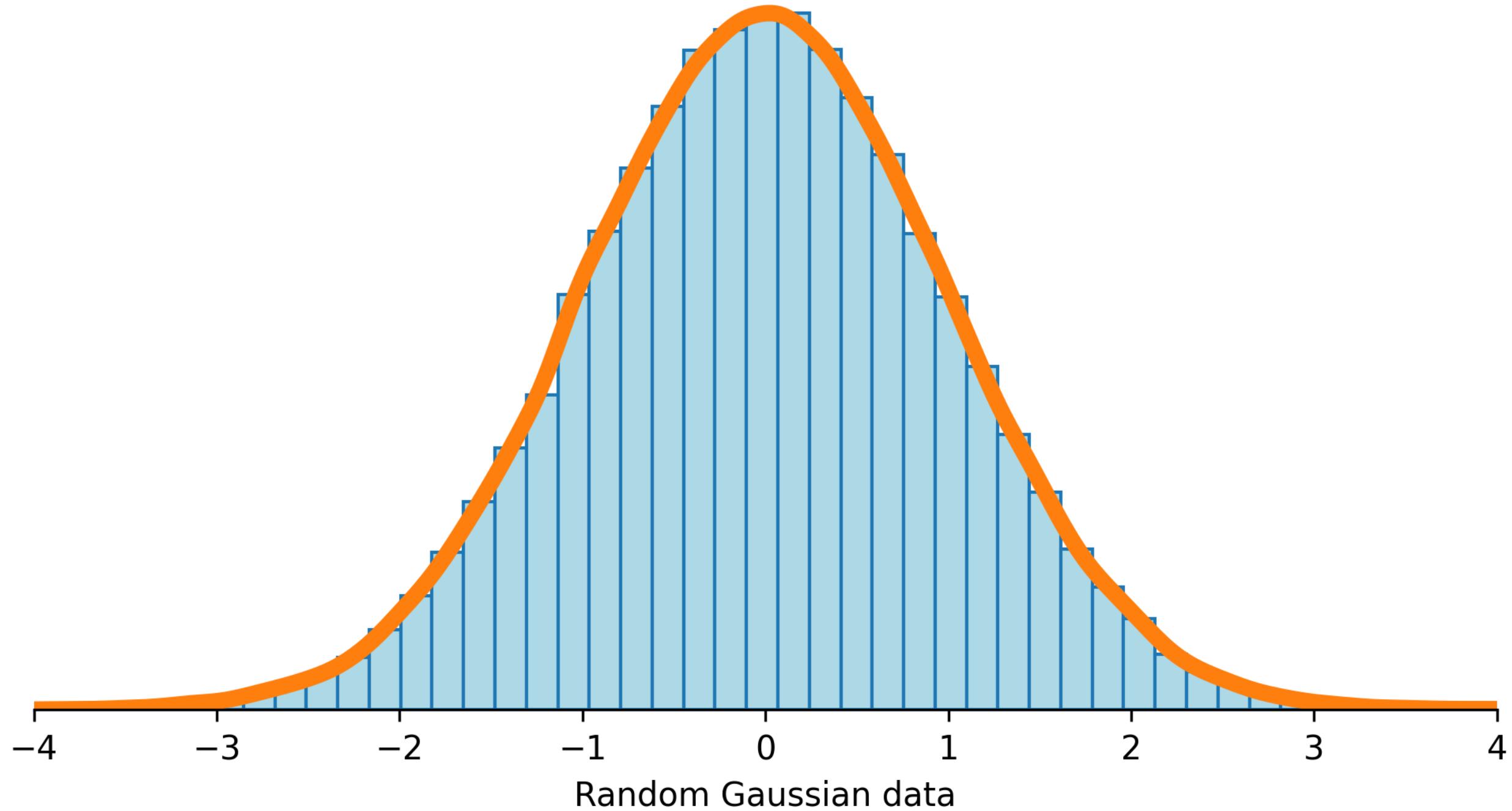
2

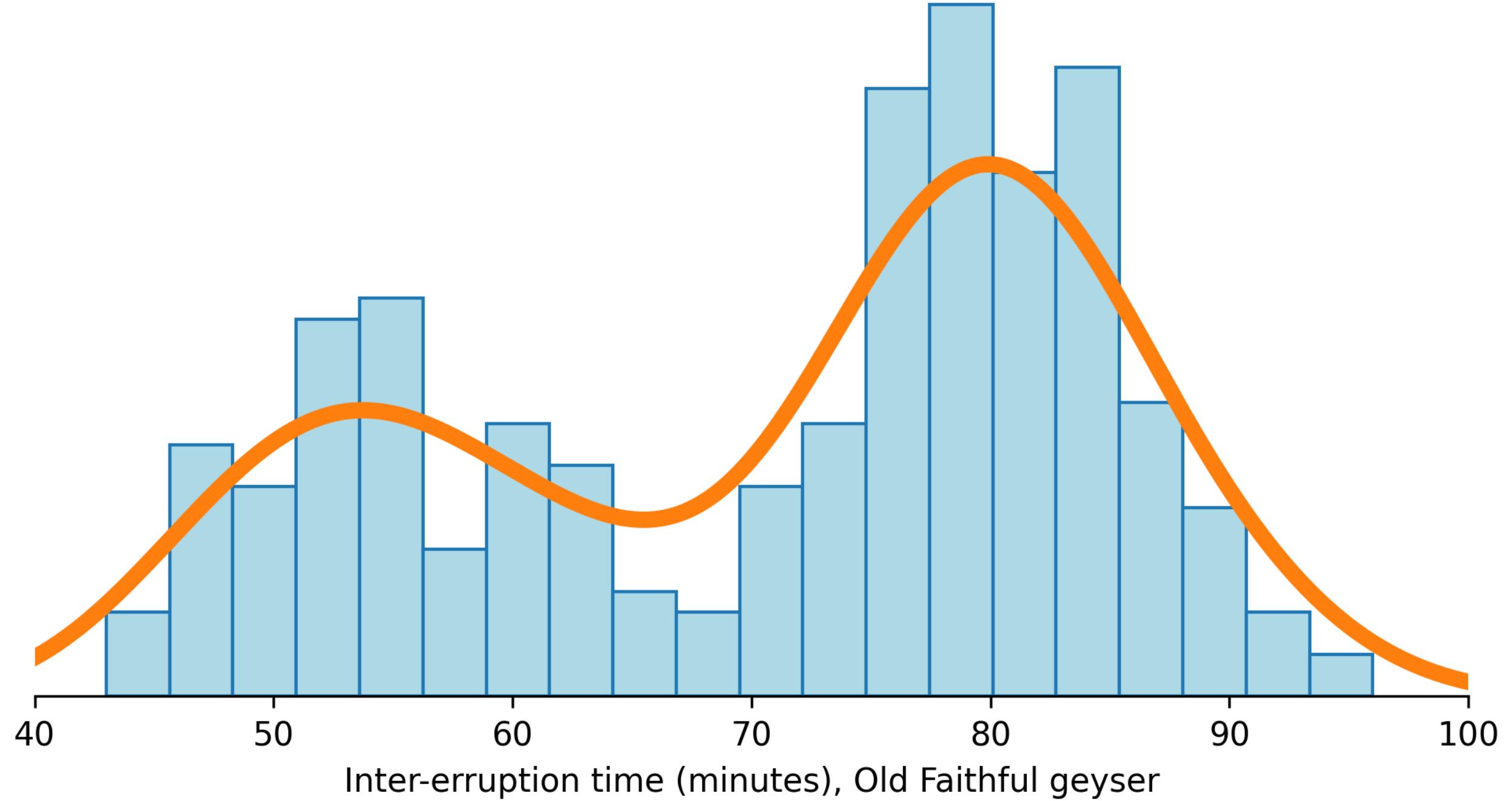
3

4

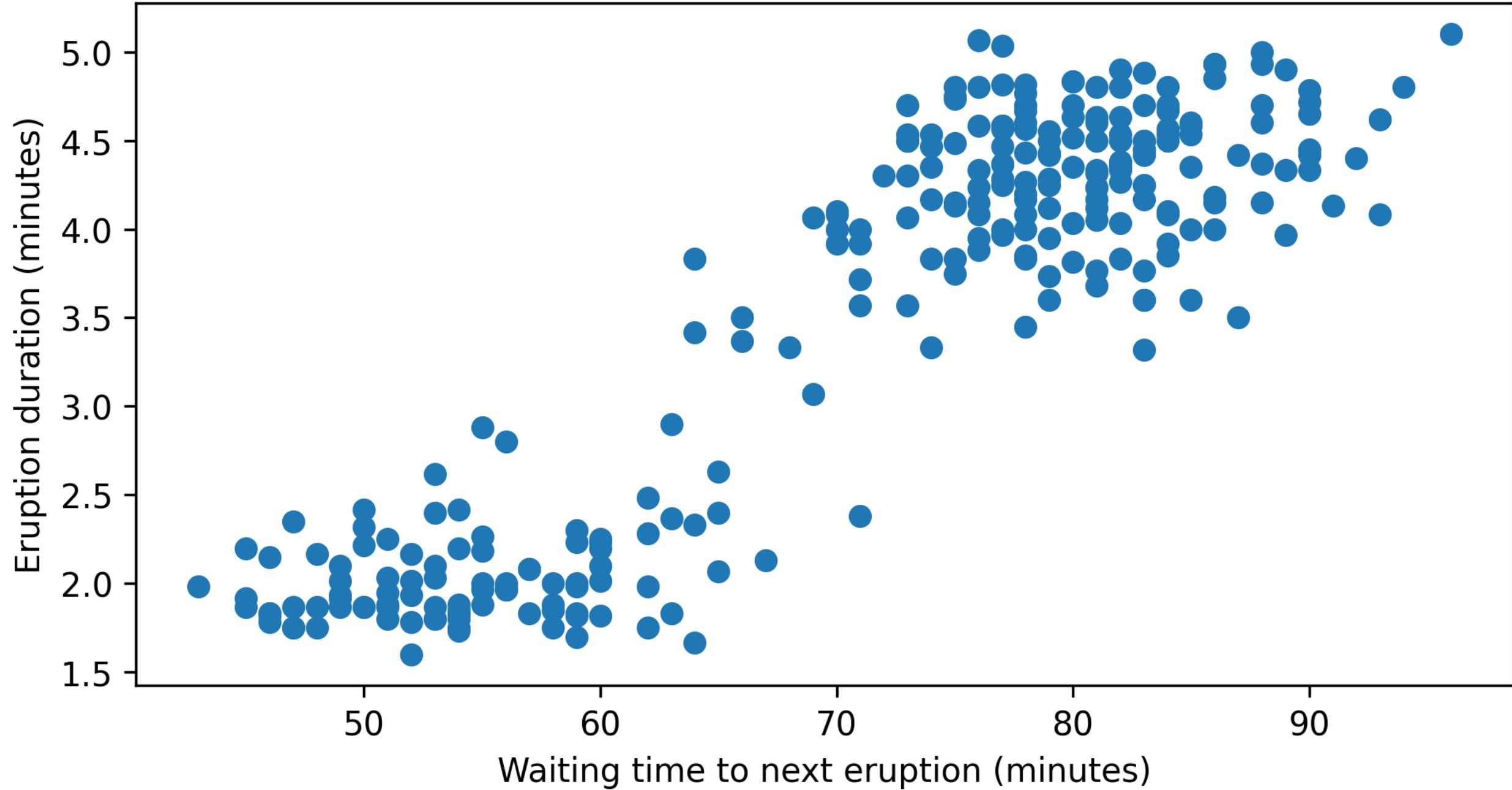
Random Gaussian data



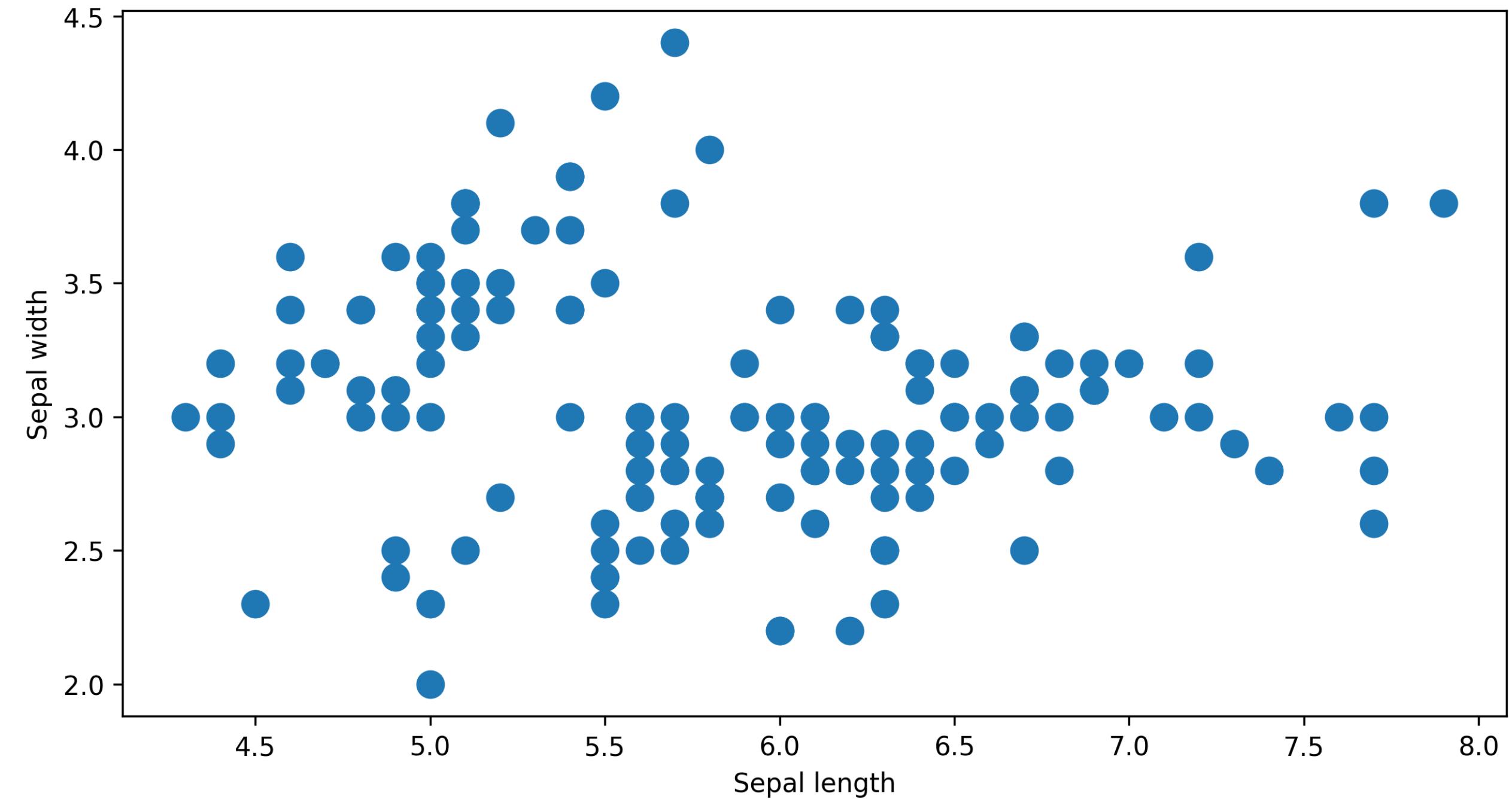




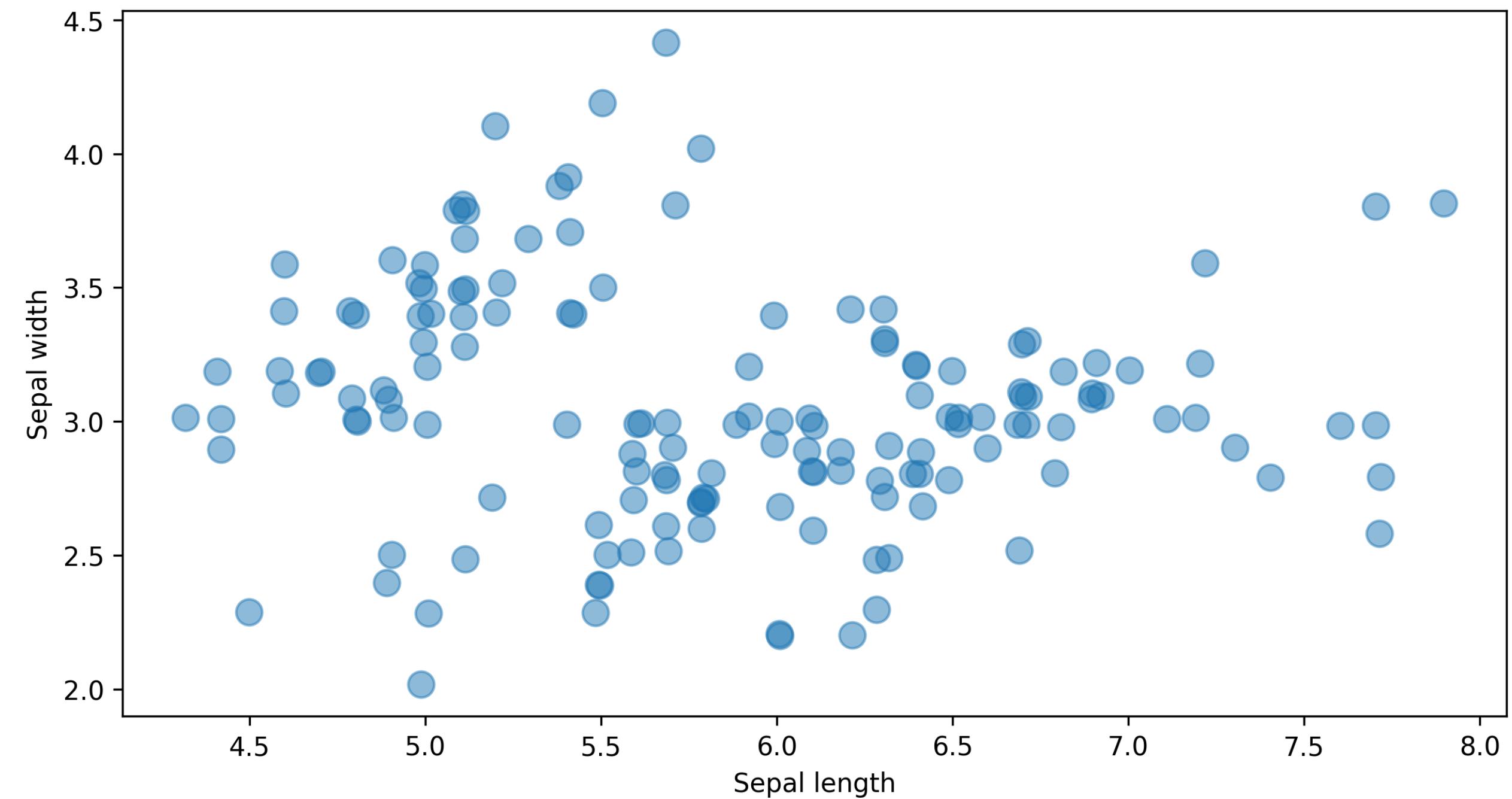
Old Faithful geyser



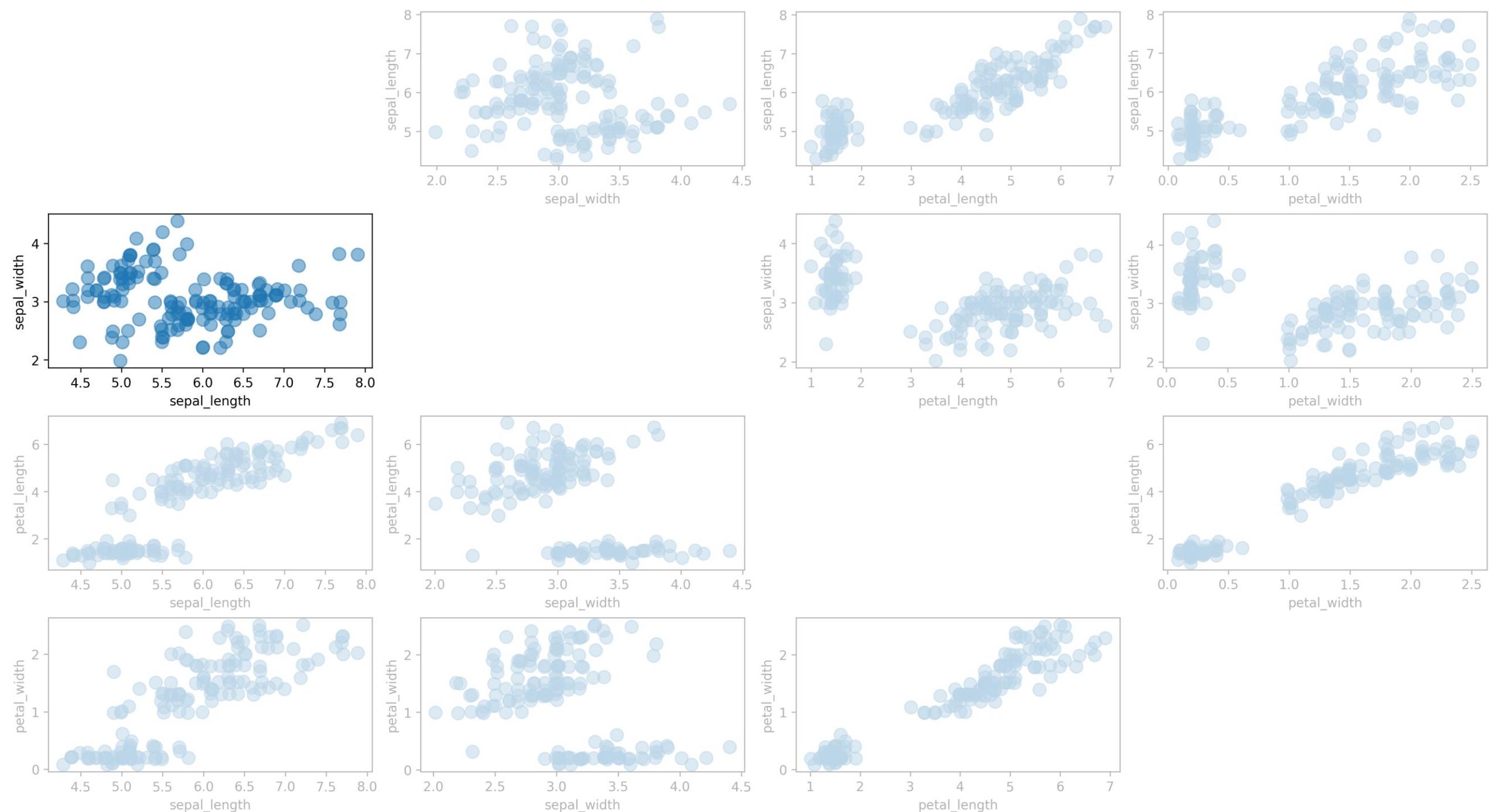
Iris



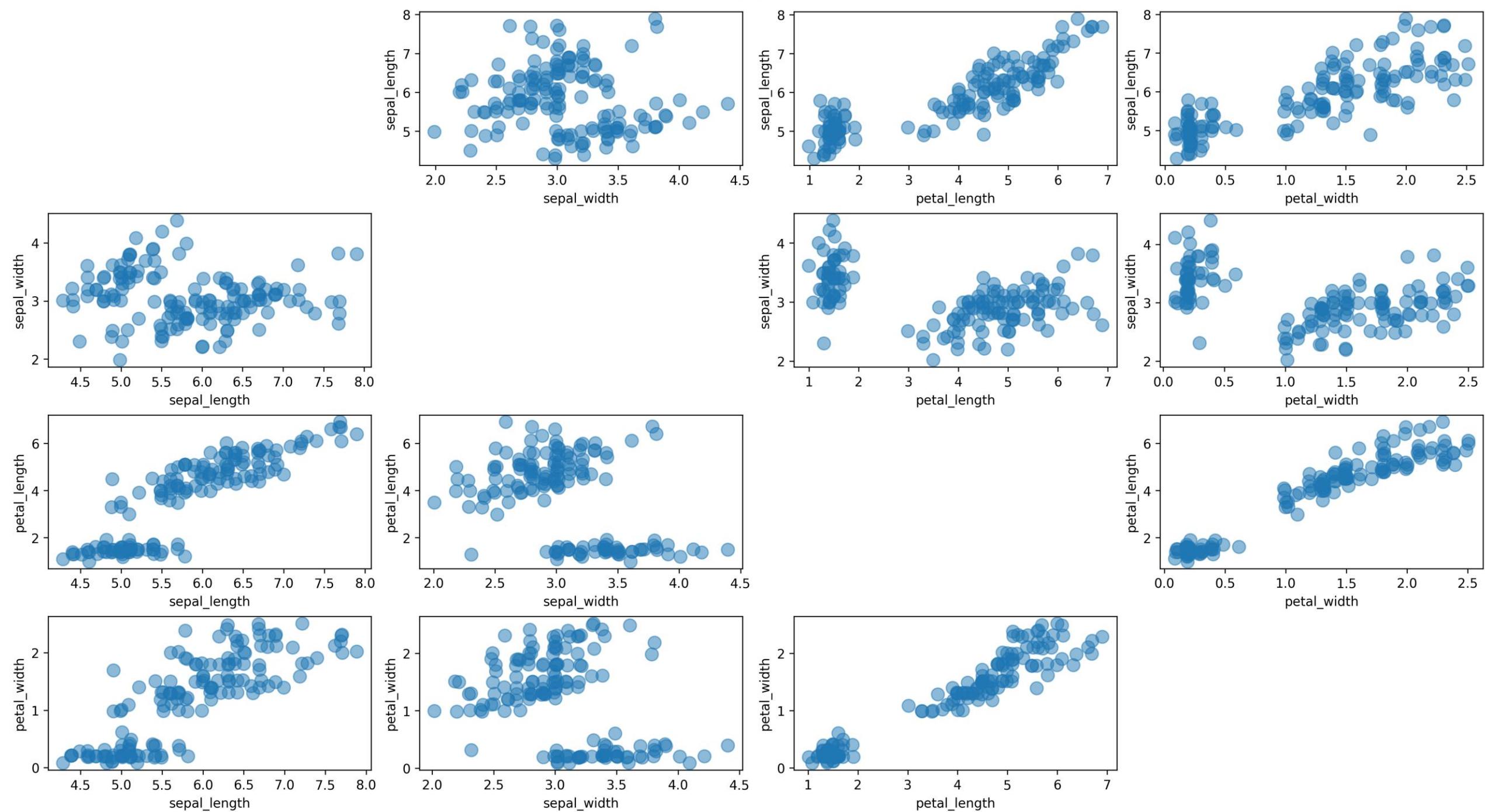
Iris



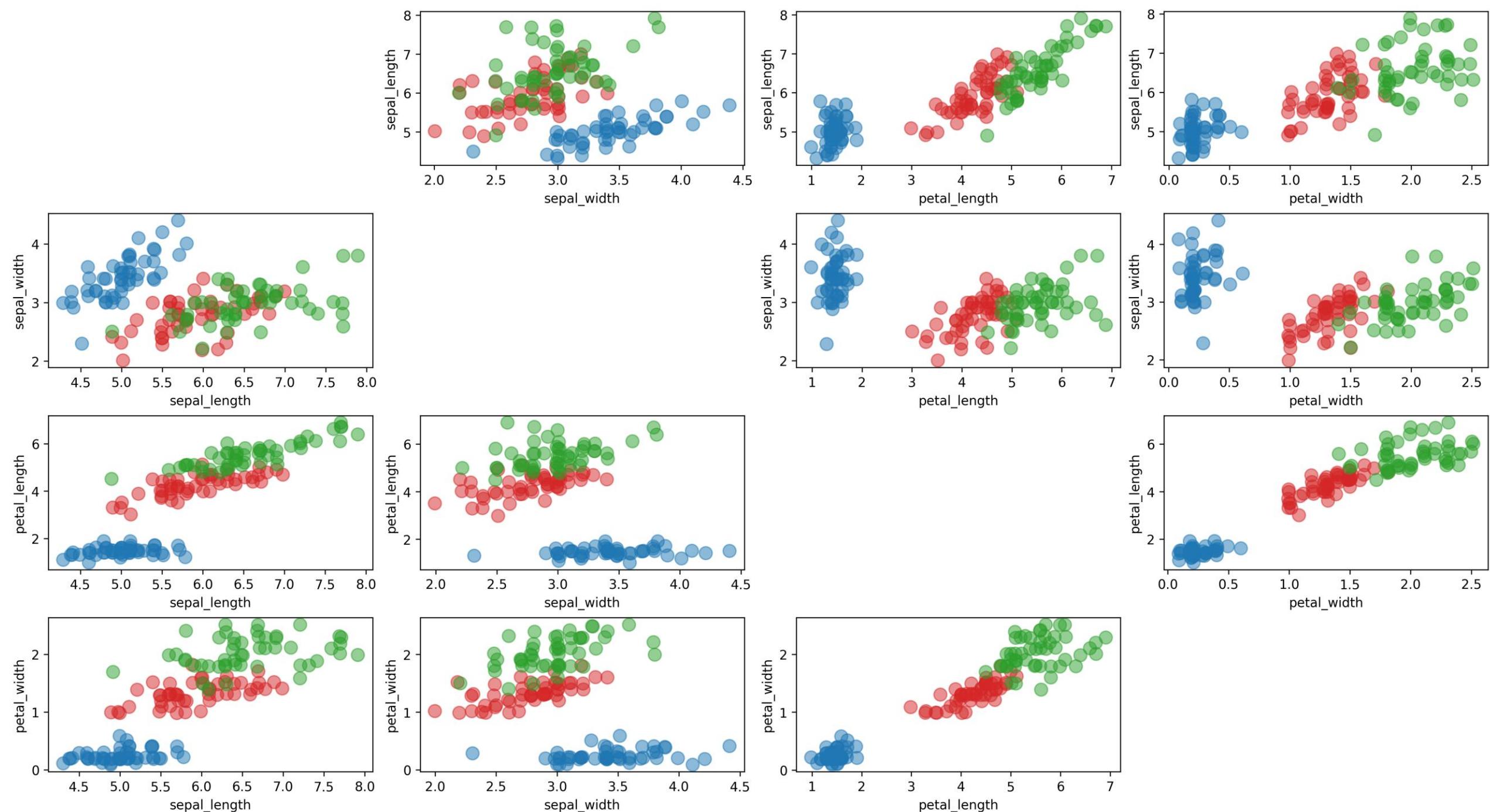
Iris

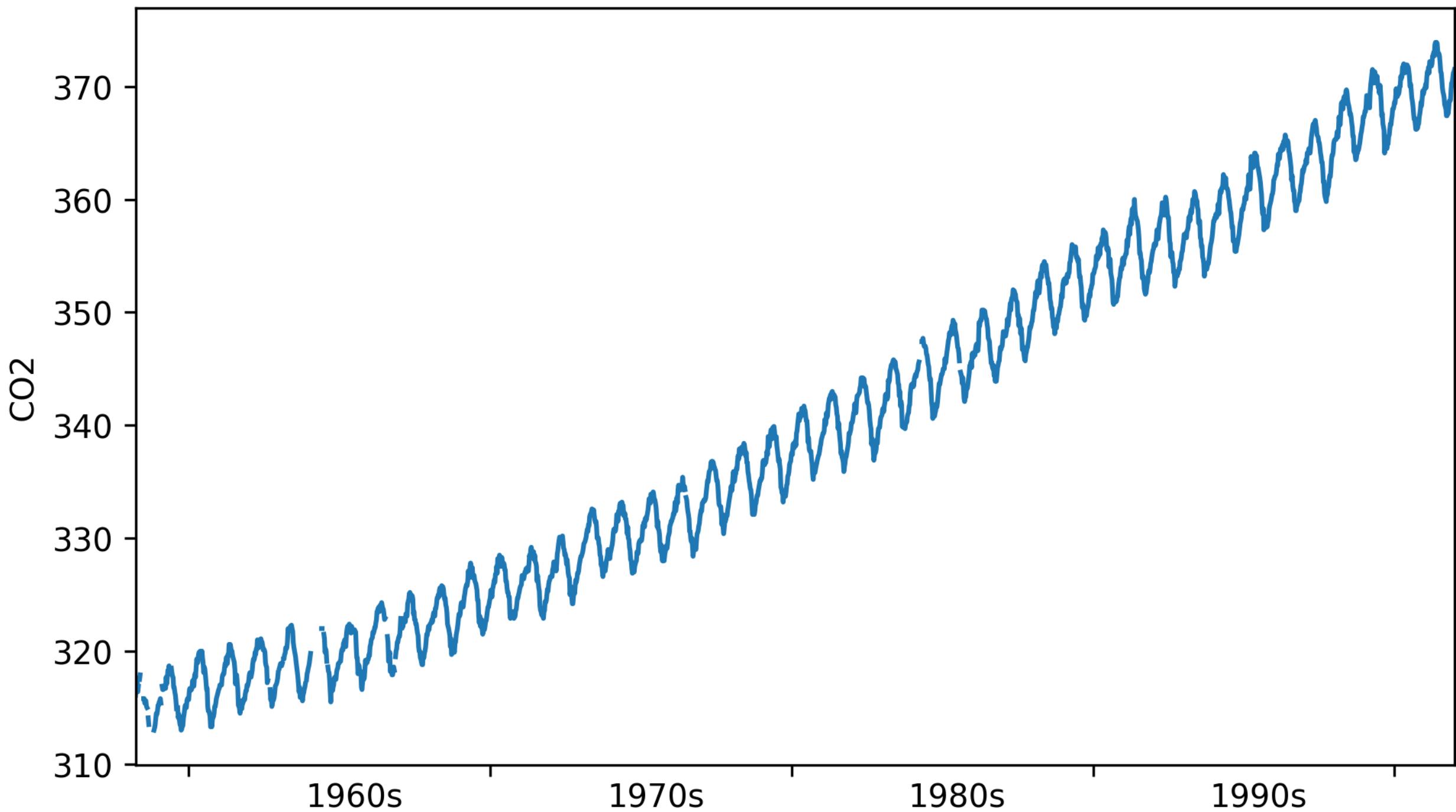


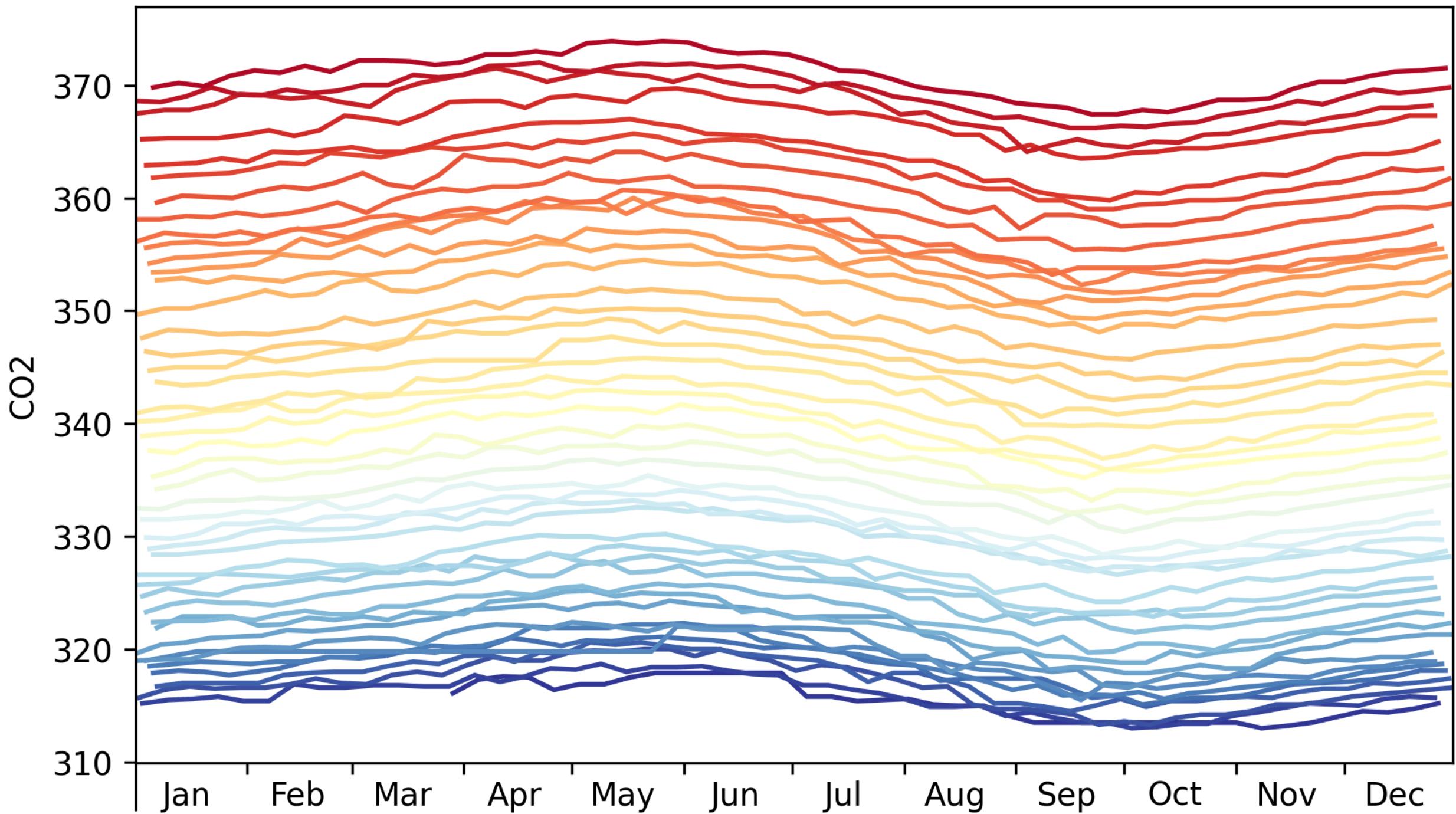
Iris



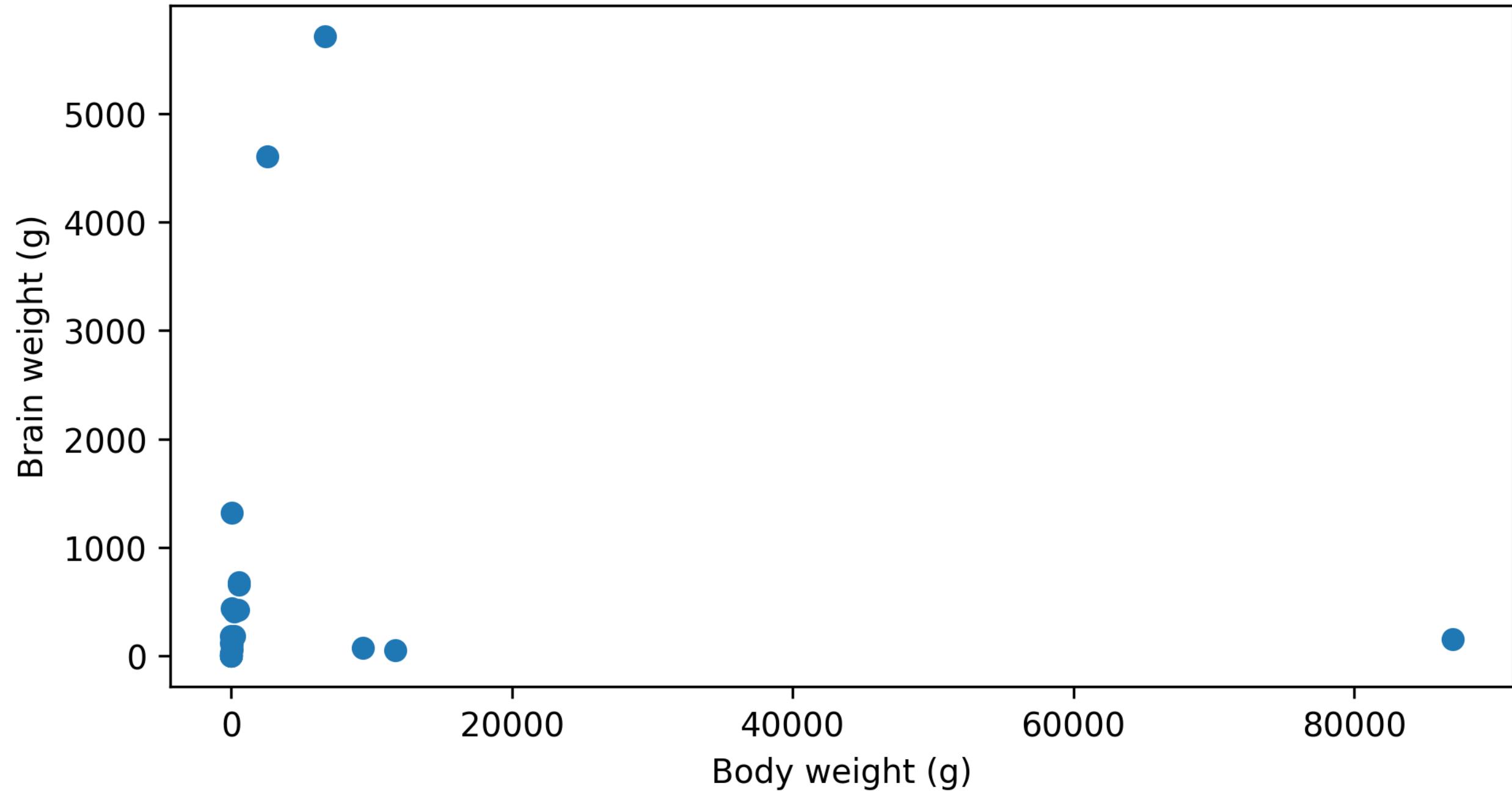
Iris



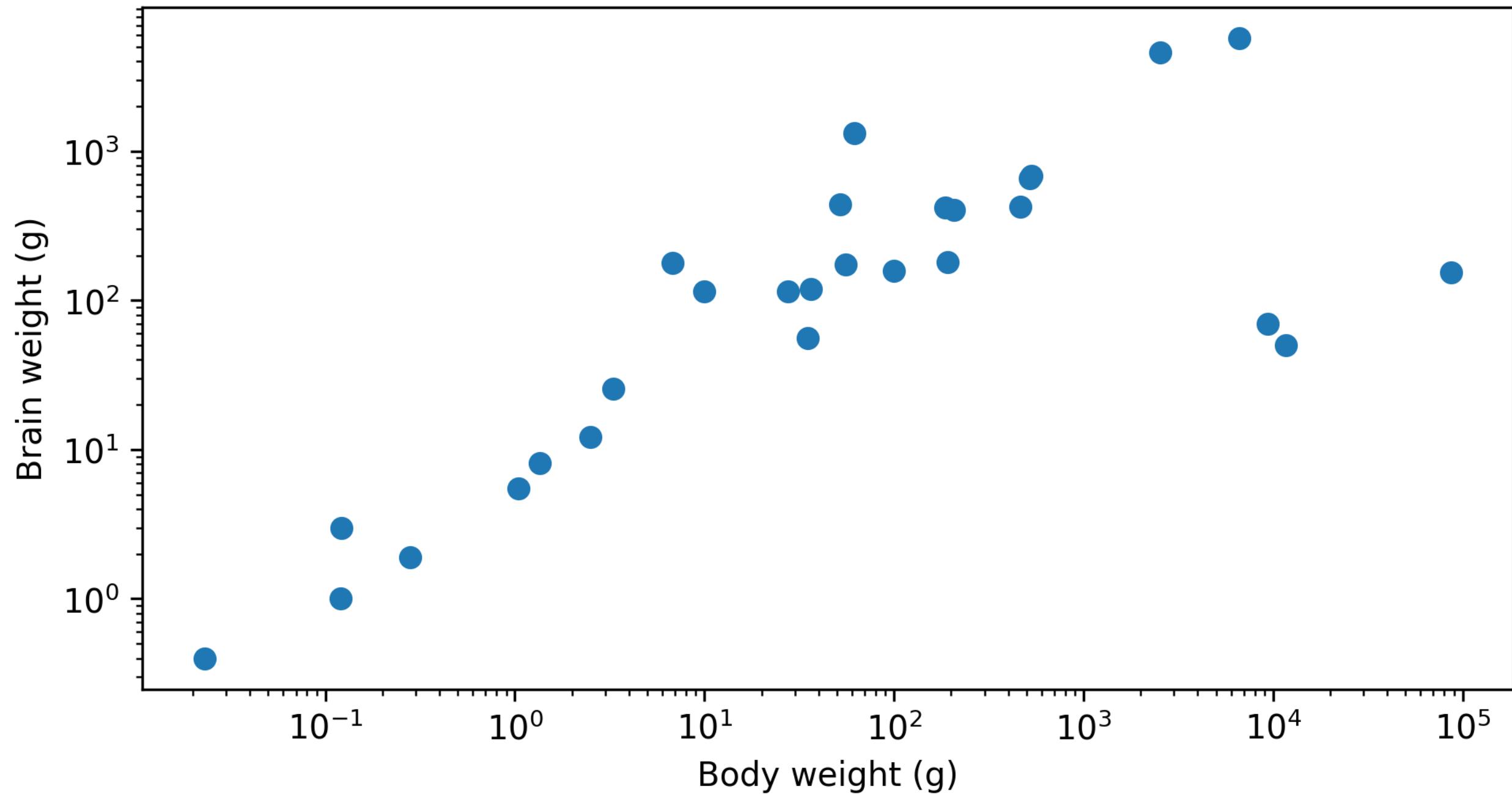




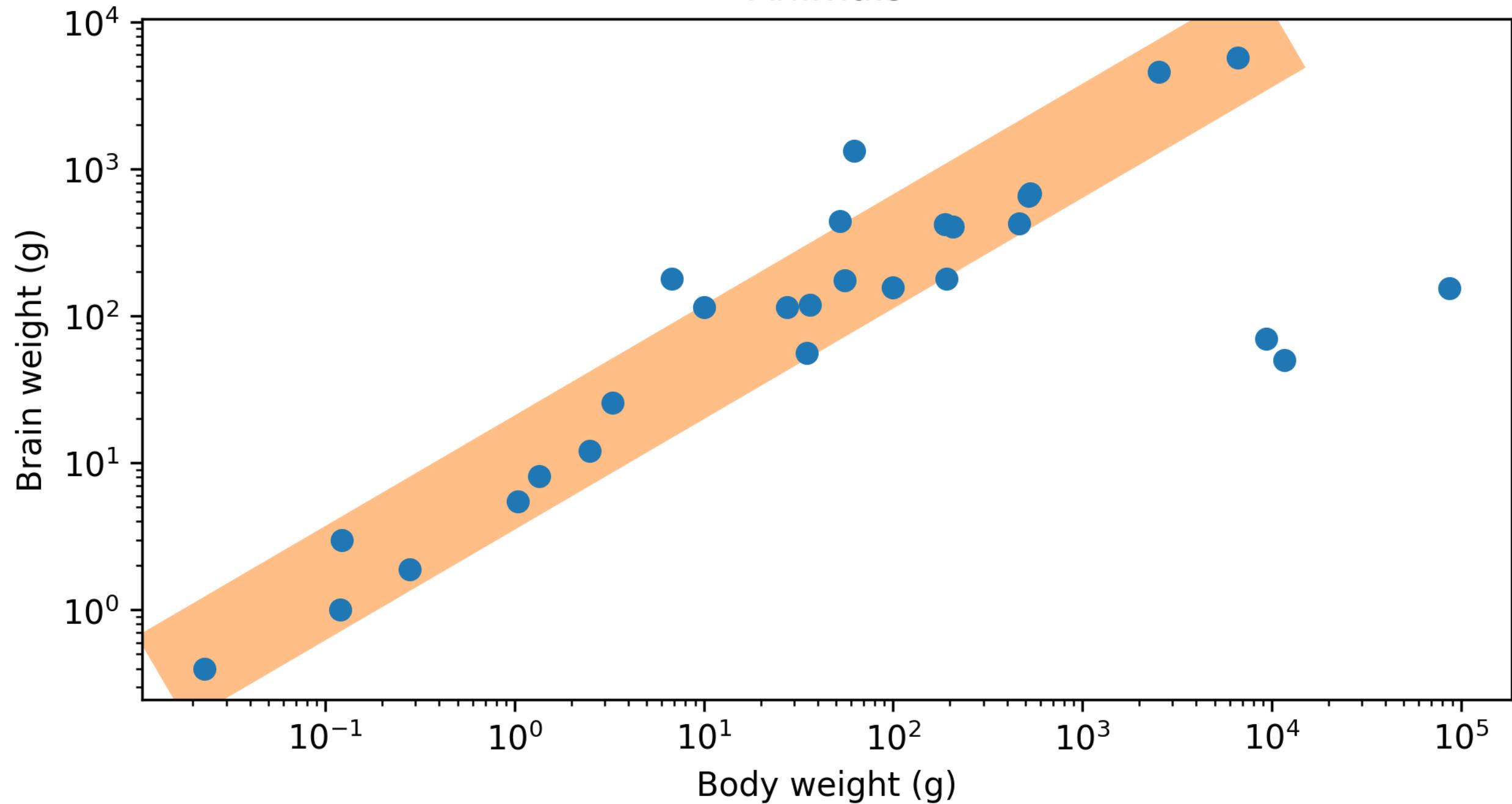
Animals



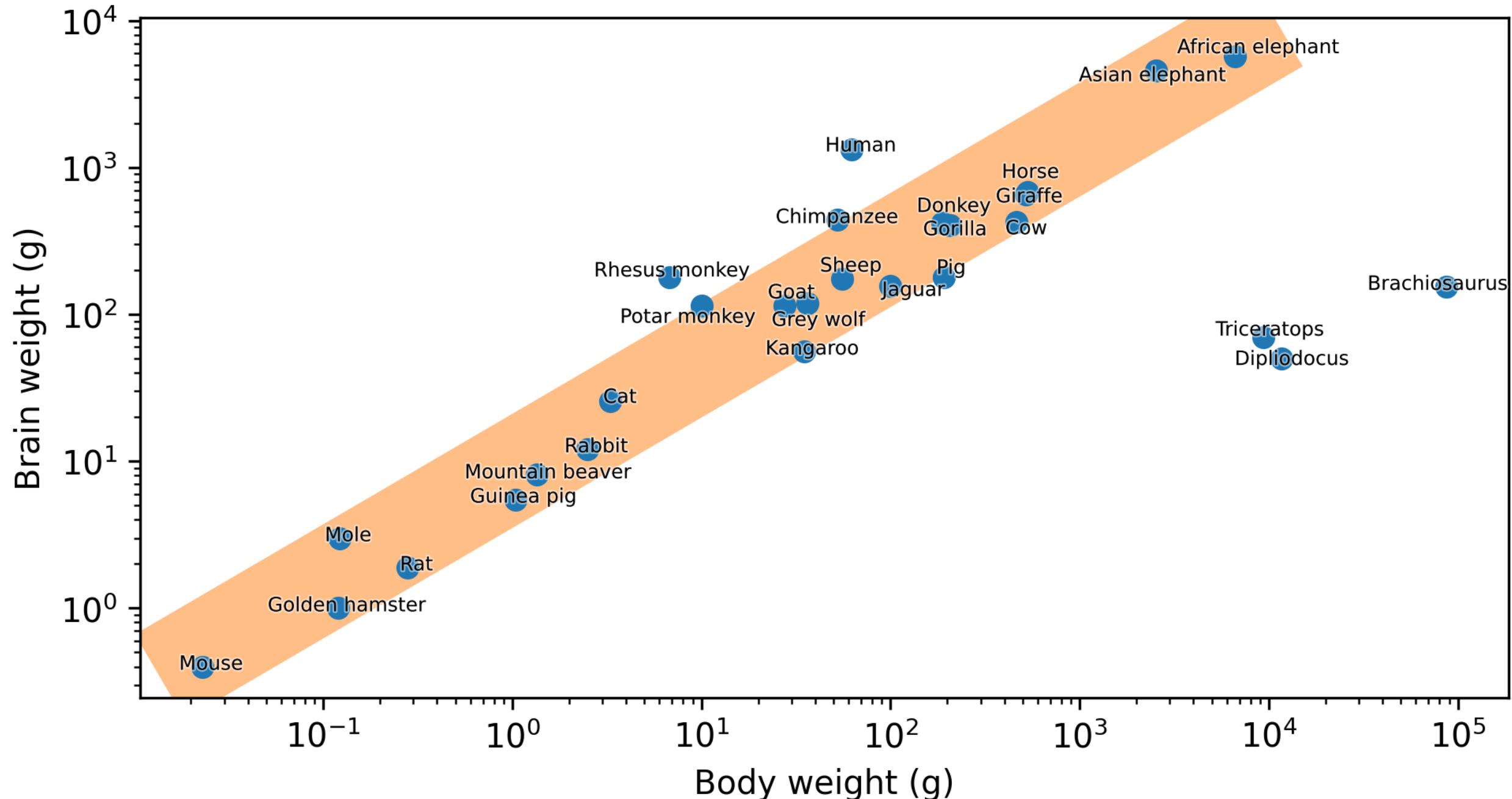
Animals



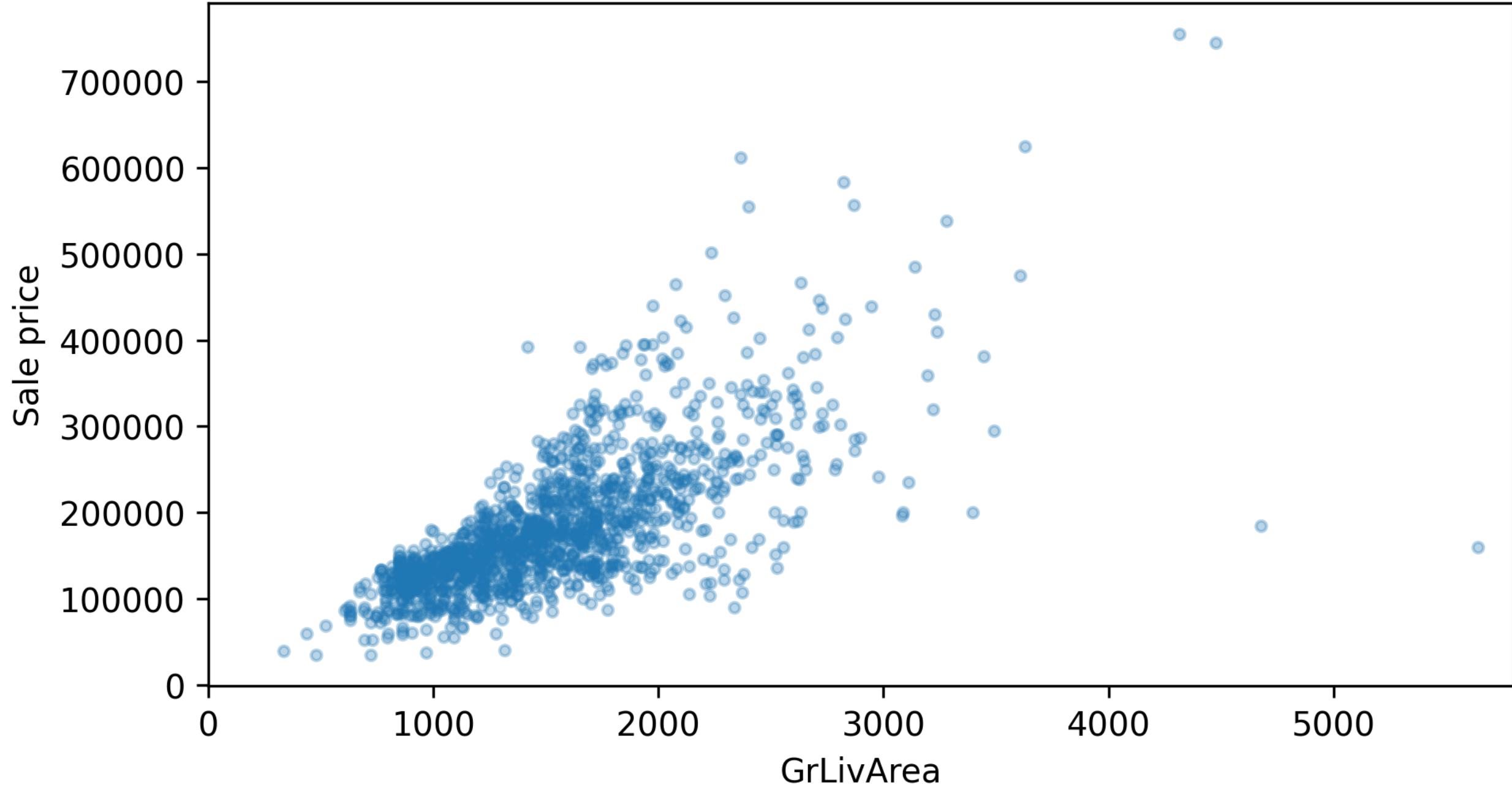
Animals



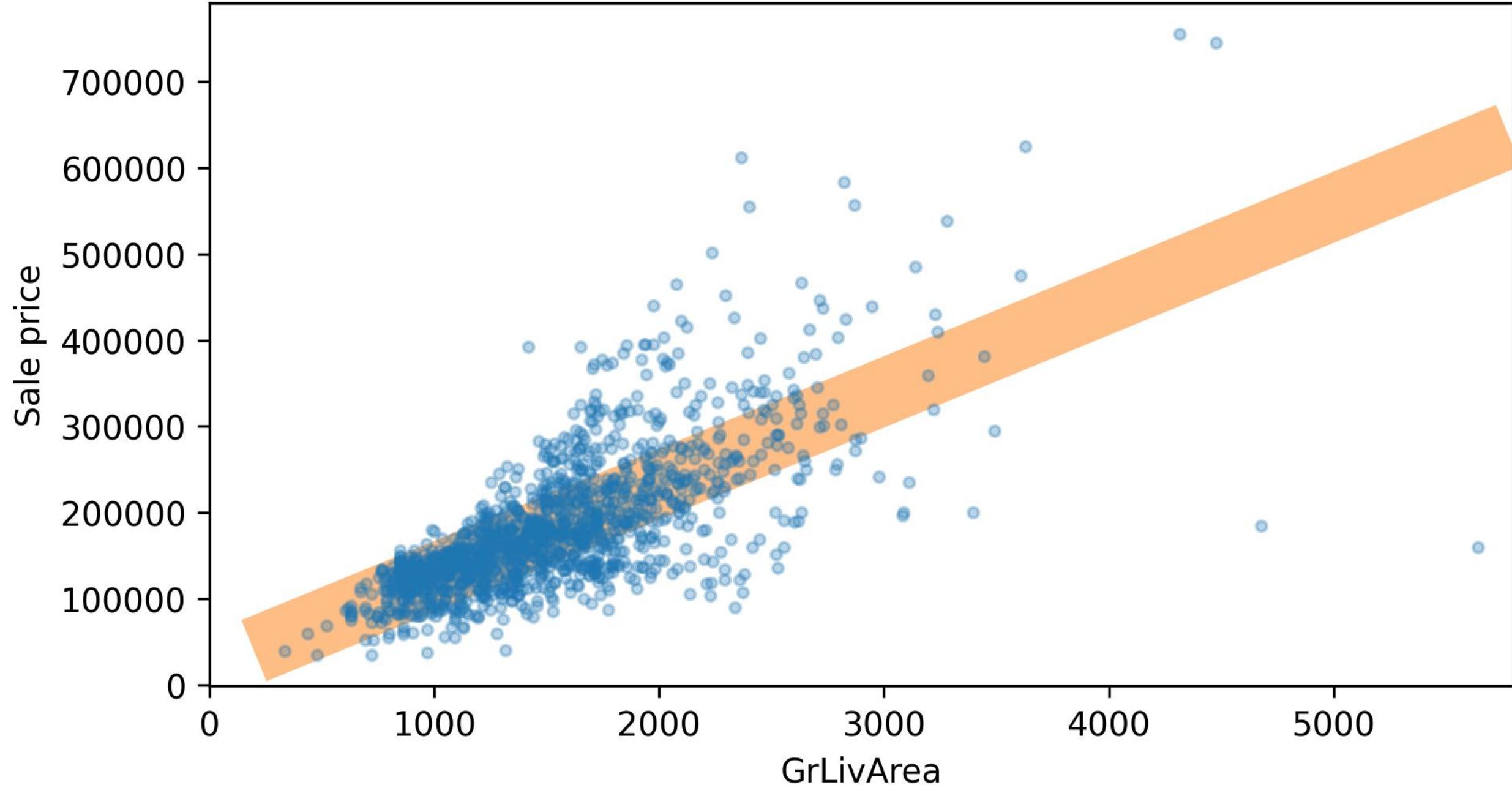
Animals



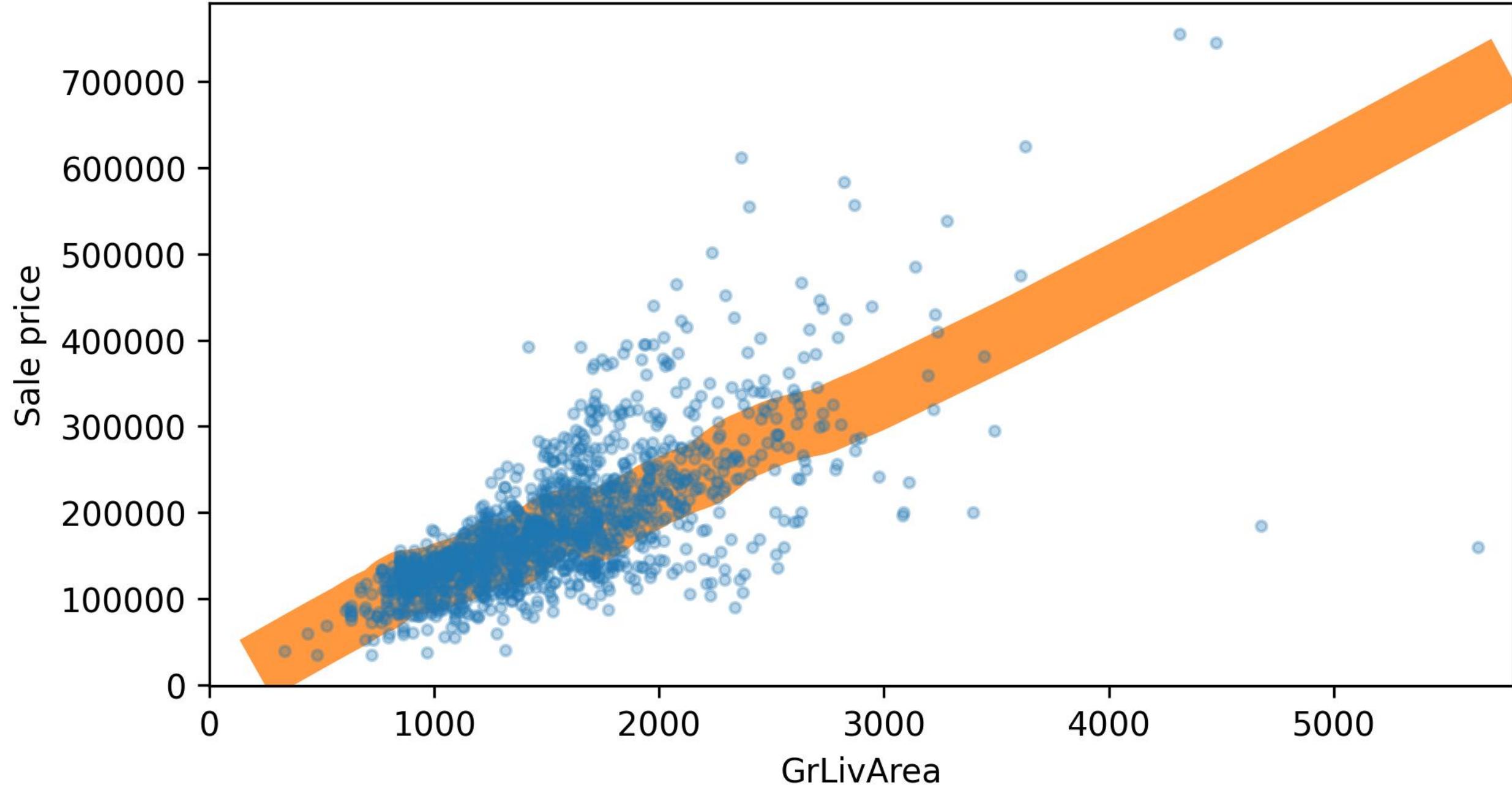
House prices (Ames, Iowa, 2006-2010)



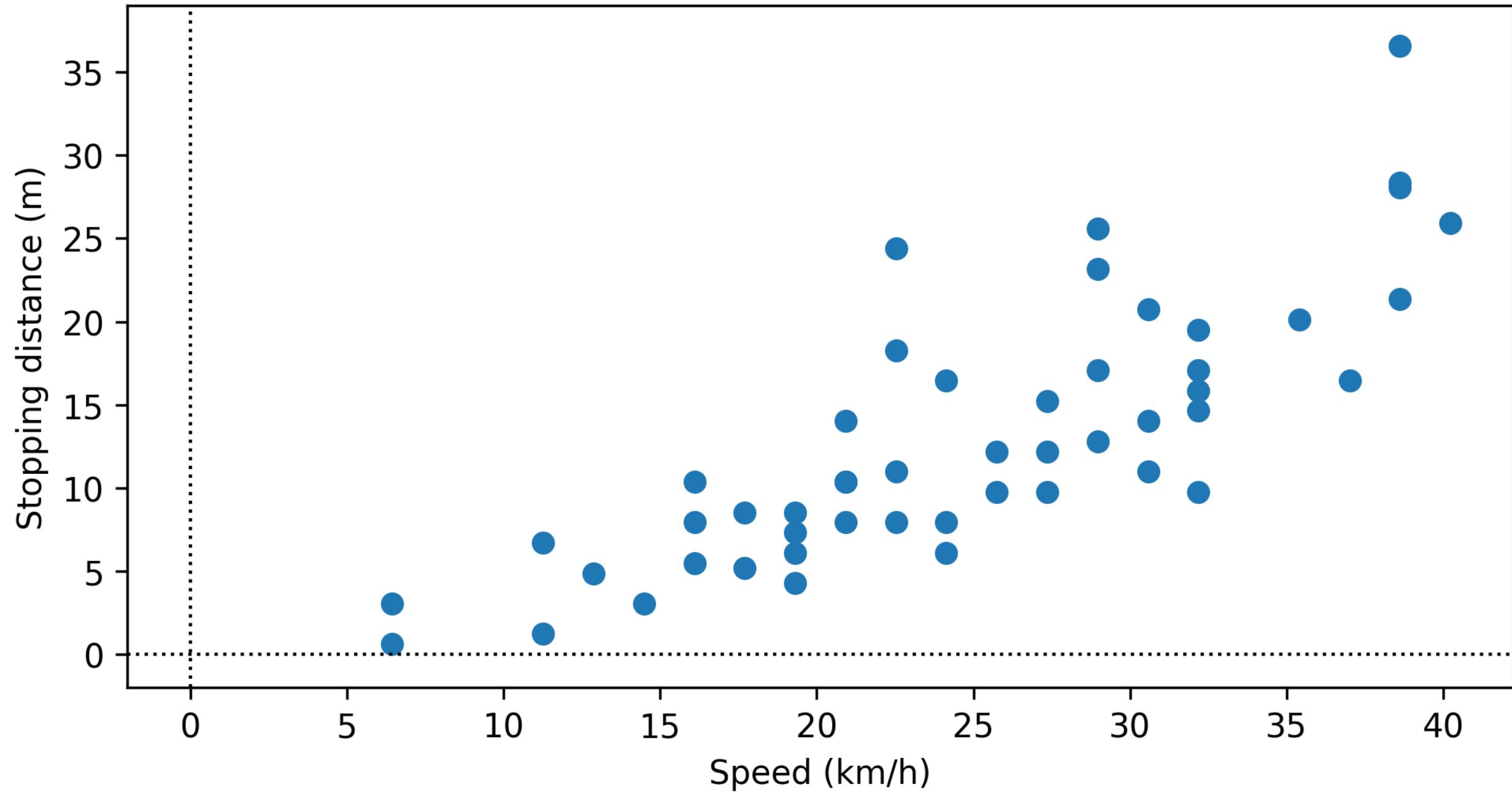
House prices (Ames, Iowa, 2006-2010)



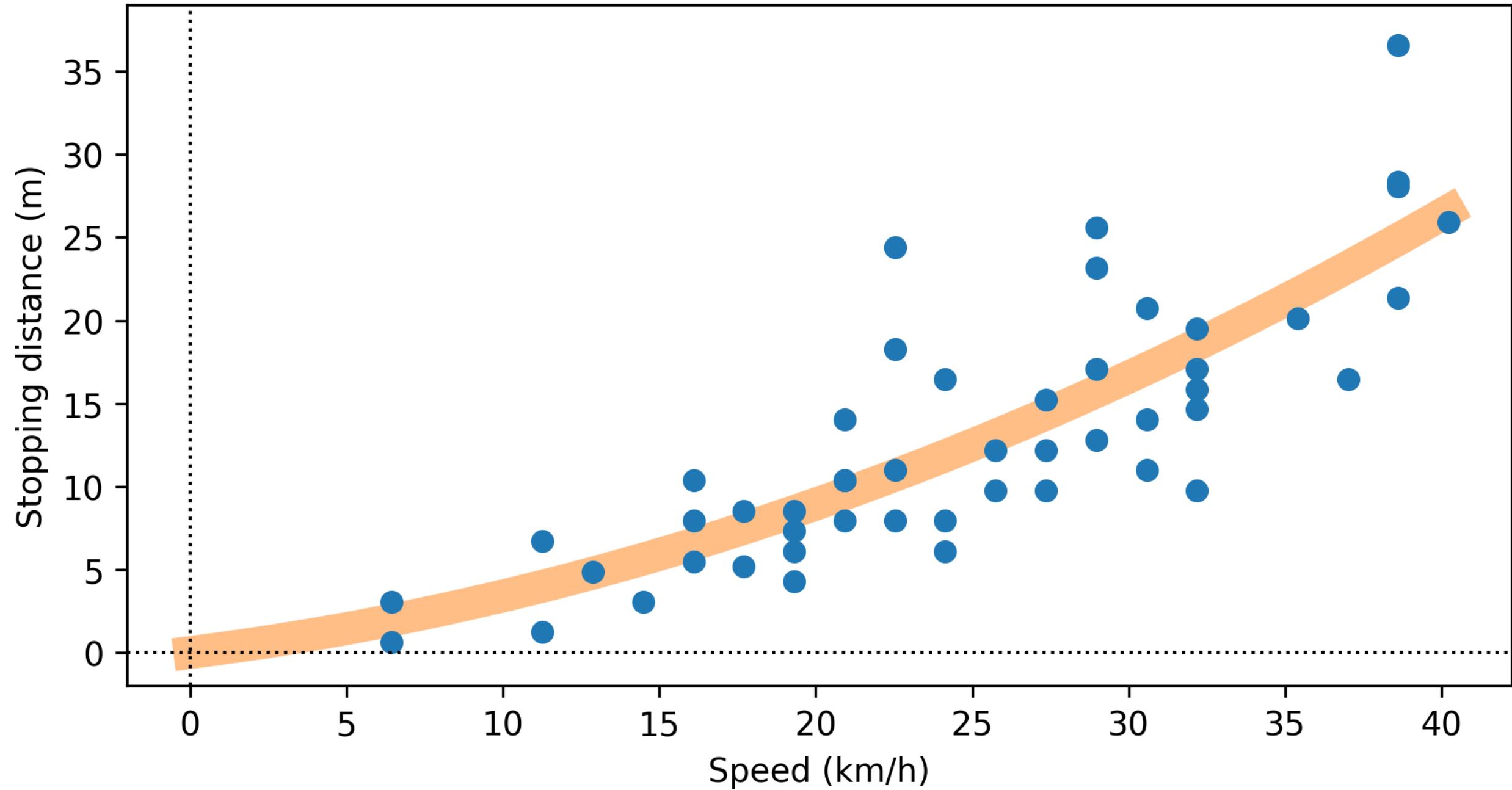
House prices (Ames, Iowa, 2006-2010)



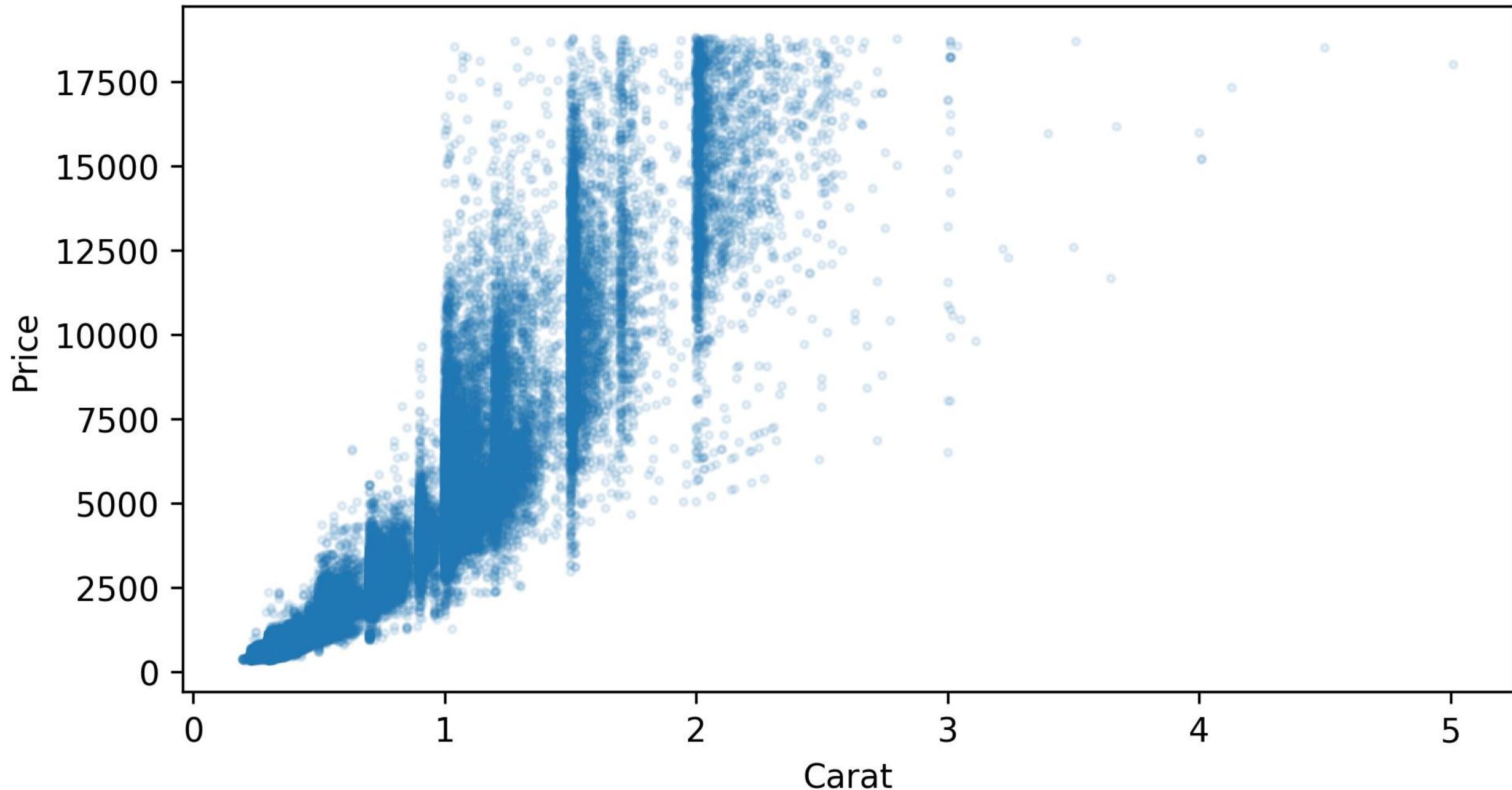
Car stopping distances

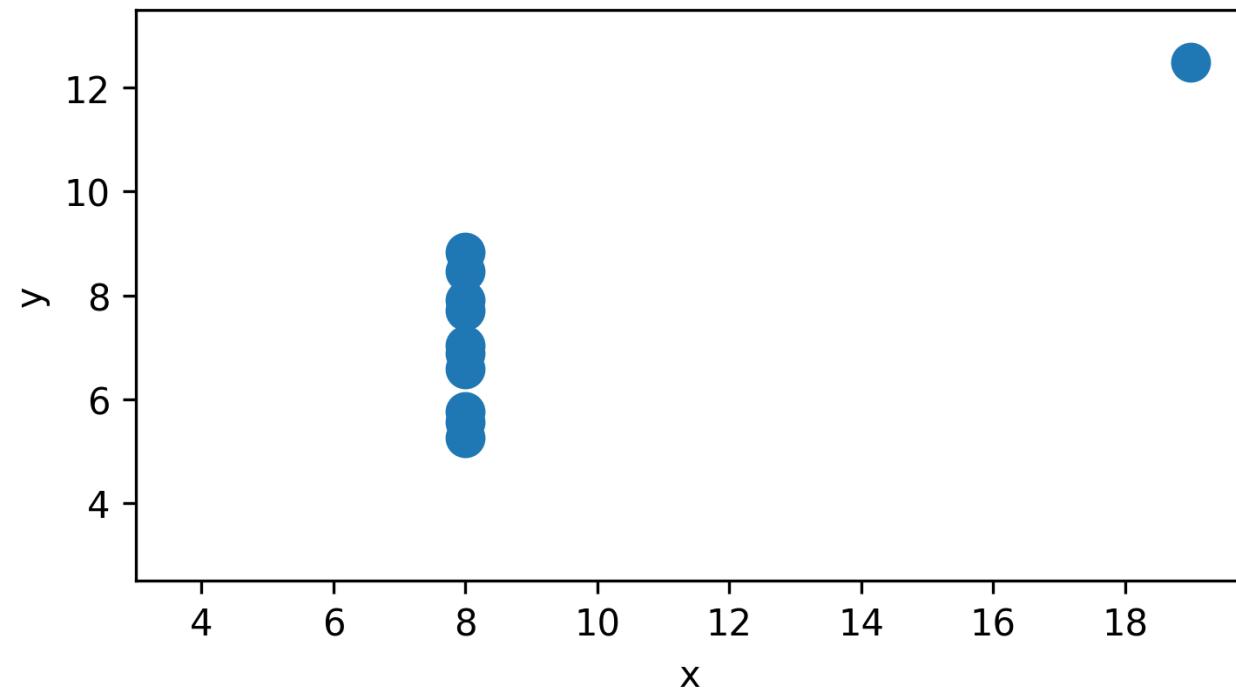
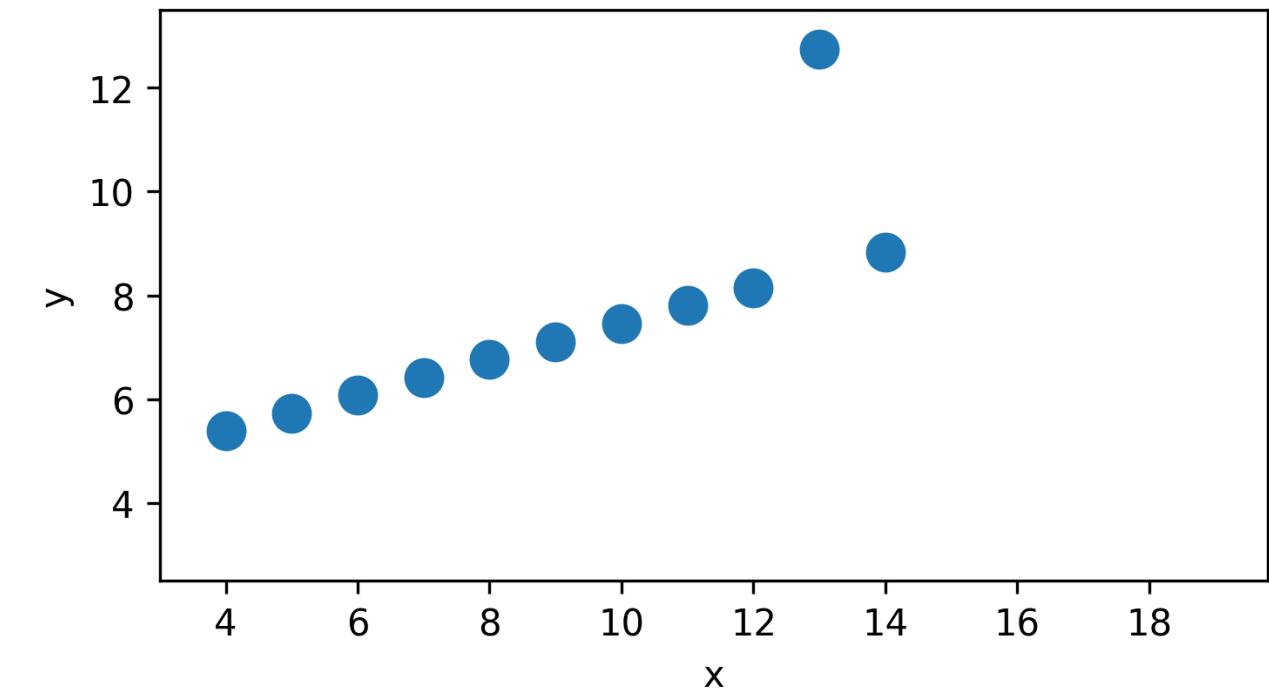
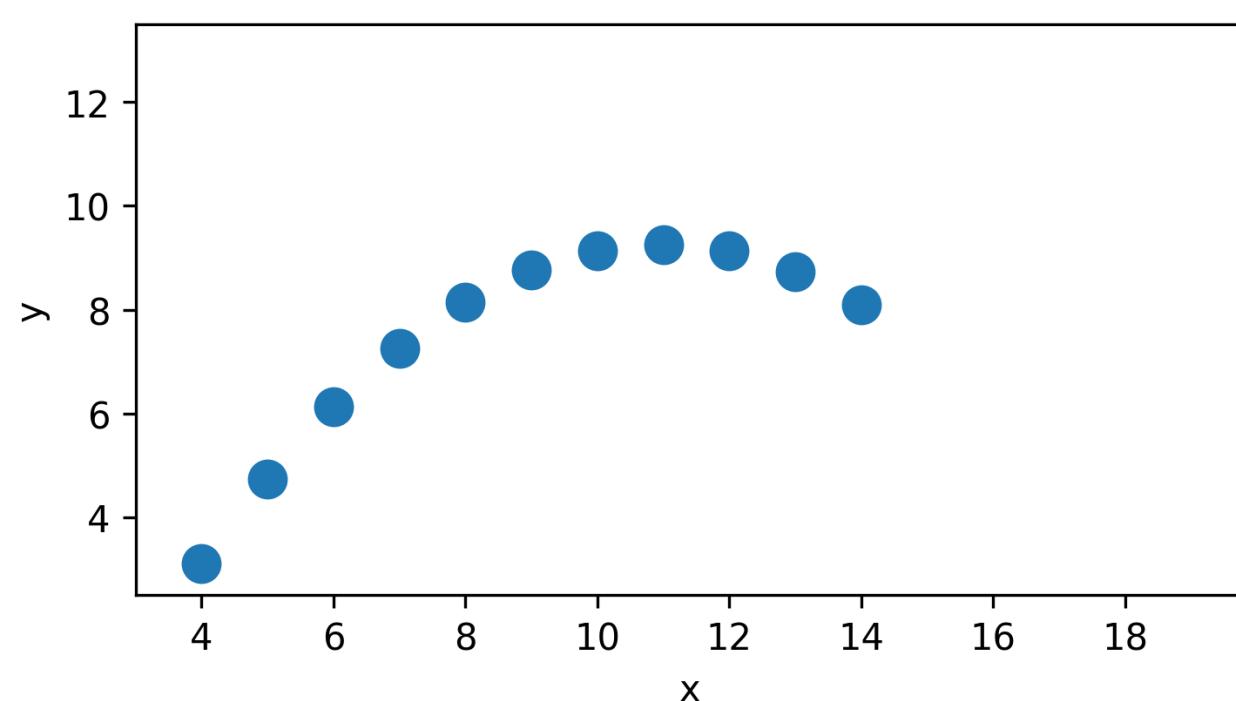
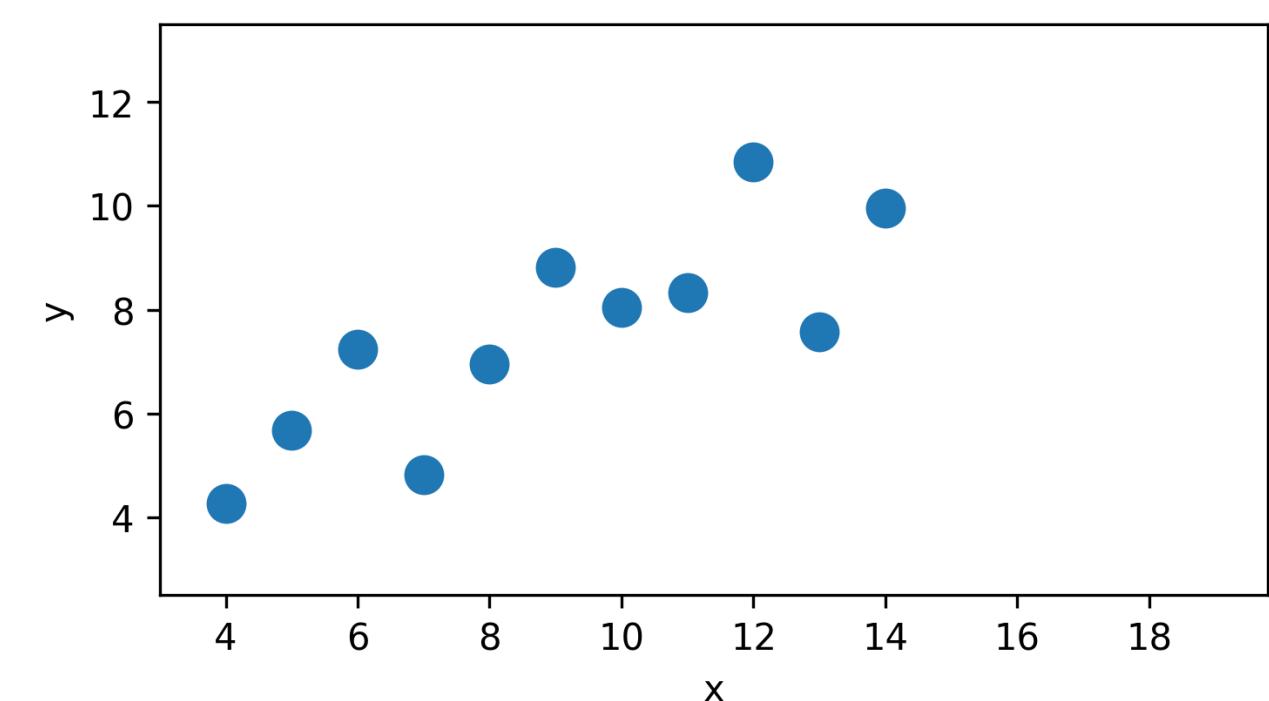


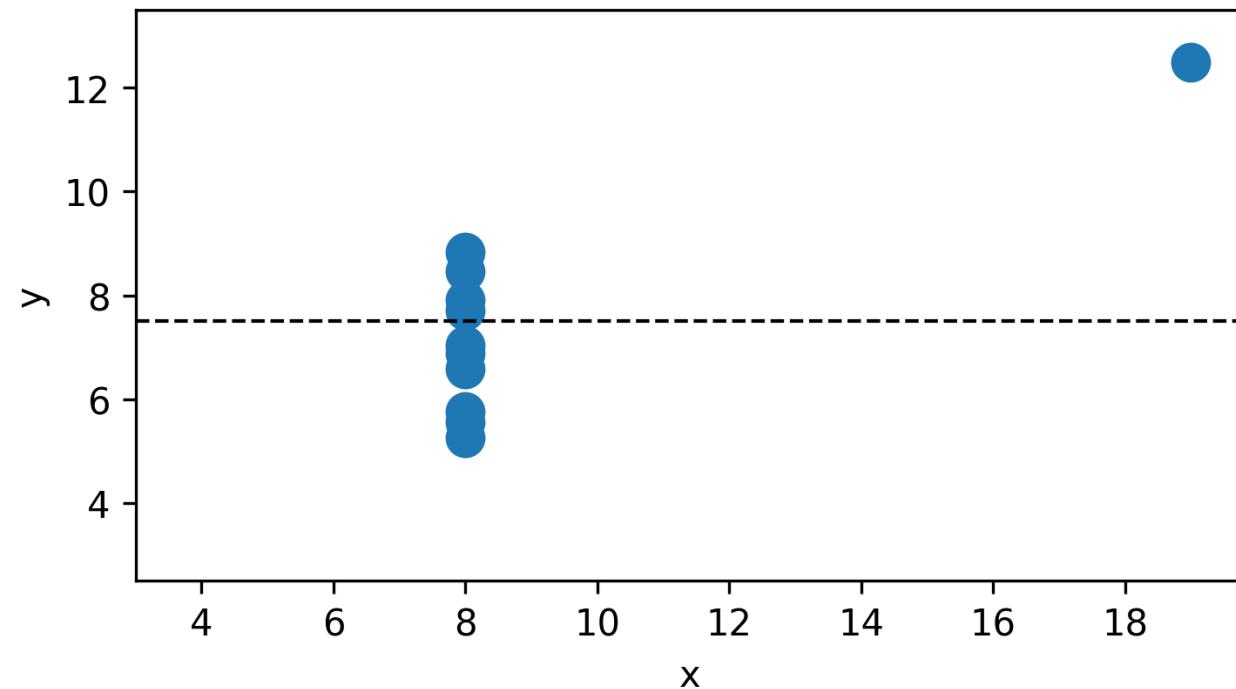
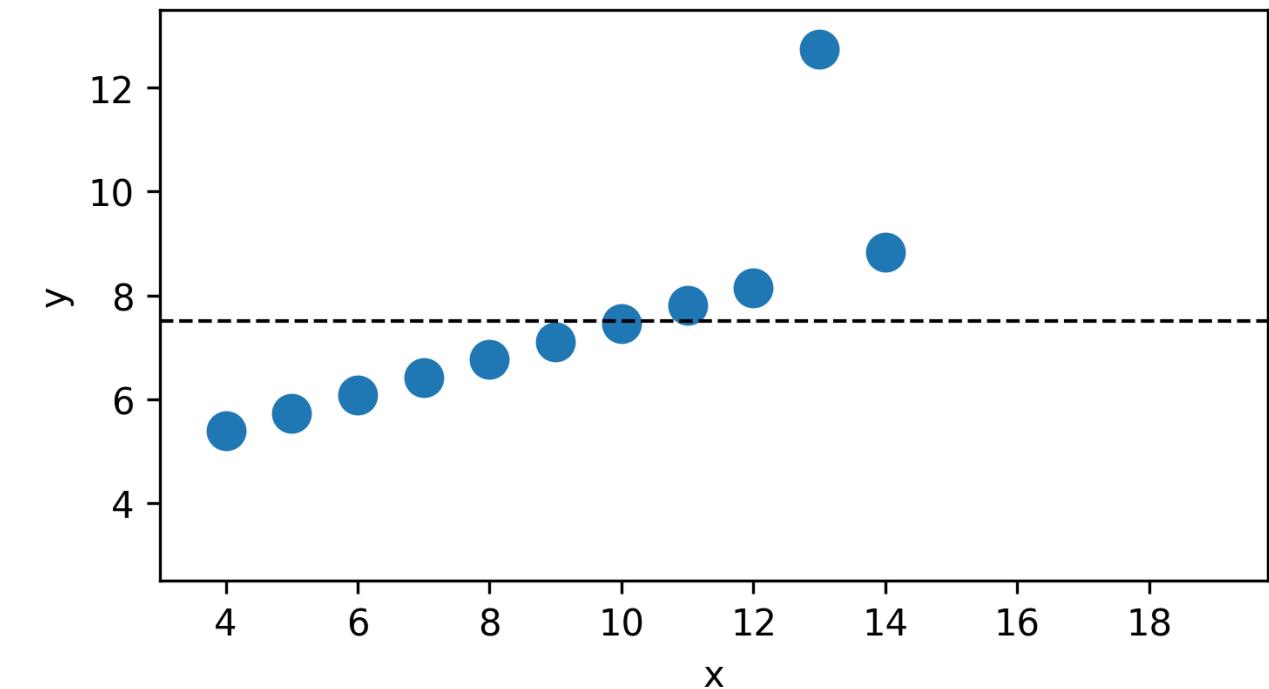
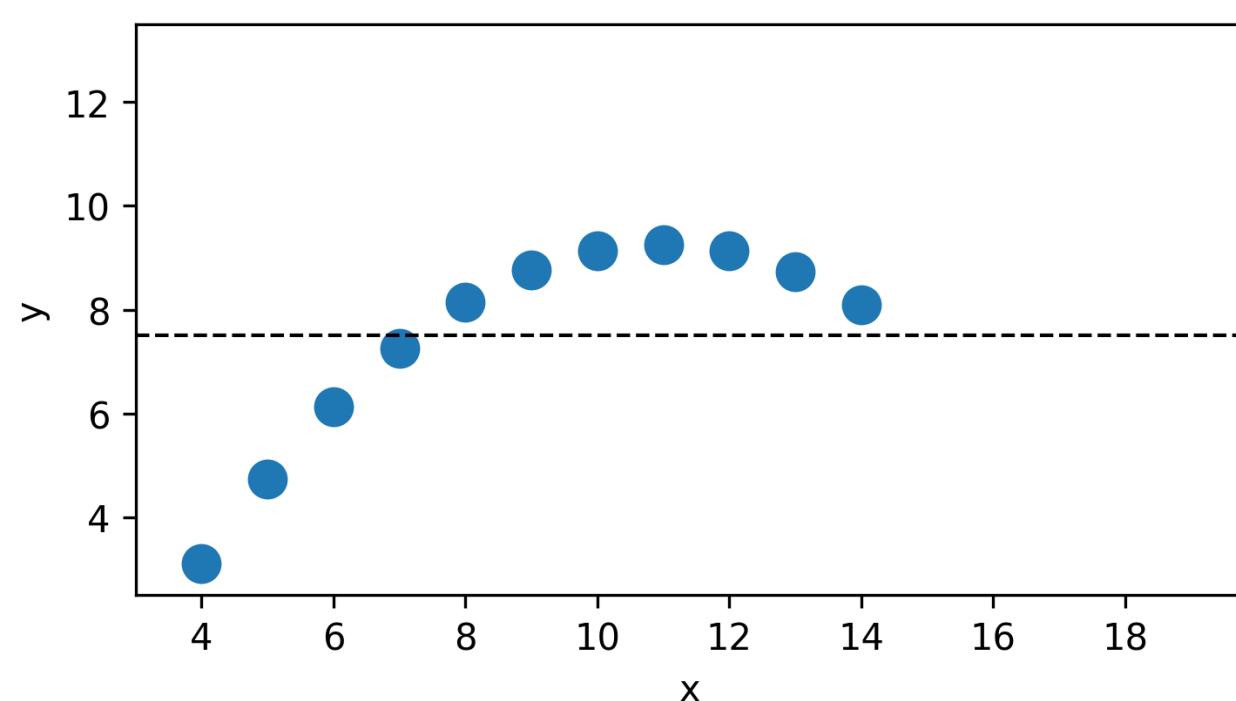
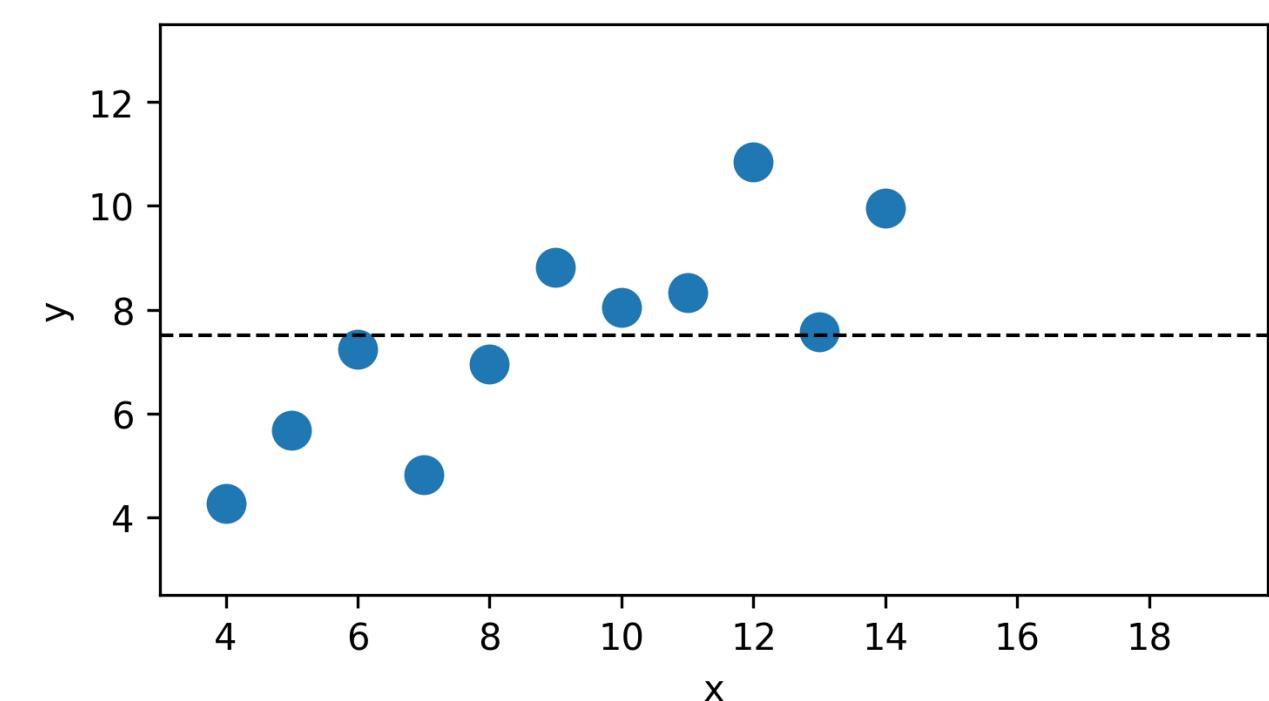
Car stopping distances

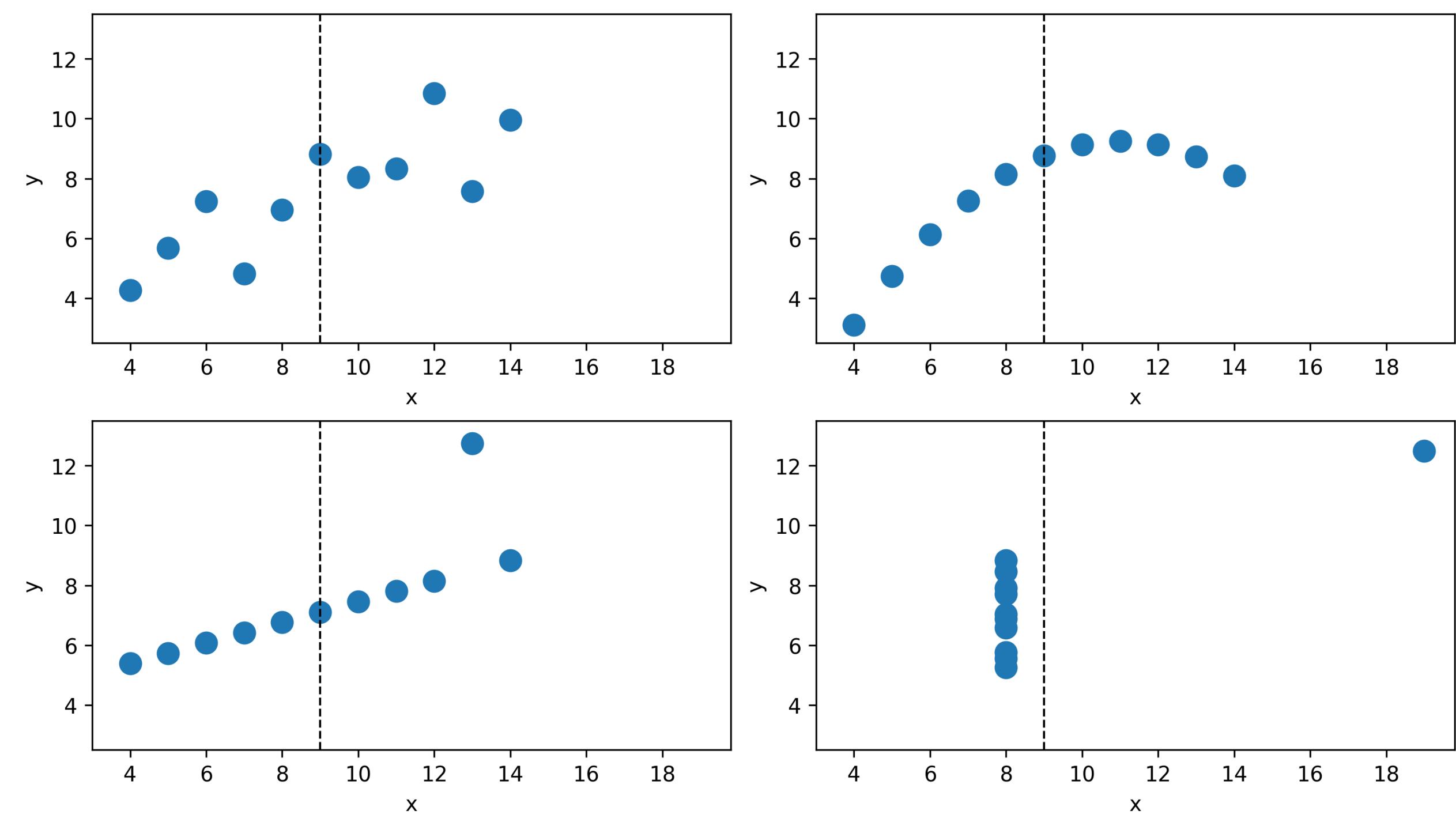


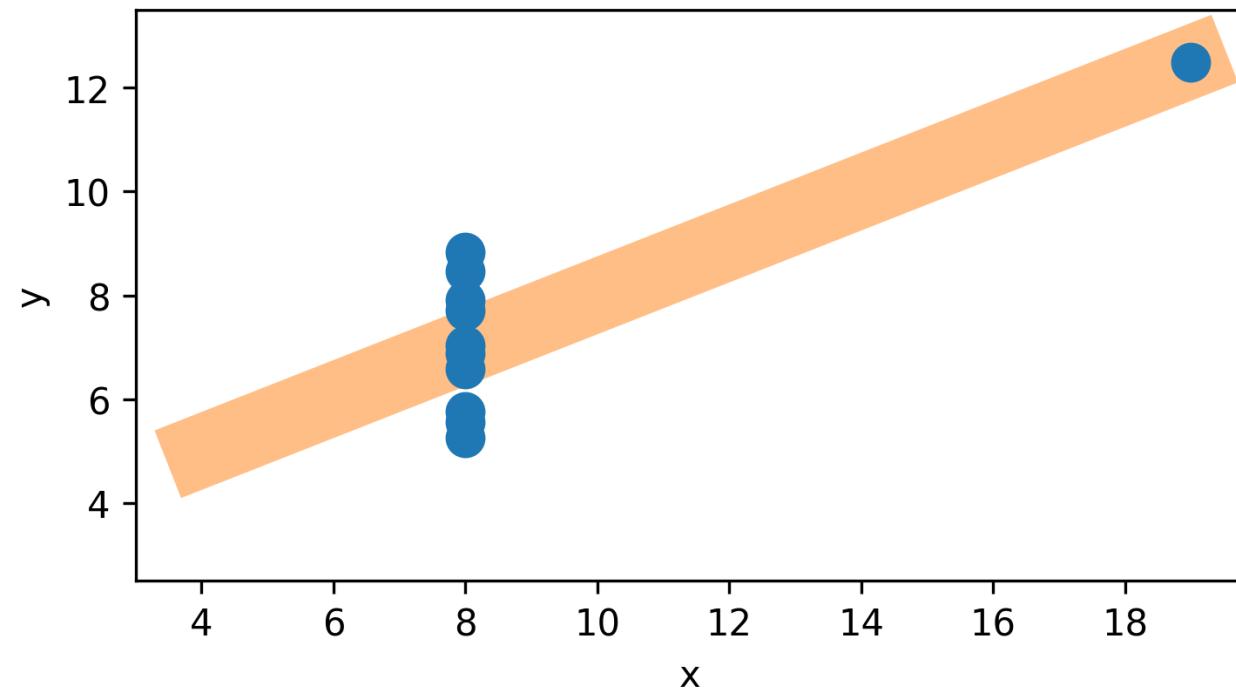
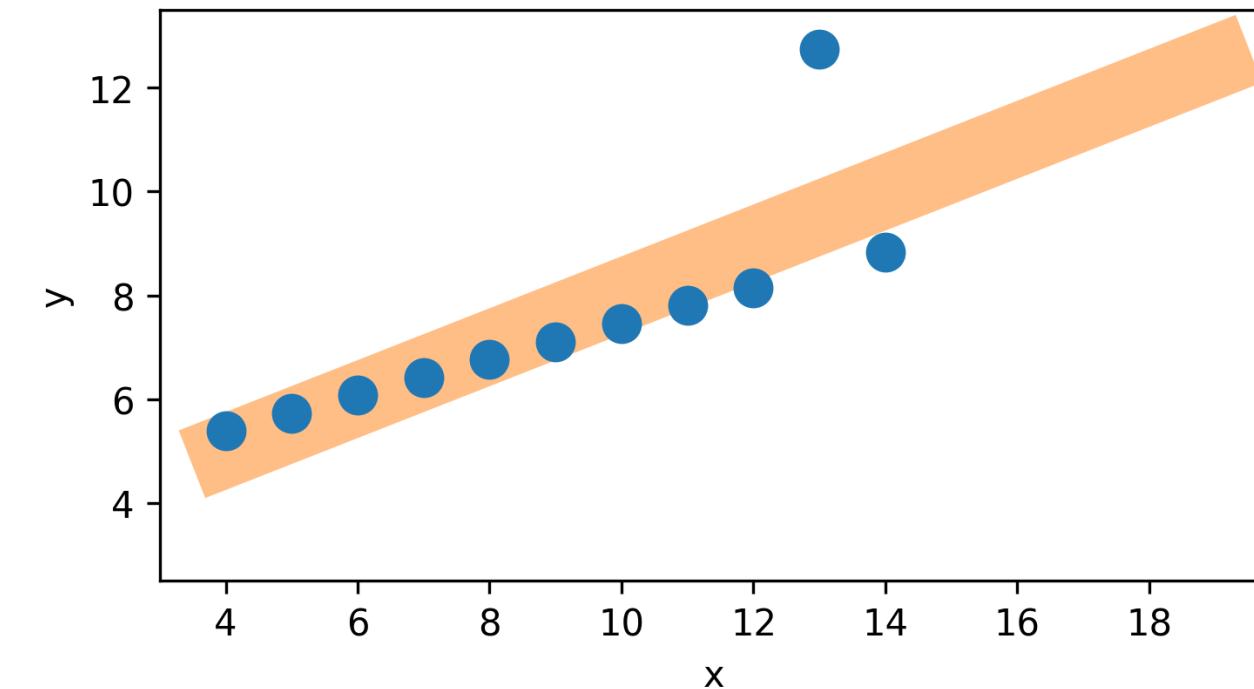
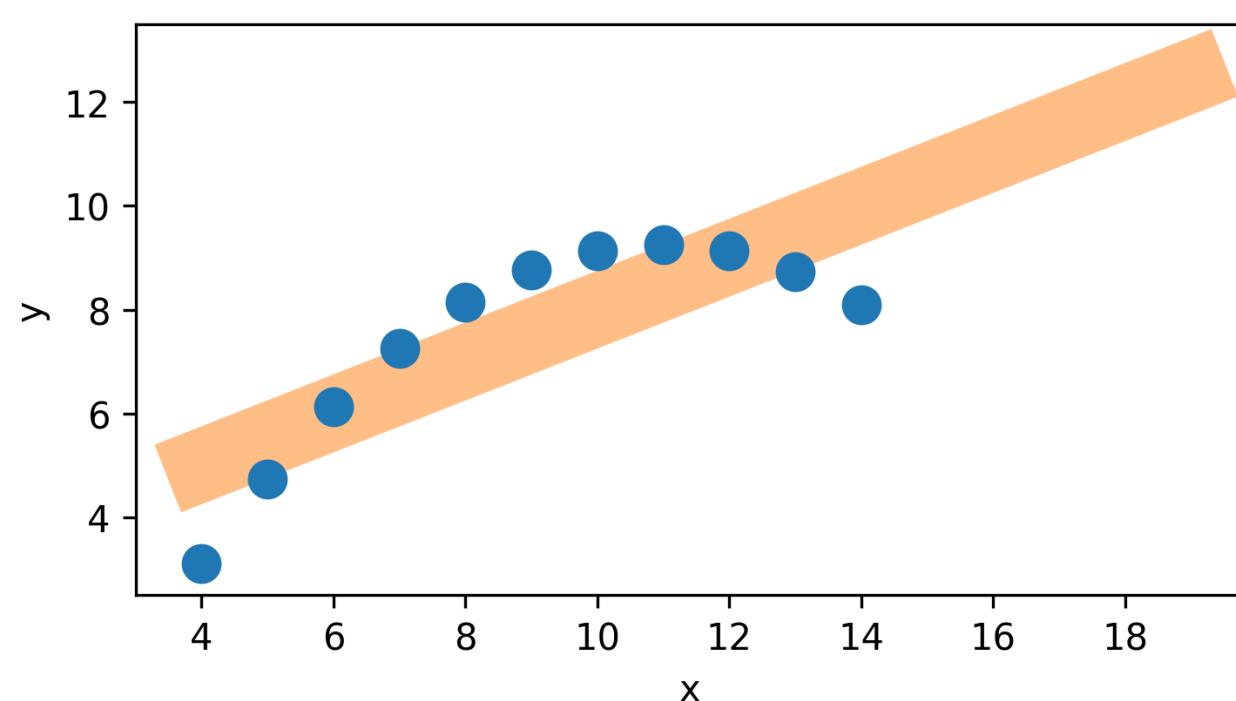
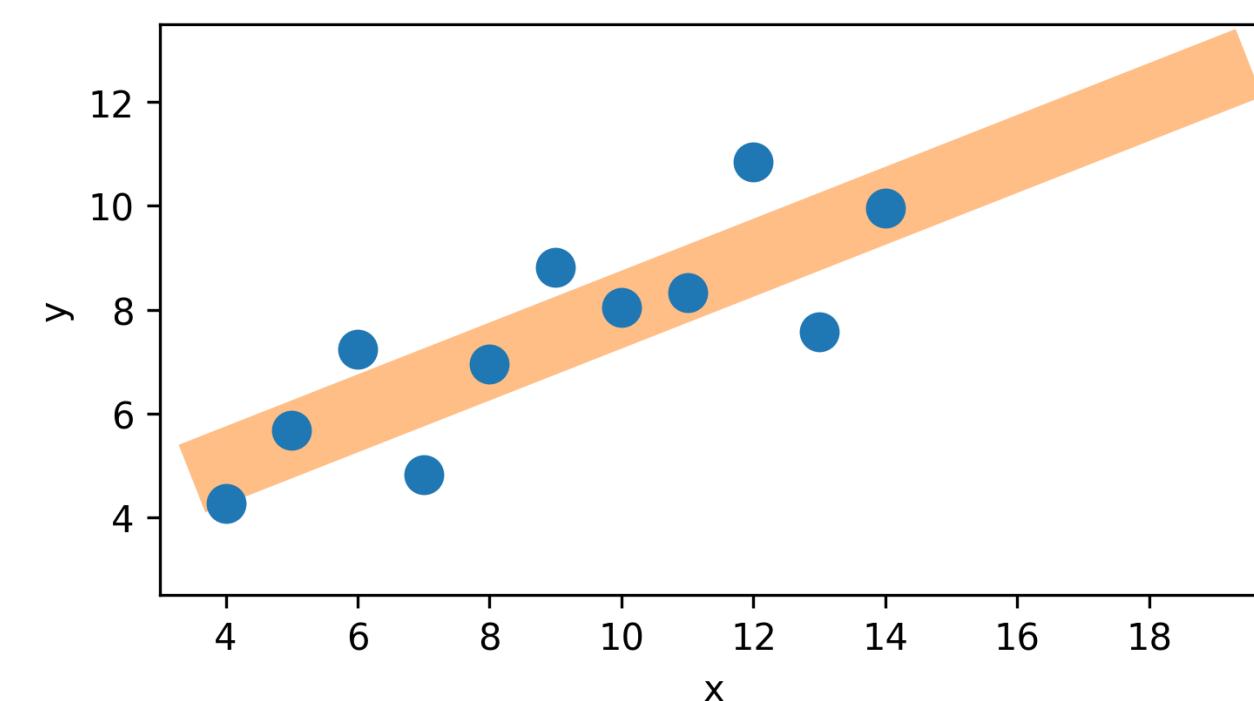
Diamonds

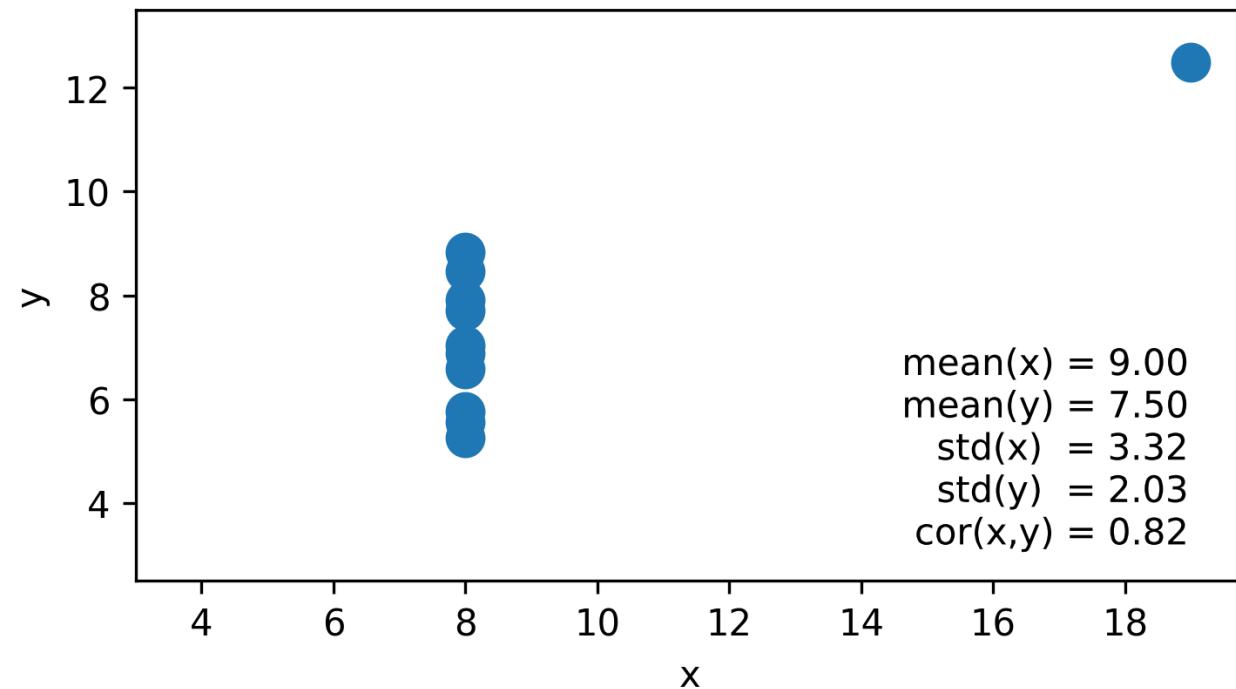
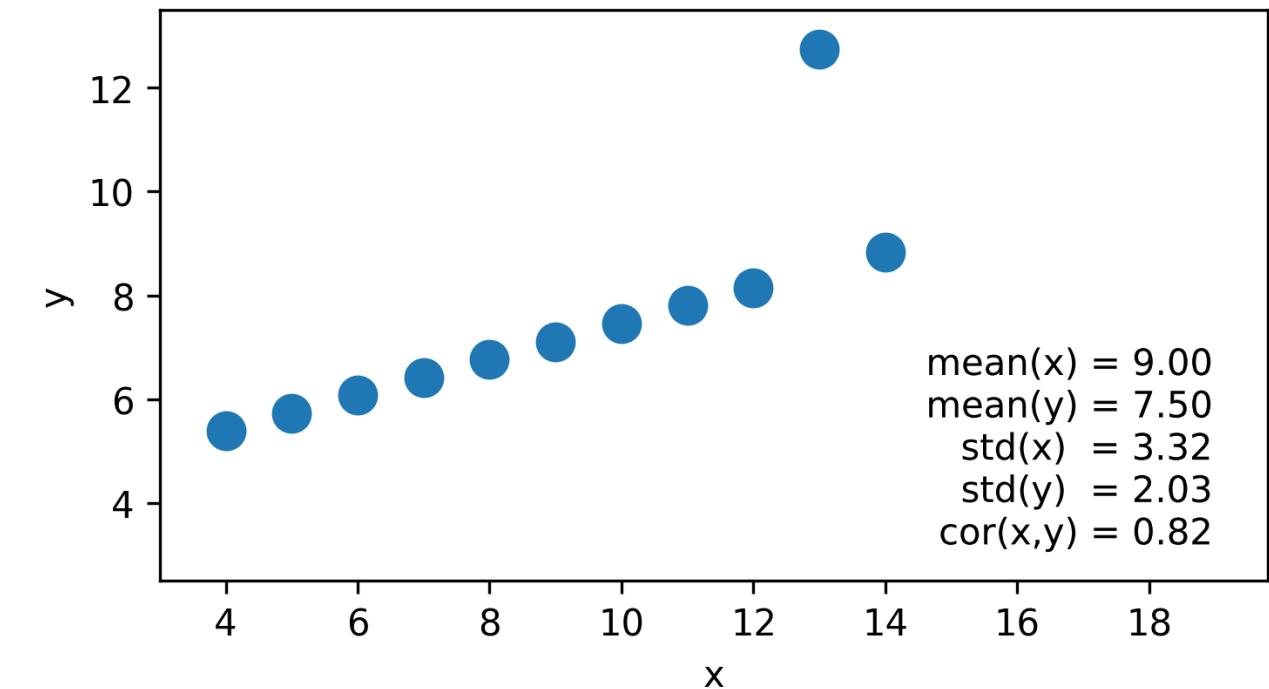
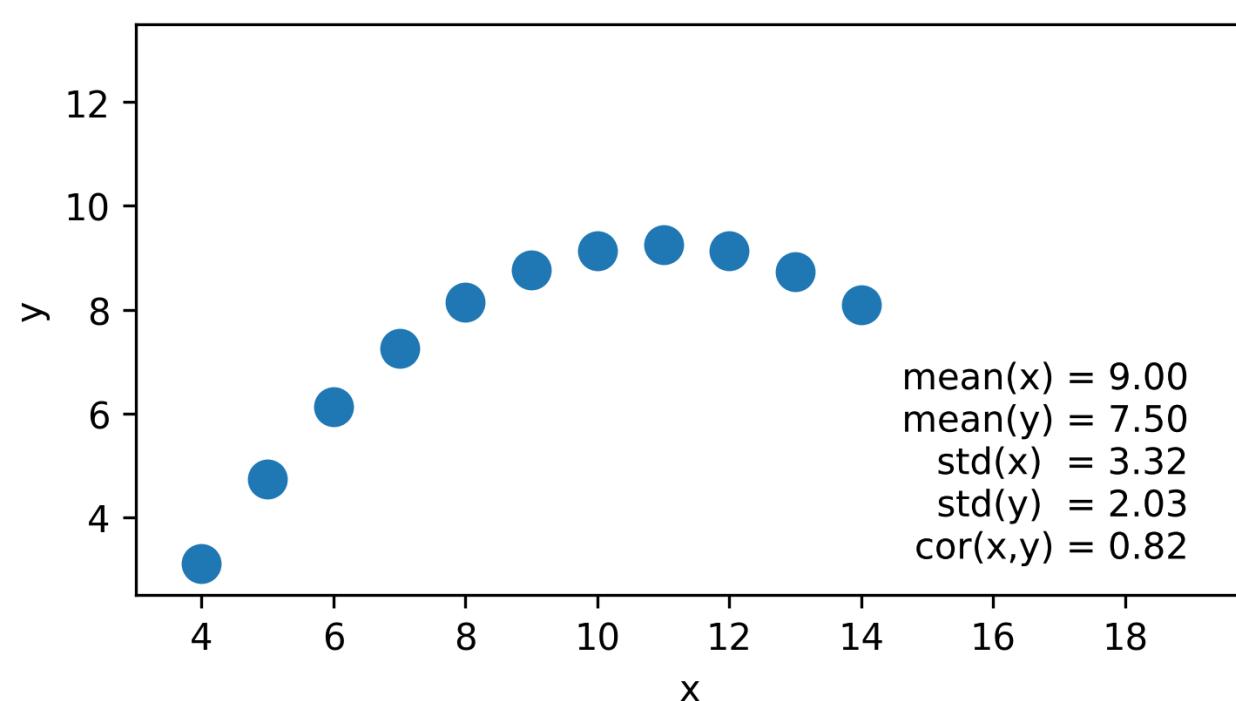
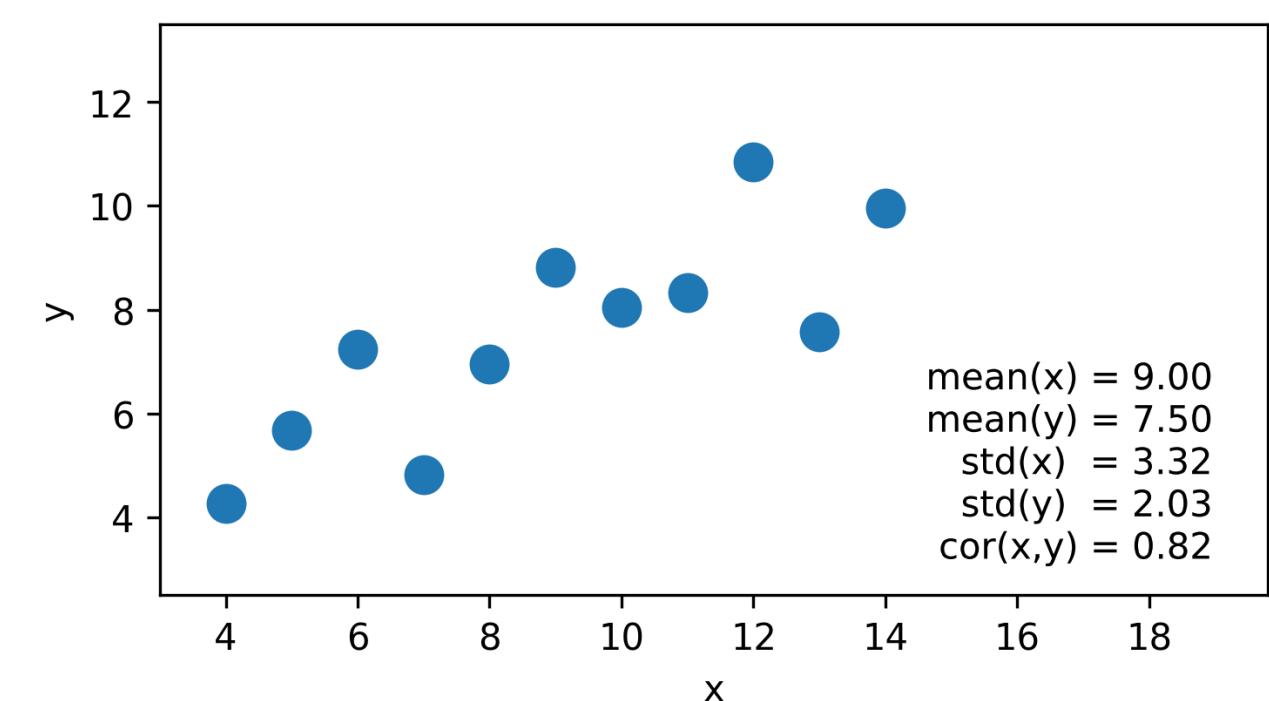


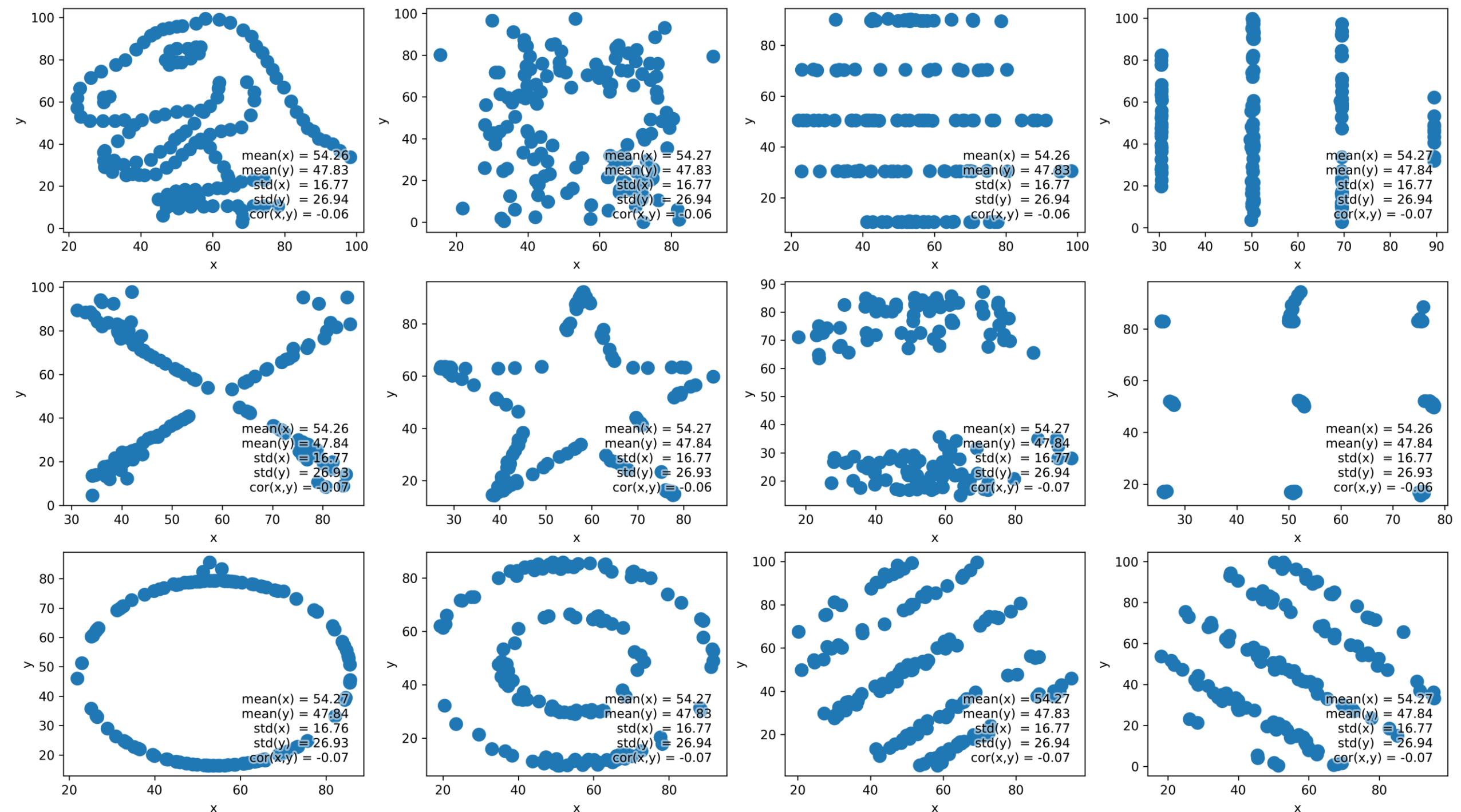


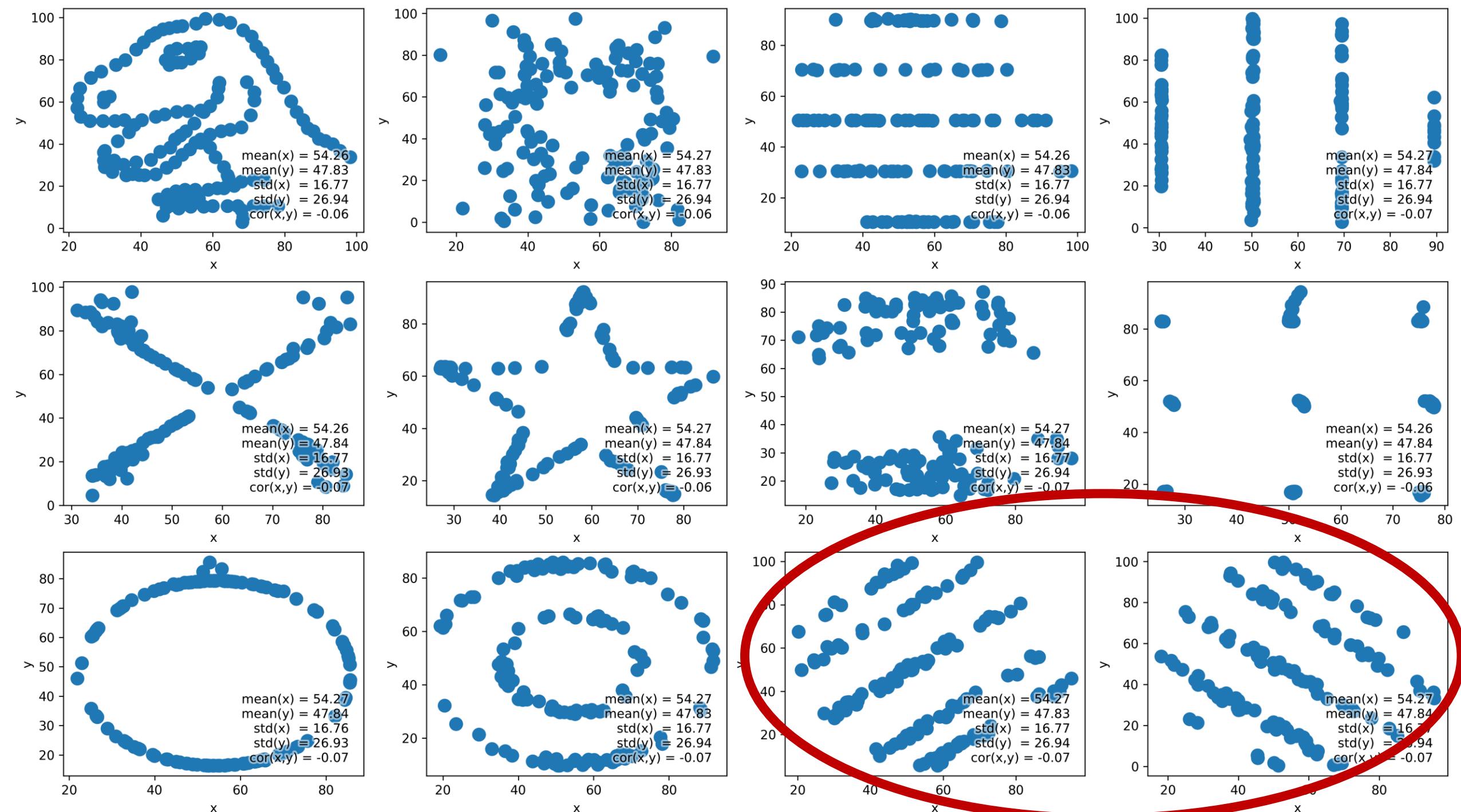




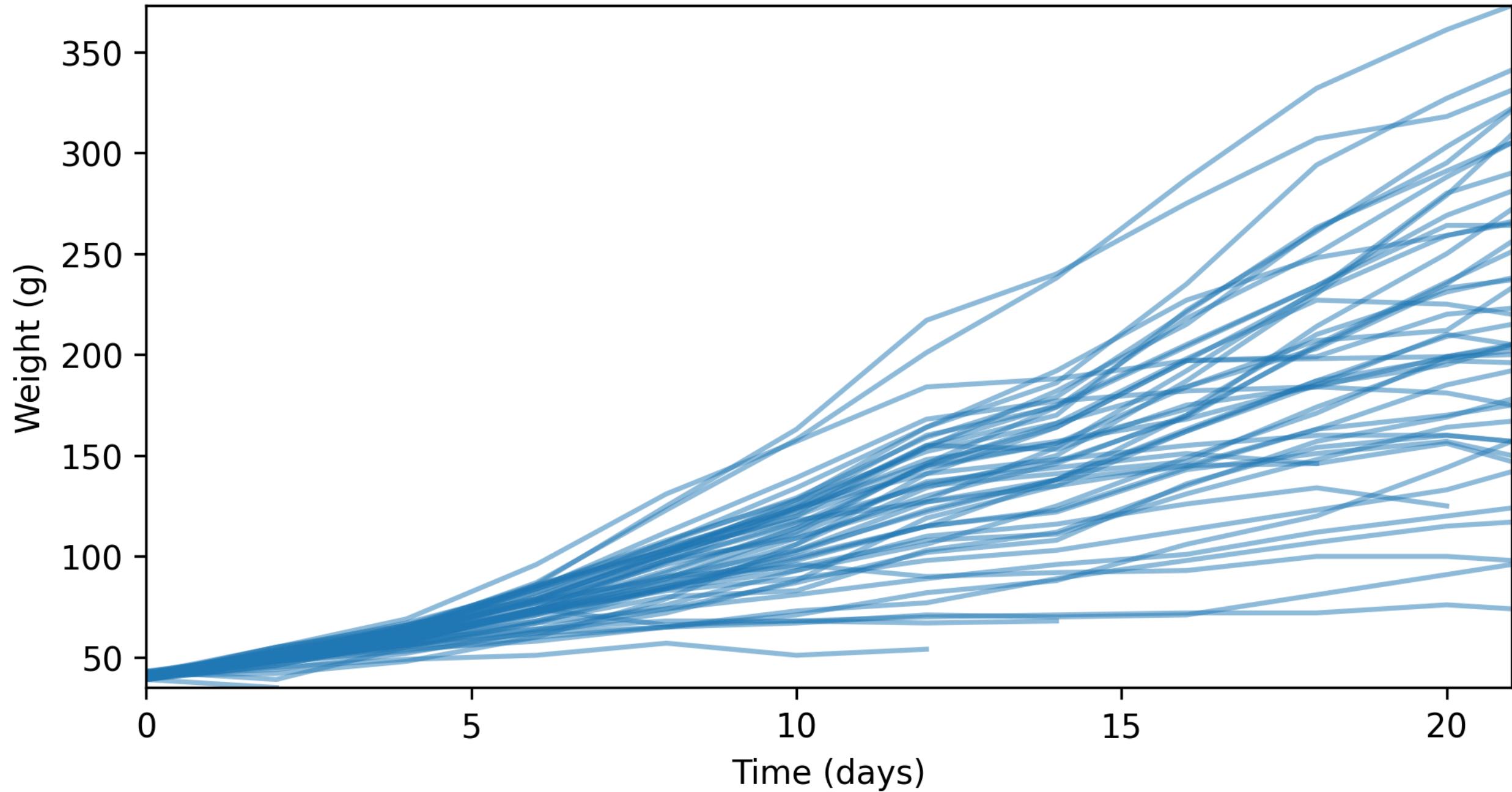




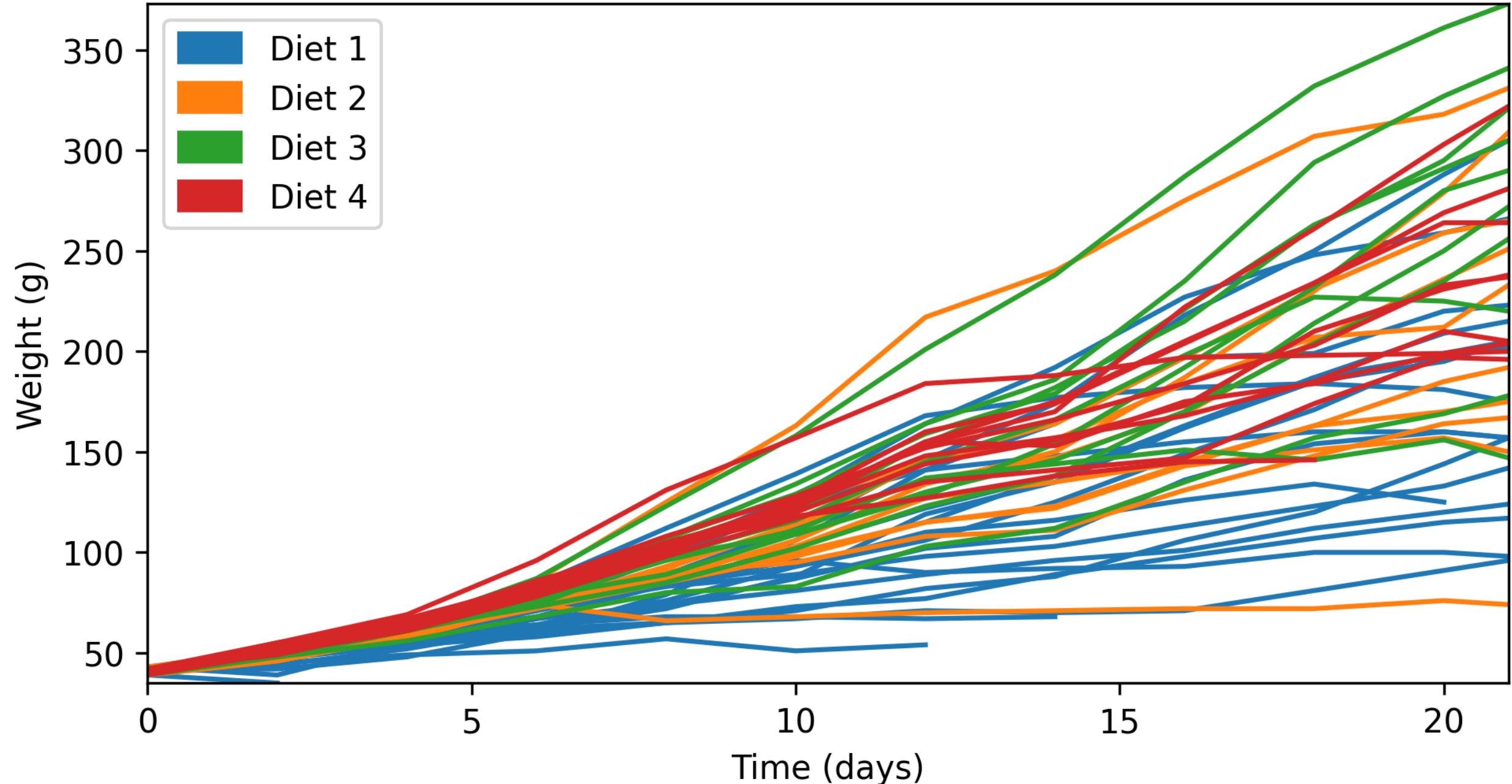




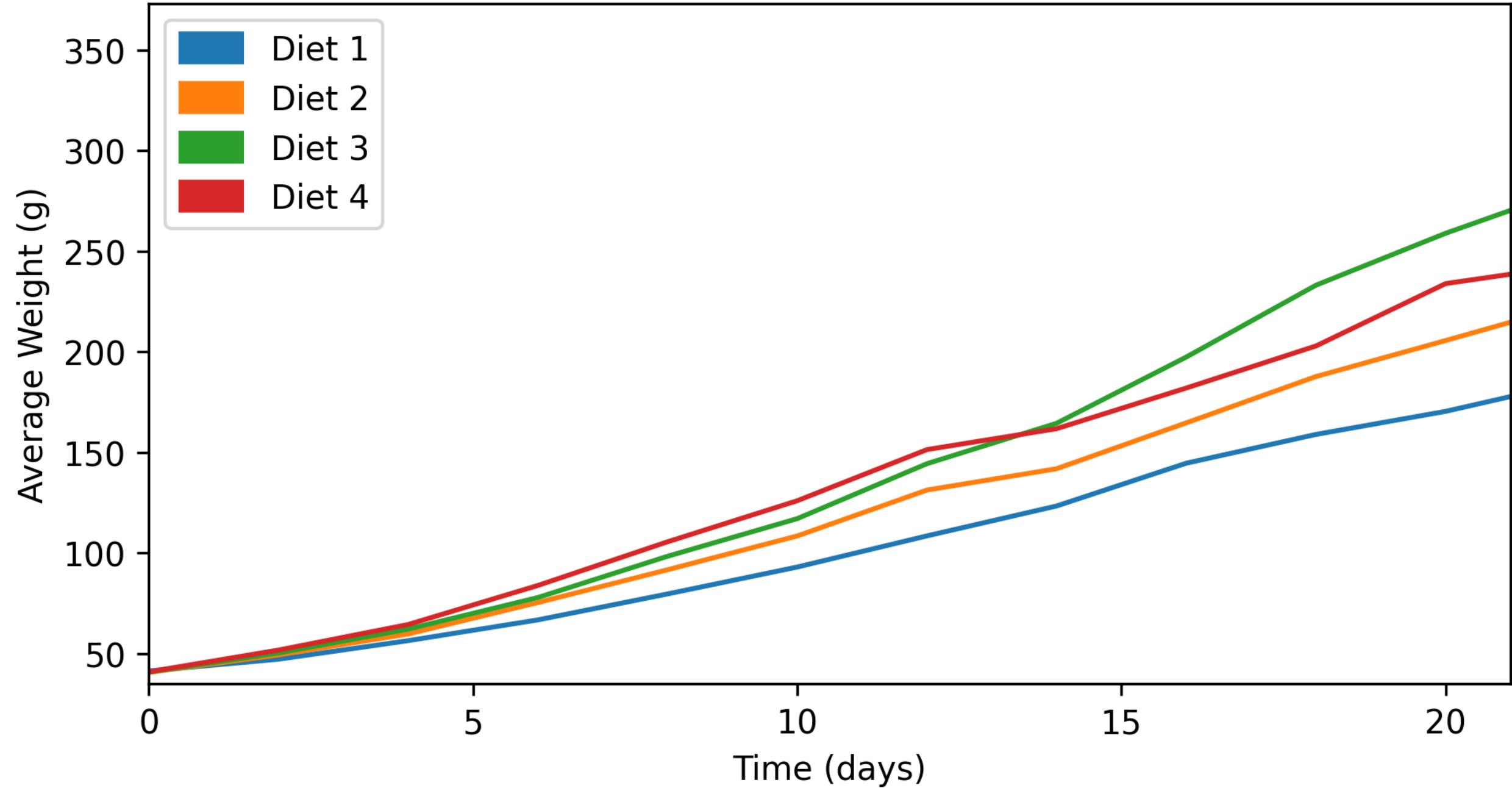
Chick weights



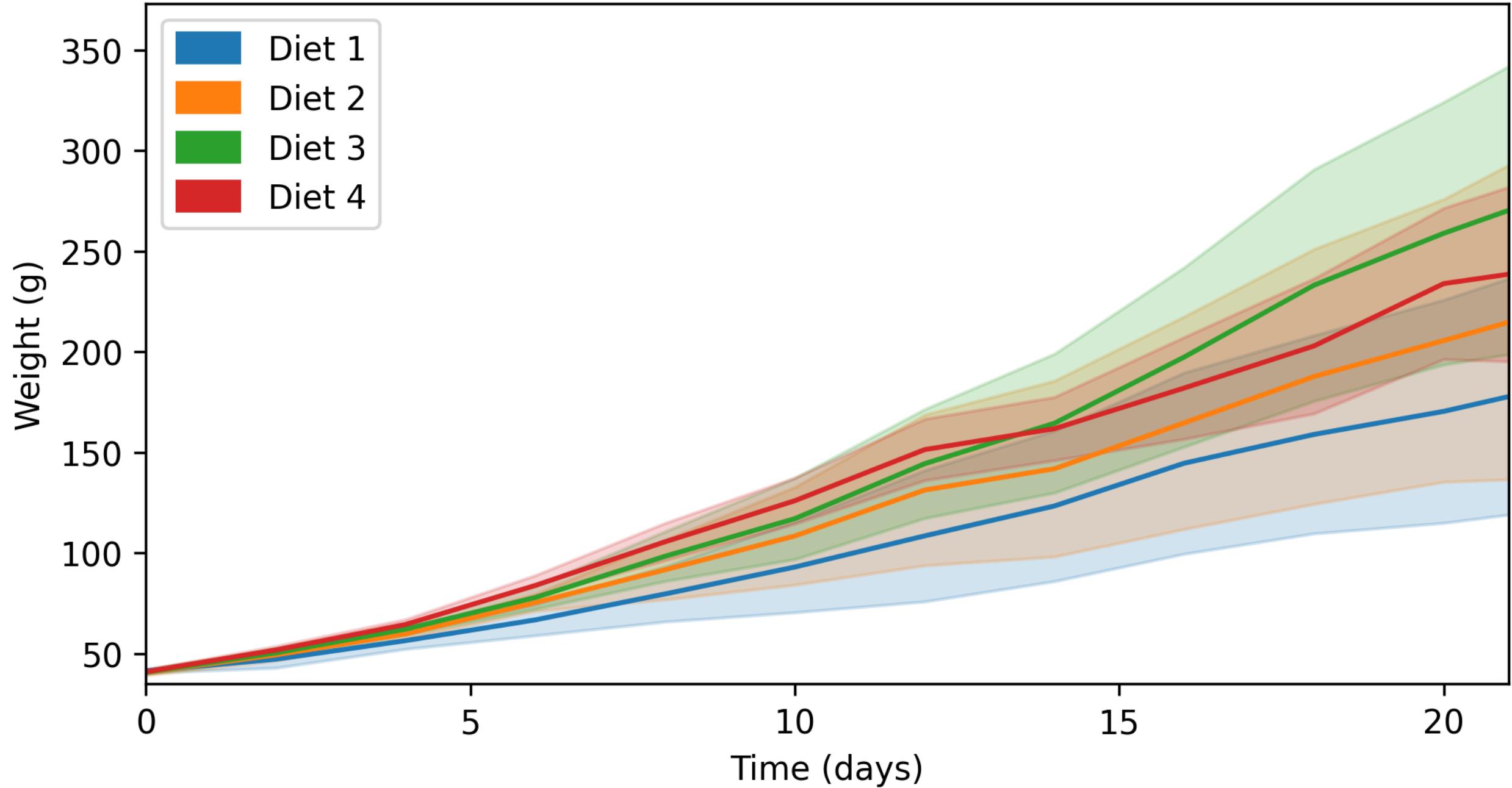
Chick weights



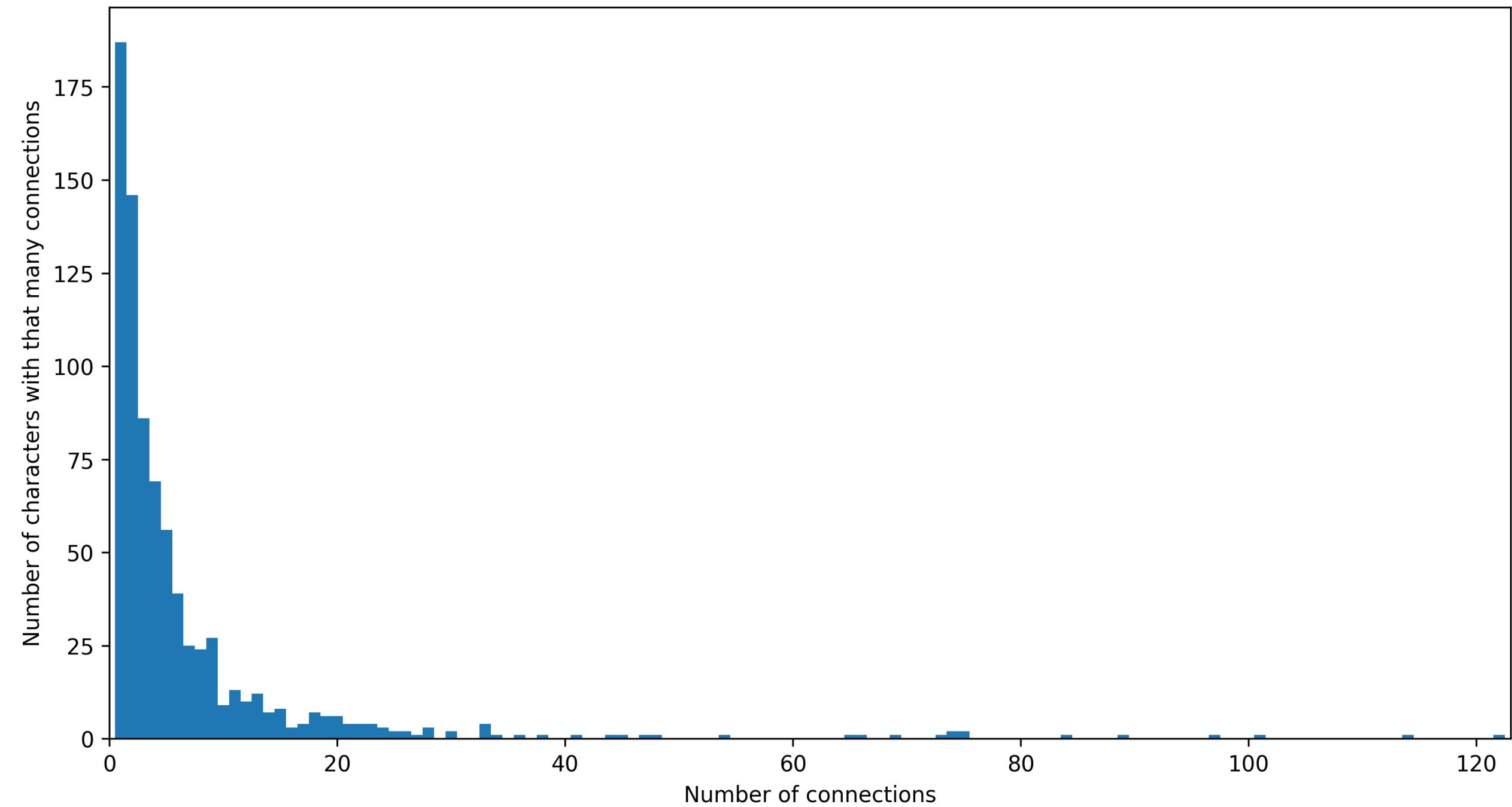
Chick weights



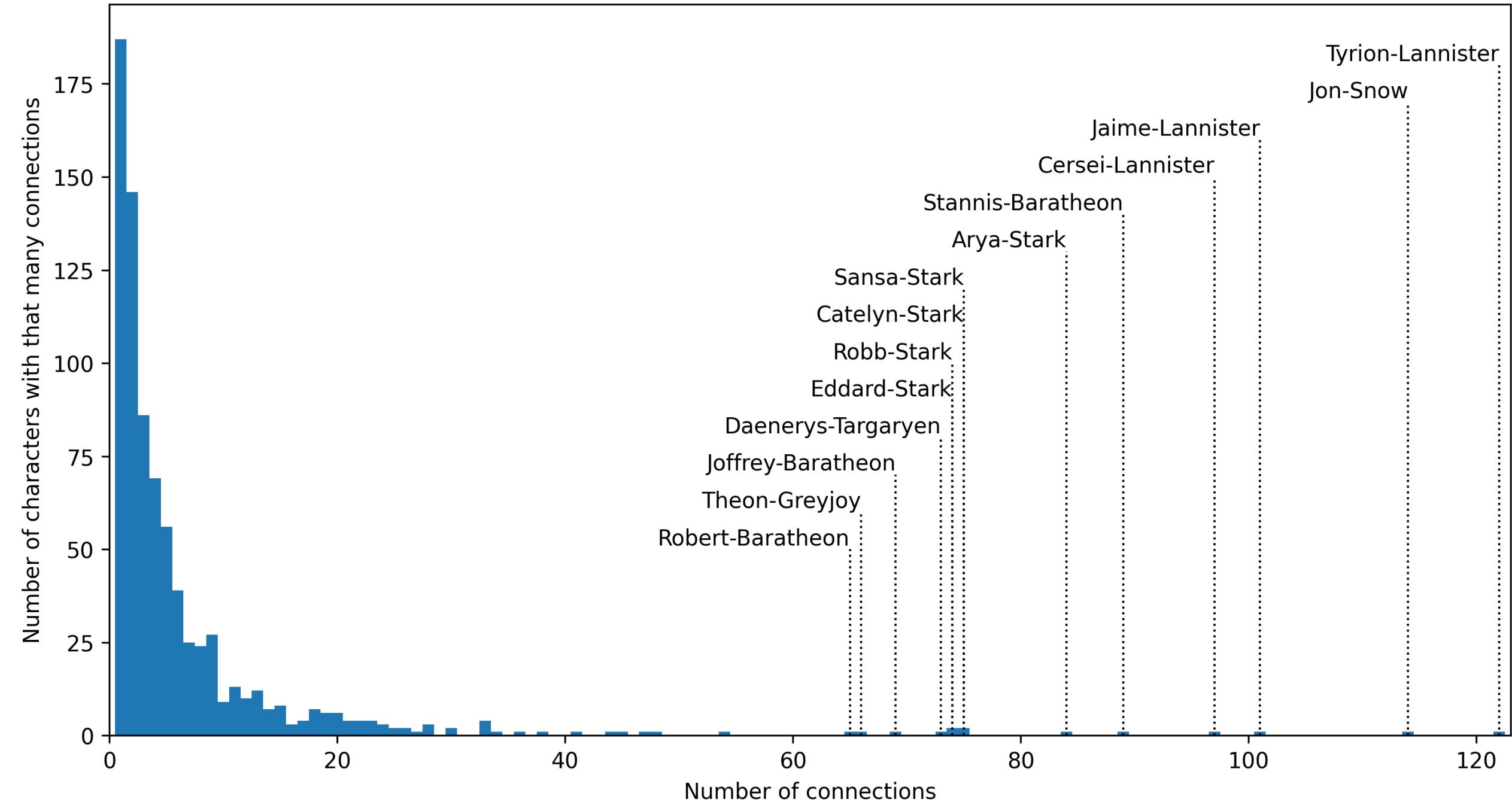
Chick weights



Game of Thrones characters



Game of Thrones characters



More plots

For more complex plots

- Check the [Matplotlib](#) or [Seaborn](#) galleries
(for inspiration, or to find the name of a plot)
- Ask an LLM (but check and fix its answers)

Section Navigation

Lines, bars and markers



Images, contours and fields



Subplots, axes and figures



Statistics



Pie and polar charts



Text, labels and annotations



Color



Shapes and collections



Style sheets



Module - pyplot



Module - axes_grid1



Module - axisartist



Showcase



Animation



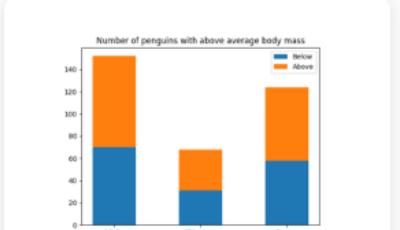
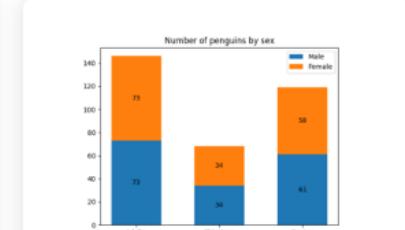
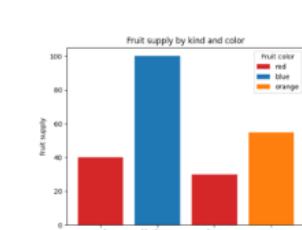
Examples

For an overview of the plotting methods we provide, see [Plot types](#)

This page contains example plots. Click on any image to see the full image and source code.

For longer tutorials, see our [tutorials page](#). You can also find [external resources](#) and a [FAQ](#) in our [user guide](#).

Lines, bars and markers



On this page

[Lines, bars and markers](#)

[Images, contours and fields](#)

[Subplots, axes and figures](#)

[Statistics](#)

[Pie and polar charts](#)

[Text, labels and annotations](#)

[Color](#)

[Shapes and collections](#)

[Style sheets](#)

[Module - pyplot](#)

[Module - axes_grid1](#)

[Module - axisartist](#)

[Showcase](#)

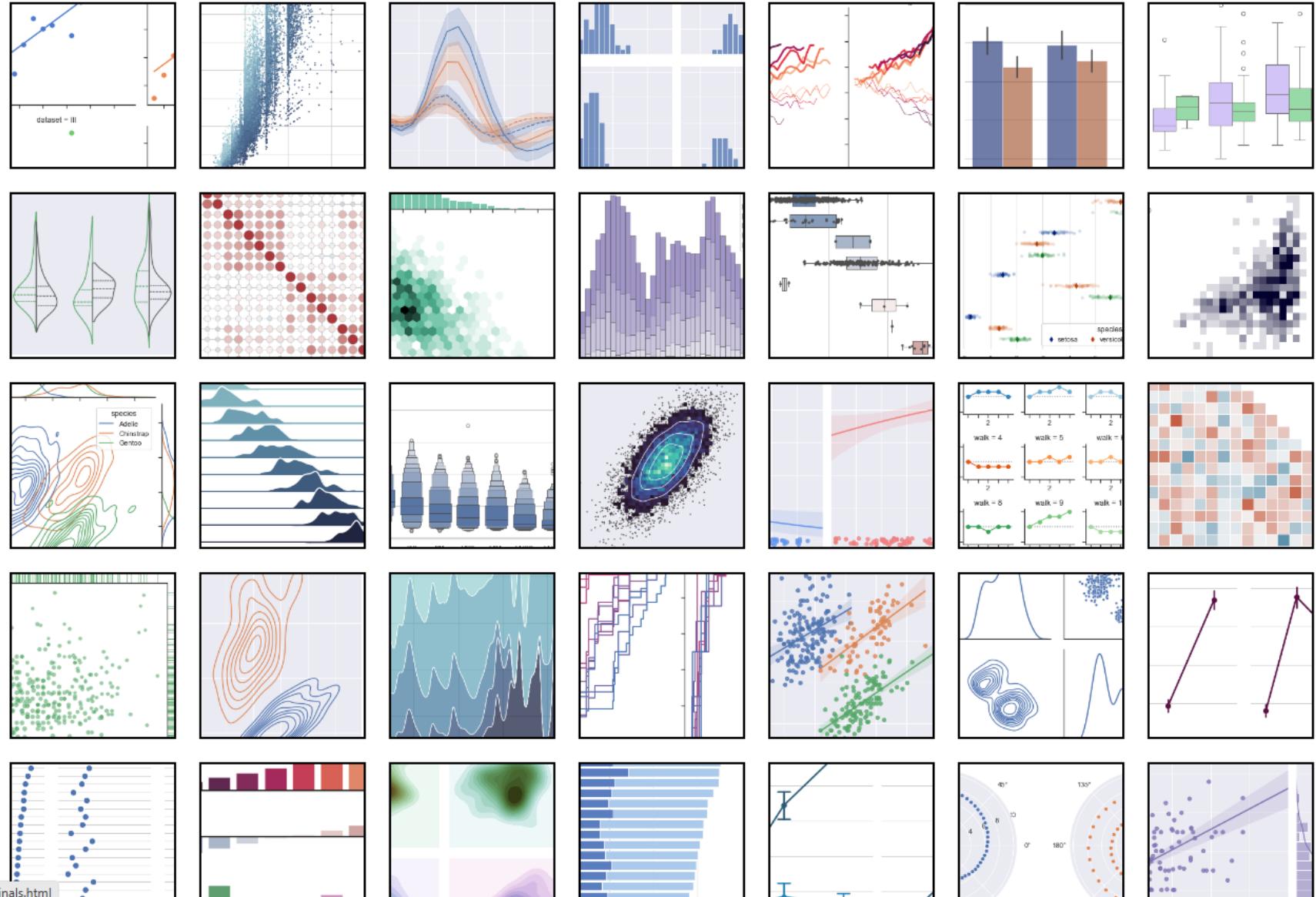
[Animation](#)

[Event handling](#)

[Miscellaneous](#)

[3D plotting](#)

[Scales](#)



I have a Pandas data-frame with the following columns: "species", "island", "bill_length_mm", "bill_depth_mm". Give me Matplotlib code to plot the data, with bill_length_mm on the horizontal axis, bill_depth_mm on the vertical axis, a different colour for each species, a different symbol for each island, a linear fit for each species, and a legend.

Certainly! Here's a Matplotlib code snippet that will help you plot your data as described:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

# Assuming df is your pandas DataFrame
# Replace 'df' with the name of your actual DataFrame variable

# Create a figure and a set of subplots
fig, ax = plt.subplots()

# Unique markers for each island
markers = {'Island1': 'o', 'Island2': 's', 'Island3': '^'}

# Unique colors for each species
colors = {'Species1': 'r', 'Species2': 'g', 'Species3': 'b'}
```

2024

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

# Assuming df is your pandas DataFrame
# Replace 'df' with the name of your actual DataFrame variable

# Create a figure and a set of subplots
fig, ax = plt.subplots()

# Unique markers for each island
markers = {'Island1': 'o', 'Island2': 's', 'Island3': '^'}

# Unique colors for each species
colors = {'Species1': 'r', 'Species2': 'g', 'Species3': 'b'}

# Plot each species
for species in df['species'].unique():
    for island in df['island'].unique():
        species_island_df = df[(df['species'] == species) & (df['island'] == island)]
        sns.regplot(x='bill_length_mm', y='bill_depth_mm', data=species_island_df,
                    scatter_kws={'color': colors[species], 'marker': markers[island]},
                    line_kws={'color': colors[species], 'label': f'Linear fit for {species}'},
                    ax=ax)

# Set the labels for the axes
ax.set_xlabel('Bill Length (mm)')
ax.set_ylabel('Bill Depth (mm)')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```

2024

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

# Assuming df is your pandas DataFrame
# Replace 'df' with the name of your actual DataFrame variable

# Create a figure and a set of subplots
fig, ax = plt.subplots()

# Unique markers for each island
markers = {'Island1': 'o', 'Island2': 's', 'Island3': '^'}

# Unique colors for each species
colors = {'Species1': 'r', 'Species2': 'g', 'Species3': 'b'}

# Plot each species
for species in df['species'].unique():
    for island in df['island'].unique():
        species_island_df = df[(df['species'] == species) & (df['island'] == island)]
        sns.regplot(x='bill_length_mm', y='bill_depth_mm', data=species_island_df,
                    scatter_kws={'color': colors[species], 'marker': markers[island]},
                    line_kws={'color': colors[species], 'label': f'Linear fit for {species}'},
                    ax=ax)

# Set the labels for the axes
ax.set_xlabel('Bill Length (mm)')
ax.set_ylabel('Bill Depth (mm)')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```

2024

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

# Assuming df is your pandas DataFrame
# Replace 'df' with the name of your actual DataFrame variable

# Create a figure and a set of subplots
fig, ax = plt.subplots()

# Unique markers for each island
markers = {'Biscoe': 'o', 'Dream': 's', 'Torgersen': '^'}

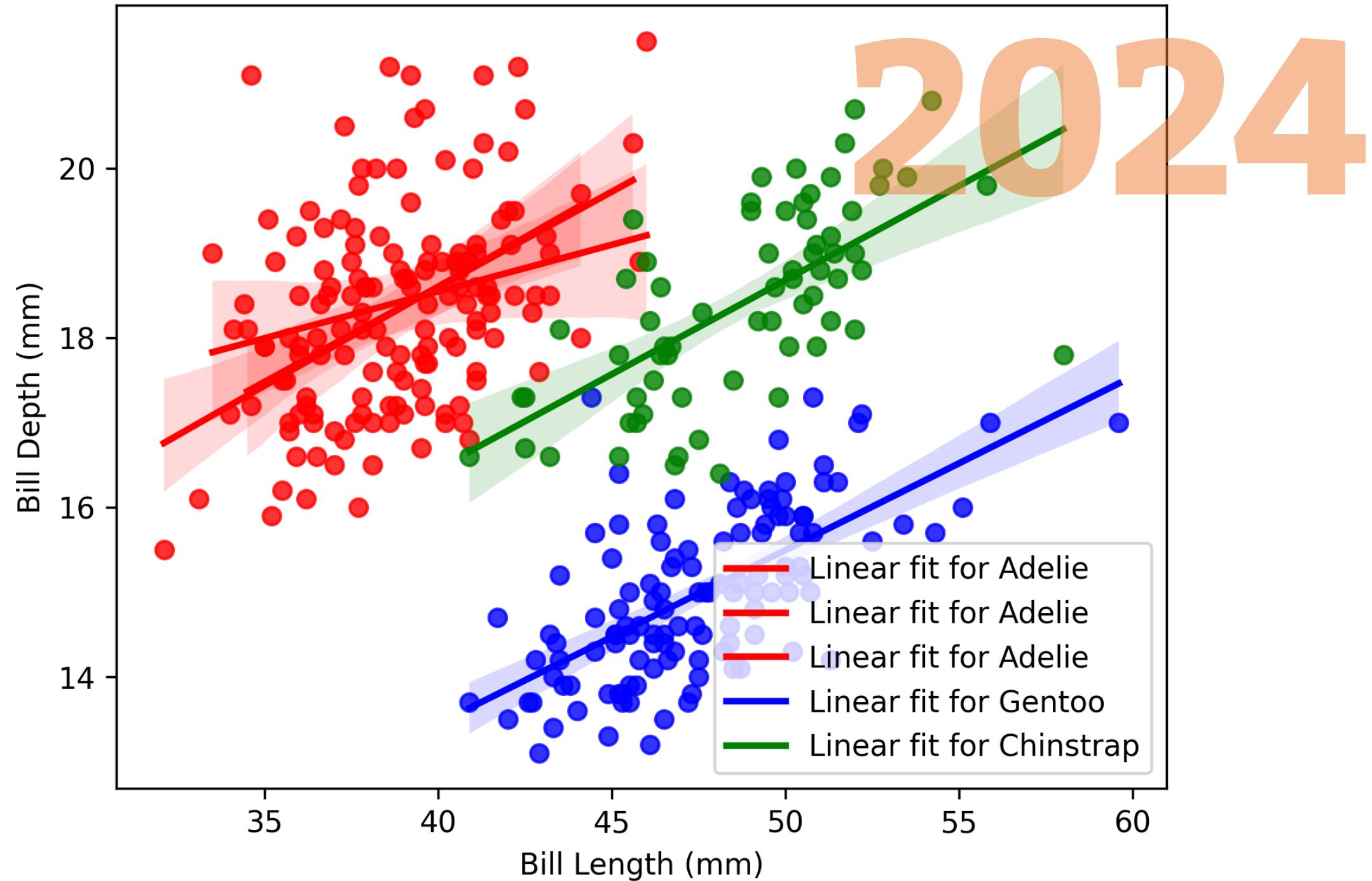
# Unique colors for each species
colors = {'Adelie': 'r', 'Chinstrap': 'g', 'Gentoo': 'b'}

# Plot each species
for species in df['species'].unique():
    for island in df['island'].unique():
        species_island_df = df[(df['species'] == species) & (df['island'] == island)]
        sns.regplot(x='bill_length_mm', y='bill_depth_mm', data=species_island_df,
                    scatter_kws={'color': colors[species], 'marker': markers[island]},
                    line_kws={'color': colors[species], 'label': f'Linear fit for {species}'},
                    ax=ax)

# Set the labels for the axes
ax.set_xlabel('Bill Length (mm)')
ax.set_ylabel('Bill Depth (mm)')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```



2025

I have a pandas data-frame, `d`, with the following columns: "species", "island", "bill_length_mm", "bill_depth_mm".

Give me matplotlib code to plot the data, with:

- bill_length_mm on the horizontal axis,
- bill_depth_mm on the vertical axis,
- a different colour for each species,
- a different symbol for each island,
- a linear fit for each species,
- and a legend.

claudie-sonnet-4.5

```
python           ⚠ Collapse ⚡ Run Save Copy
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from scipy import stats
4
5 # Get unique species and islands
6 species_list = d['species'].unique()
7 islands_list = d['island'].unique()
8
9 # Define colors and markers
10 colors = plt.cm.tab10(np.linspace(0, 1, len(s
11 markers = ['o', 's', '^', 'D', 'v', '<', '>',
12
13 # Create the plot
14 fig, ax = plt.subplots(figsize=(10, 6))
15
```

gpt-5.1

```
python           ⚠ Collapse ⚡ Run Save Copy
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Assume your DataFrame is called d and has c
5 # "species", "island", "bill_length_mm", "bil
6
7 fig, ax = plt.subplots(figsize=(8, 6))
8
9 # Distinct markers for islands and colors for
10 markers = ['o', 's', '^', 'D', 'v', 'P', 'X',
11 islands = d['island'].unique()
12 marker_map = {isl: markers[i % len(markers)]}
13
14 species_list = d['species'].unique()
15 colors = plt.cm.tab10.colors # or any colorw
```

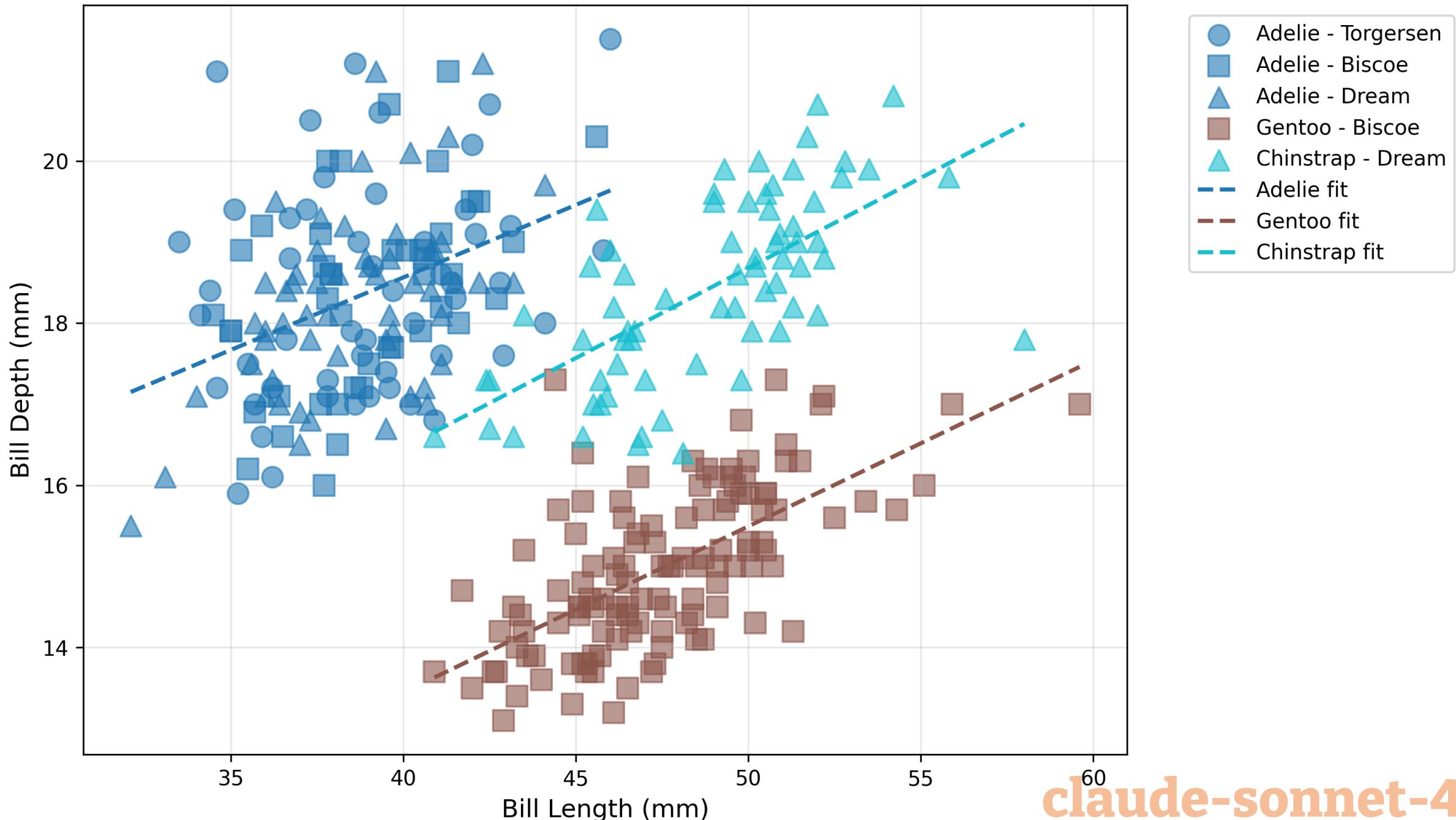
gemini-3-pro

Here is the code to generate the plot using `matplotlib`.

Since standard matplotlib requires a bit of manual setup to handle categorical mapping for colors and markers simultaneously, the code below iterates through the unique combinations to ensure the legend and fit lines correspond correctly.

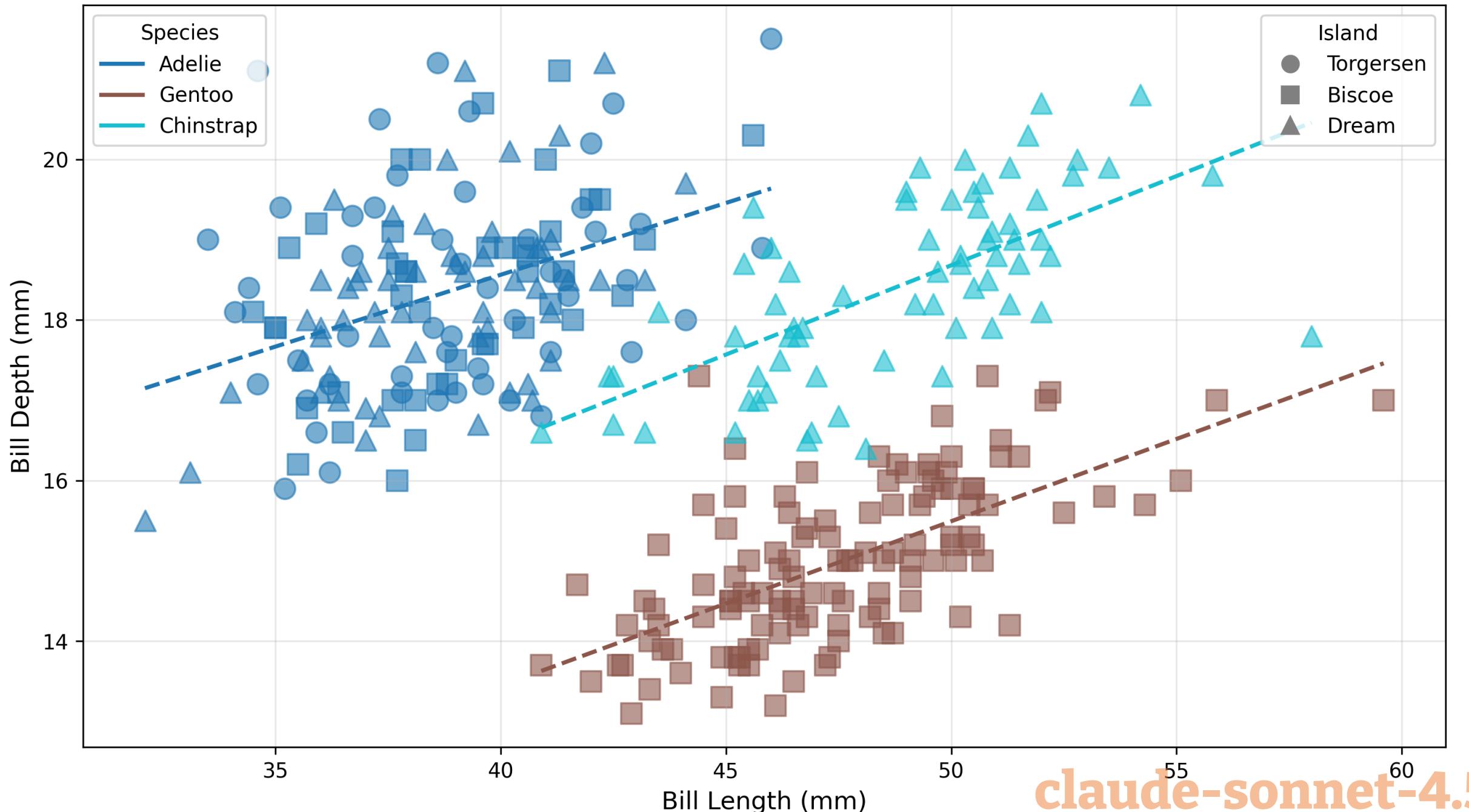
```
python           ⚠ Collapse ⚡ Run Save Copy
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import pandas as pd
4
5 # Assuming your dataframe is named 'd'
6 # Example data loading (for reproducibility,
7 # import seaborn as sns
```

Bill Dimensions by Species and Island

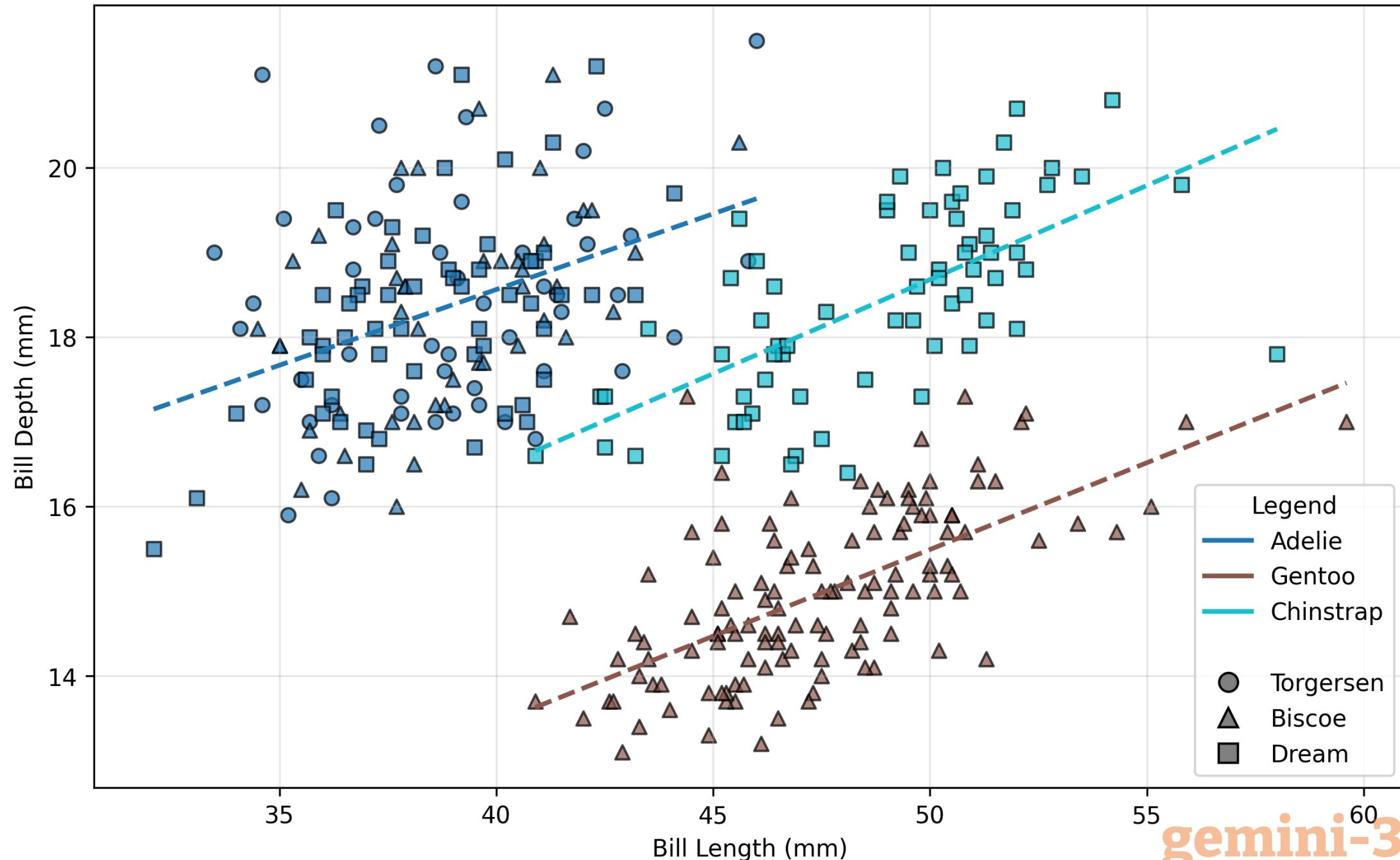


claude-sonnet-4.5

Bill Dimensions by Species and Island

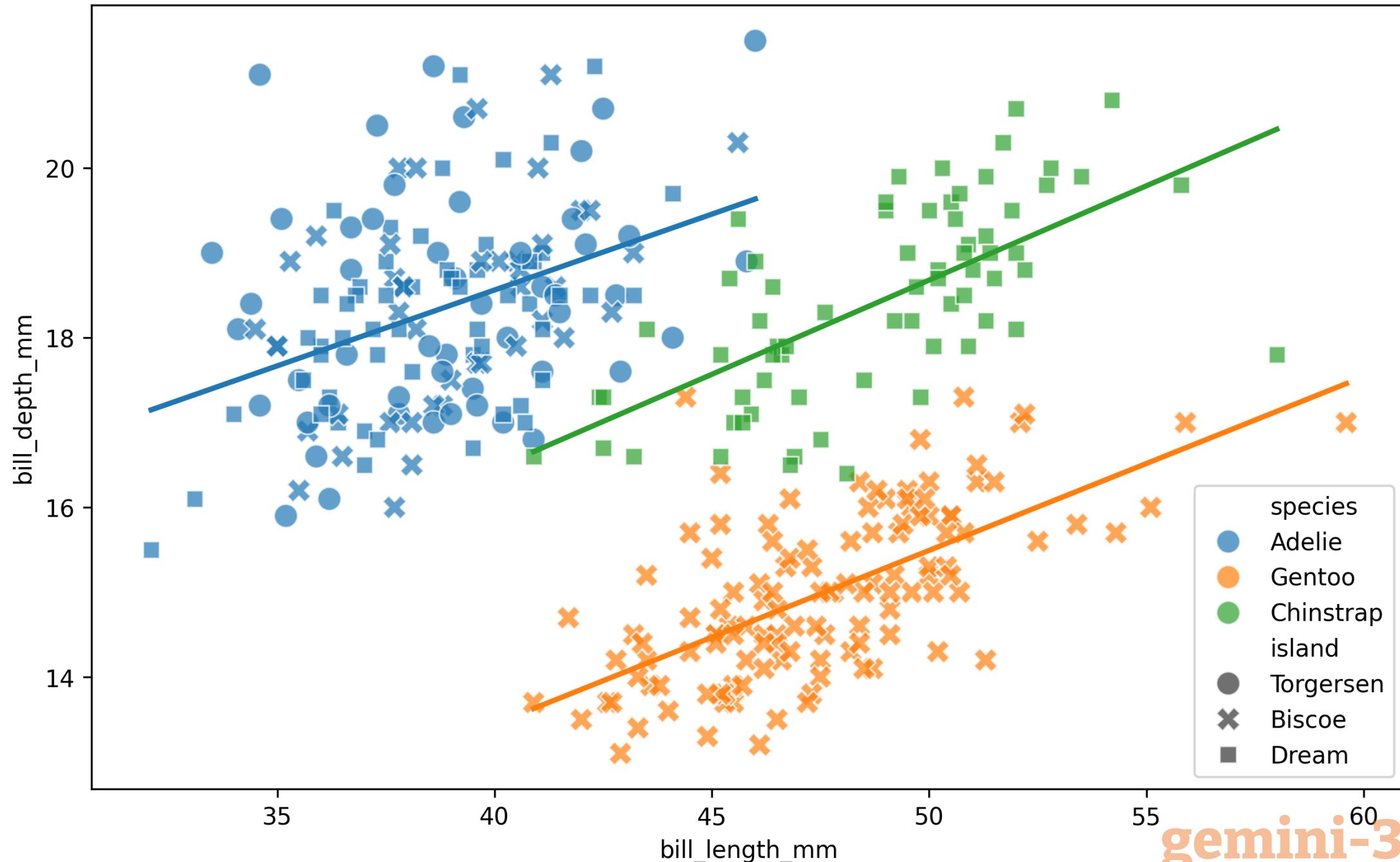


Bill Dimensions by Species and Island



gemini-3-pro

Bill Dimensions by Species and Island



gemini-3-pro

```
LinAlgError                                     Traceback (most recent call last)
Cell In[12], line 45
  43 if len(x) < 2:
  44     continue # not enough points to fit
--> 45 coeffs = np.polyfit(x, y, 1)
  46 y_fit = np.polyval(coeffs, x_fit)
  47 ax.plot(
  48     x_fit, y_fit,
  49     color=color_map[sp],
(...):    52     label=f"{sp} fit"
  53 )

File ~/github/adialab-winter-school-2025/.venv/lib/python3.11/site-packages/numpy/lib/_polynomial_impl.py:675, in polyfit(x, y, deg, rcond, full, w, cov)
  673 scale = NX.sqrt((lhs * lhs).sum(axis=0))
  674 lhs /= scale
--> 675 c, resids, rank, s = lstsq(lhs, rhs, rcond)
  676 c = (c.T / scale).T # broadcast scale coefficients
  678 # warn on rank reduction, which indicates an ill conditioned matrix

File ~/github/adialab-winter-school-2025/.venv/lib/python3.11/site-packages/numpy/linalg/_linalg.py:2564, in lstsq(a, b, rcond)
 2560     b = zeros(b.shape[:-2] + (m, n_rhs + 1), dtype=b.dtype)
 2562 with errstate(call=_raise_linalgerror_lstsq, invalid='call',
 2563                 over='ignore', divide='ignore', under='ignore'):
-> 2564     x, resids, rank, s = _umath_linalg.lstsq(a, b, rcond,
...
File ~/github/adialab-winter-school-2025/.venv/lib/python3.11/site-packages/numpy/linalg/_linalg.py:175, in _raise_linalgerror_lstsq(err, flag)
 174 def _raise_linalgerror_lstsq(err, flag):
--> 175     raise LinAlgError("SVD did not converge in Linear Least Squares")

LinAlgError: SVD did not converge in Linear Least Squares
```

tmp.ipynb u X

notebooks > tmp.ipynb > + empty cell

+ Code + Markdown | ▶ Run All ⚡ Restart ⌂ Clear All Outputs ...

.venv (Python 3.11.10)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from palmerpenguins import load_penguins
d = load_penguins()
```

[2] ✓ 0.0s Python

▶ [] Python

>Create matplotlib plot for data-frame + ⏎ ...

tmp.ipynb In cell 1, we have a pandas data-frame, `d`, with the following columns: "species", "island", "bill_length_mm", "bill_depth_mm"

Thinking ^
will be controlled by custom handles.
Next, I'll build the species legend handles using Line2D. I'll create a list of handles for species with their designated colors. For islands, I'll do something similar but use a neutral color for their markers, allowing the main plot to reflect the species' colors. This way, the legends will remain clear and helpful

cursor

tmp.ipynb U X

notebooks > tmp.ipynb > from matplotlib.lines import Line2D

+ Code + Markdown | Run All Restart Clear All Outputsvenv (Python 3.11.10)

```
# Linear fit (1st degree) for this species
x = df_sp["bill_length_mm"].to_numpy()
y = df_sp["bill_depth_mm"].to_numpy()
m, b = np.polyfit(x, y, 1)
xline = np.linspace(x.min(), x.max(), 100)
yline = m * xline + b
ax.plot(xline, yline, color=species_to_color[sp], linewidth=2)

# Legends: one for species (colors) and one for island (markers)
species_handles = [
    Line2D([0], [0], color=species_to_color[sp], lw=2, label=sp) for sp in species_list
]
island_handles = [
    Line2D(
        [0], [0], linestyle="None", marker=island_to_marker[isl],
        markerfacecolor="white", markeredgecolor="black", markersize=8, label=isl
    )
    for isl in island_list
]

leg1 = ax.legend(handles=species_handles, title="Species", loc="upper left")
leg2 = ax.legend(handles=island_handles, title="Island", loc="lower right")
ax.add_artist(leg1)

ax.set_xlabel("Bill length (mm)")
ax.set_ylabel("Bill depth (mm)")
ax.set_title("Bill dimensions by species and island")
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

1 / 1 Undo Keep Undo Ctrl+N Keep Ctrl+Shift+Y Python

Create matplotlib plot for data-frame + ⏺ ...

In cell 1, we have a pandas data-frame, `d`, with the following columns: "species", "island", "bill_length_mm", "bill_depth_mm"

```
from matplotlib.lines import Line2D

# Drop rows with missing values in rel
_d = d.dropna(subset=["species", "isl"])

species_list = df["species"].unique()
```

- I added the plotting code to cell 2 of notebooks/tmp.ipynb that:
 - uses bill_length_mm (x) and bill_depth_mm (y),
 - colors points by species,
 - uses different markers by island,
 - draws a linear fit per species,
 - and includes separate legends for species (colors) and island (markers).

Run cell 2 to render the plot.

1 File Review

tmp.ipynb +67 -1

Reject, suggest, follow up?

∞ gpt-5 @ 🔍 🗂️ 🎤

SSH: vincent.qromatiq.dev main* adialab-winter-school-2025 0 0 0 1.00ms Cursor Tab Spaces: 4 LF {} Cell 2 of 2

cursor

notebooks > tmp.ipynb > from matplotlib.lines import Line2D

+ Code + Markdown | Run All Restart Clear All Outputs ...

.venv (Python 3.11.10)

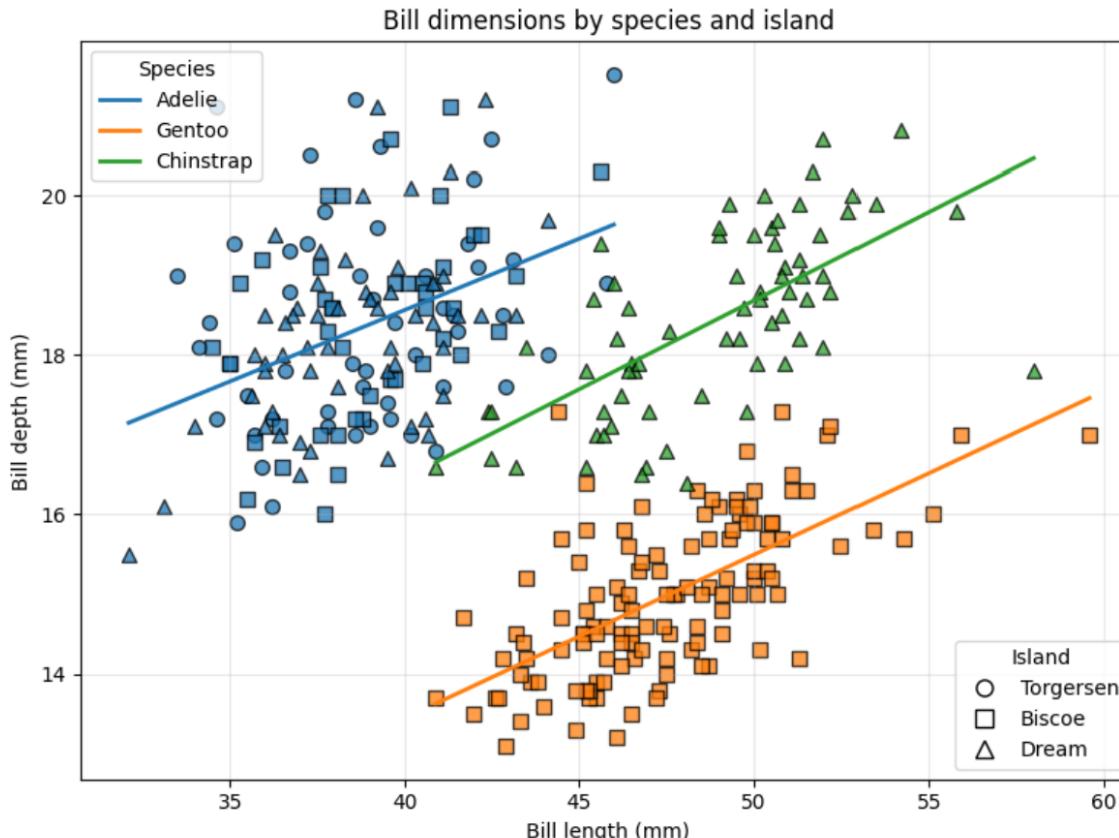
```
ax.set_xlabel("Bill length (mm)")
ax.set_ylabel("Bill depth (mm)")
ax.set_title("Bill dimensions by species and island")
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

[3]

✓ 0.1s

Python

...



In cell 1, we have a pandas data-frame, `d`, with the following columns: "species", "island", "bill_length_mm", "bill_depth_mm".

I'll add the matplotlib plotting code to cell 2 of notebooks/tmp.ipynb to satisfy the plotting requirements and legends.

tmp.ipynb +67 -1

```
from matplotlib.lines import Line2D

# Drop rows with missing values in rel
_d = d.dropna(subset=["species", "isl
species_list = _df["species"].unique()
```

- I added the plotting code to cell 2 of notebooks/tmp.ipynb that:

- uses `bill_length_mm` (x) and `bill_depth_mm` (y),
- colors points by `species`,
- uses different markers by `island`,
- draws a linear fit per `species`,
- and includes separate legends for species (colors) and island (markers).

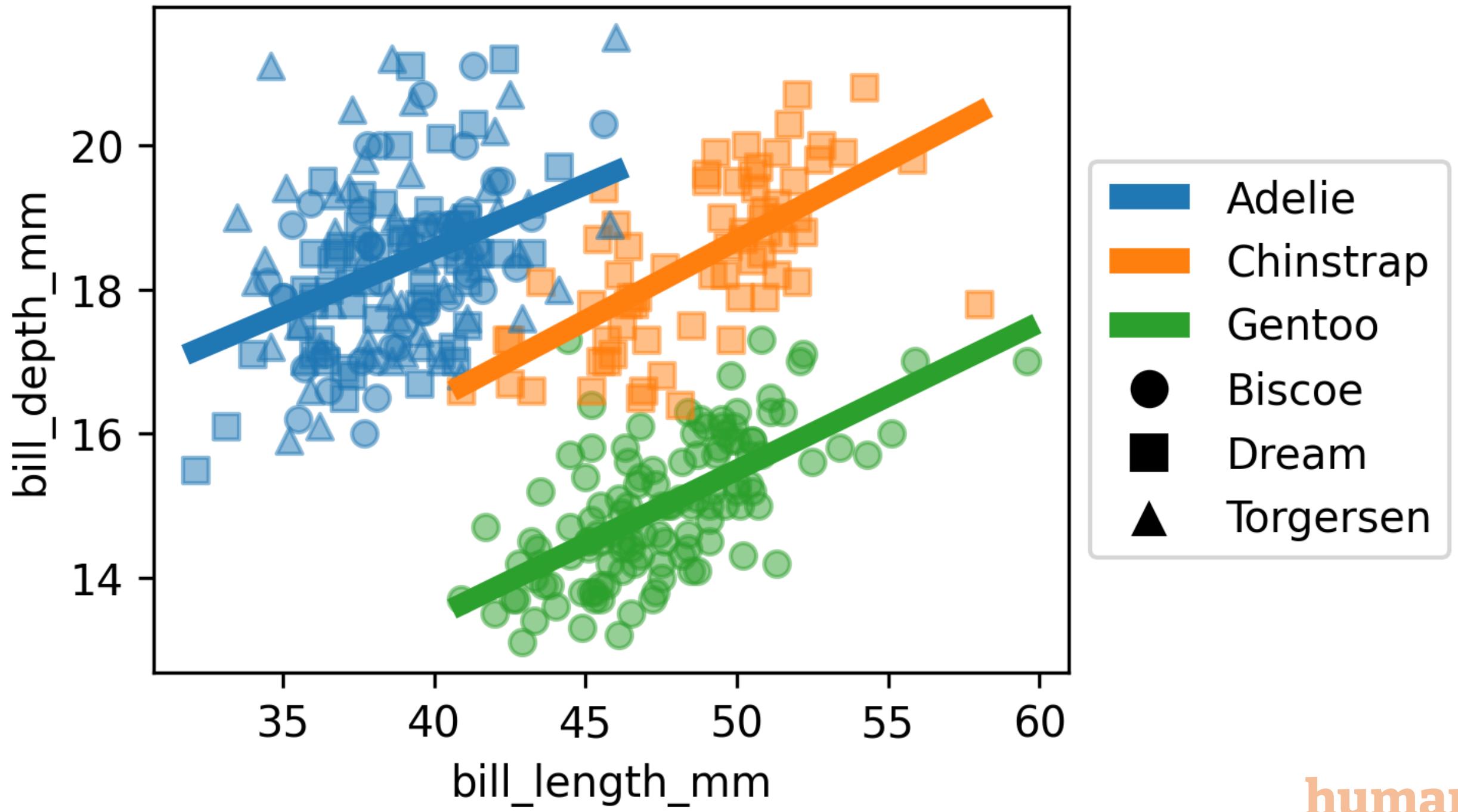
Run cell 2 to render the plot.

Plan, @ for context, / for commands

∞ v gpt-5 v

@ 🌐 📸 🎧

cursor

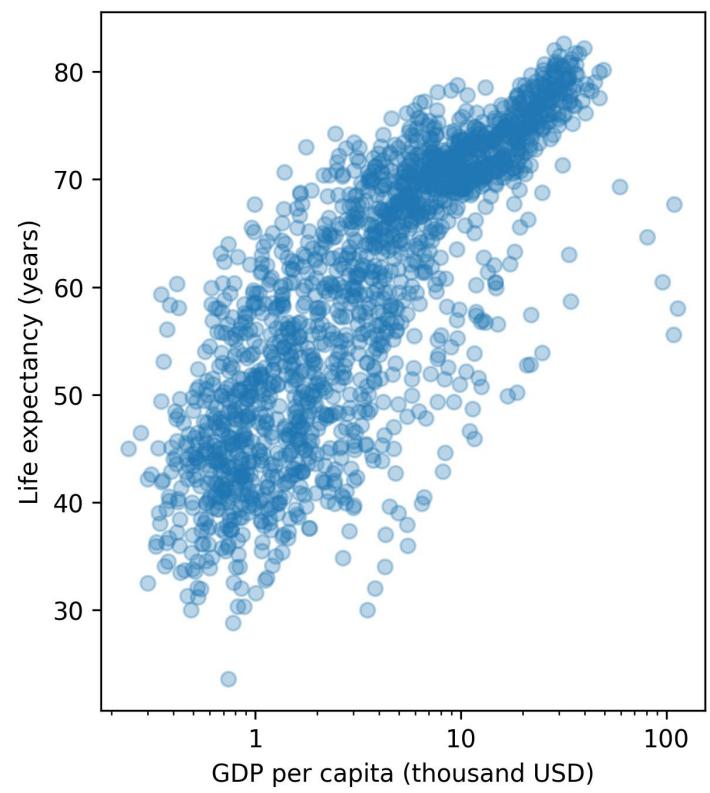
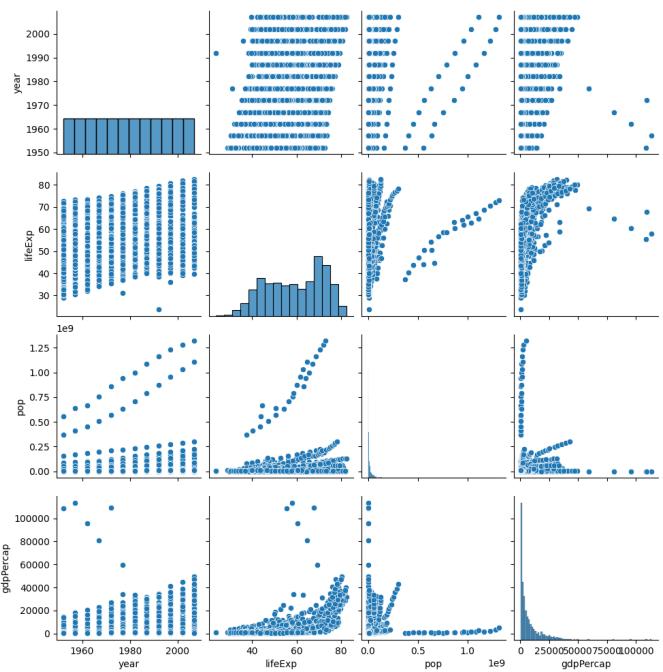
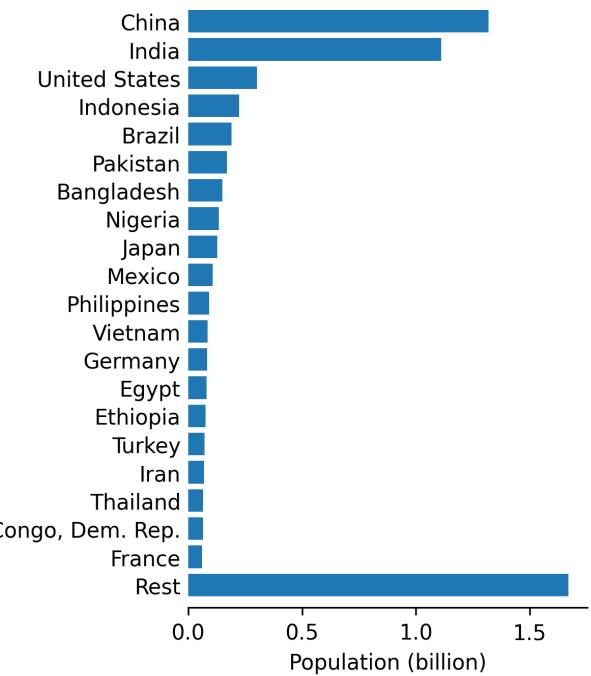
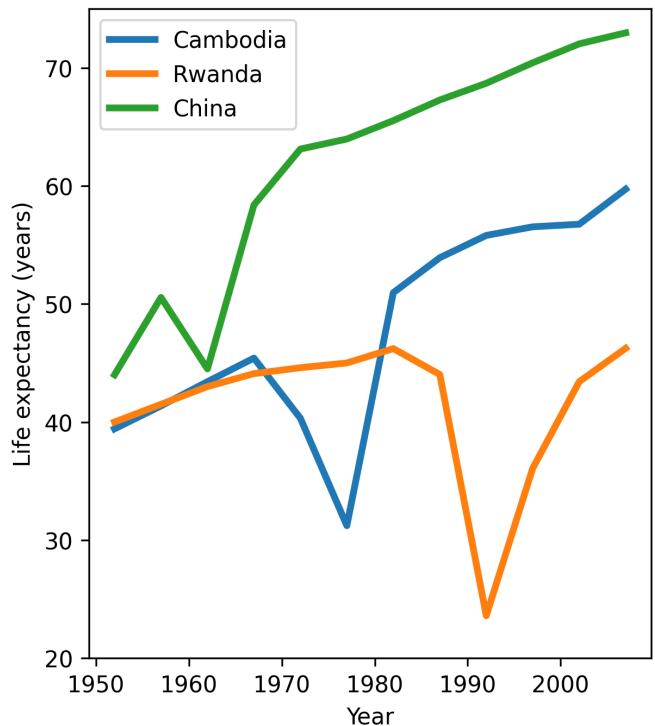
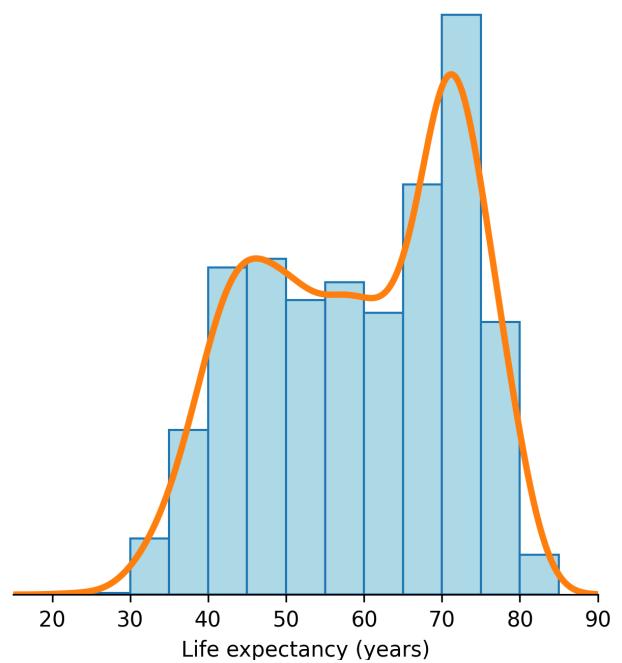


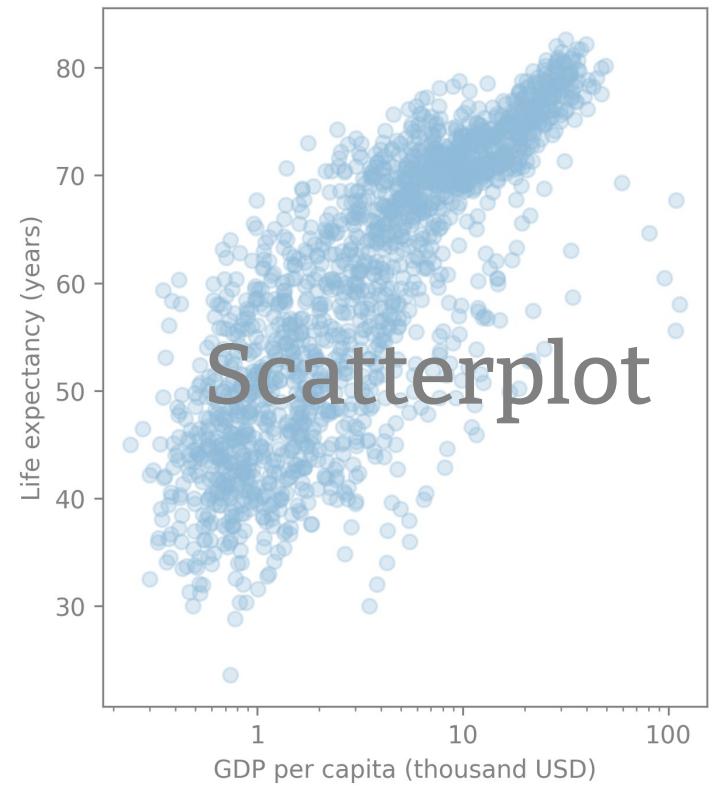
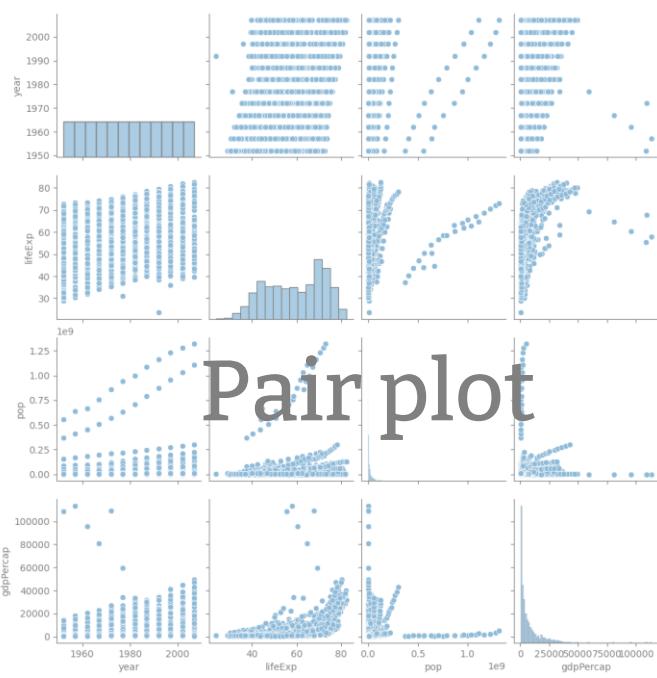
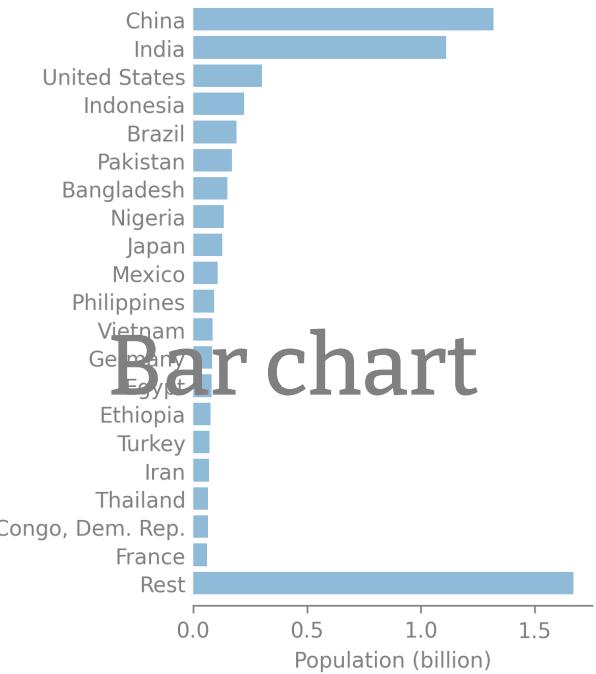
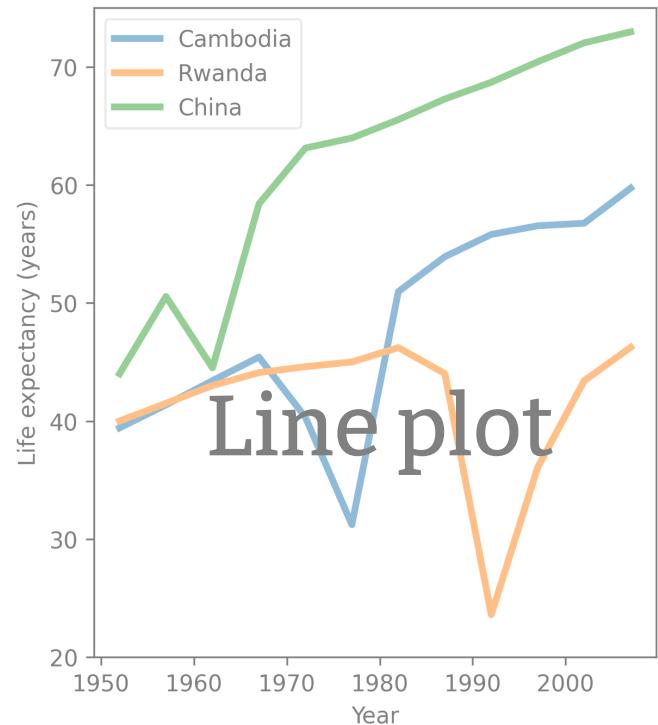
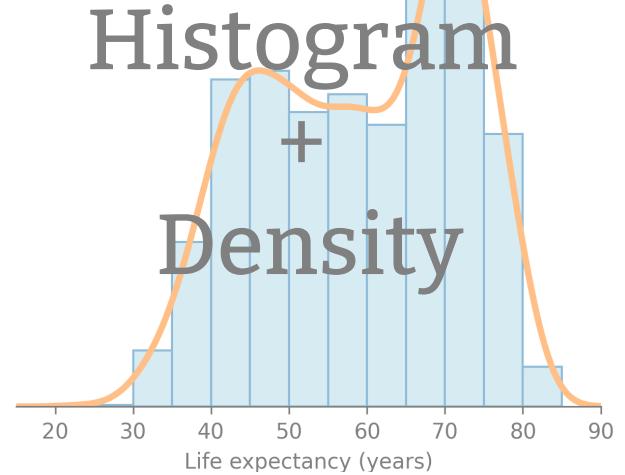
human

Conclusion

Summary

Number of variables	Types	Plots
1	Numeric	Histogram, density
1	Qualitative	Bar
2	Numeric + Numeric	Scatter
2	Time + Numeric	Line
2	Qualitative + Numeric	Bar, boxplots
2	Qualitative + Qualitative	Mosaic
3+	Numeric	Pair, Correlation





References

References

Learn X in Y minutes, where X=Python

<https://learnxinyminutes.com/docs/python/>

Statistical plots

V. Zoonekynd (2020)

<https://www.youtube.com/watch?v=YrSjtxGGhX0>

Modern Data Visualization with R

R. Kabacoff

<https://rkabacoff.github.io/datavis/Advice.html>

An introduction to statistical learning

G. James et al. (2013, 2023)

<https://www.statlearning.com/>

Scientific visualization, Python & Matplotlib

N.P. Rougier (2021)

<https://github.com/rougier/scientific-visualization-book>

Matplotlib, Pandas, NumPy cheat sheets

<https://matplotlib.org/cheatsheets/>

https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

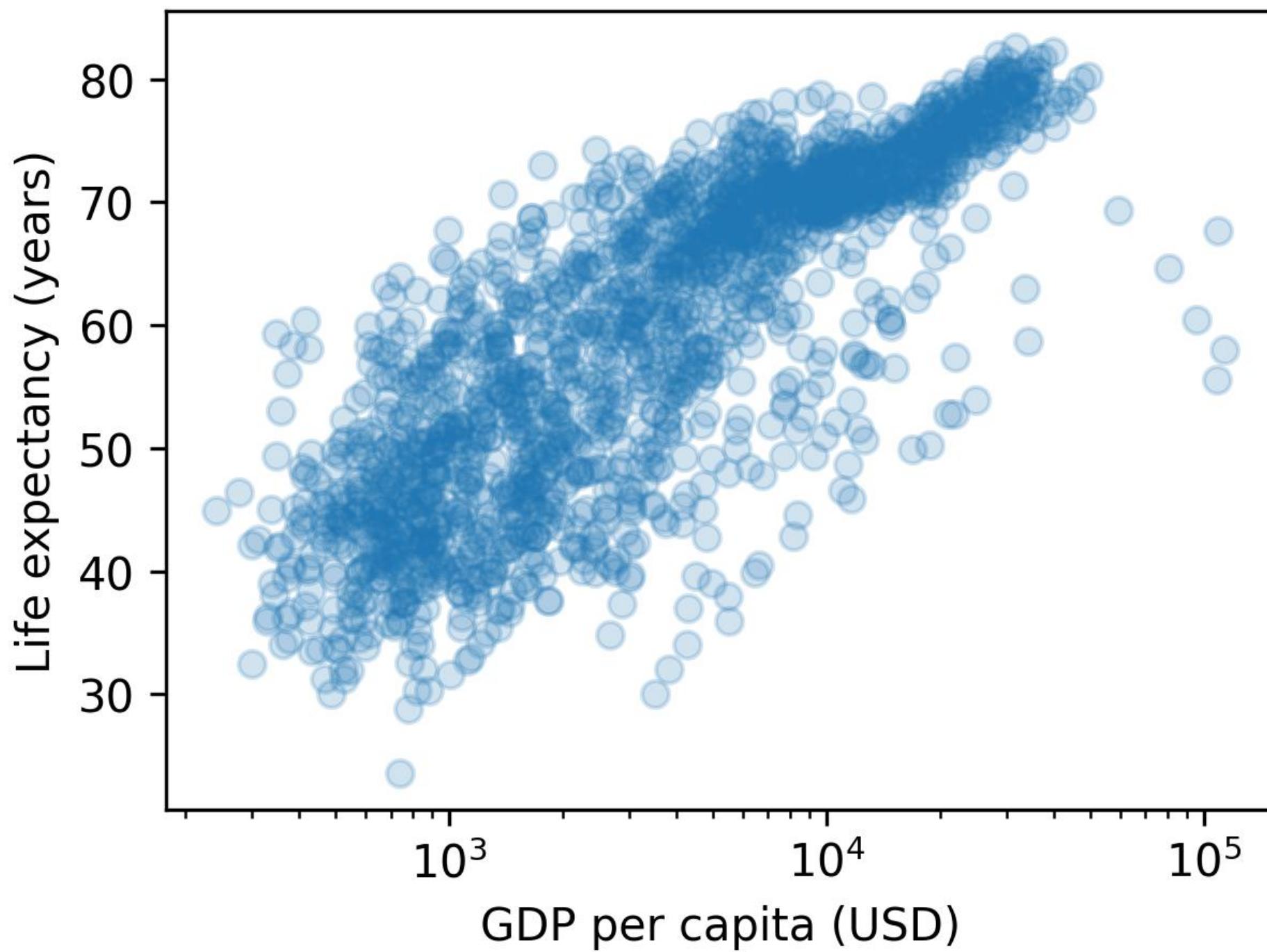
<https://www.datacamp.com/cheat-sheet/pandas-cheat-sheet-for-data-science-in-python>

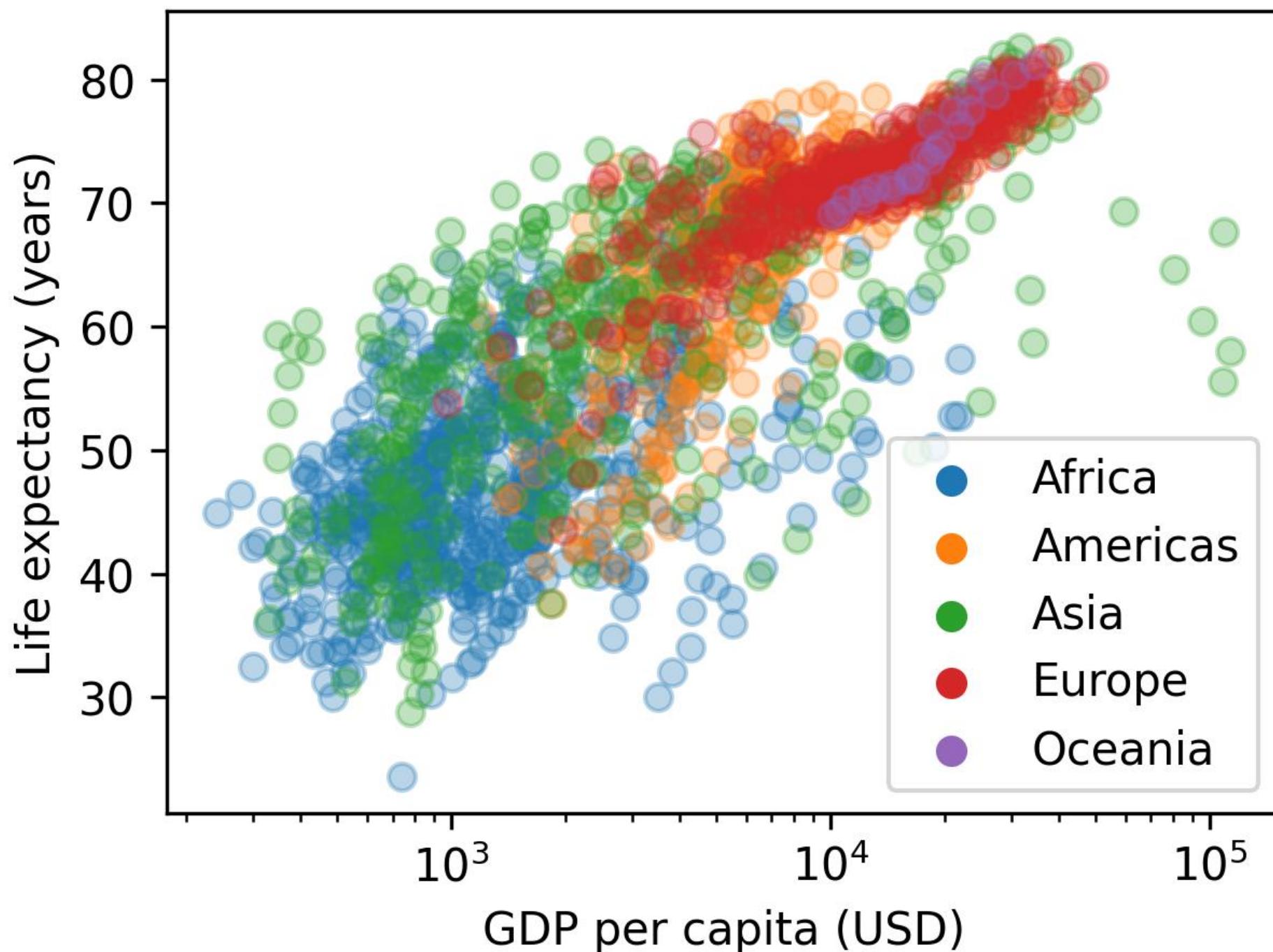
<https://www.datacamp.com/cheat-sheet/numpy-cheat-sheet-data-analysis-in-python>

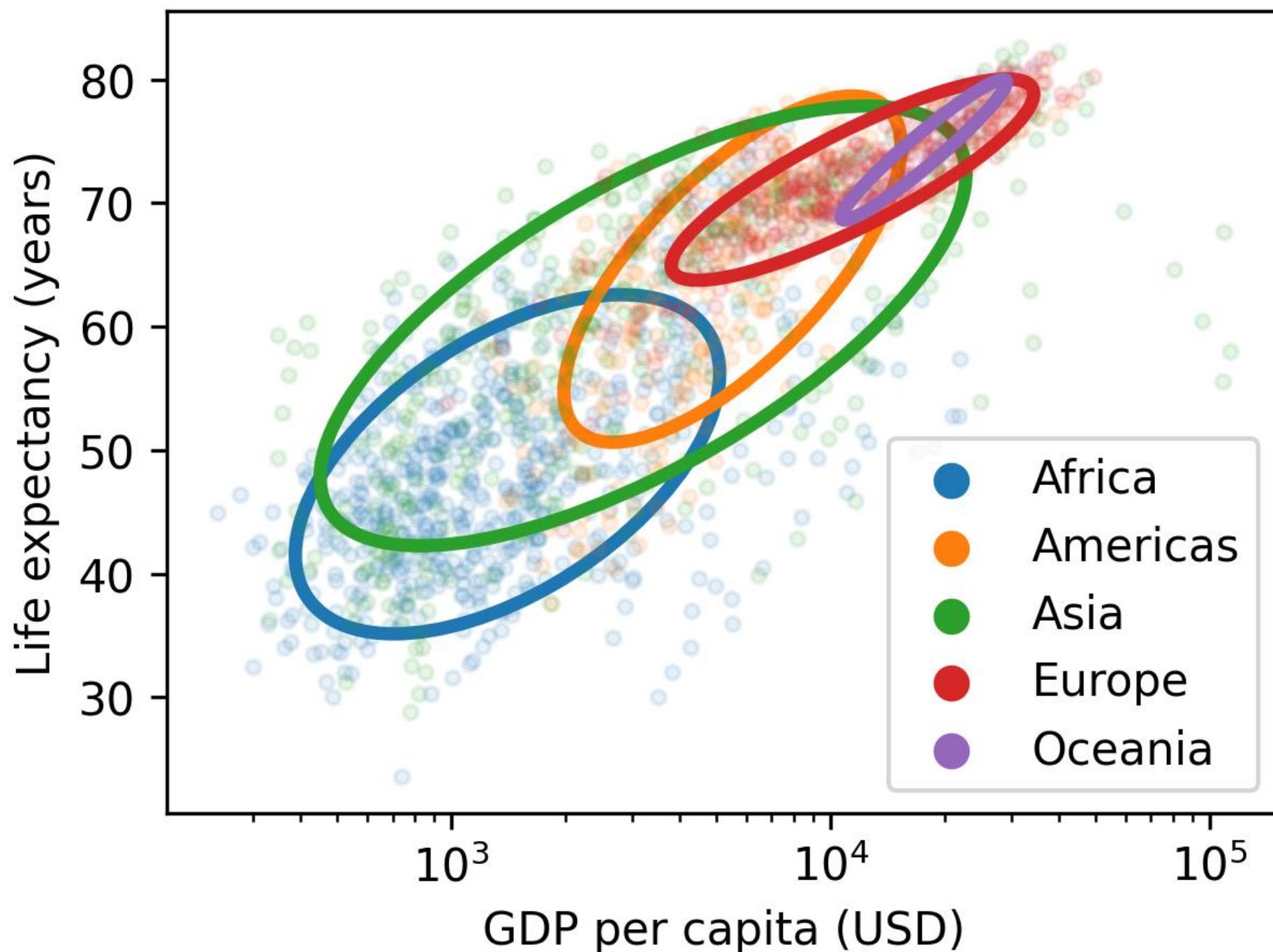
**Extra
Exercise**

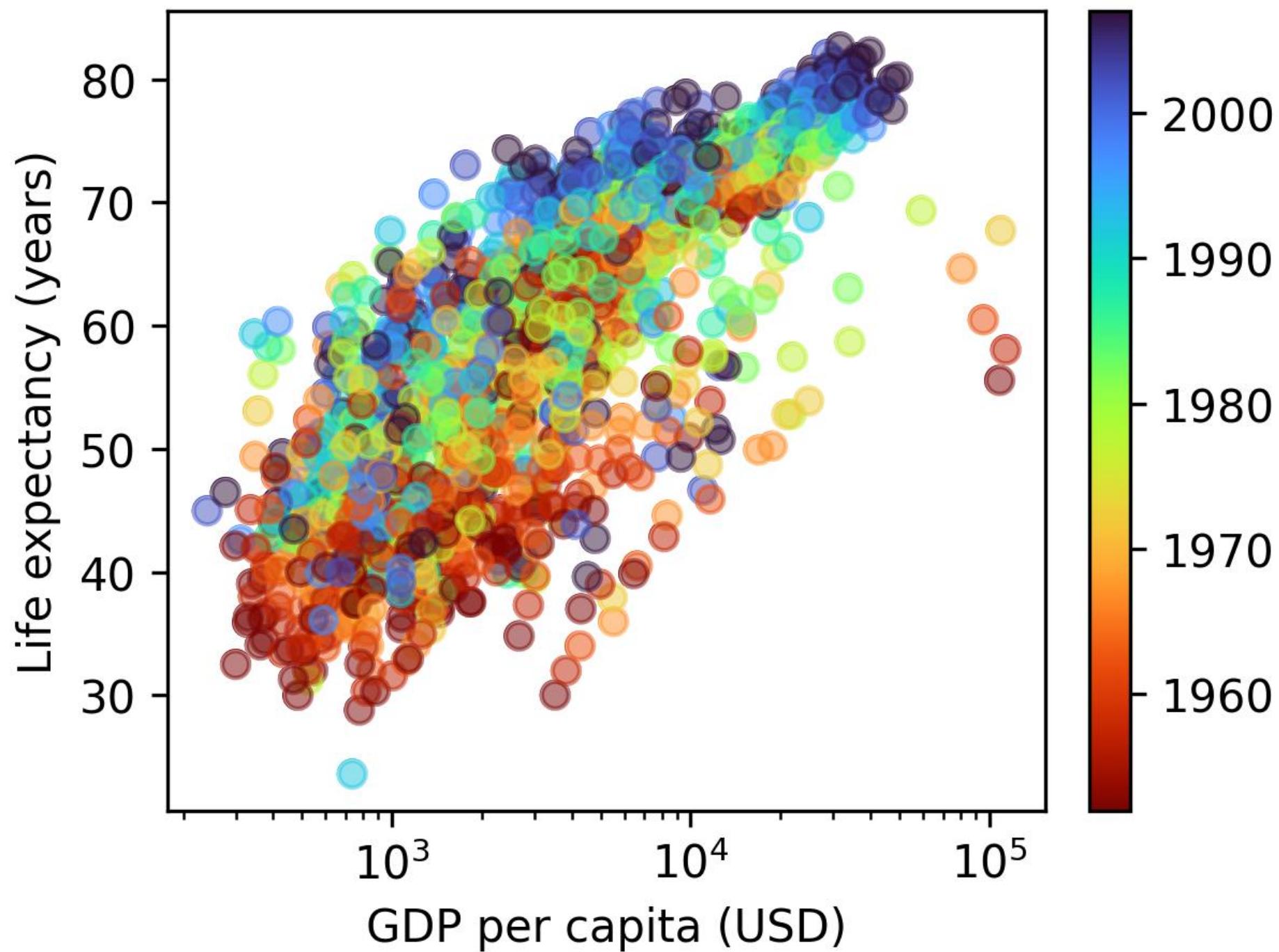
Exercise

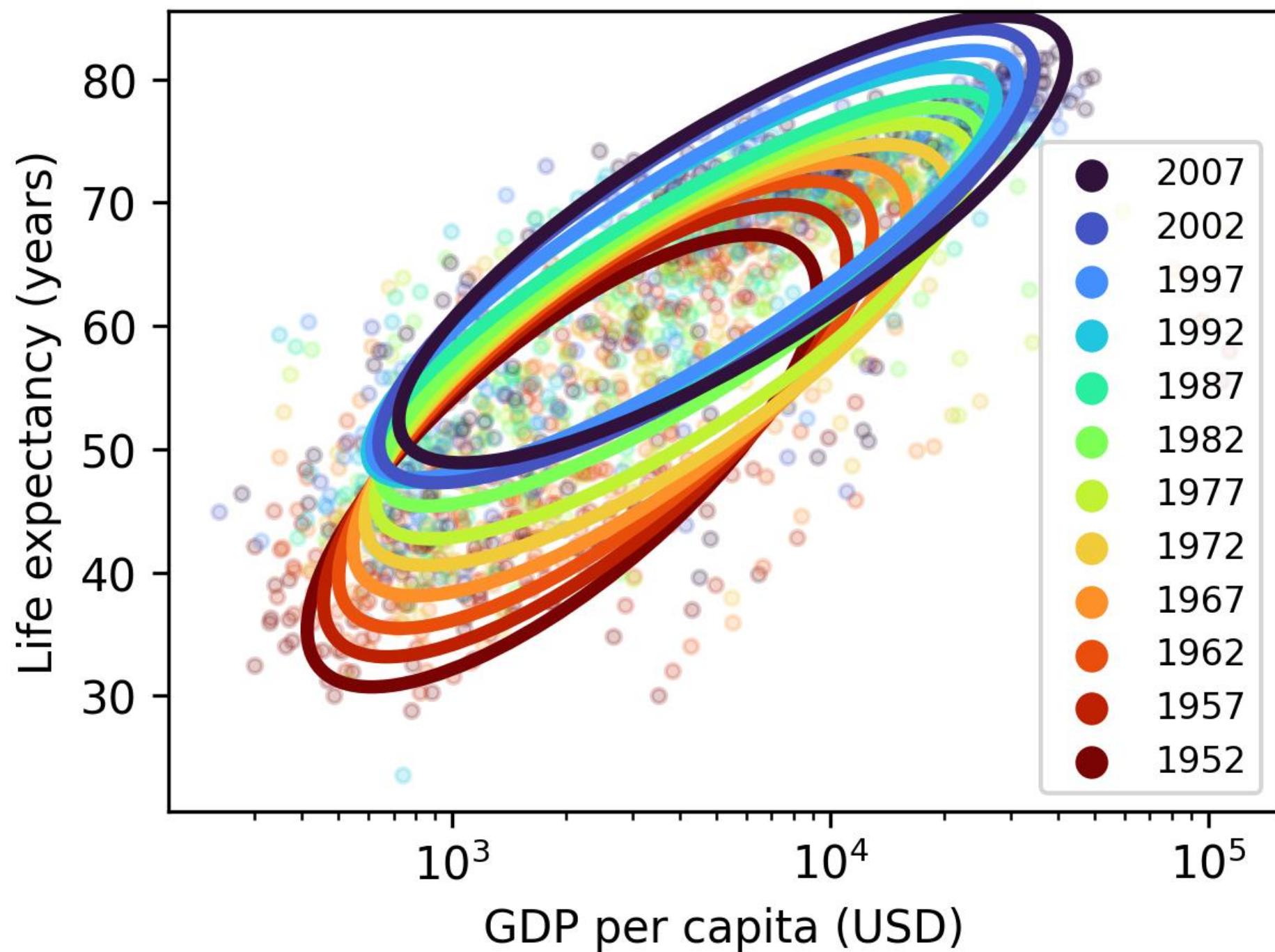
- Find other interesting visualizations of the Gapminder dataset



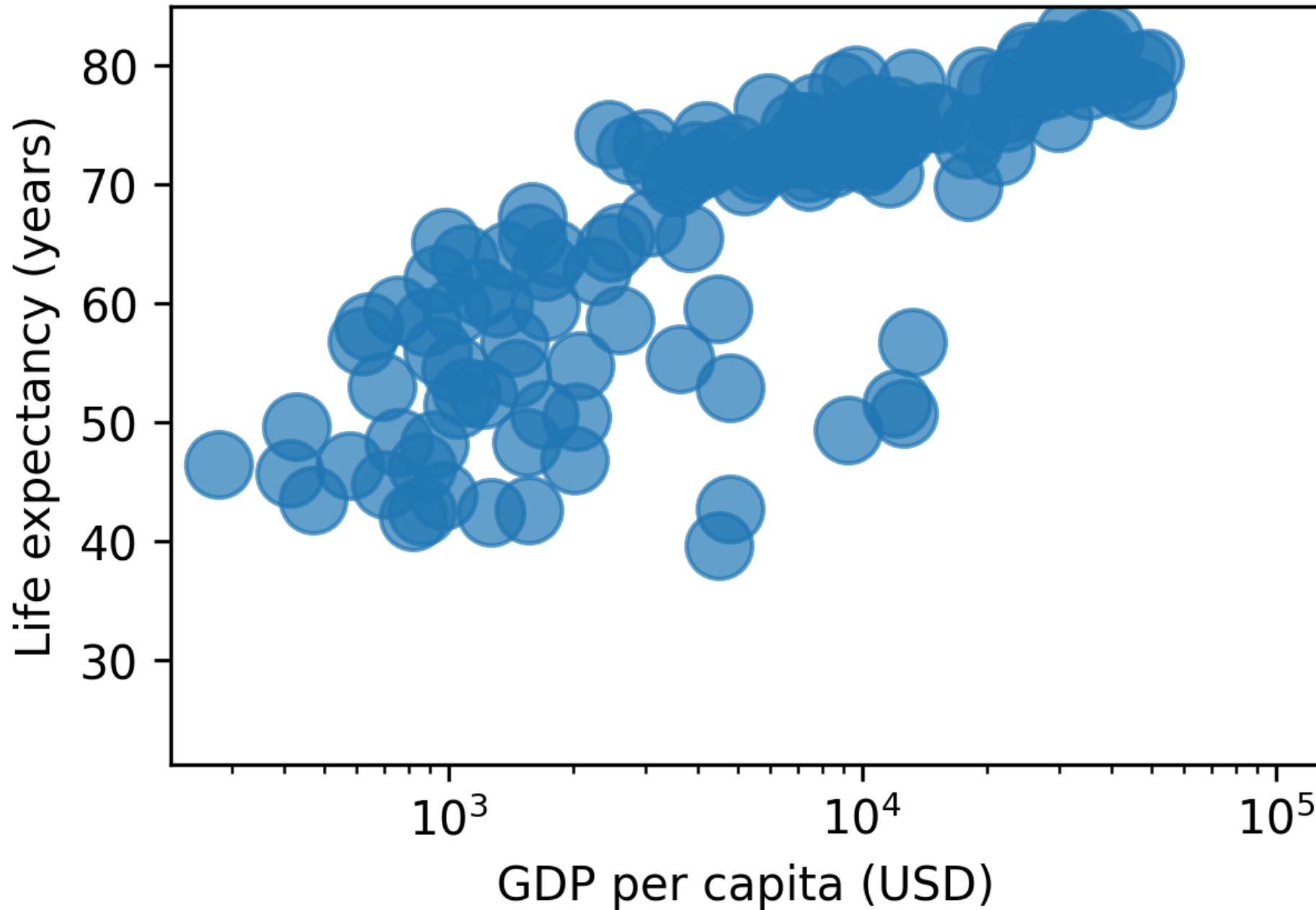




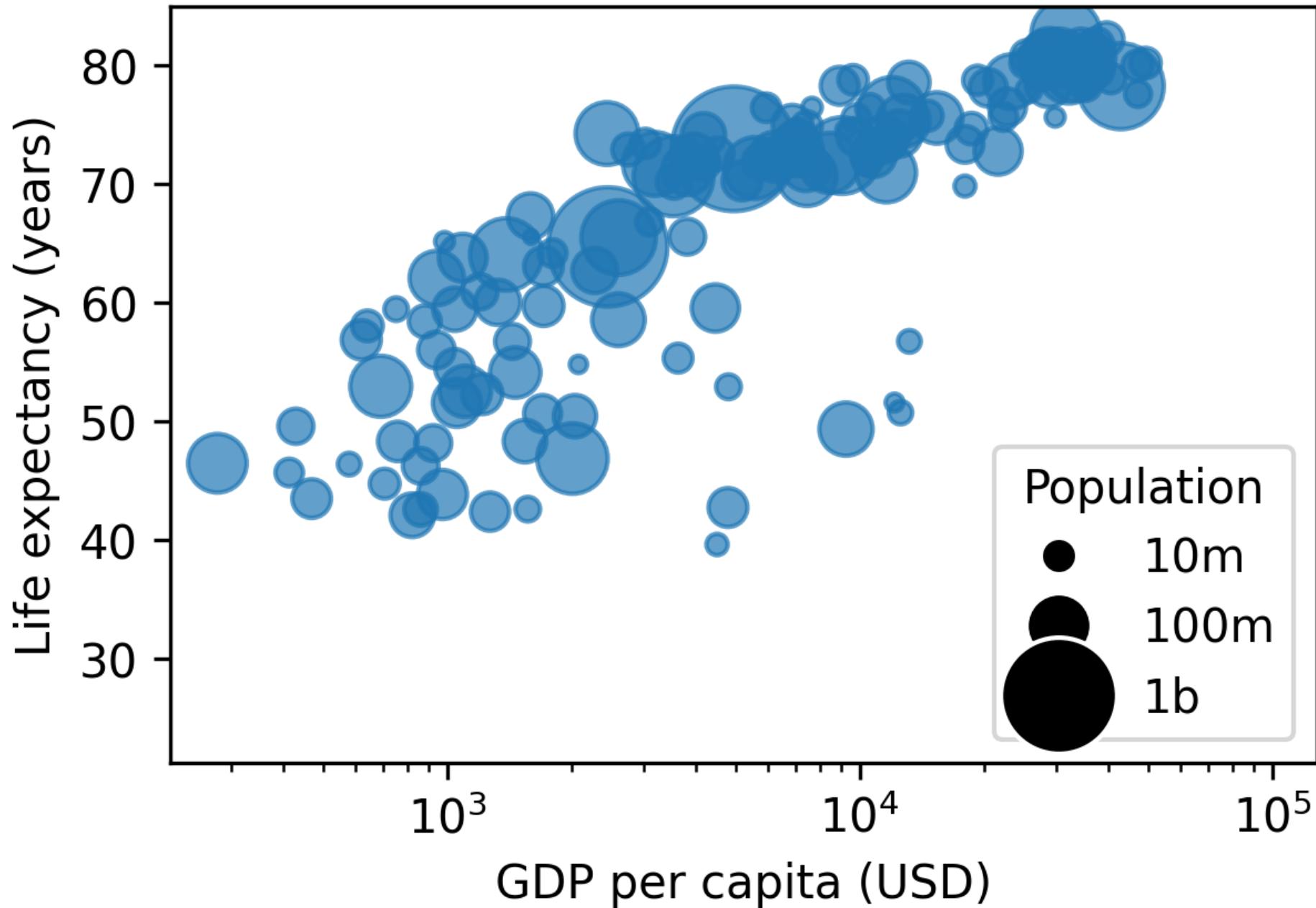




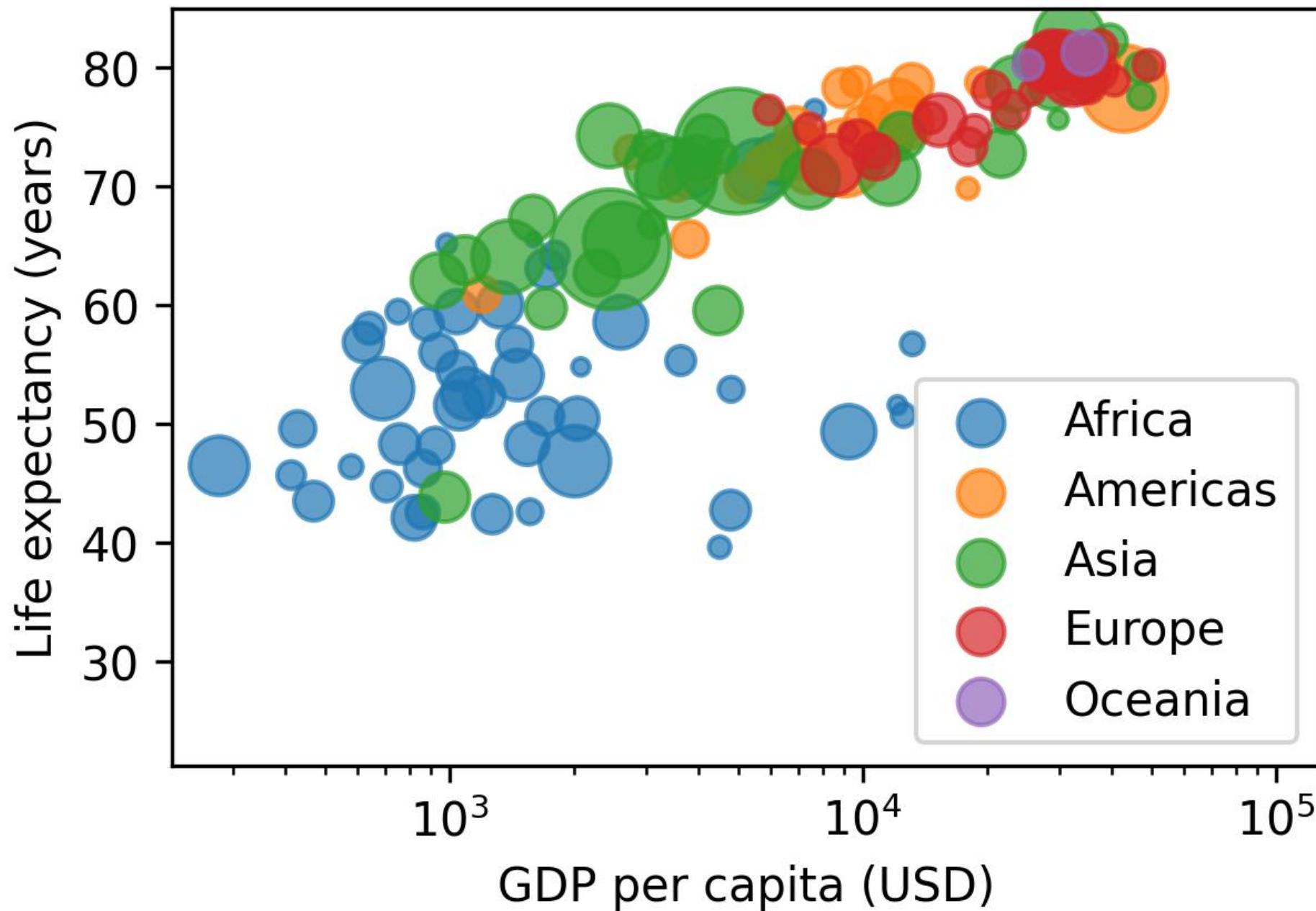
2007

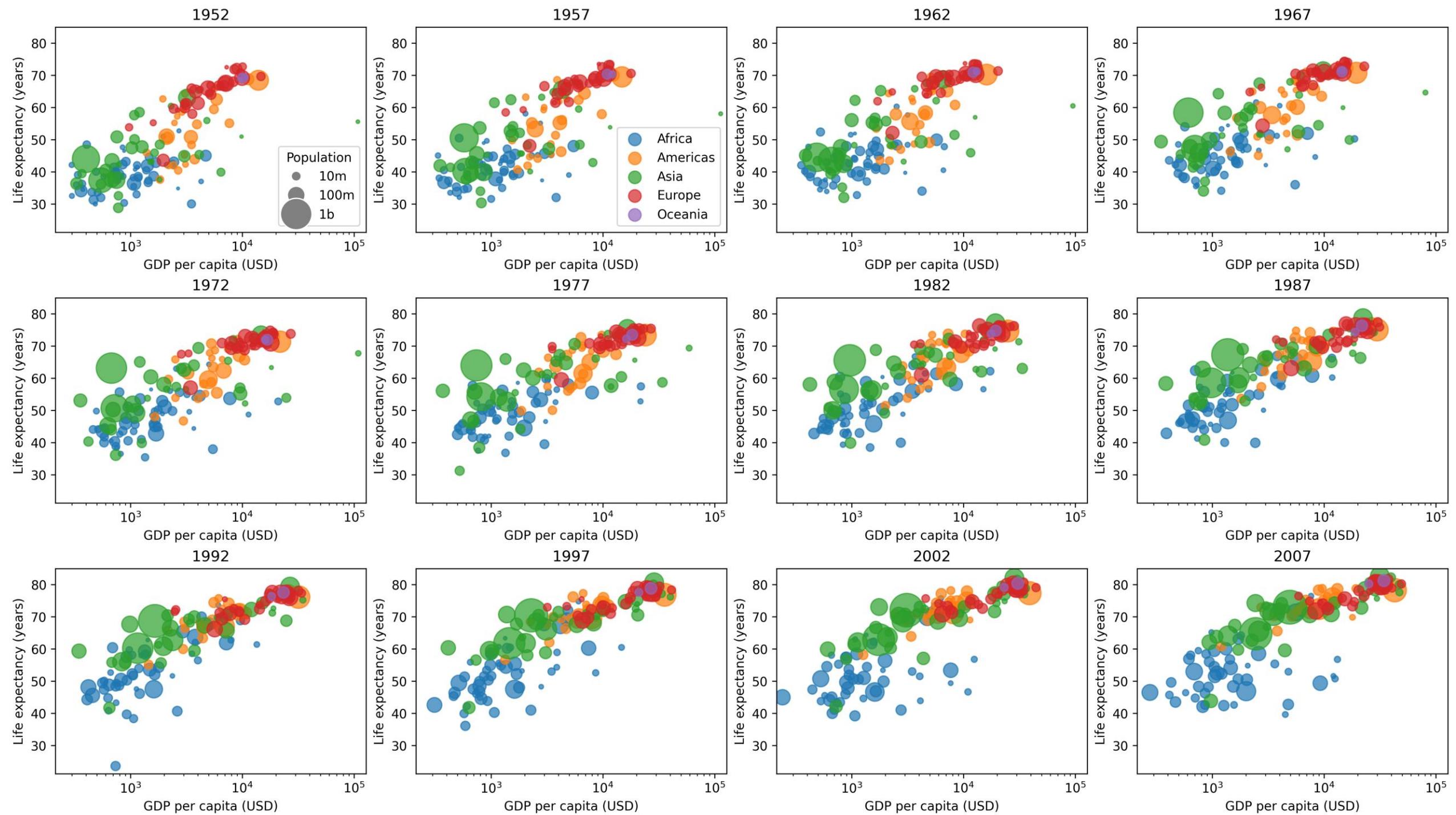


2007



2007





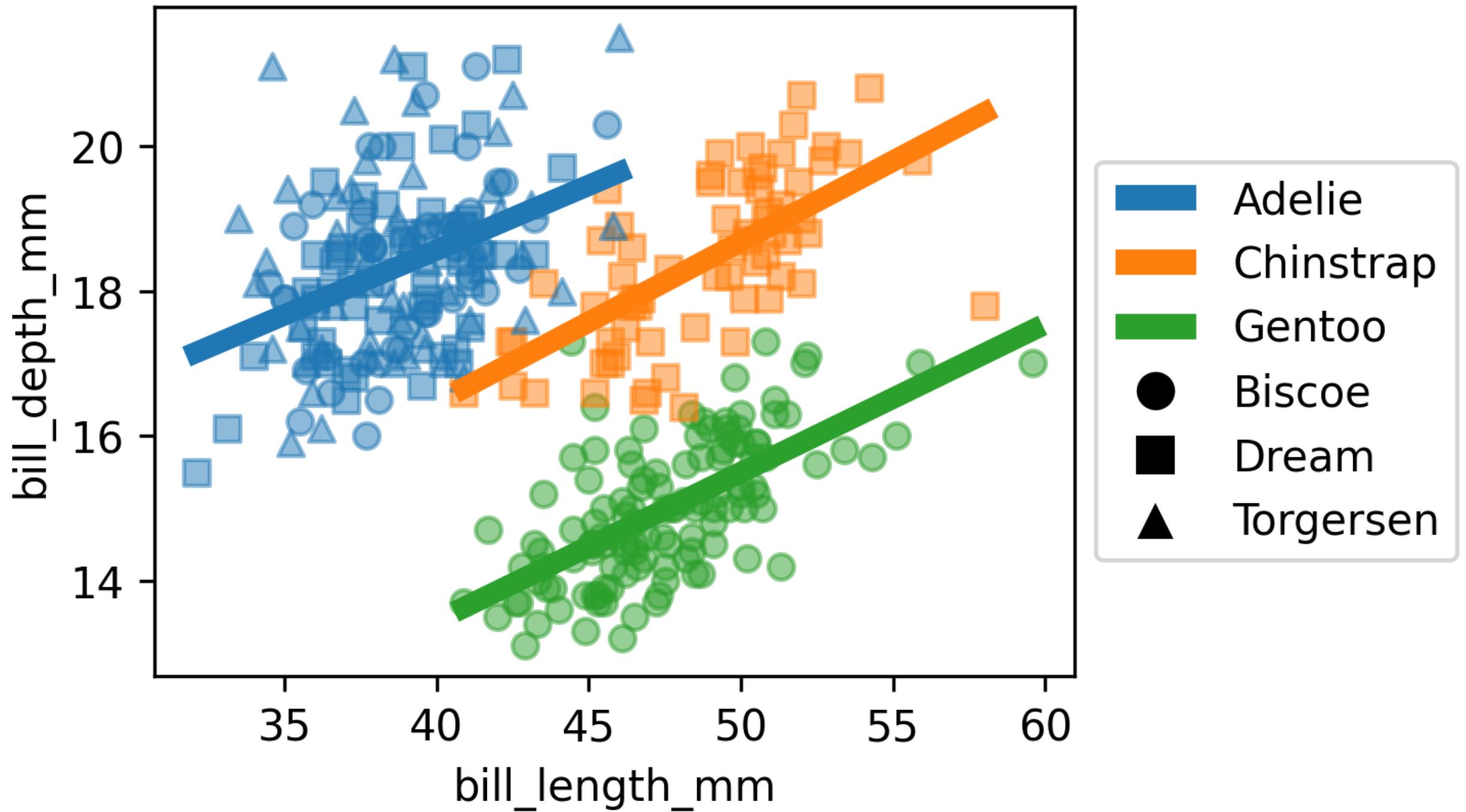
**Extra
Exercise**



species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007

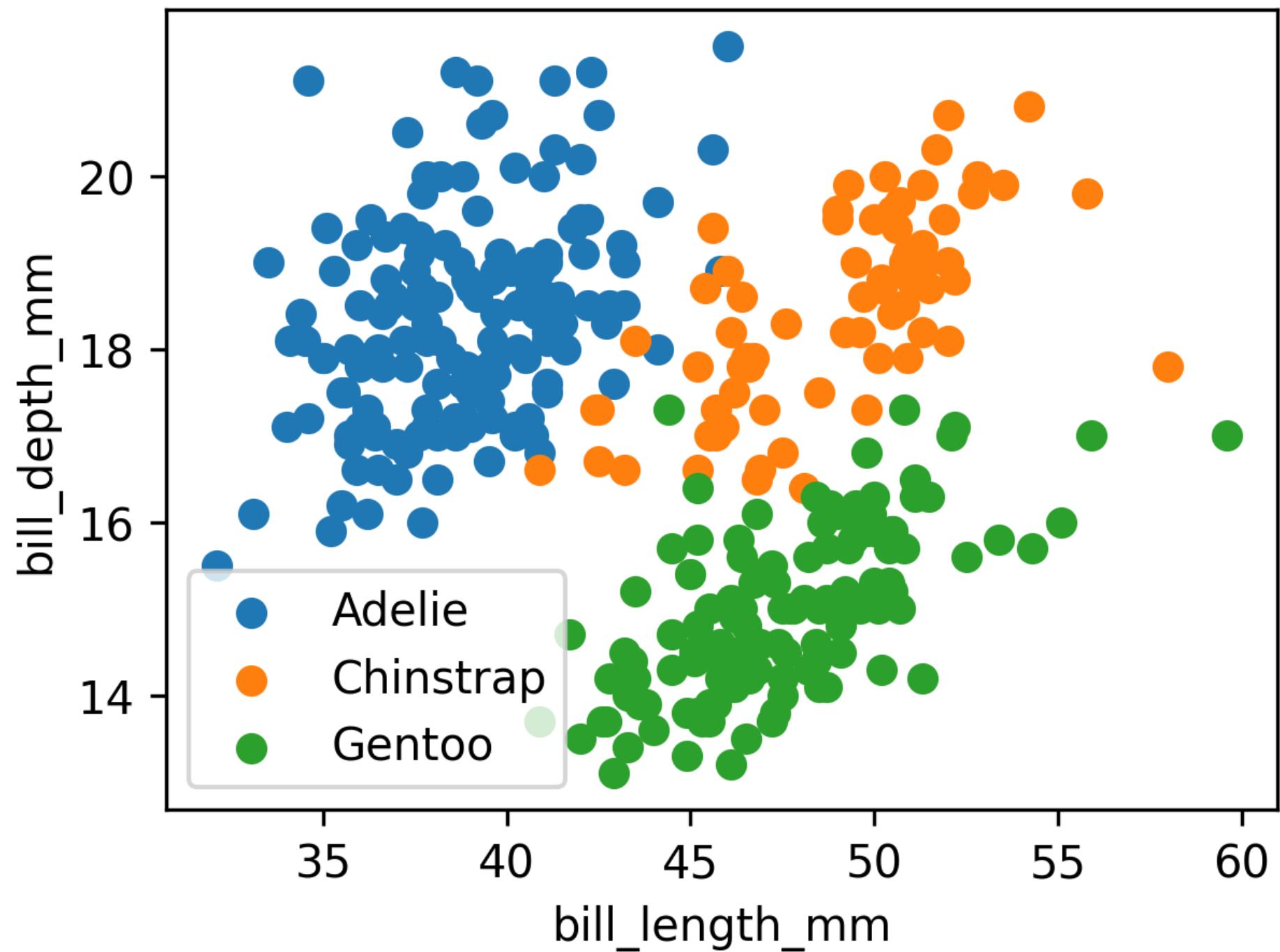
Exercise

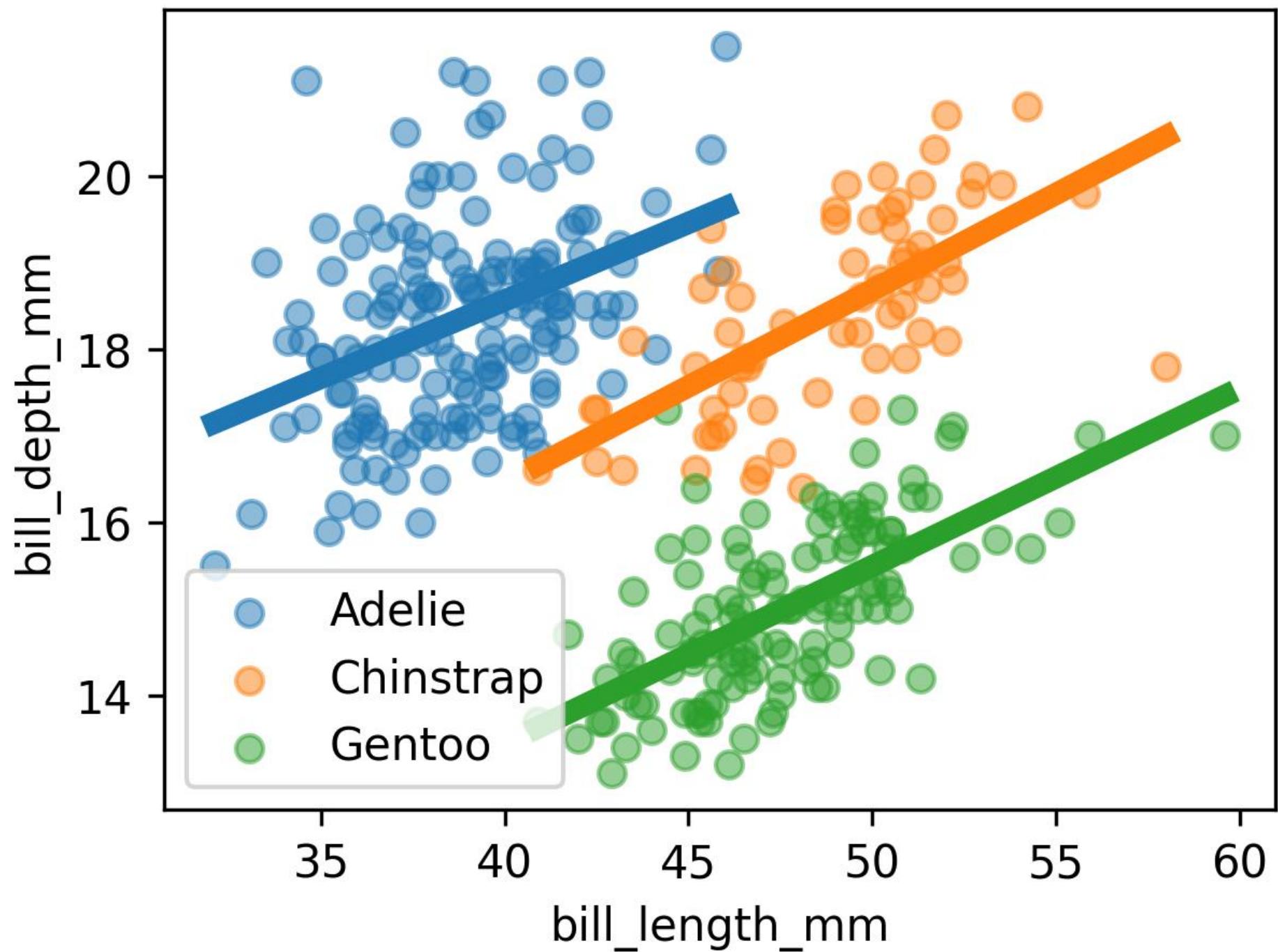
- Reproduce the following plot:
 - Scatterplot, with labeled axes
 - One colour per species
 - One marker per island
 - Fit a line for each species
 - Legend

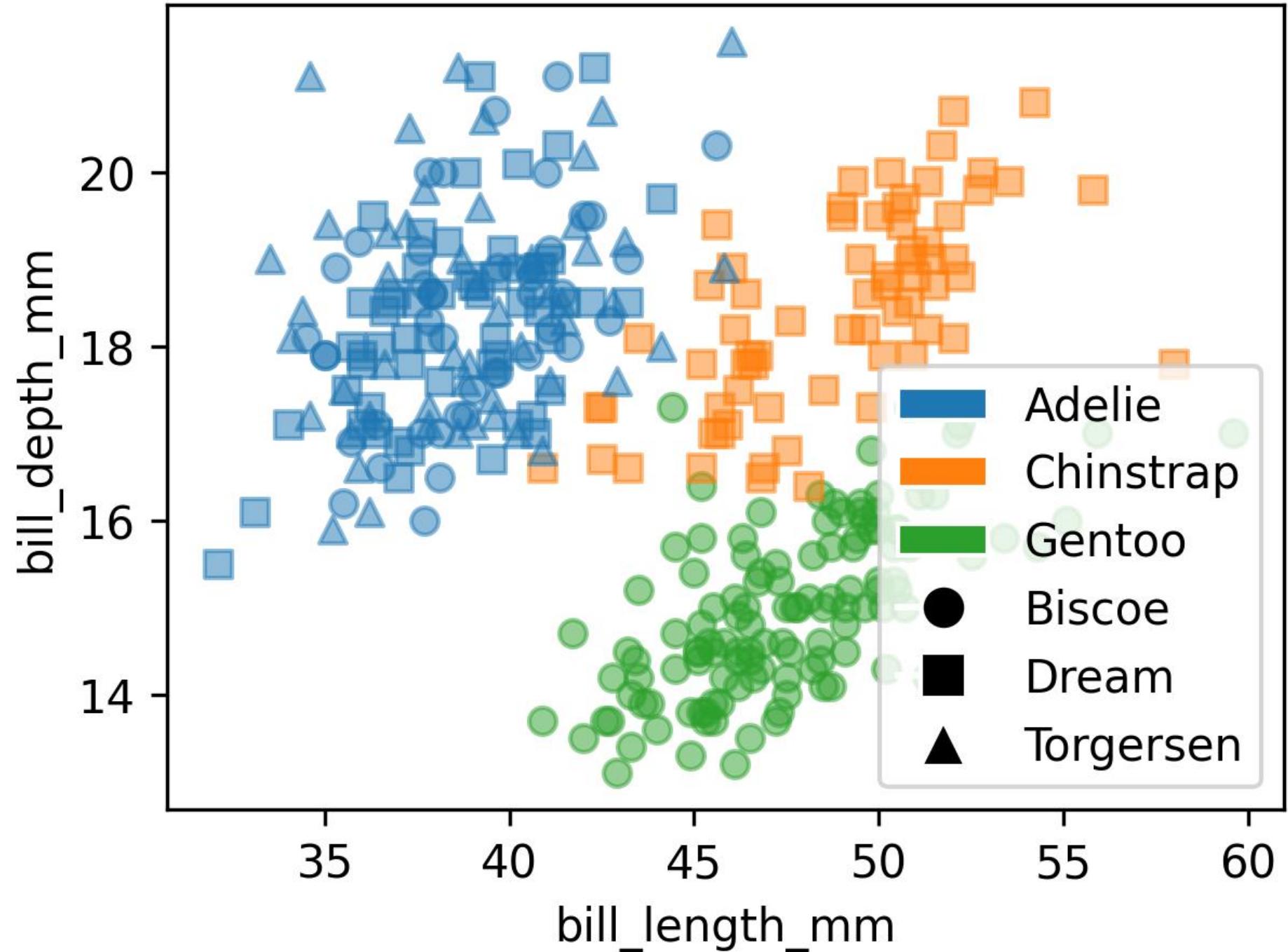


Exercise

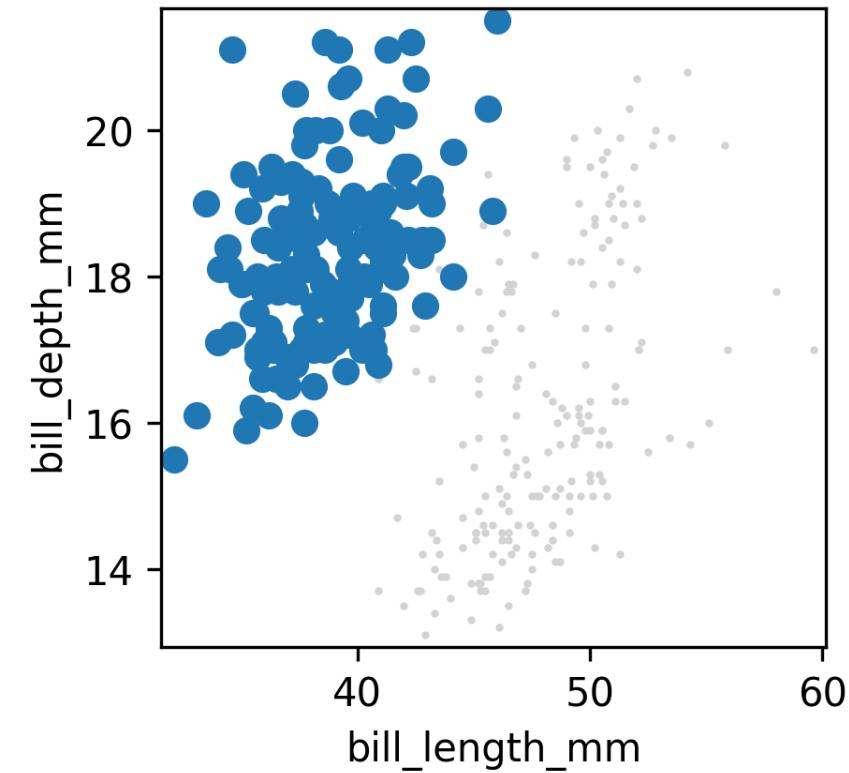
- Find other interesting visualizations of the Penguin dataset



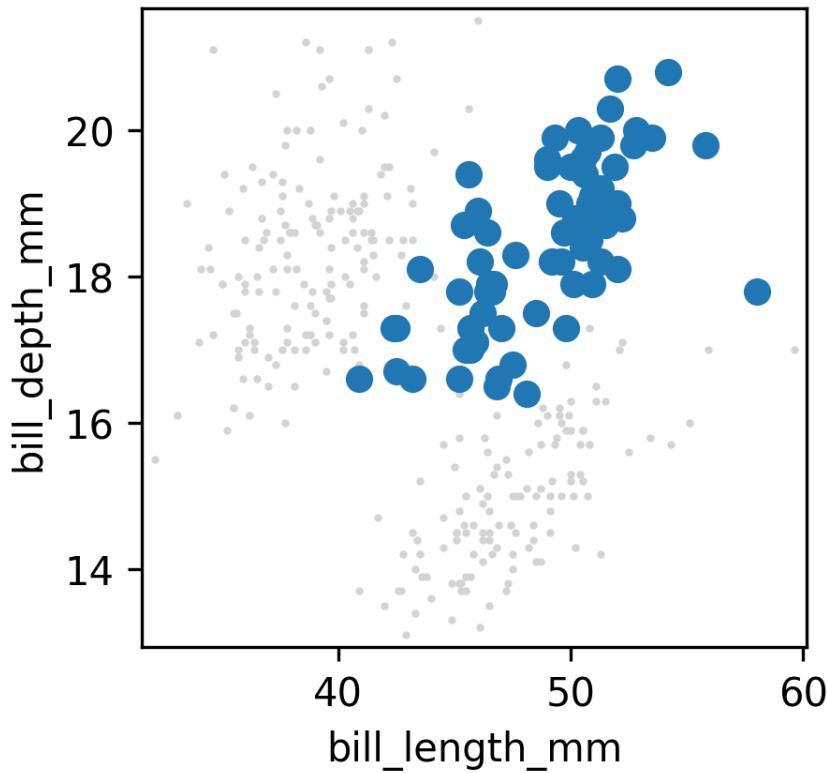




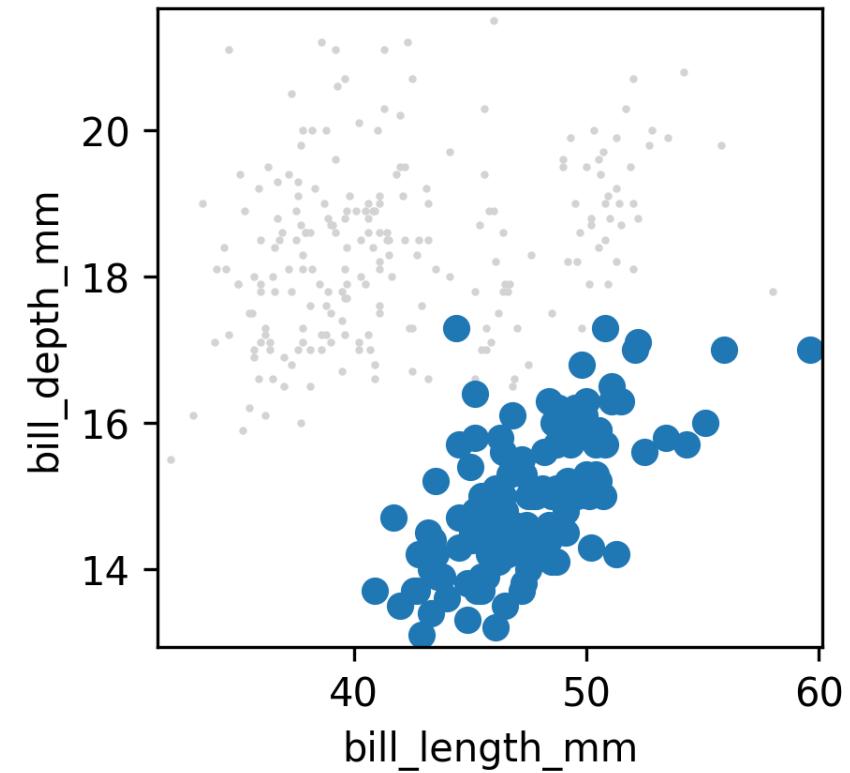
Adelie

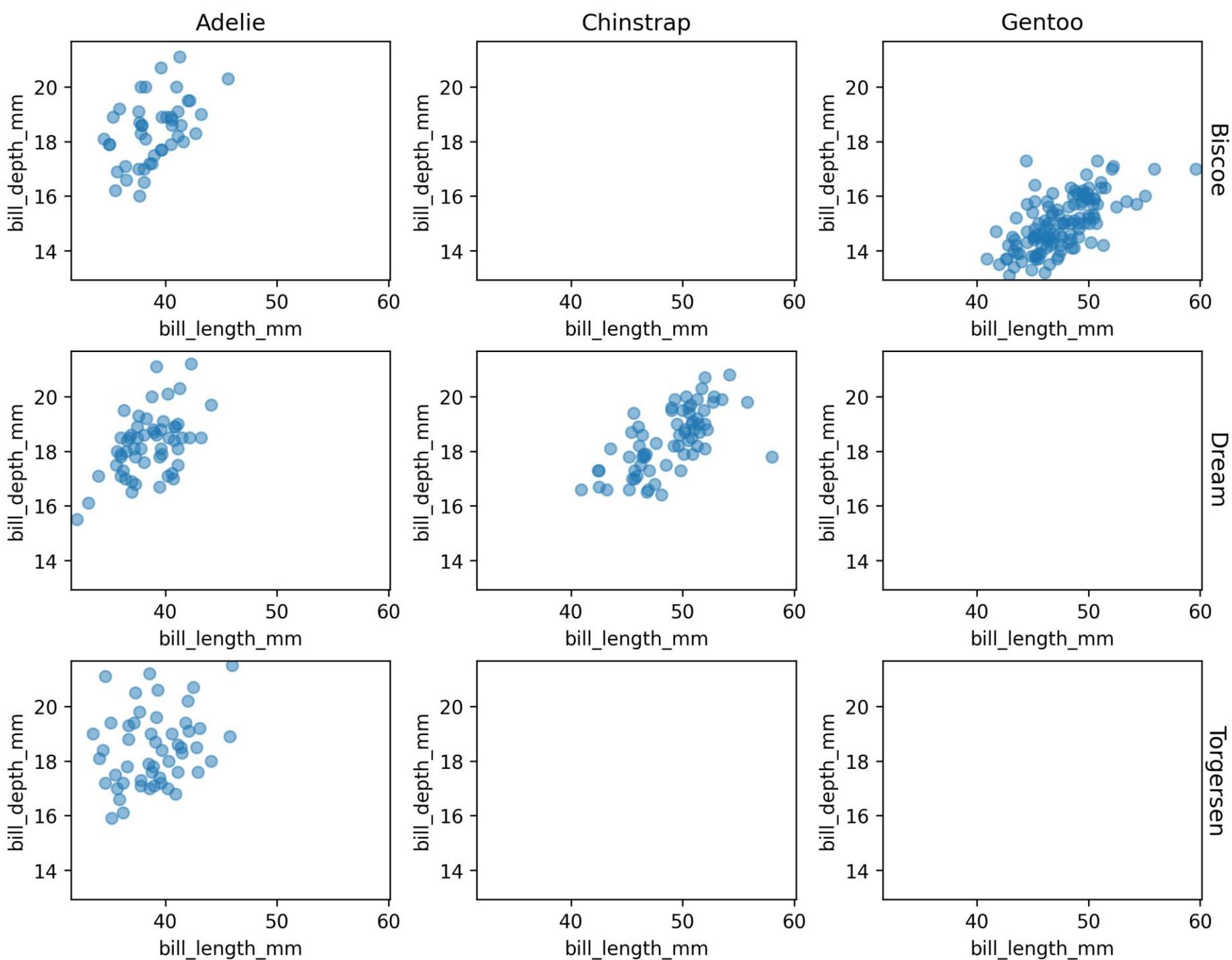


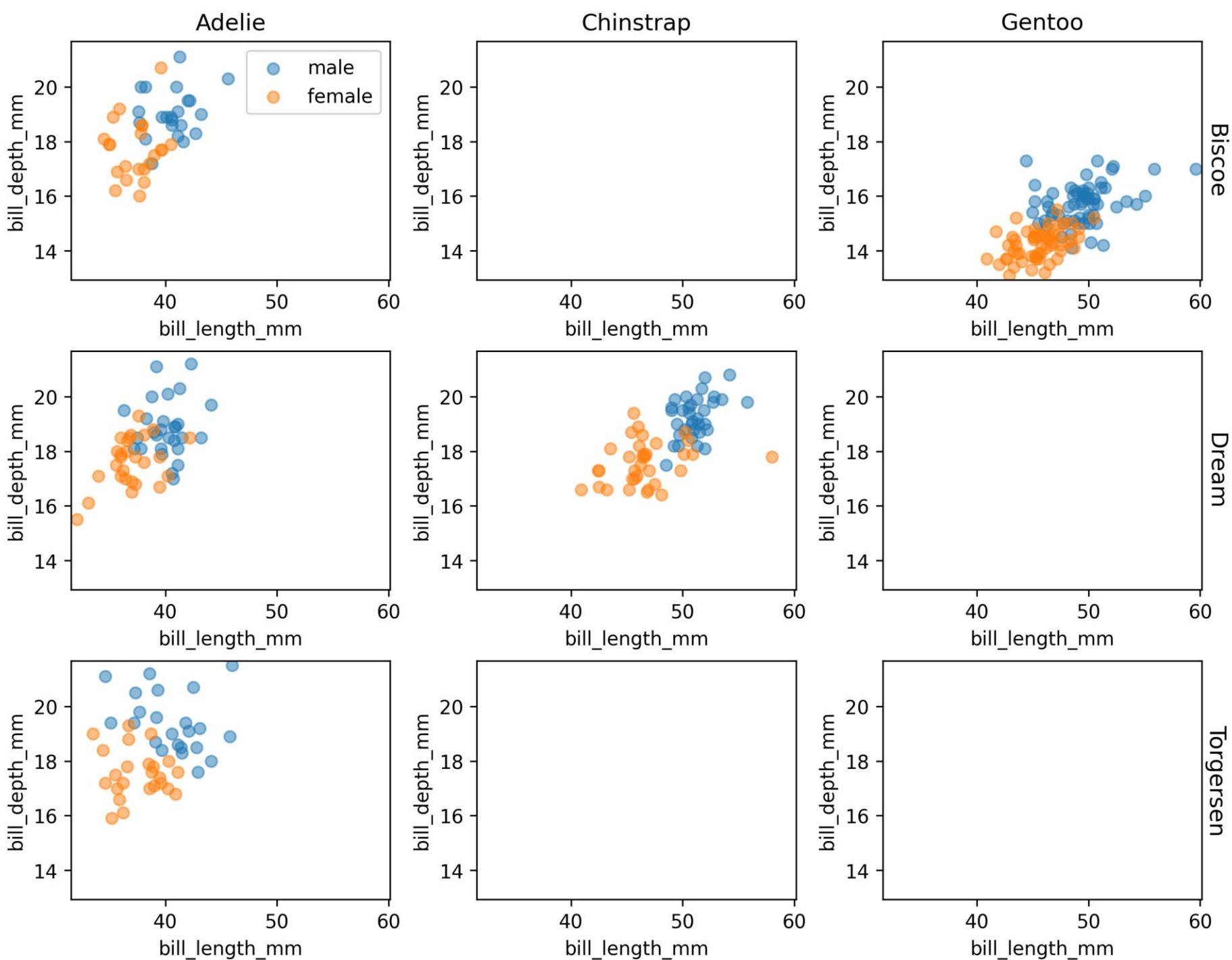
Chinstrap



Gentoo





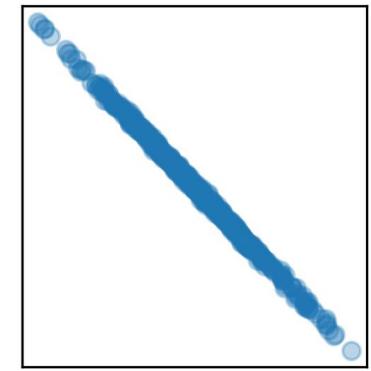


Extra

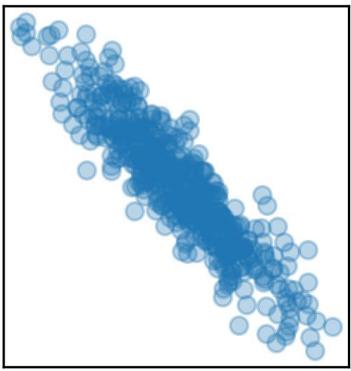
Slides

Correlation

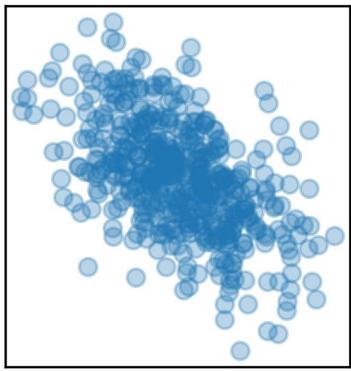
$\rho=-1.0$



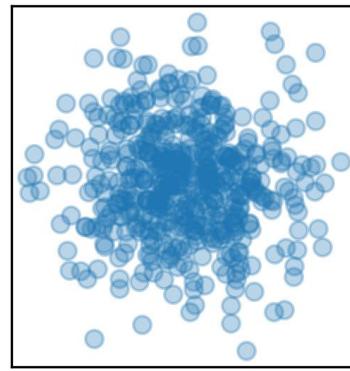
$\rho=-0.9$



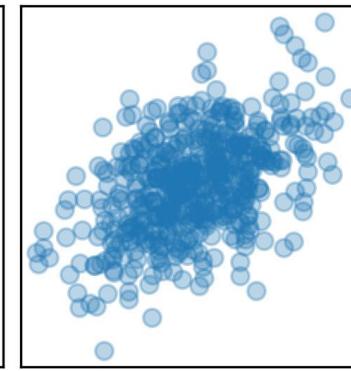
$\rho=-0.5$



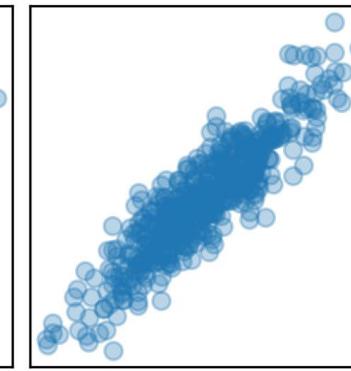
$\rho=0.0$



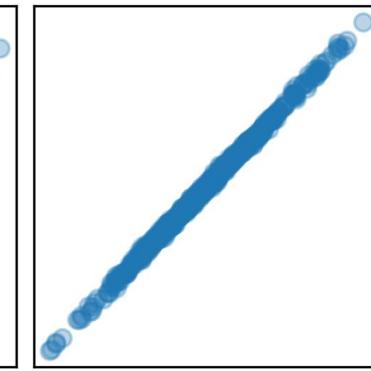
$\rho=0.5$



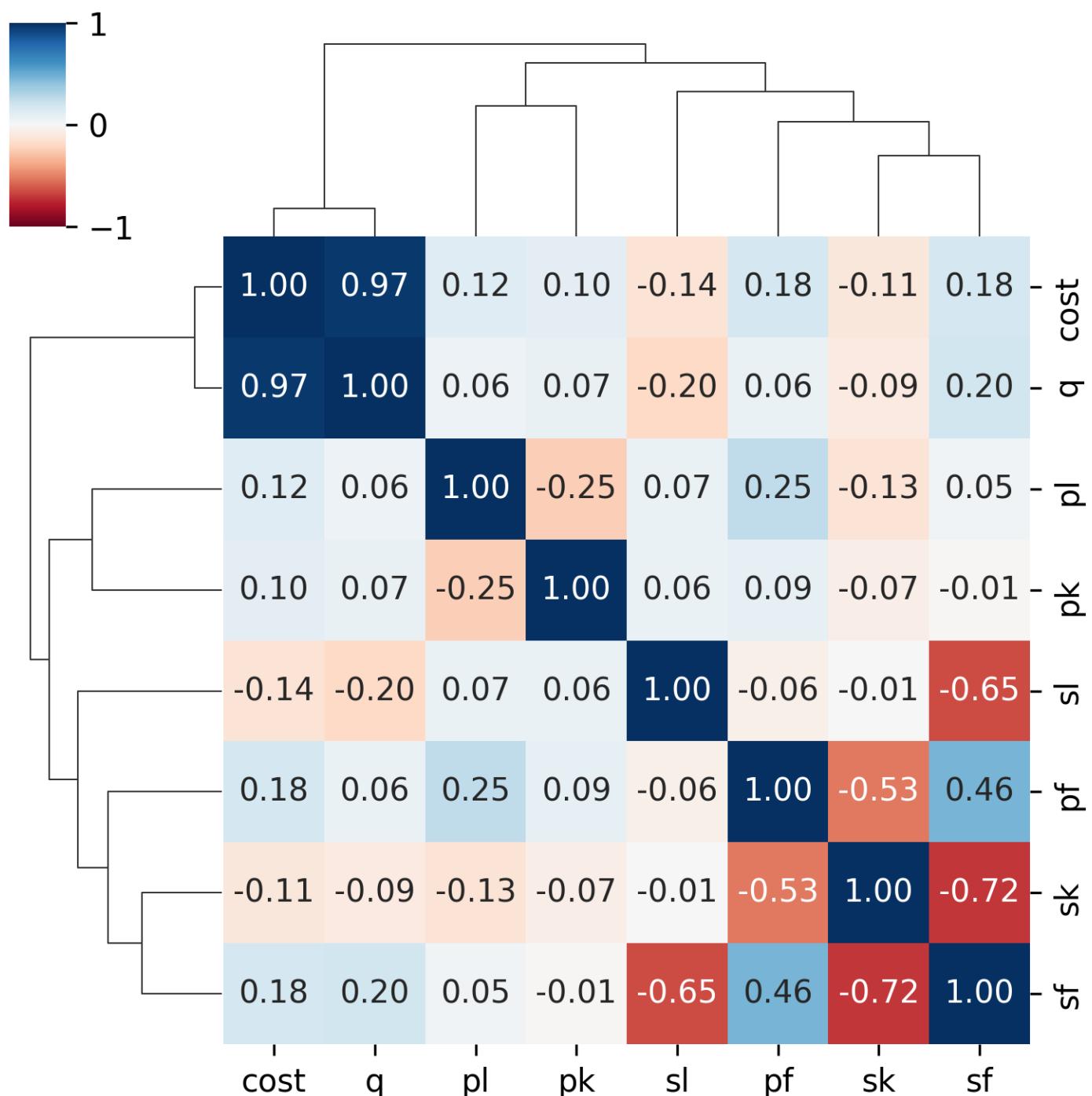
$\rho=0.9$



$\rho=1.0$

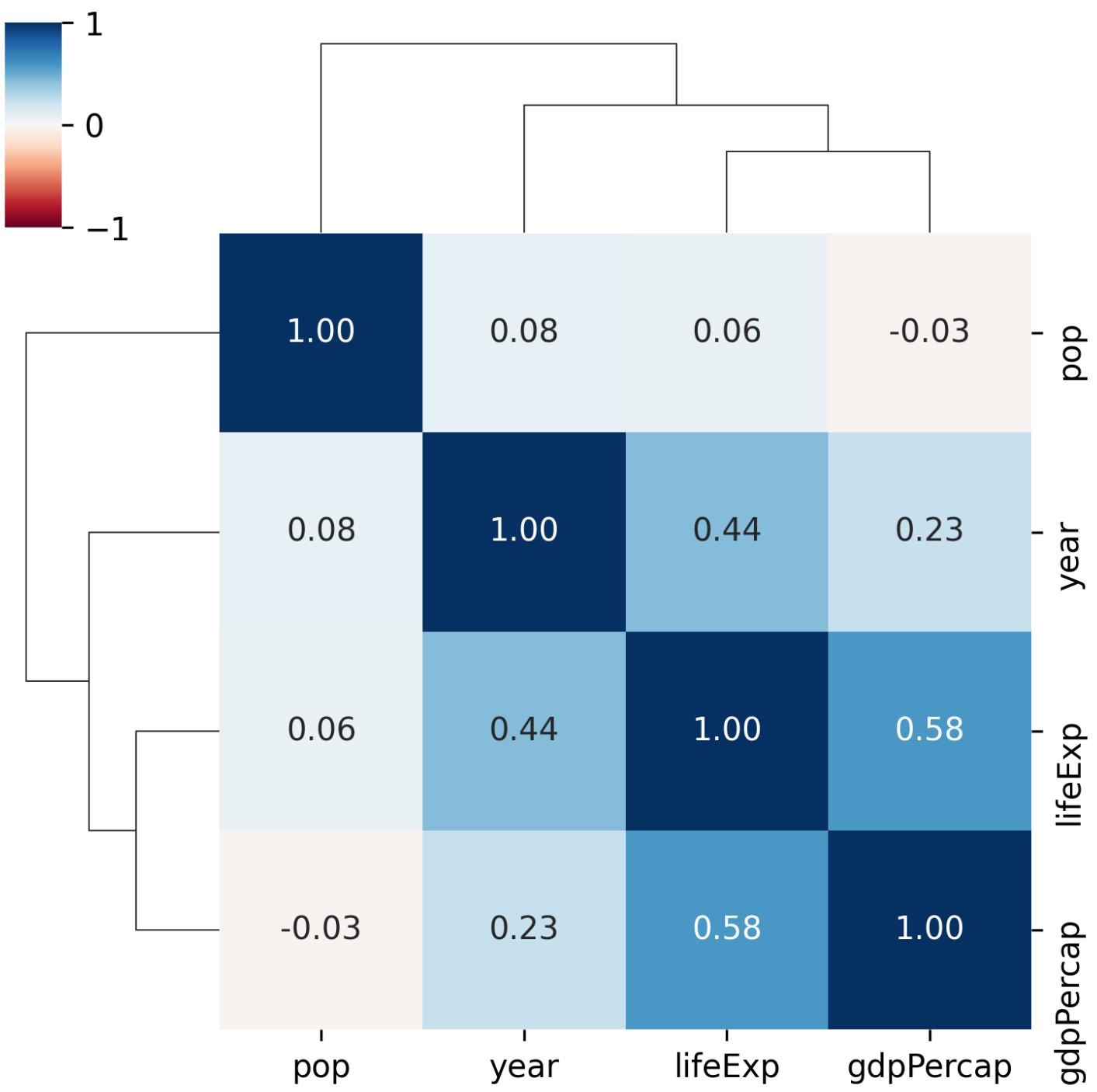


```
sns.clustermap(  
    d.corr(),  
    vmin = -1,  
    vmax = +1,  
    cmap = 'RdBu',  
    figsize = (5, 5),  
    annot = True,  
    fmt = ".2f",  
)
```



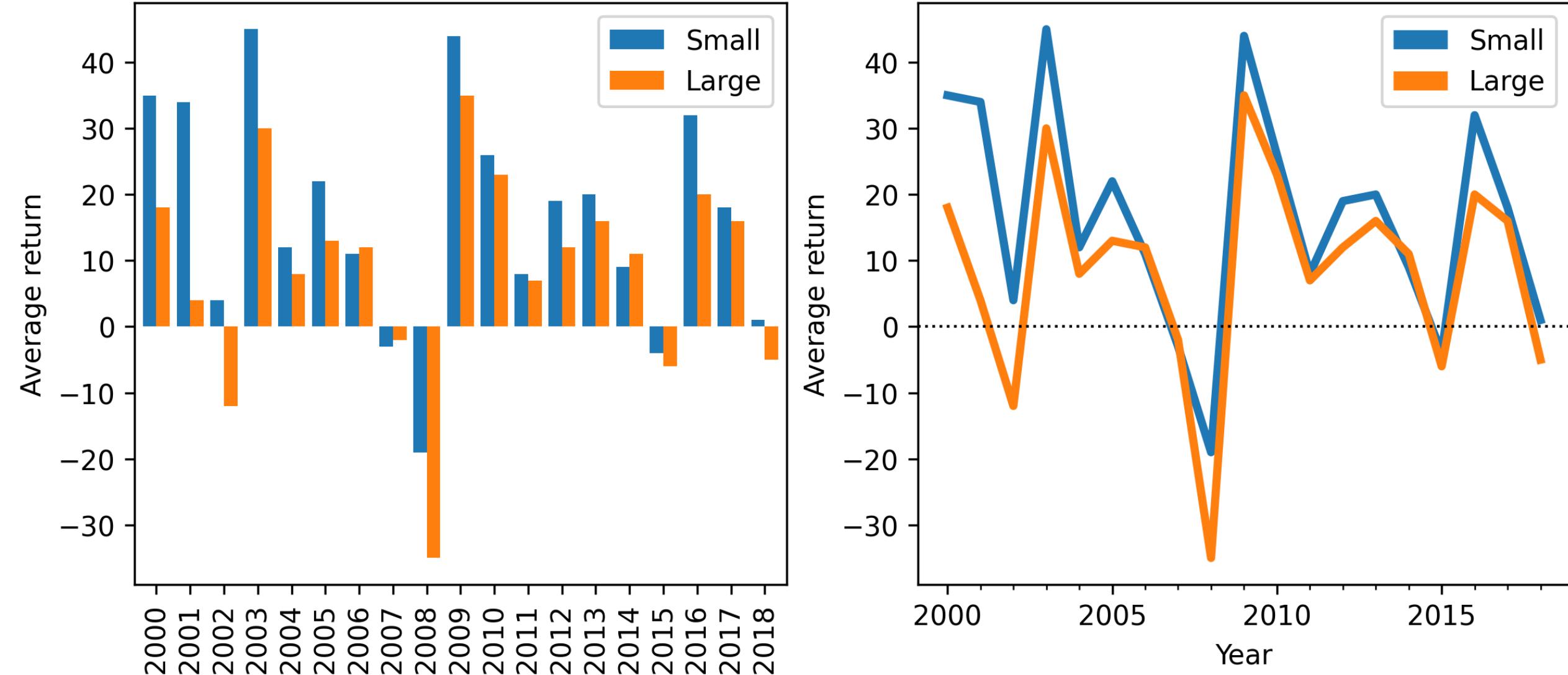
Exercise

- Plot the correlation matrix of the dataset
- Is there anything noteworthy?

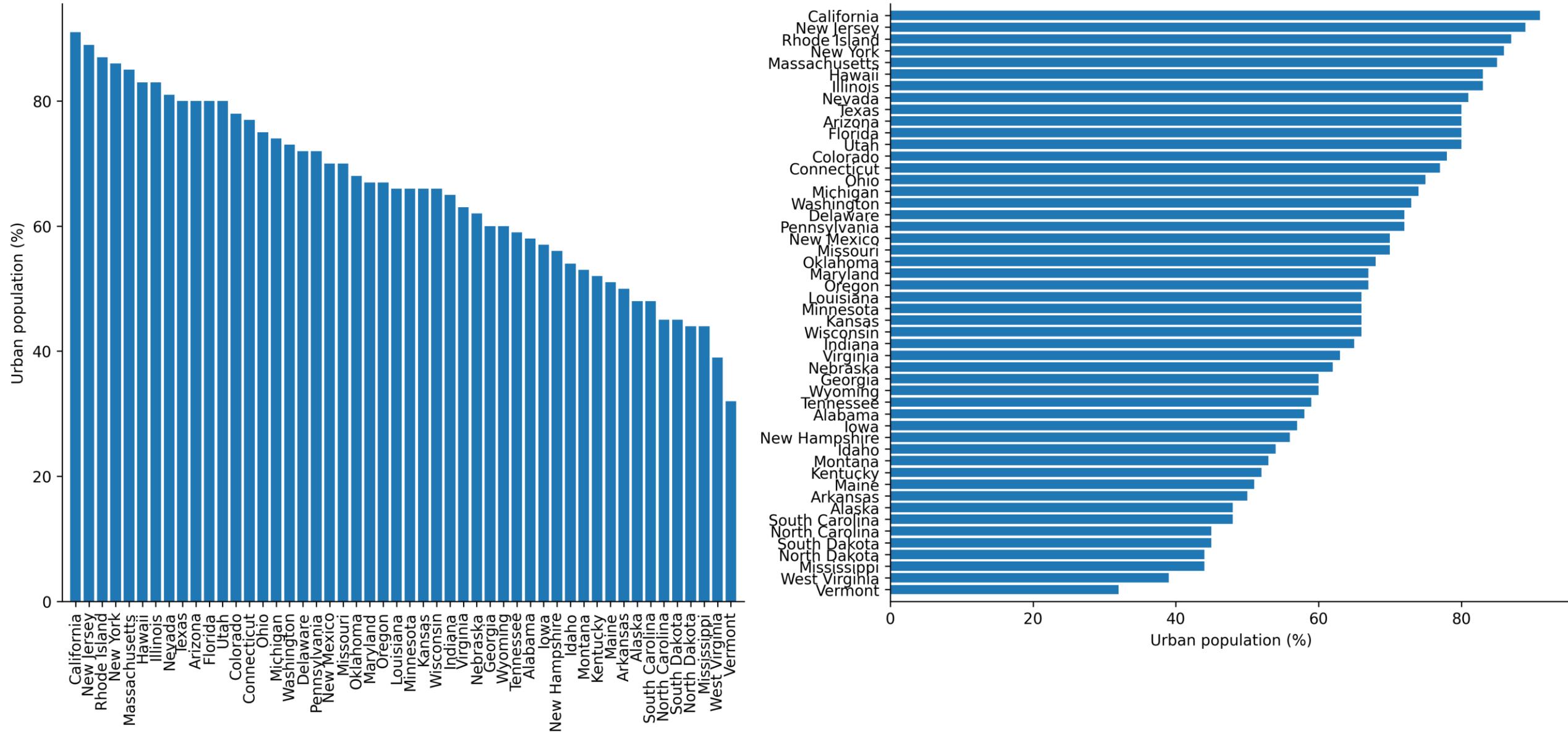


Plotting Advice

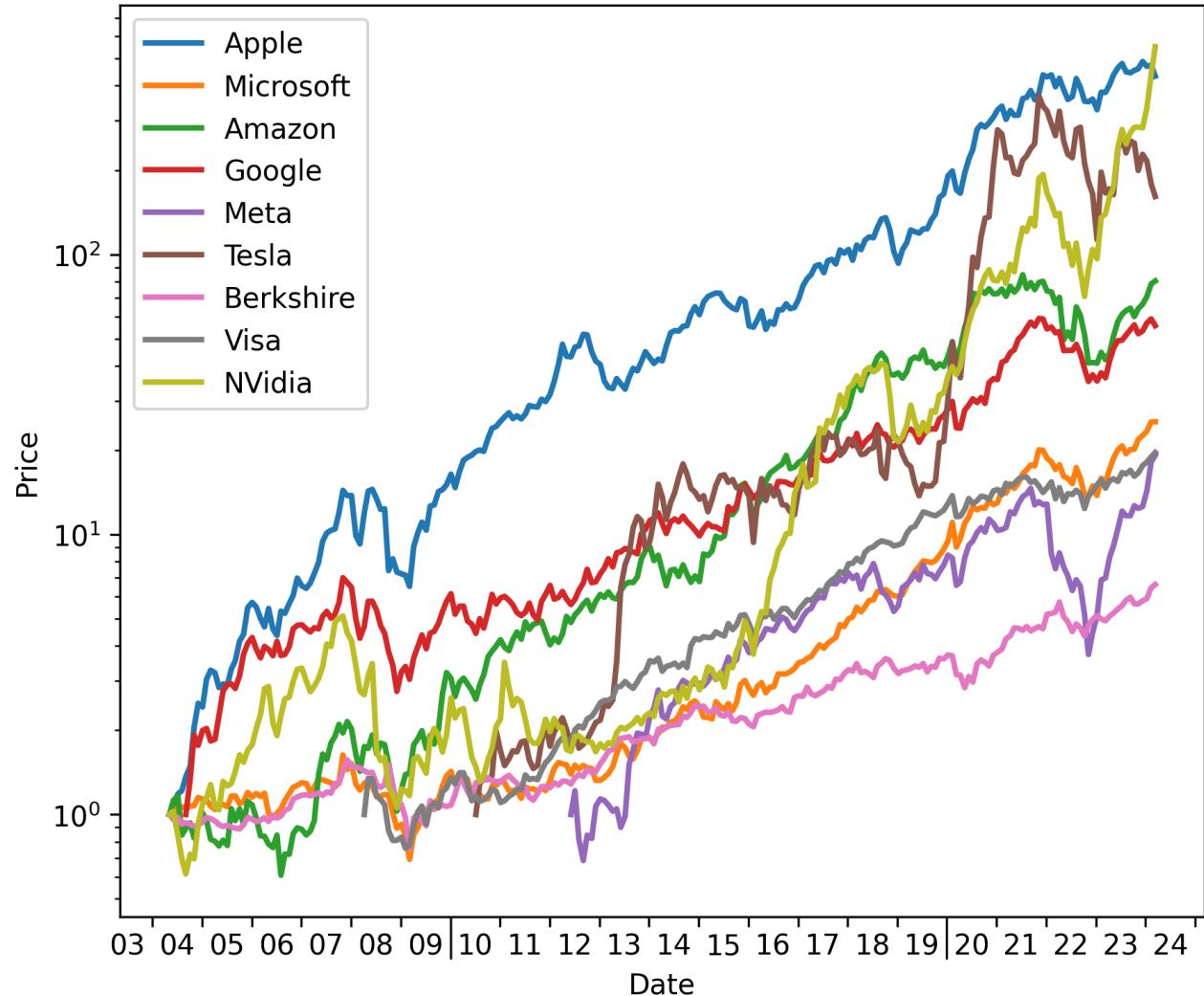
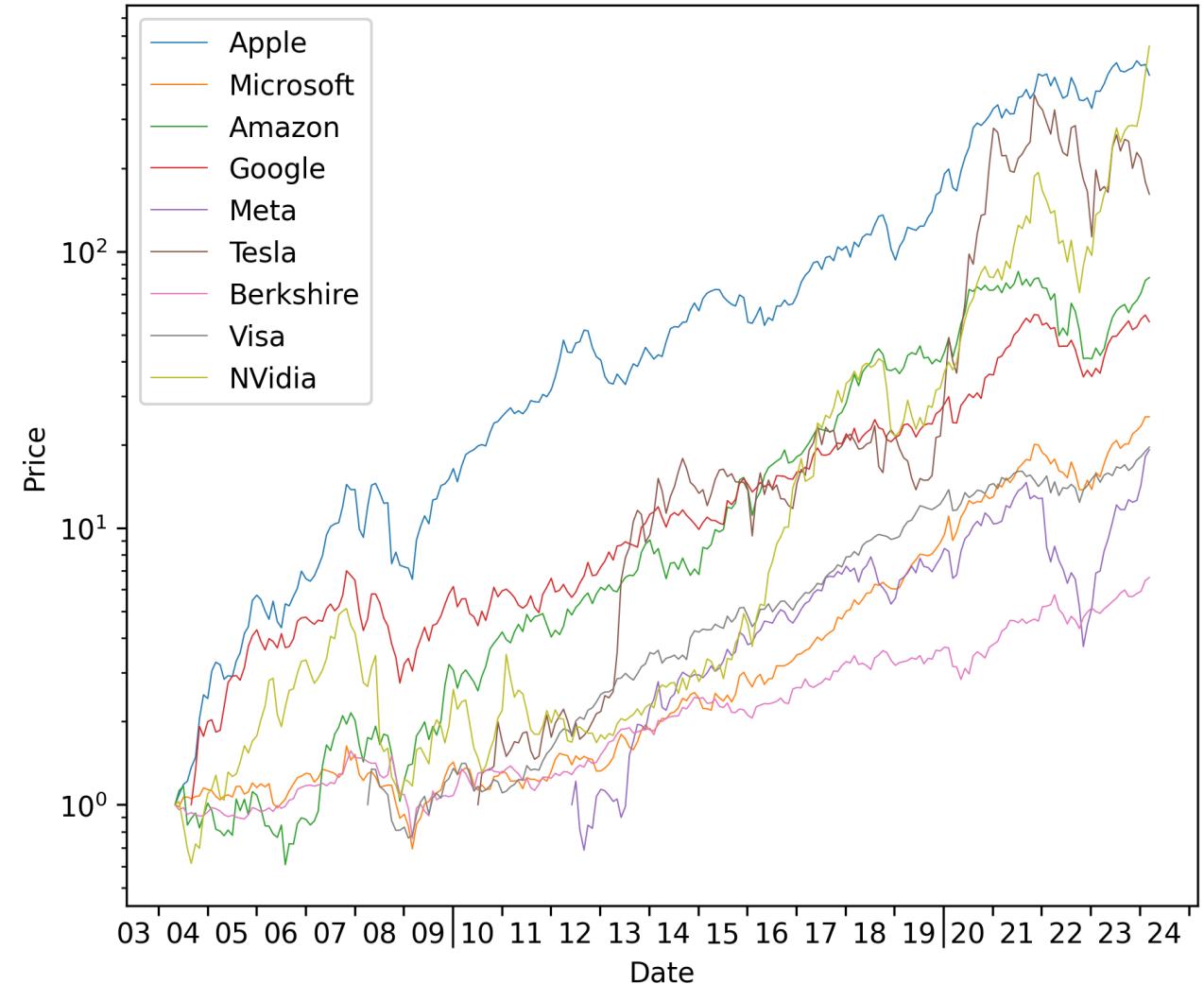
Line Plots



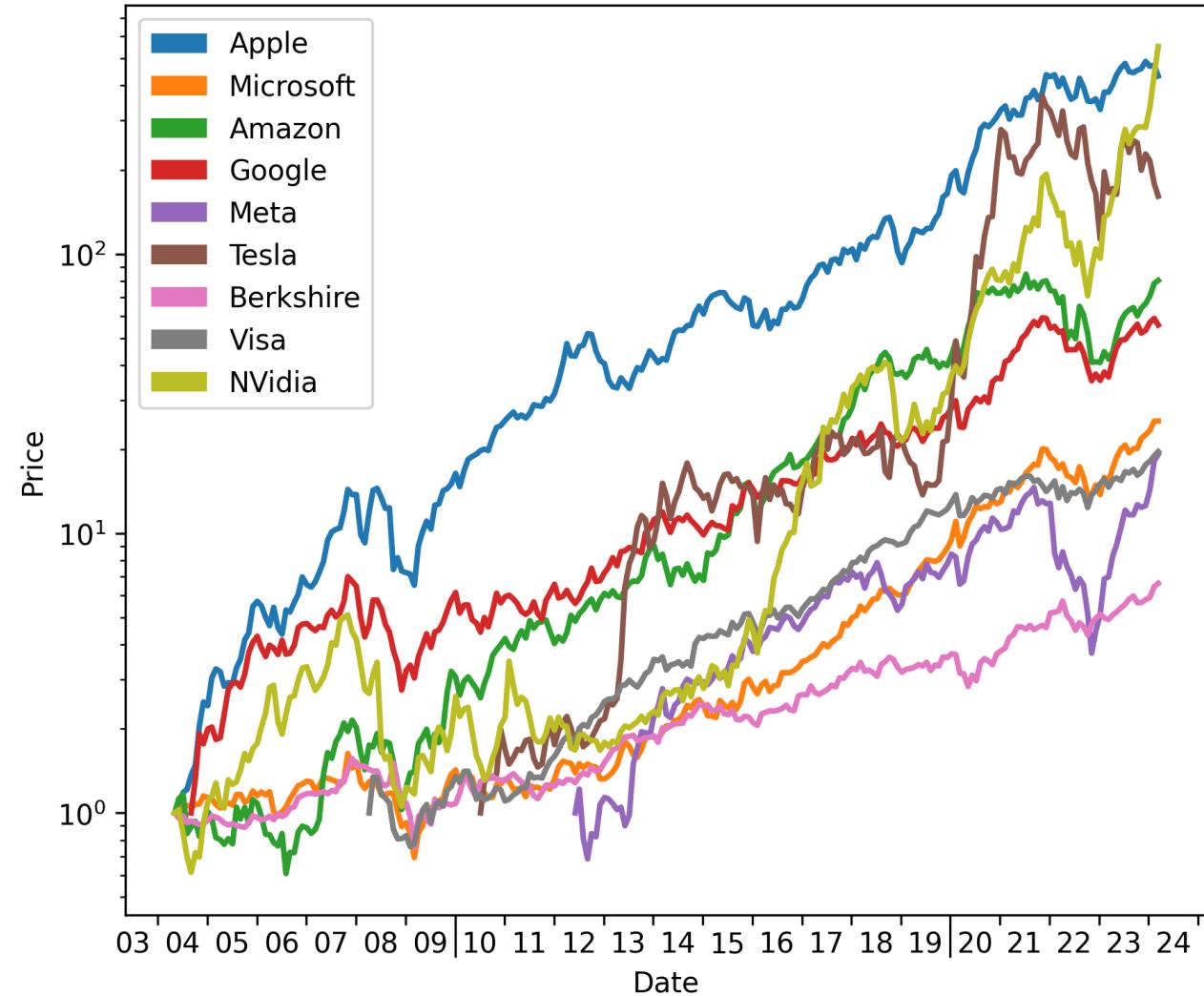
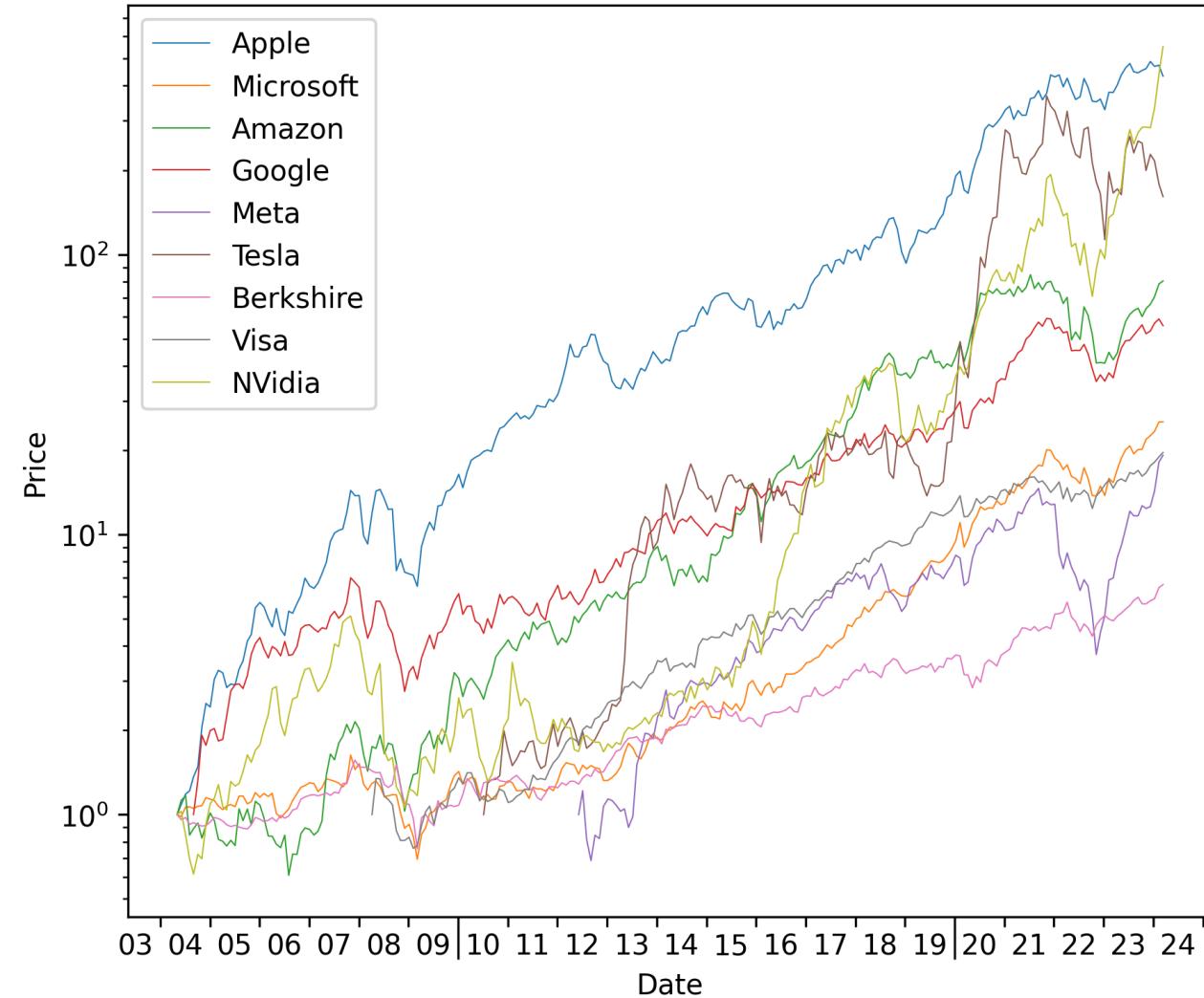
Horizontal Labels



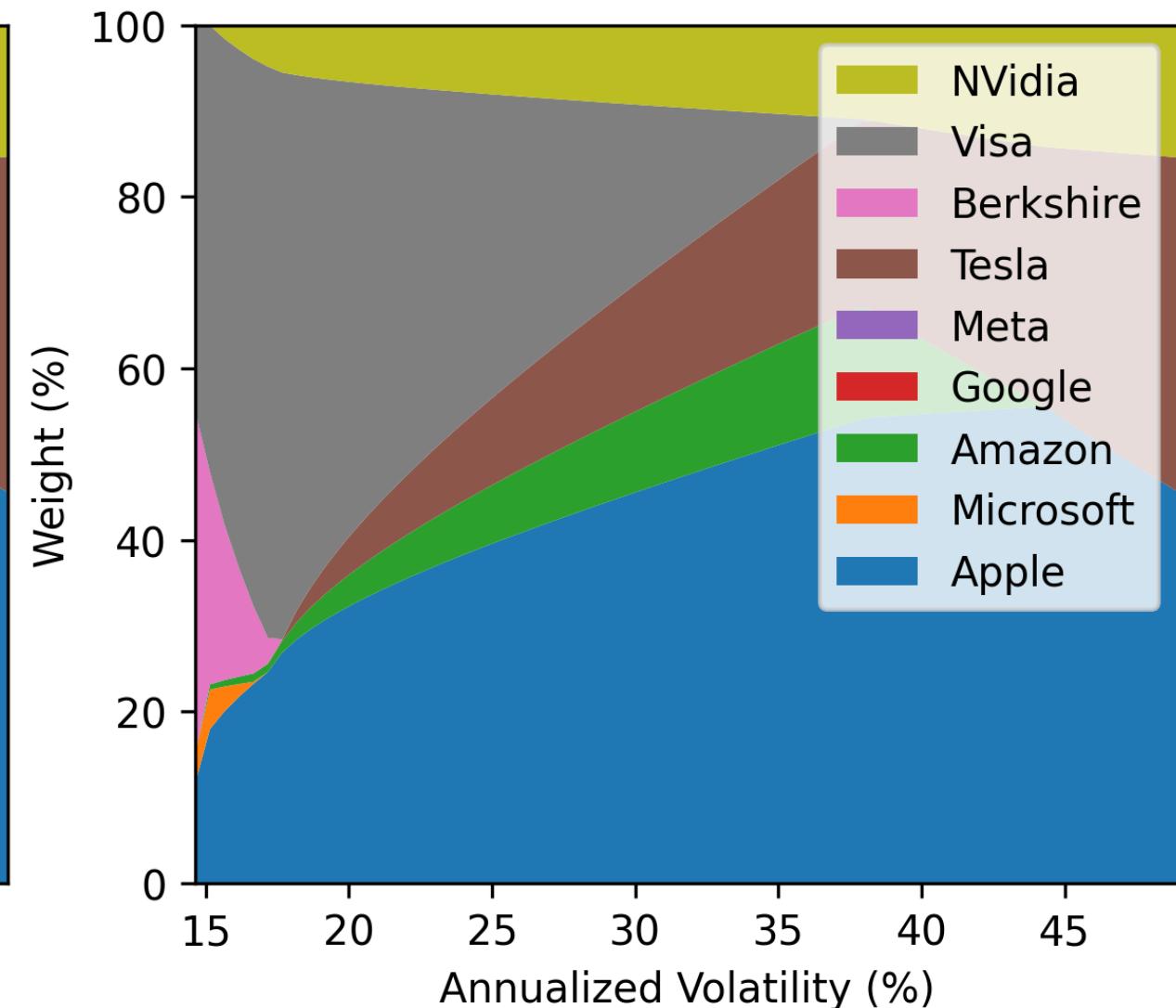
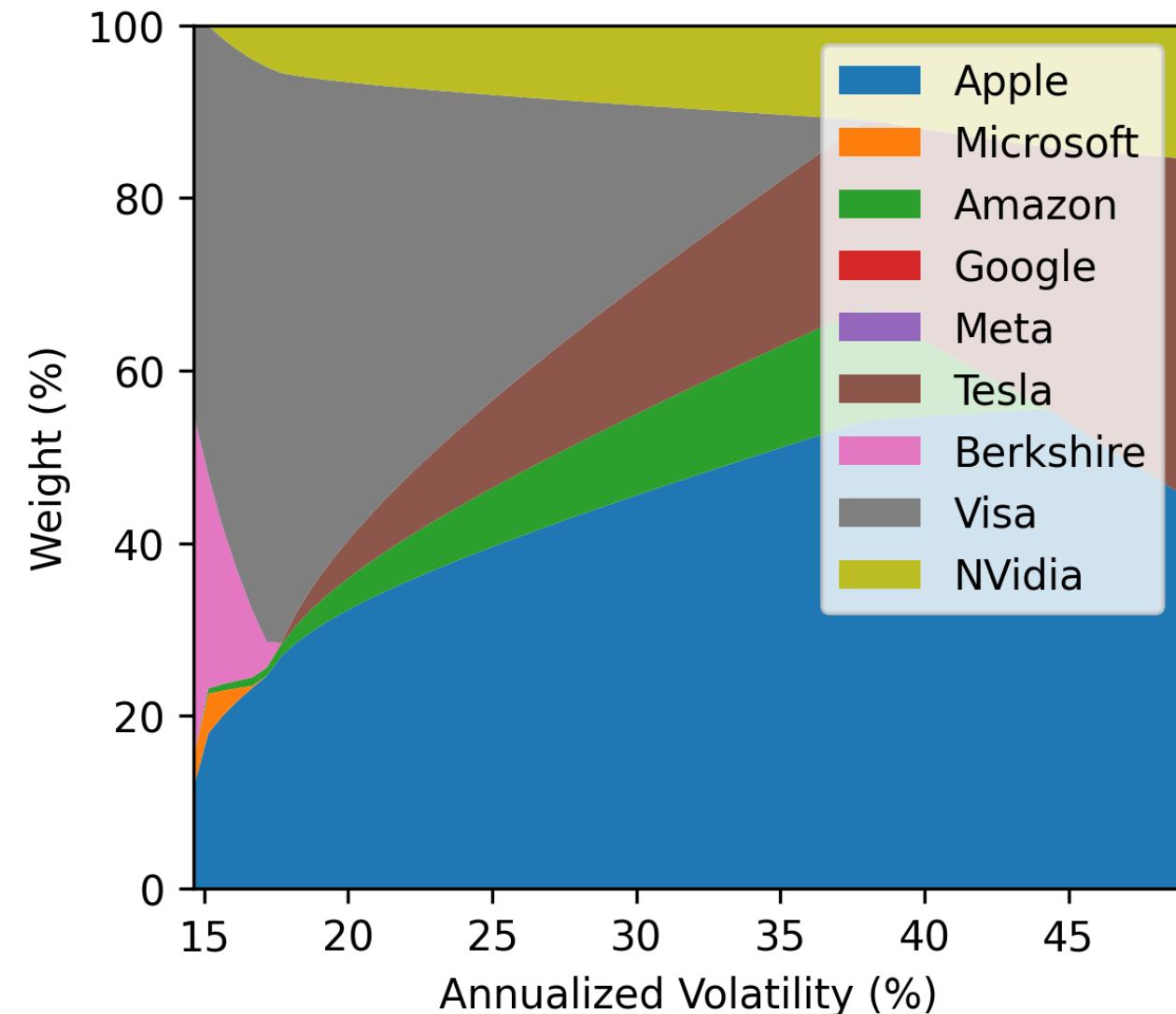
Thick lines



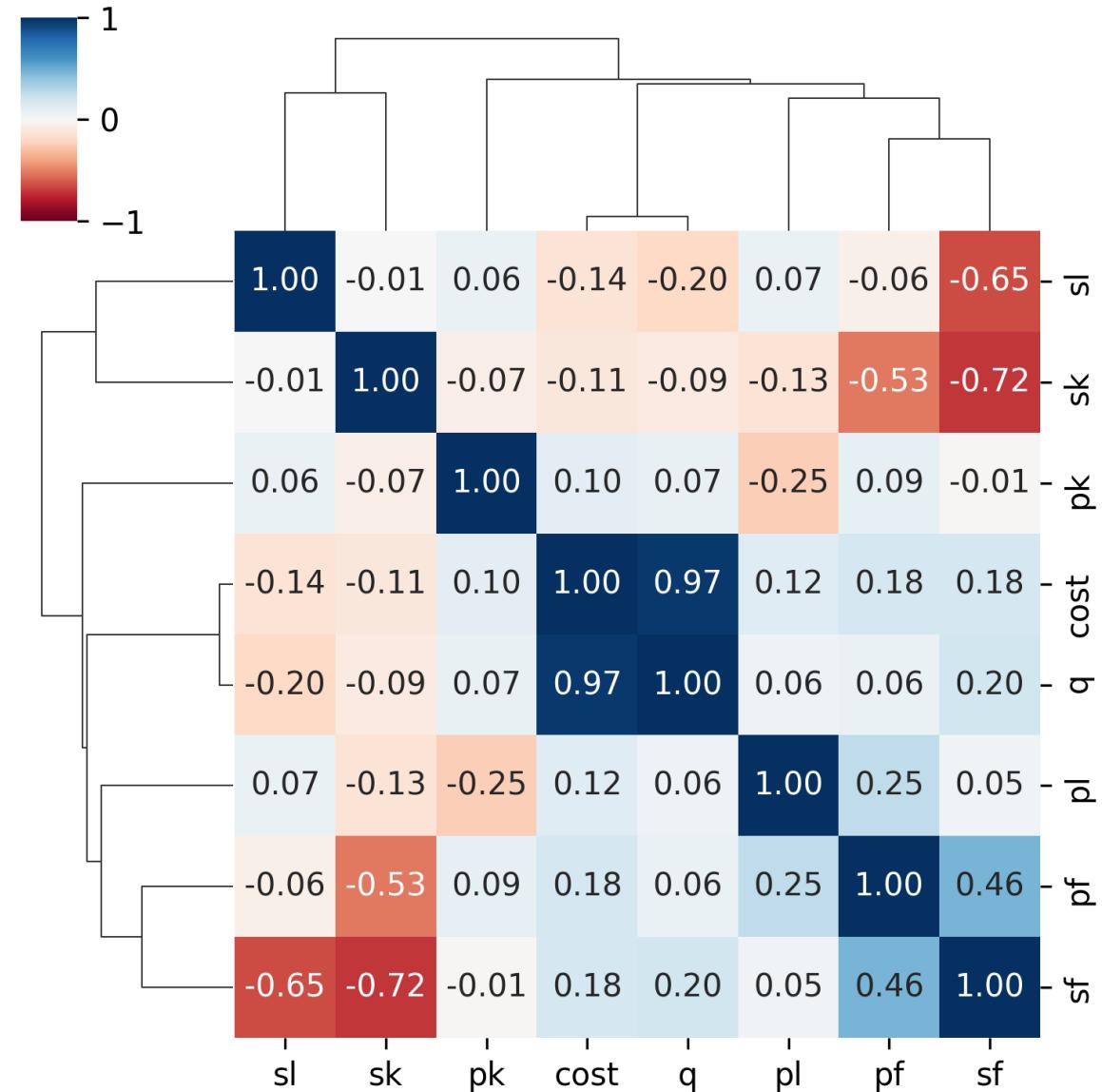
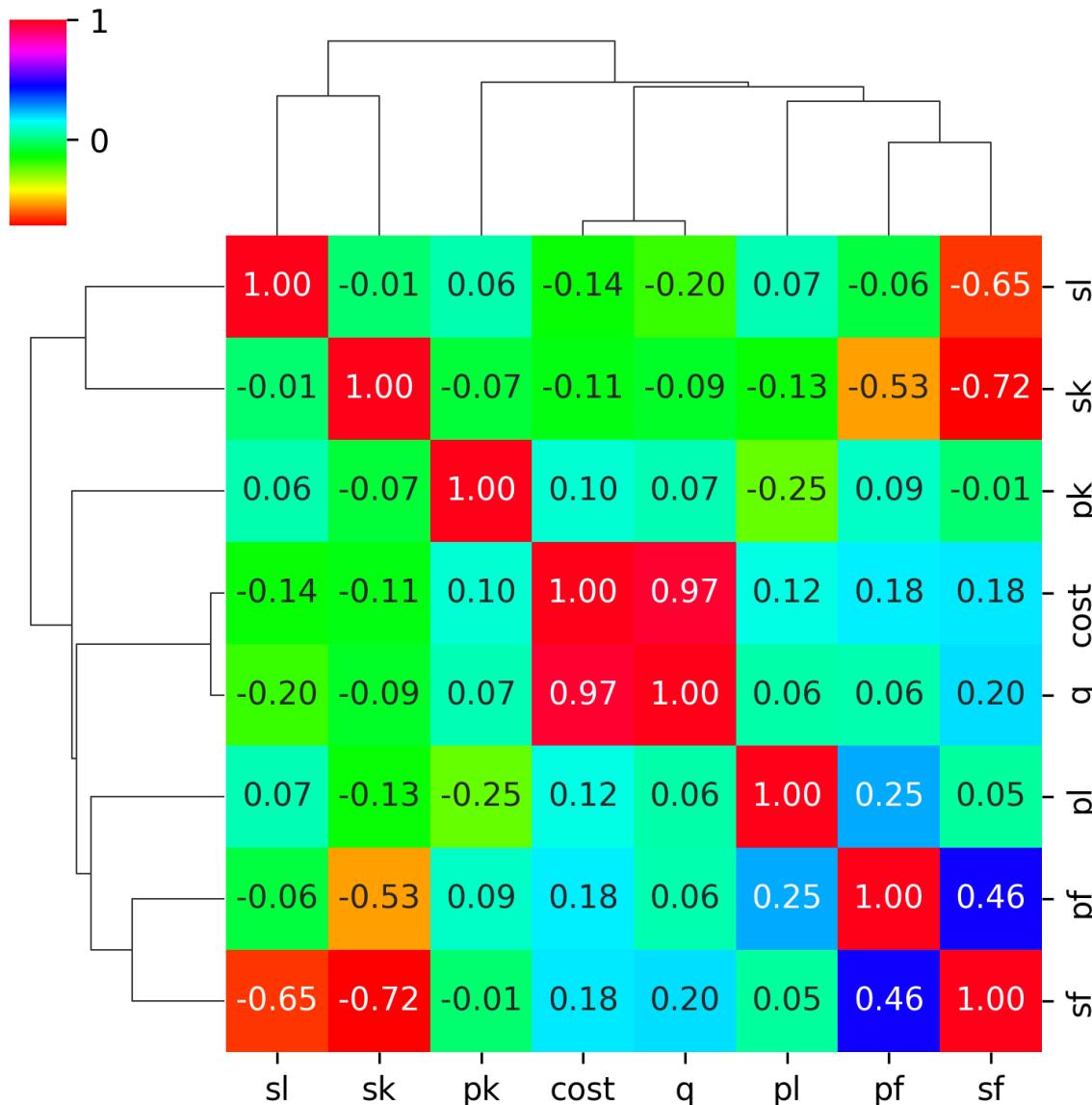
Thicker lines in the legend



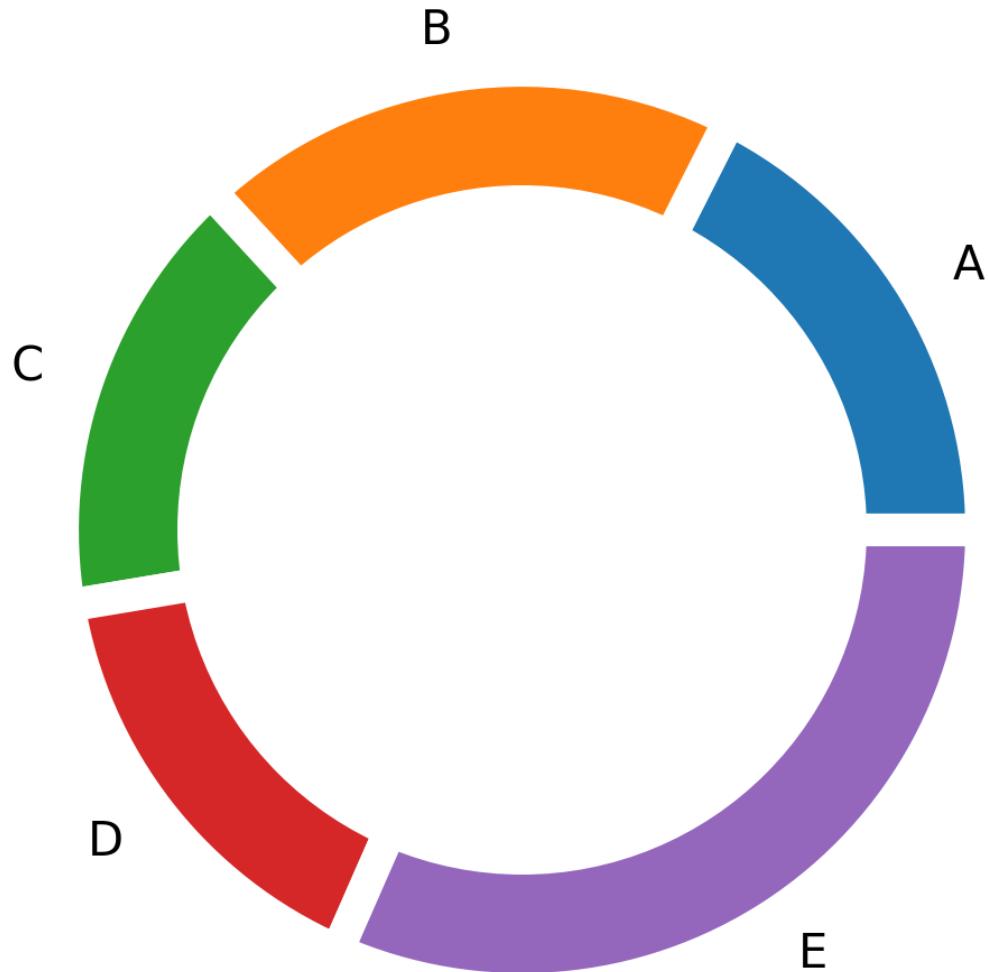
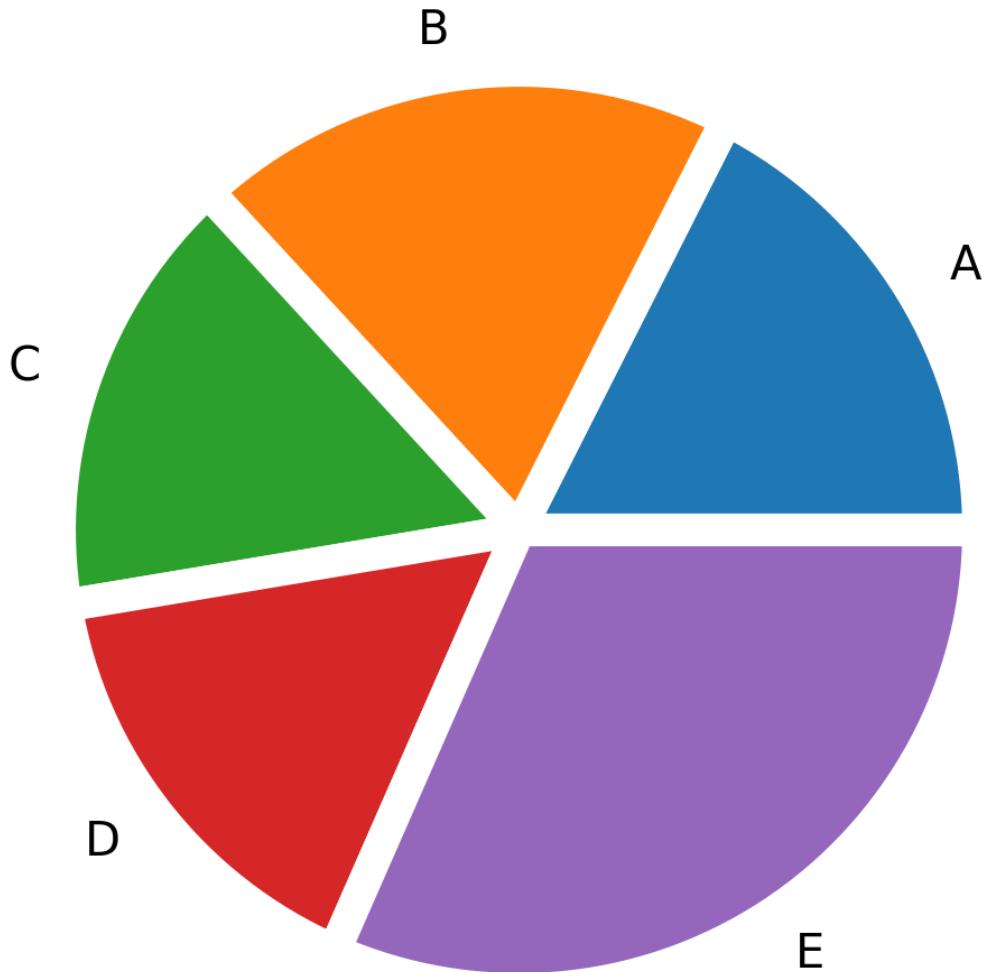
Legend order



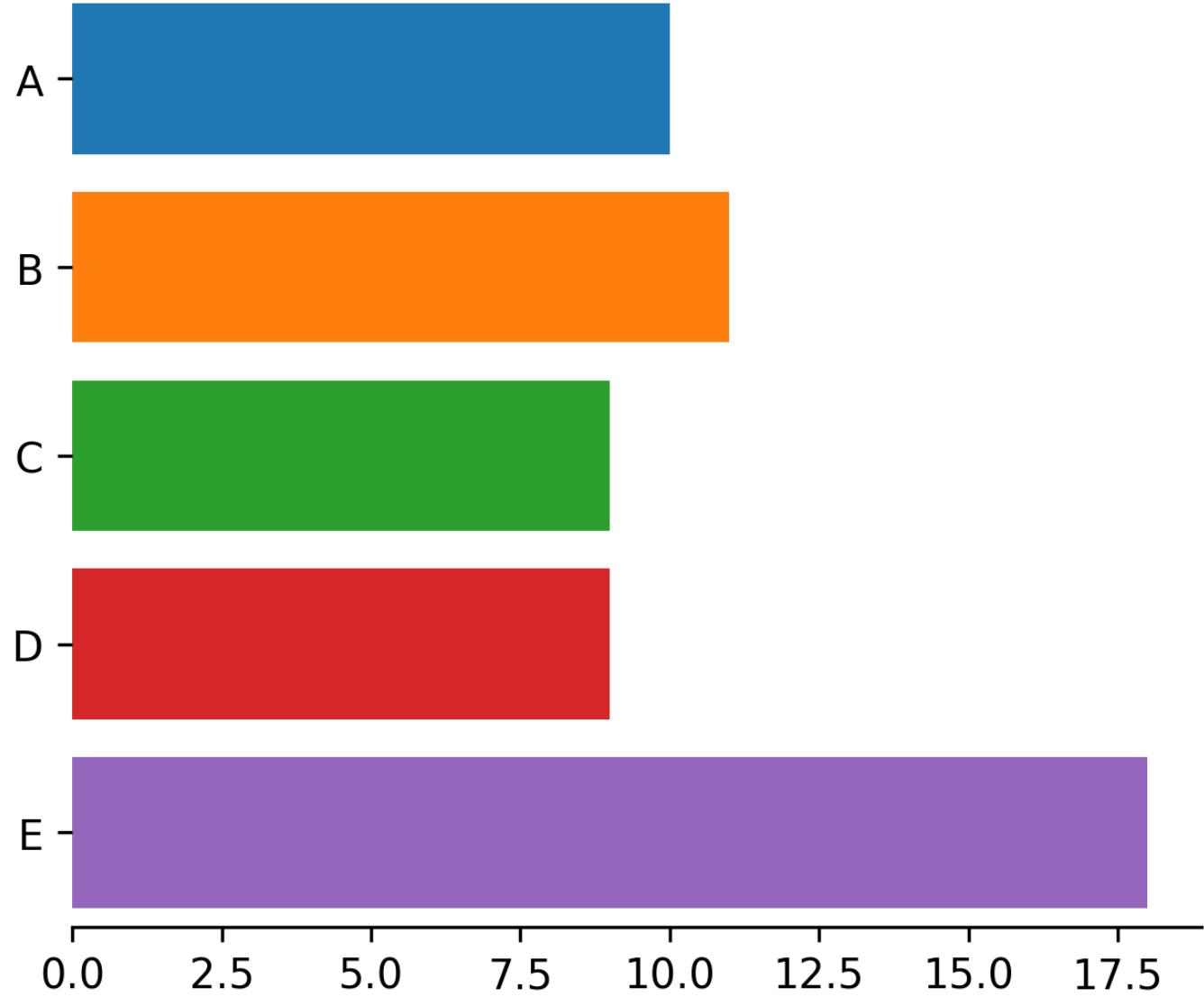
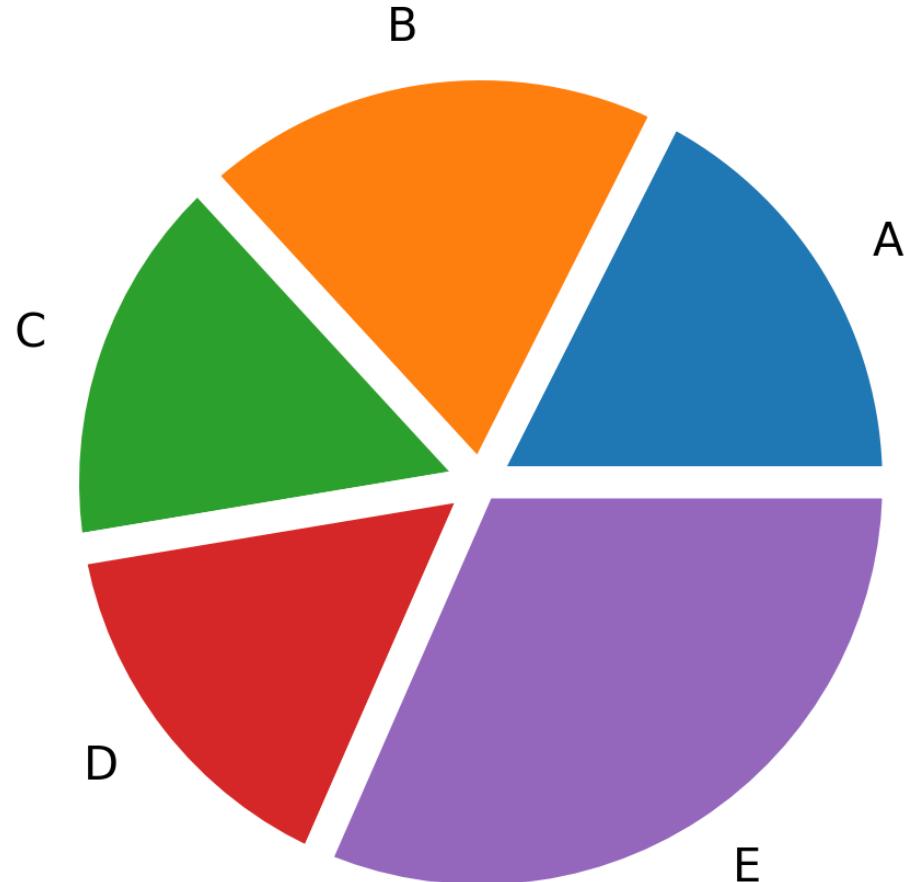
Colour palettes



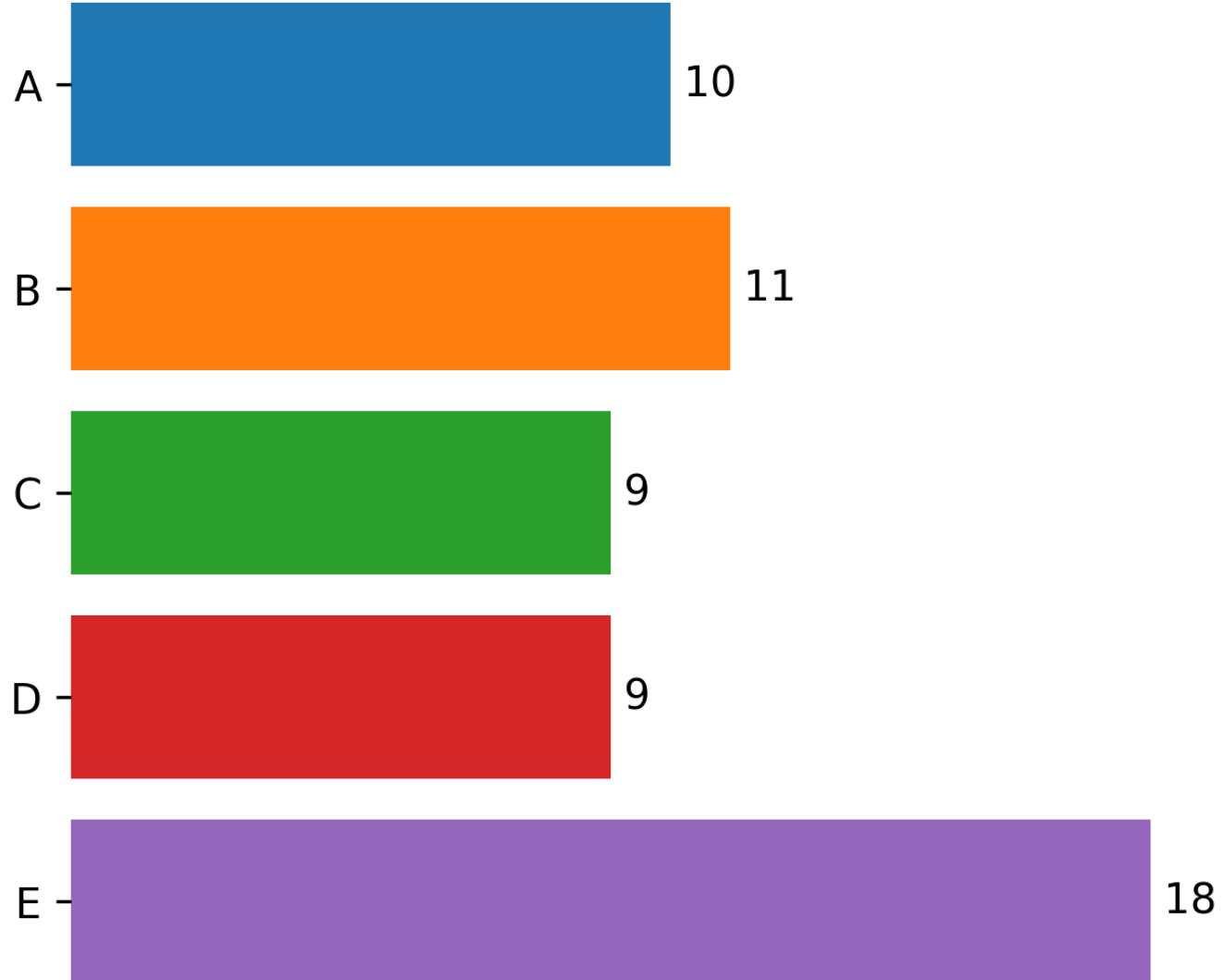
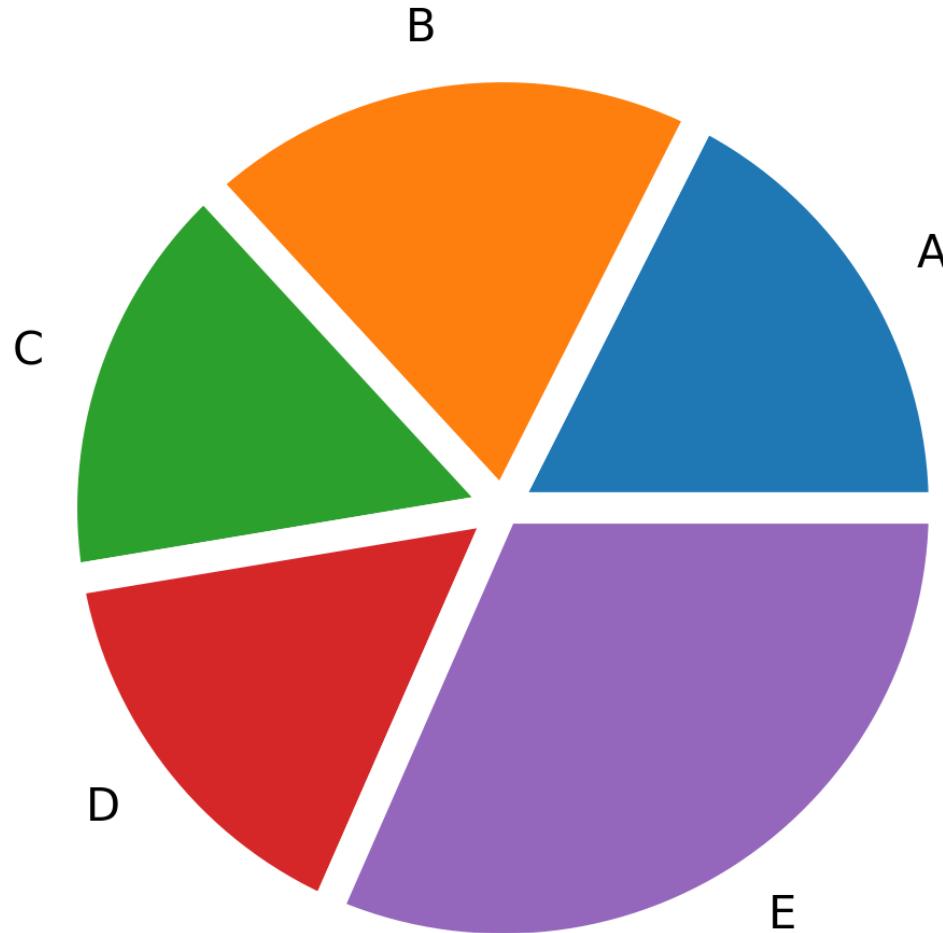
Avoid pie charts



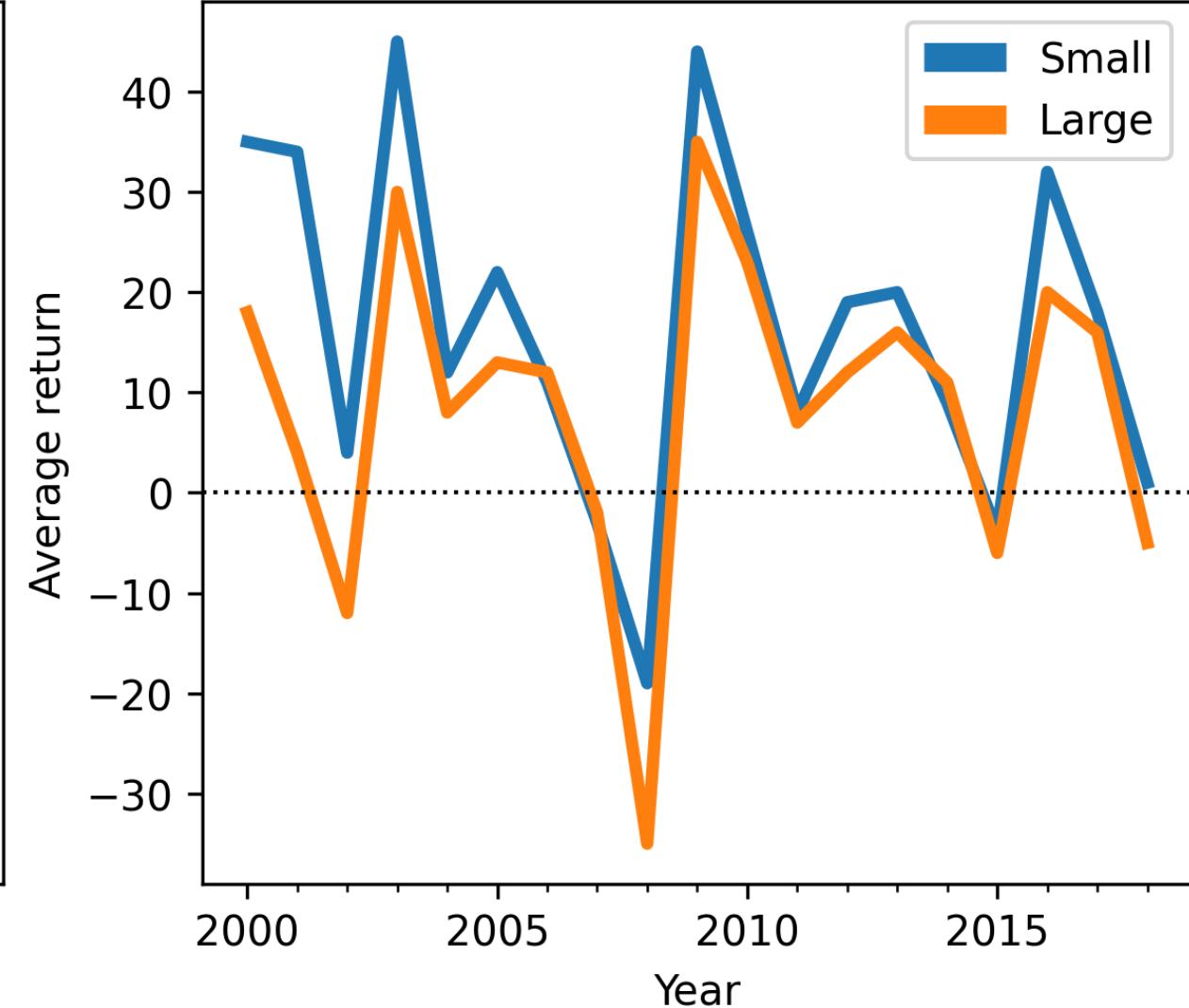
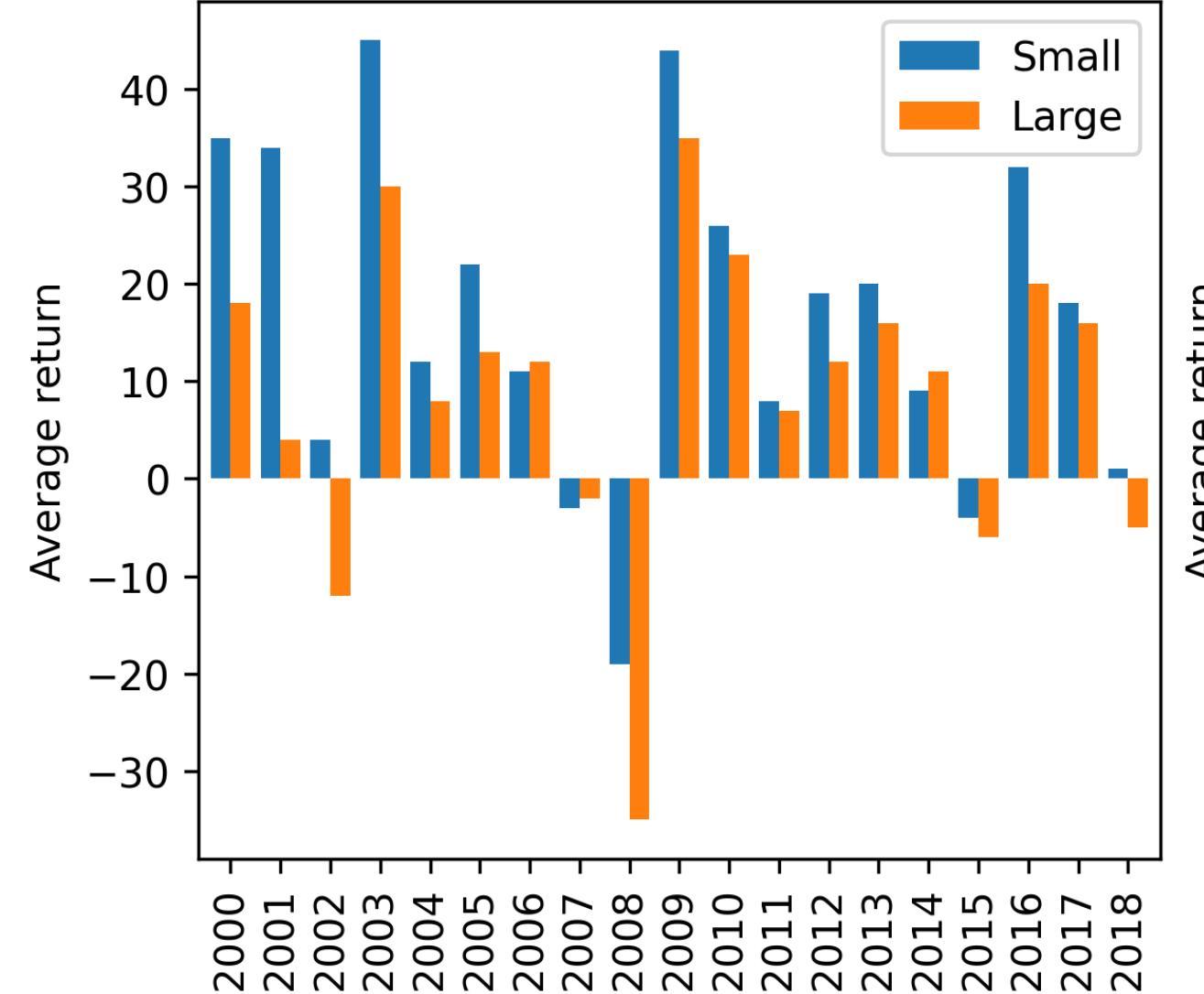
Avoid pie charts



Avoid pie charts



Avoid “Excel plots”



Datasets

old Faithful

waiting

79

54

74

62

85

55

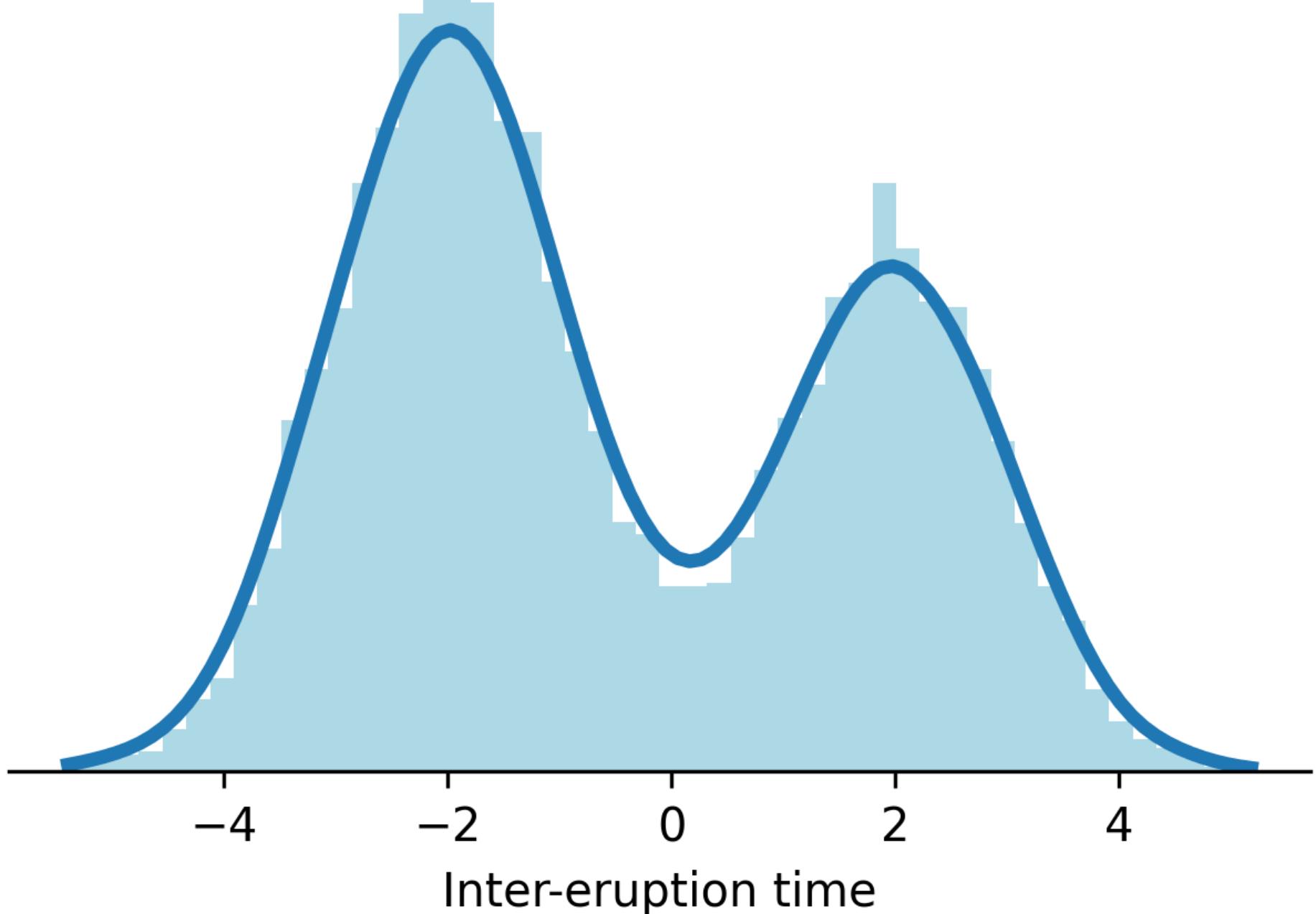
88

85

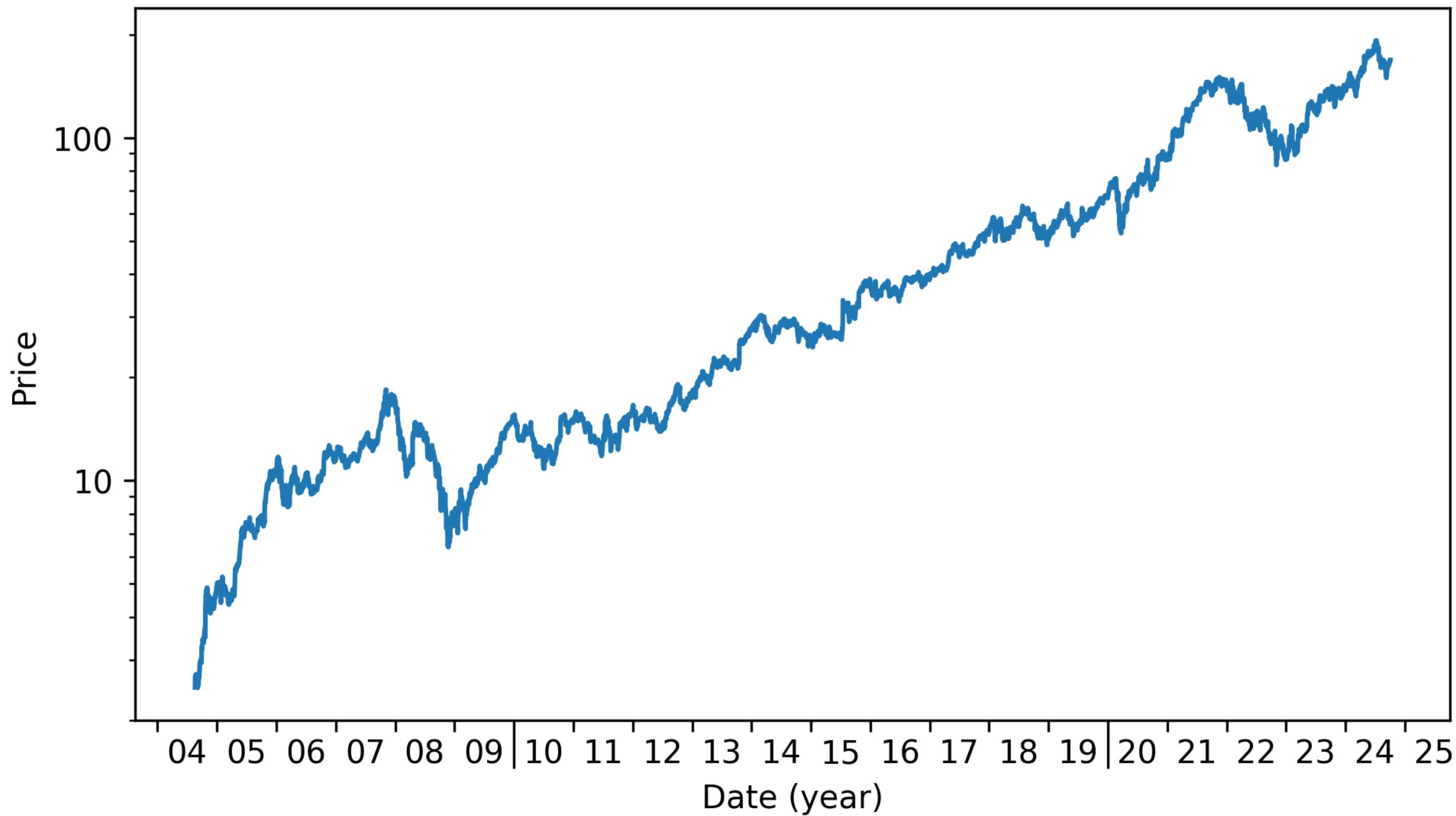
51

85

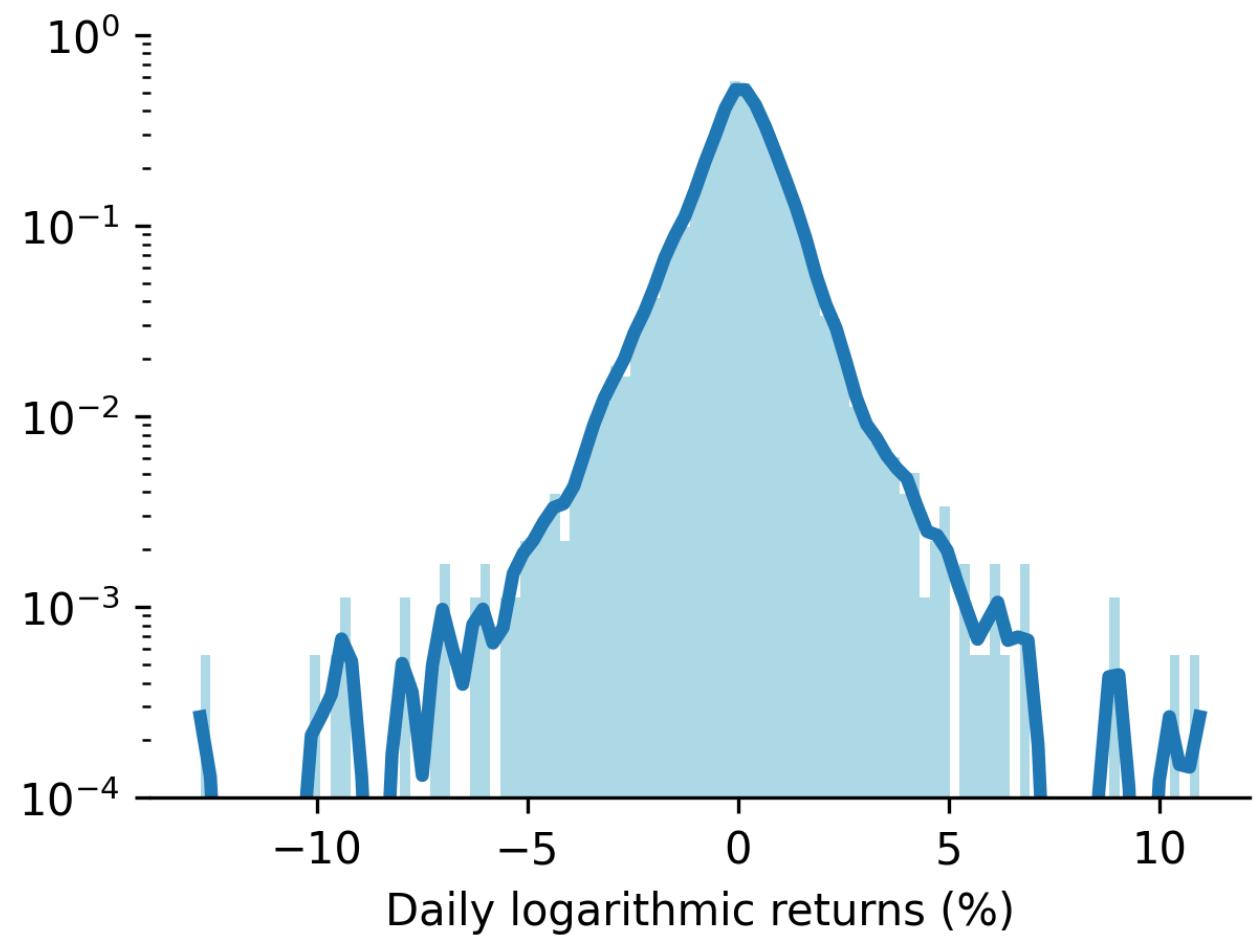
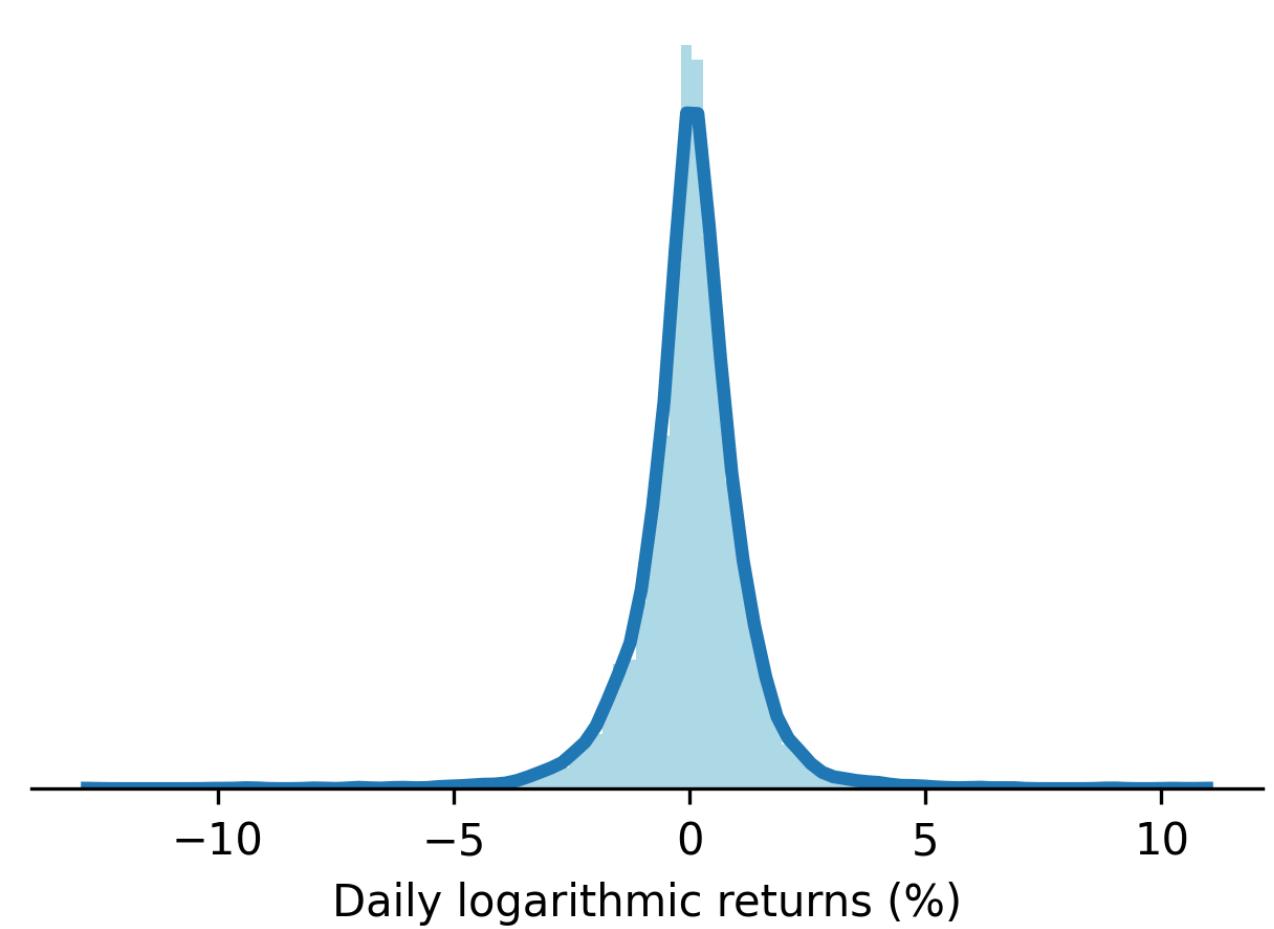
54



Stock Prices

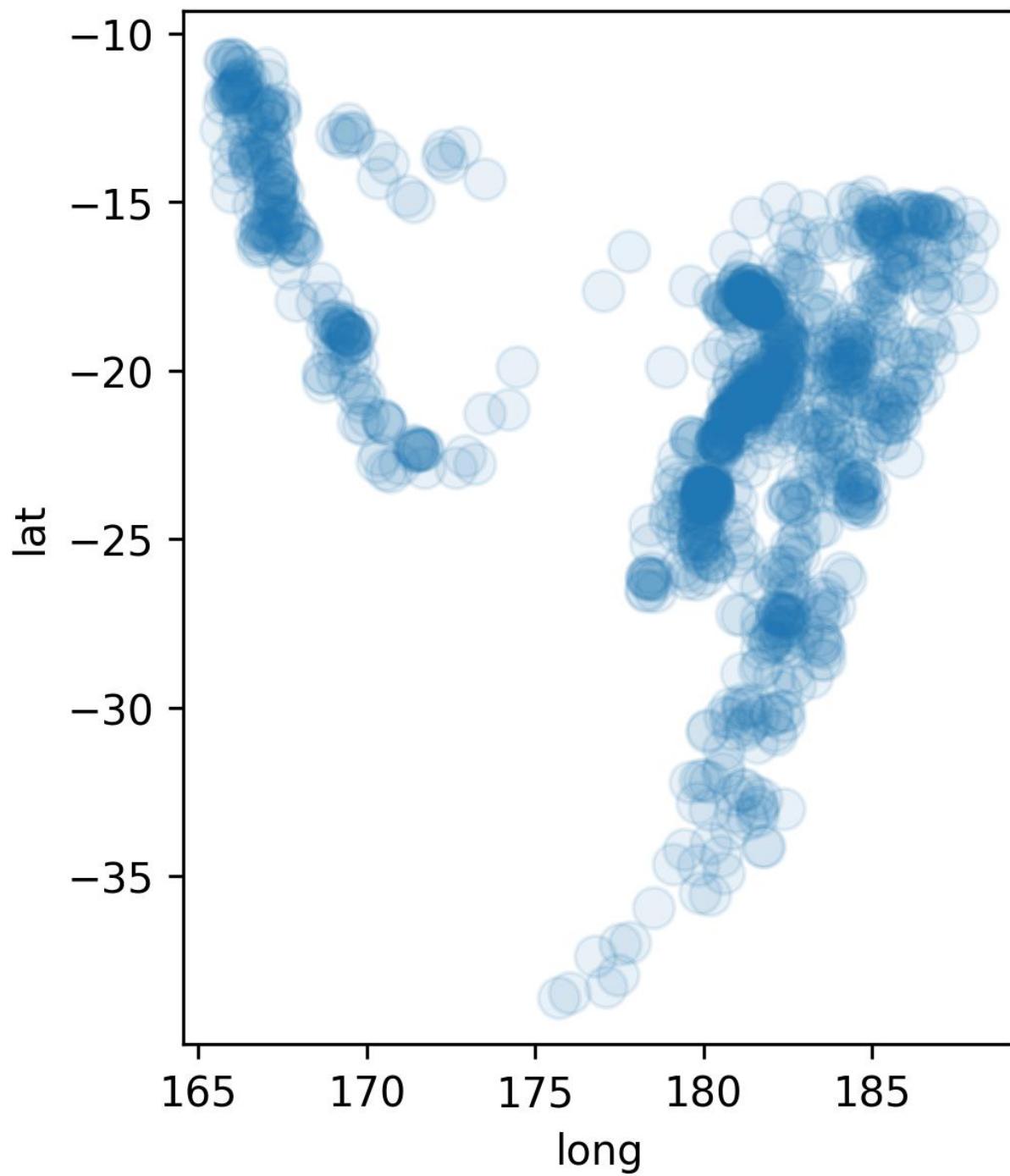


Stock Returns

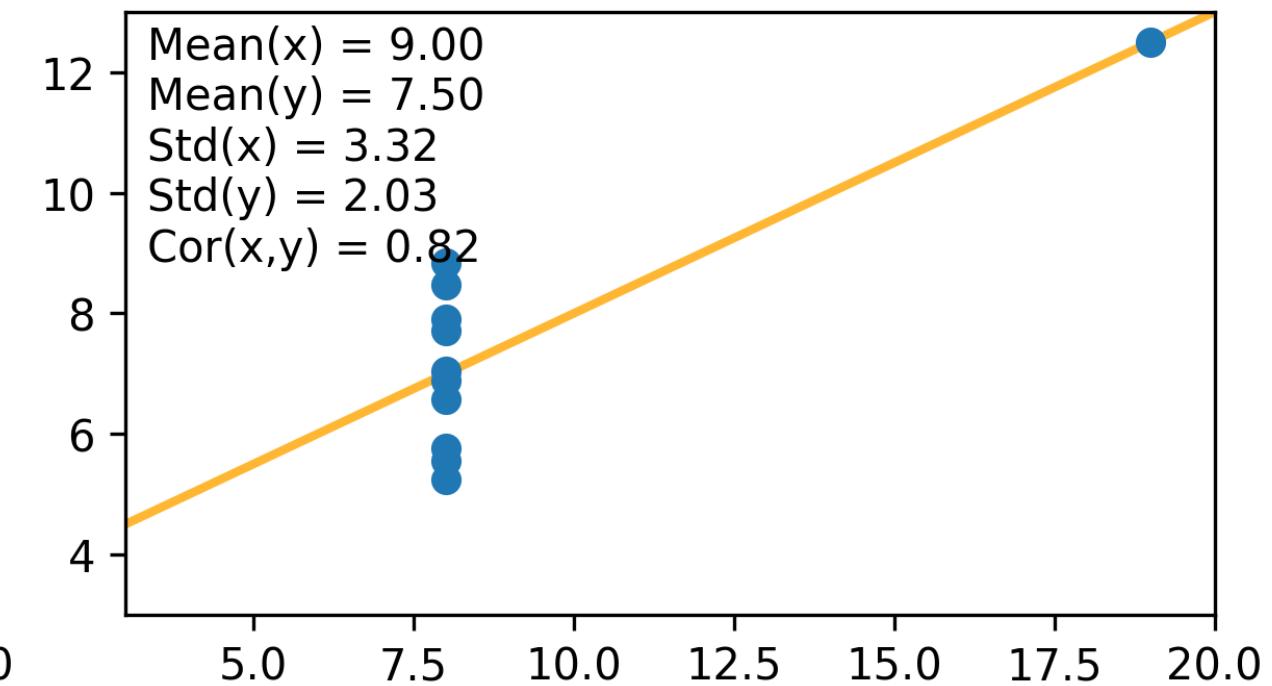
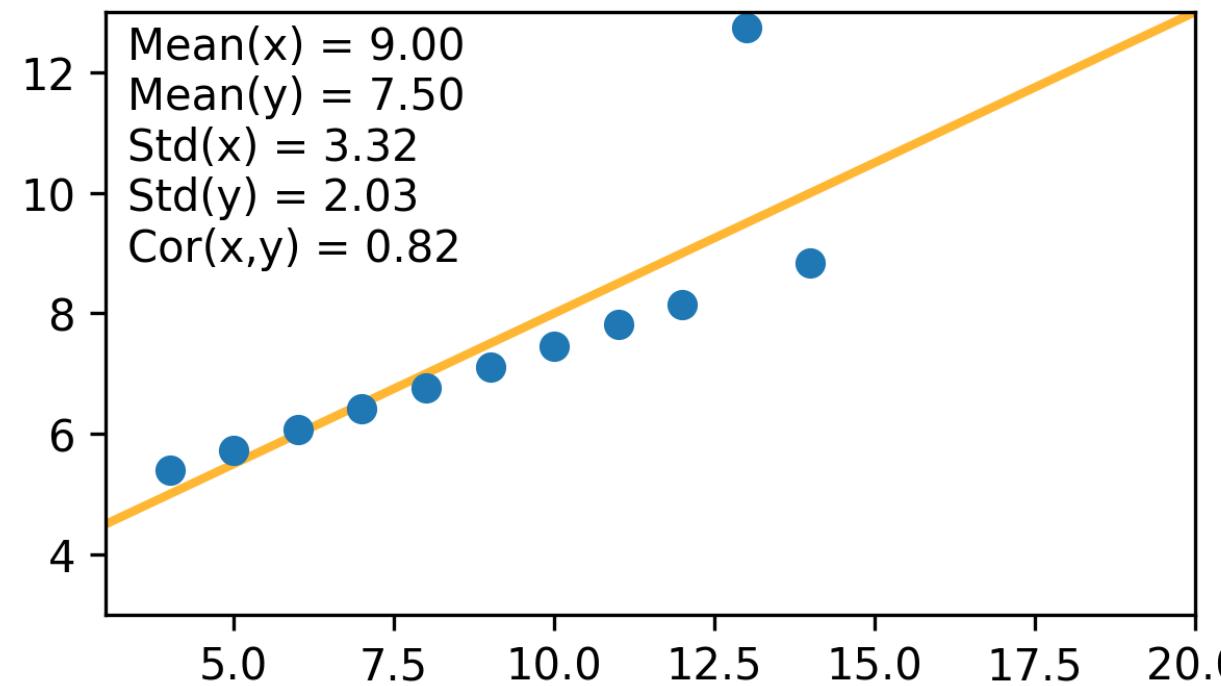
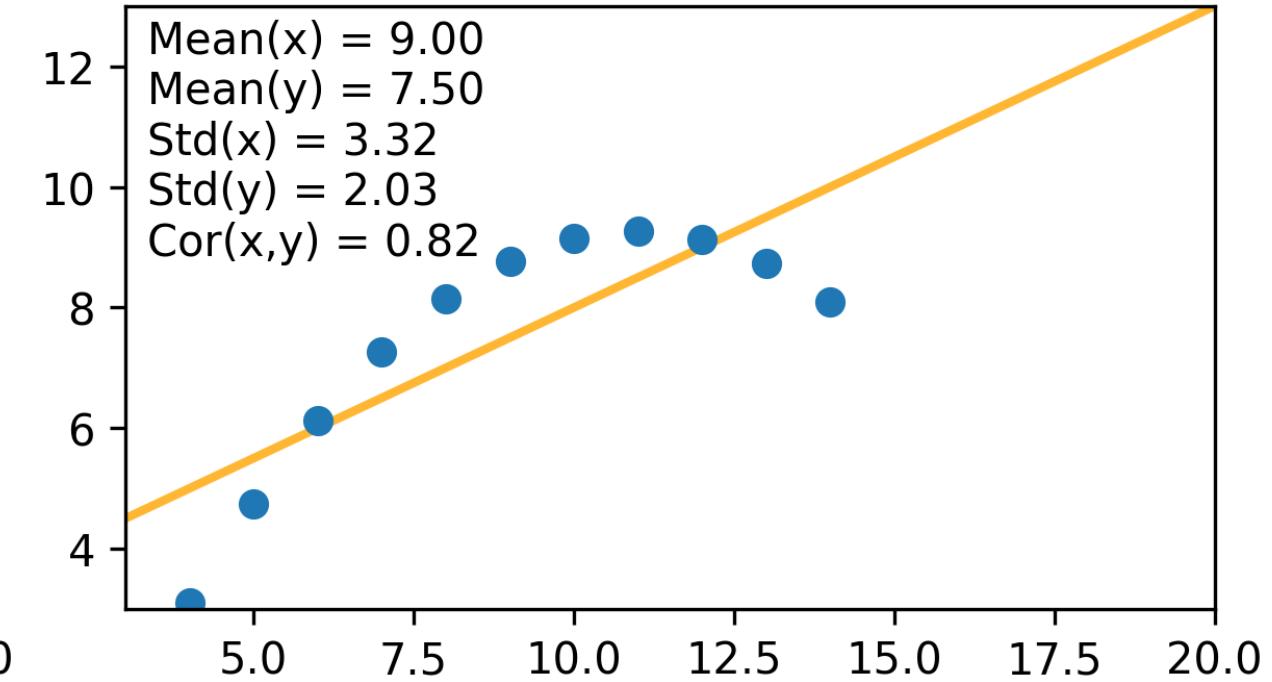
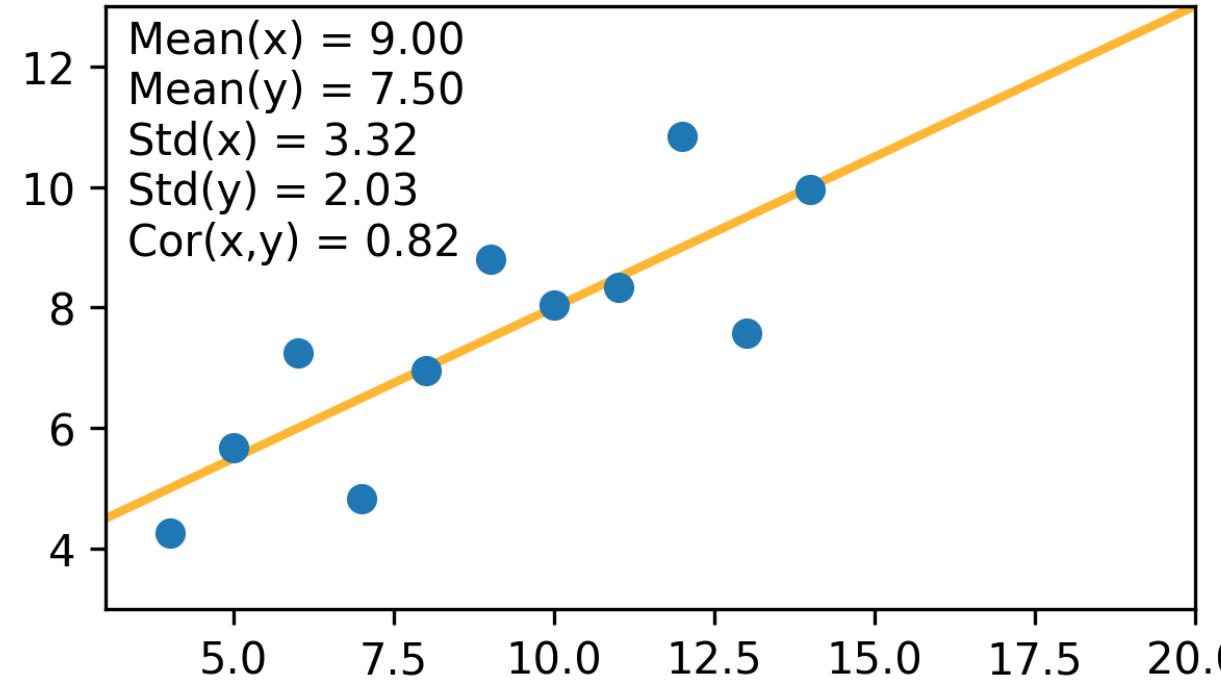


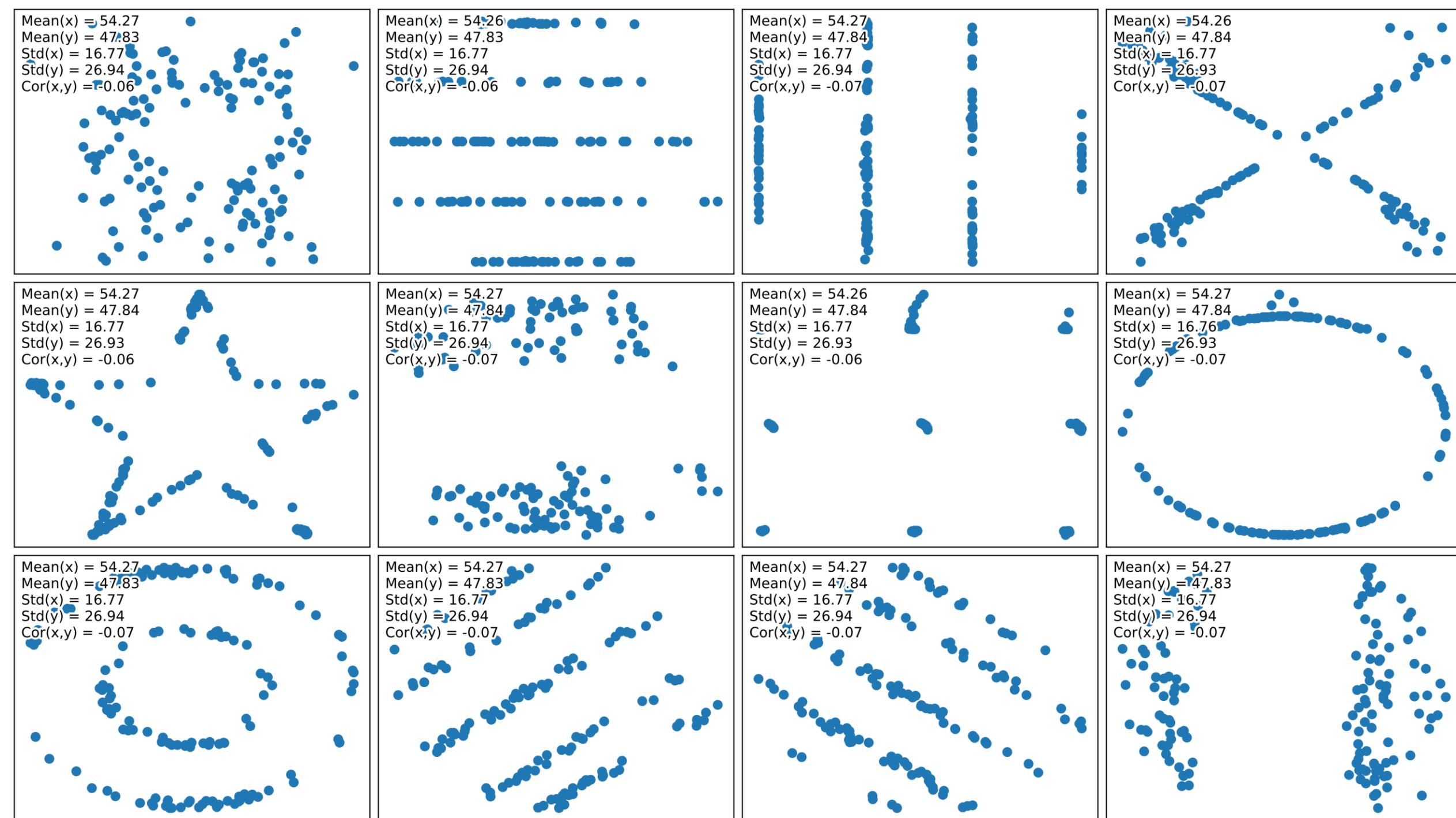
Quakes

lat	long
-20.42	181.62
-20.62	181.03
-26.00	184.10
-17.97	181.66
-20.42	181.96
-19.68	184.31
-11.70	166.10
-28.11	181.93
-28.74	181.74
-17.47	179.59
-21.44	180.69
-12.26	167.00
-18.54	182.11



**Anscombe,
data-saurus**





Diamonds

Clarity

clarity

SI2

SI1

VS1

VS2

SI2

VVS2

VVS1

SI1

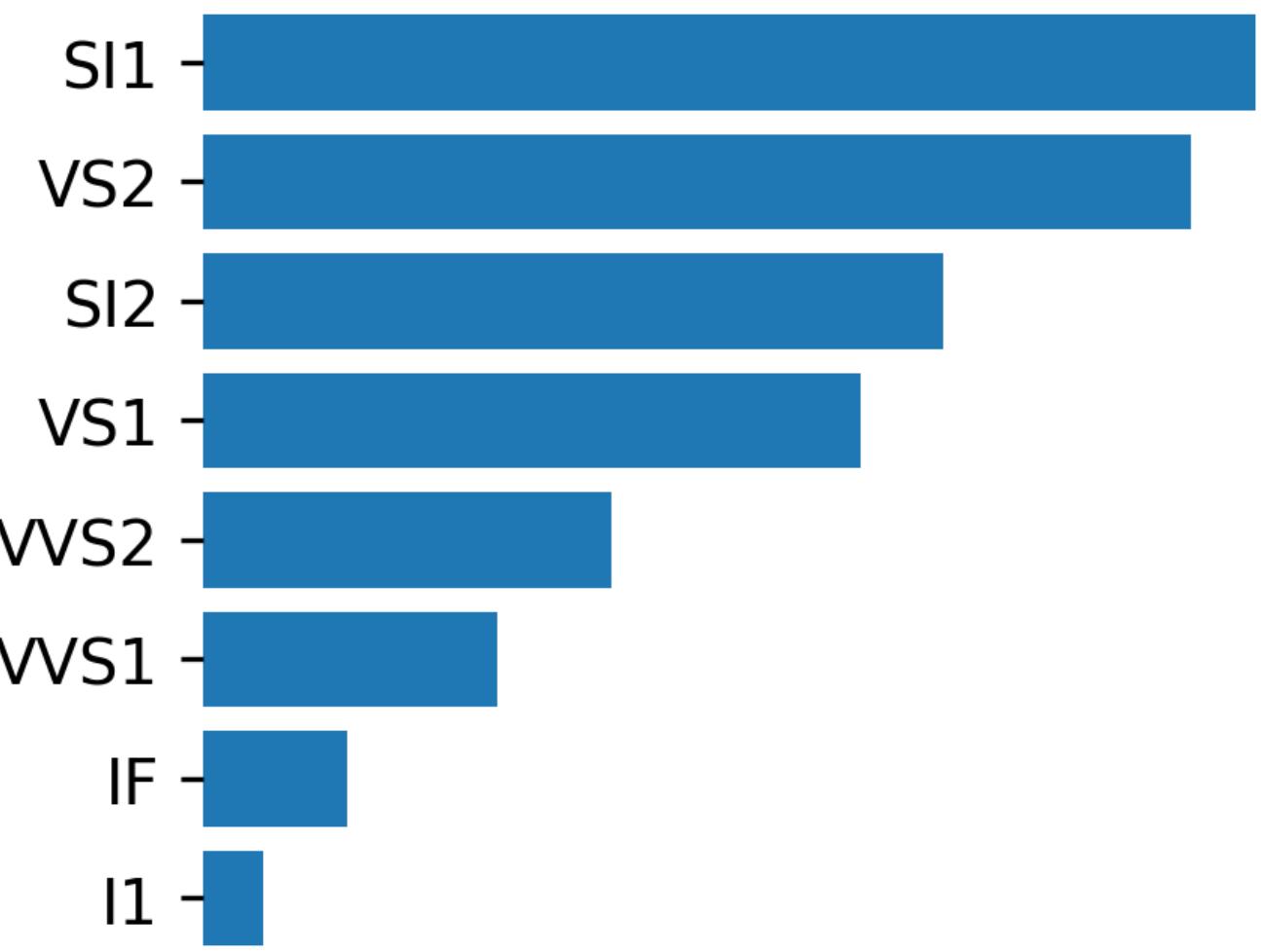
VS2

VS1

SI1

VS1

SI1

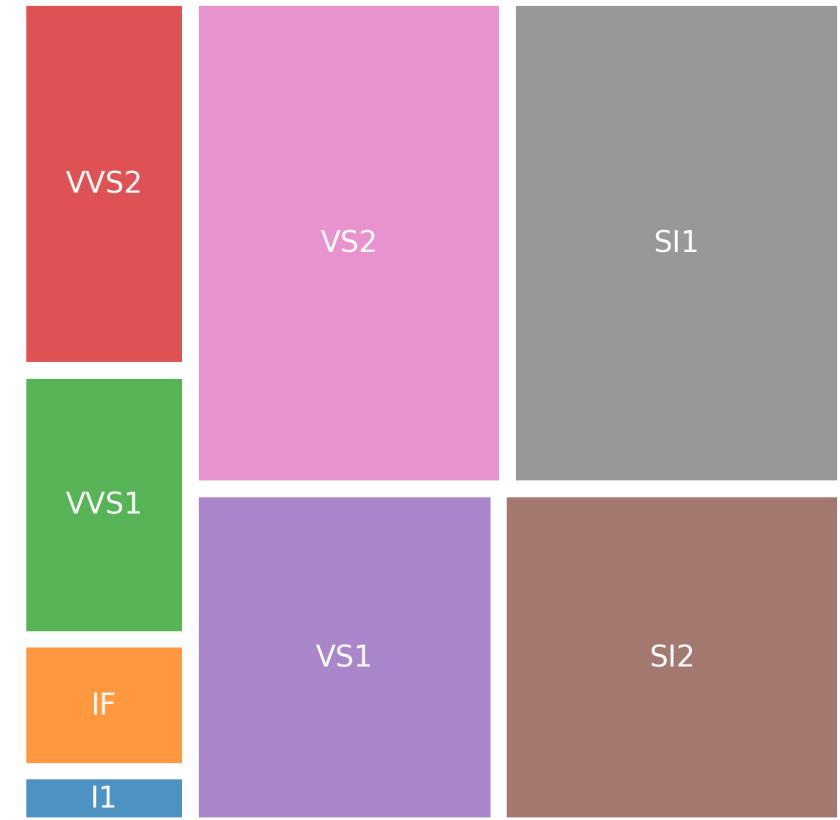
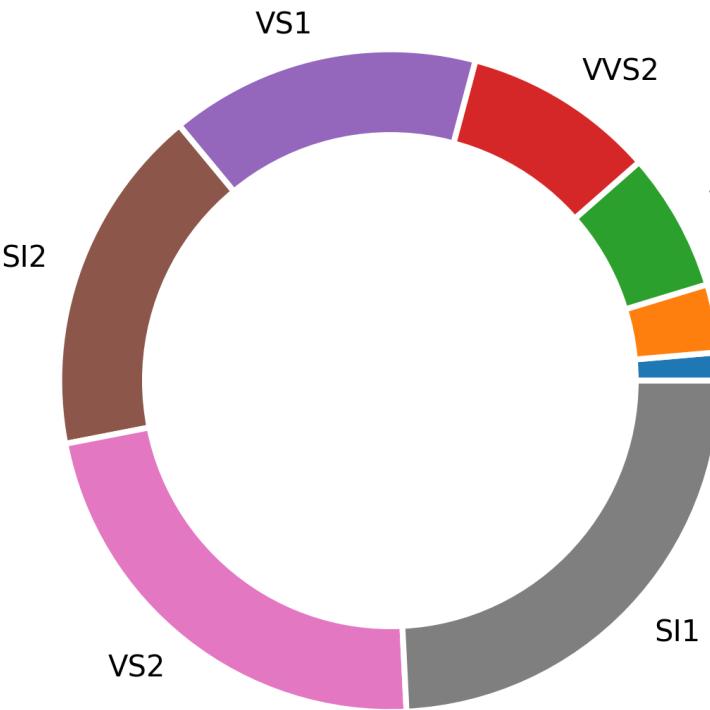
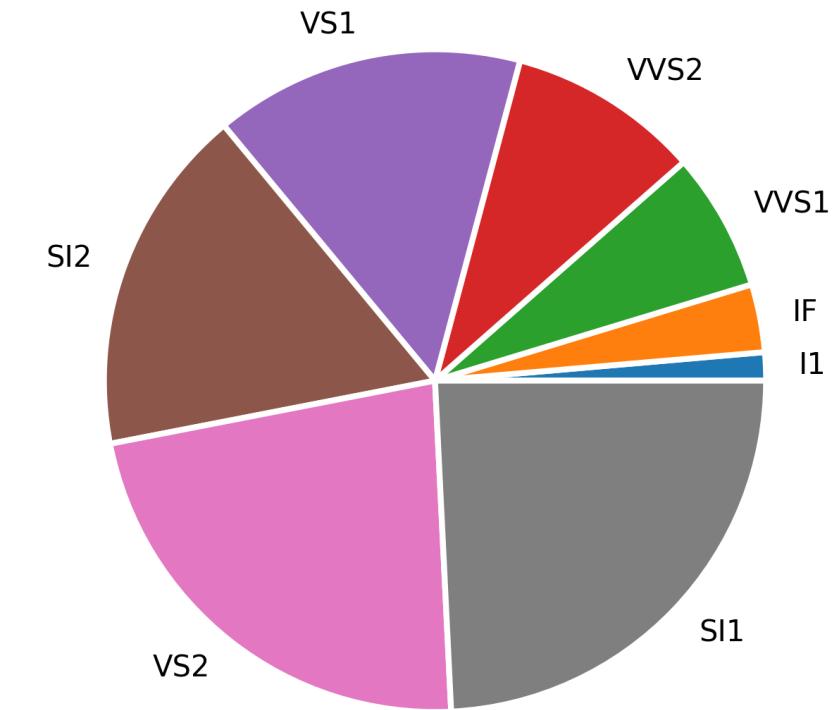


0

5000

10000

Count



clarity color

SI2 E

SI1 E

VS1 E

VS2 I

SI2 J

VVS2 J

VVS1 I

SI1 H

VS2 E

VS1 H

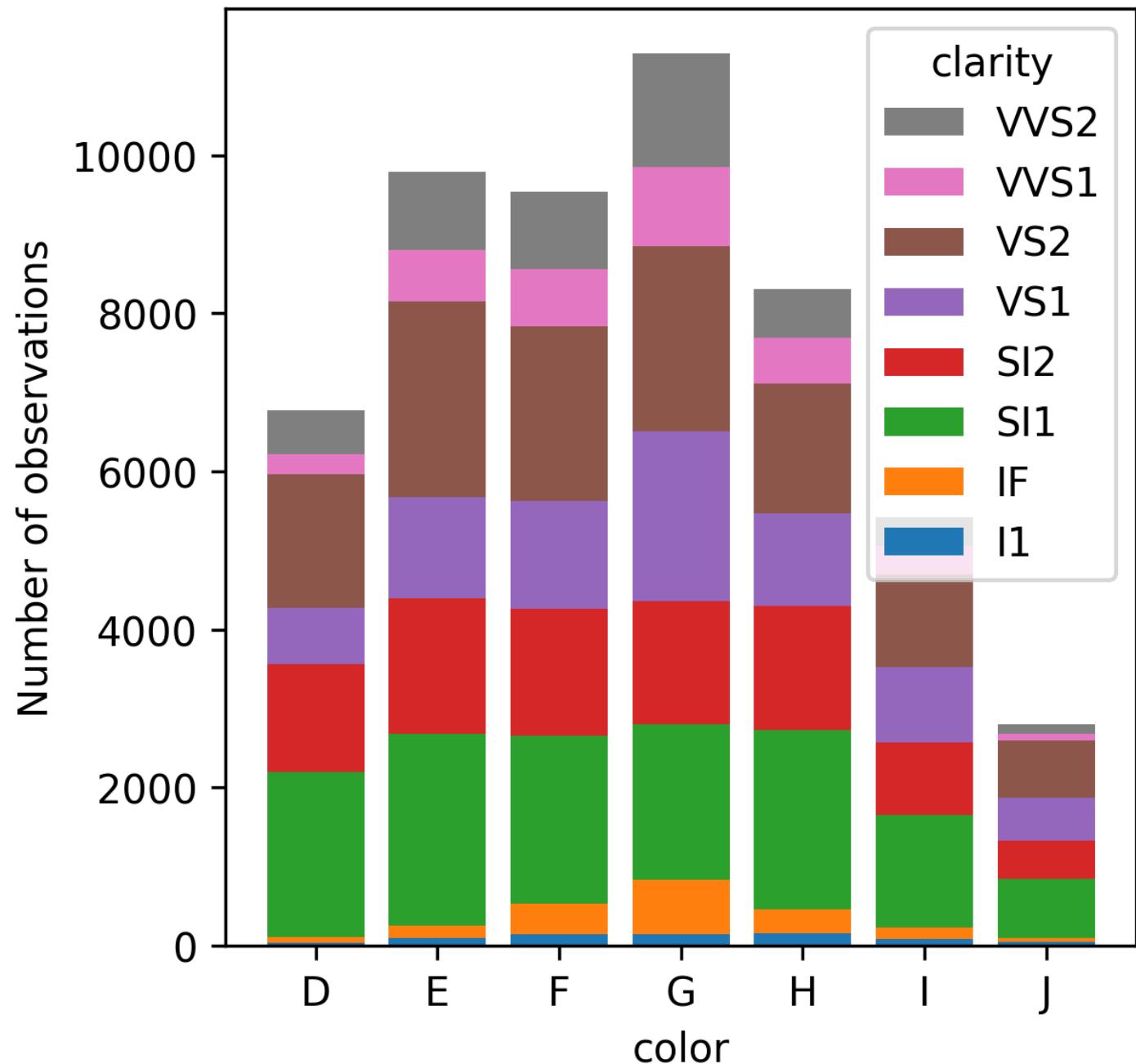
SI1 J

VS1 J

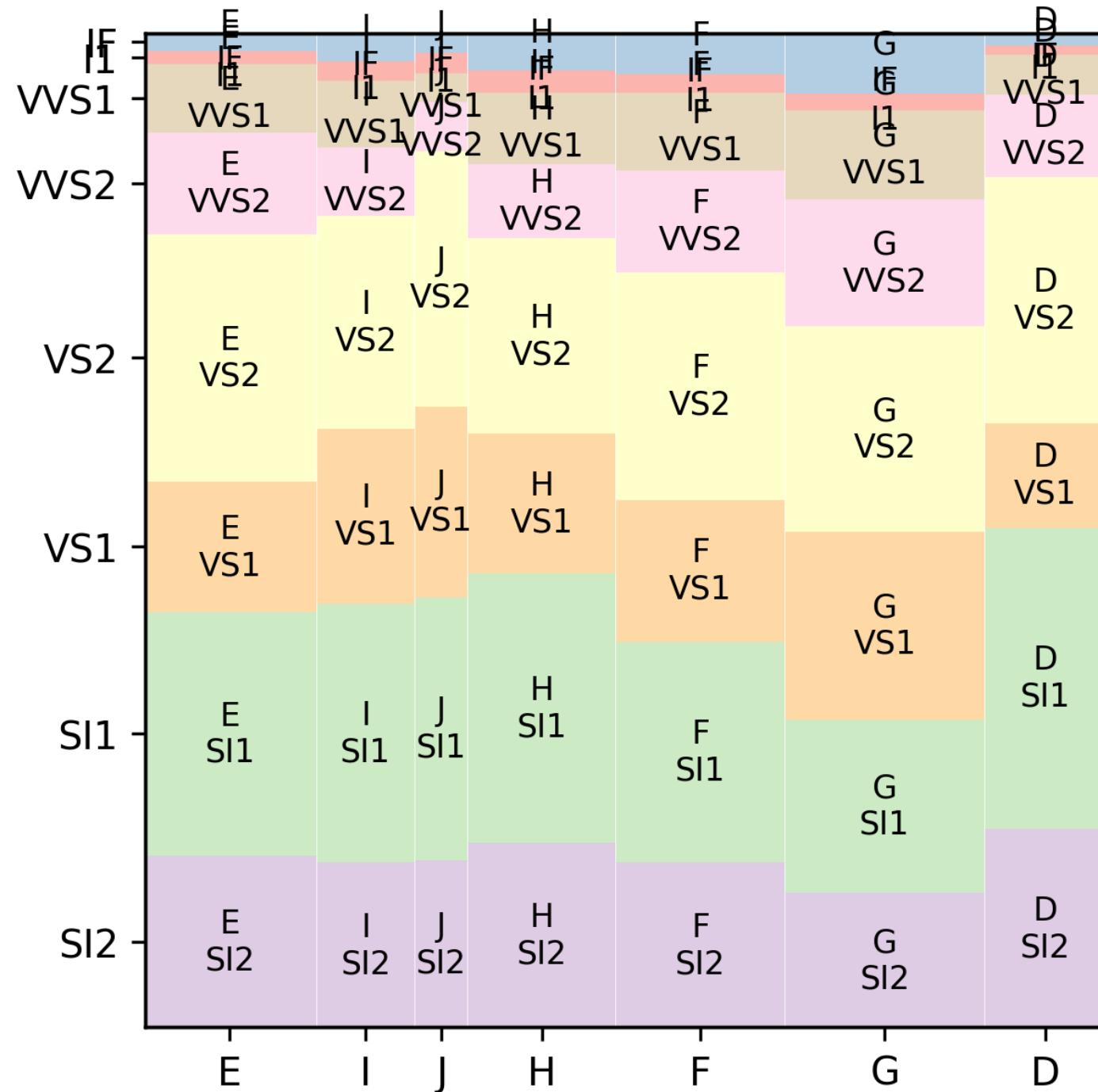
SI1 F

Diamond clarity and colour

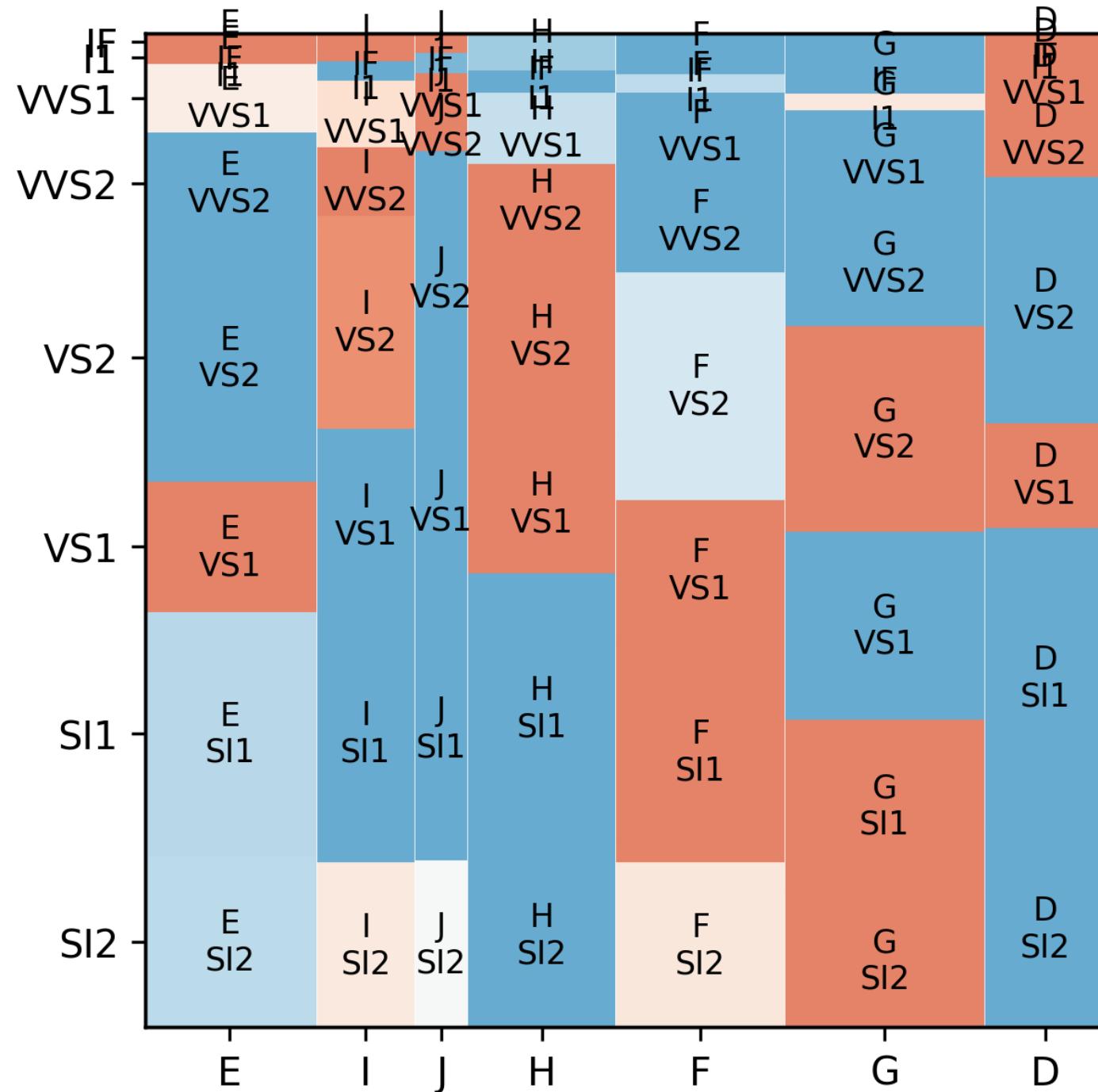
clarity	I1	IF	SI1	SI2	VS1	VS2	VVS1	VVS2
color								
D	42	73	2083	1370	705	1697	252	553
E	102	158	2426	1713	1281	2470	656	991
F	143	385	2131	1609	1364	2201	734	975
G	150	681	1976	1548	2148	2347	999	1443
H	162	299	2275	1563	1169	1643	585	608
I	92	143	1424	912	962	1169	355	365
J	50	51	750	479	542	731	74	131



clarity	I1	IF	SI1	SI2	VS1	VS2	VVS1	VVS2
color								
D	42	73	2083	1370	705	1697	252	553
E	102	158	2426	1713	1281	2470	656	991
F	143	385	2131	1609	1364	2201	734	975
G	150	681	1976	1548	2148	2347	999	1443
H	162	299	2275	1563	1169	1643	585	608
I	92	143	1424	912	962	1169	355	365
J	50	51	750	479	542	731	74	131



clarity	I1	IF	SI1	SI2	VS1	VS2	VVS1	VVS2
color								
D	42	73	2083	1370	705	1697	252	553
E	102	158	2426	1713	1281	2470	656	991
F	143	385	2131	1609	1364	2201	734	975
G	150	681	1976	1548	2148	2347	999	1443
H	162	299	2275	1563	1169	1643	585	608
I	92	143	1424	912	962	1169	355	365
J	50	51	750	479	542	731	74	131



Words

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversations?"

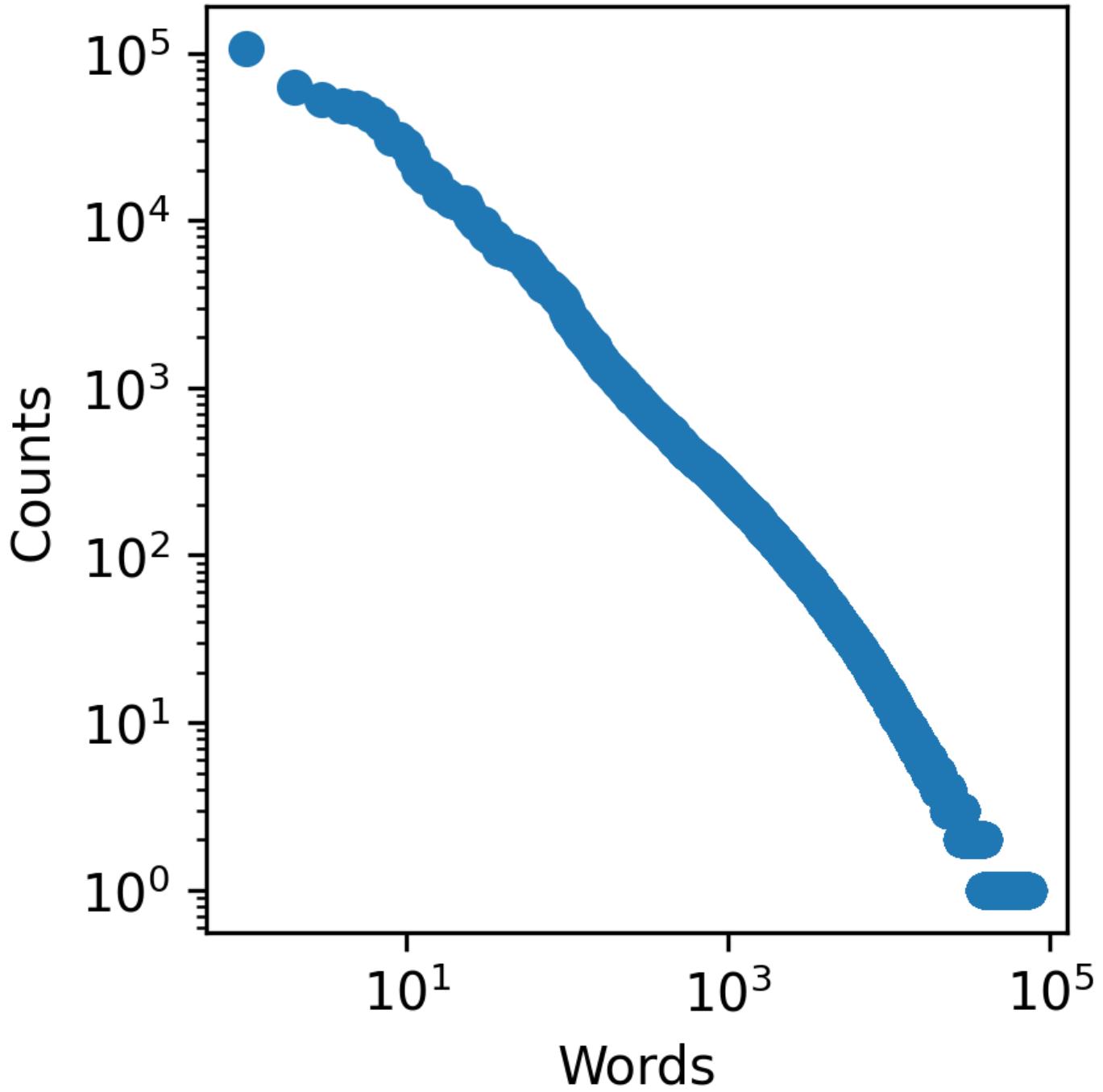
So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

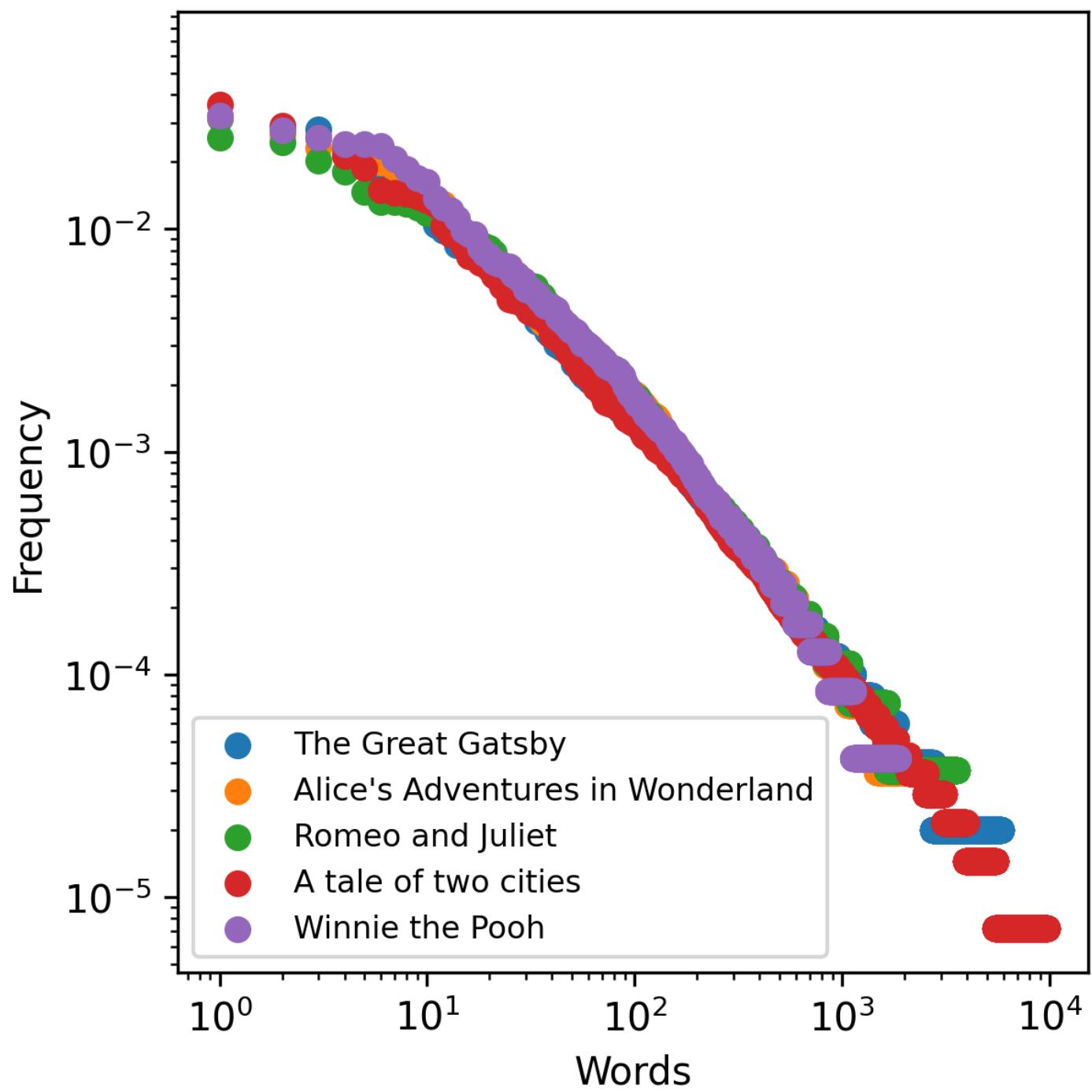
There was nothing so _very_ remarkable in that; nor did Alice think it so _very_ much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually took a watch out of its waistcoat-pocket, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

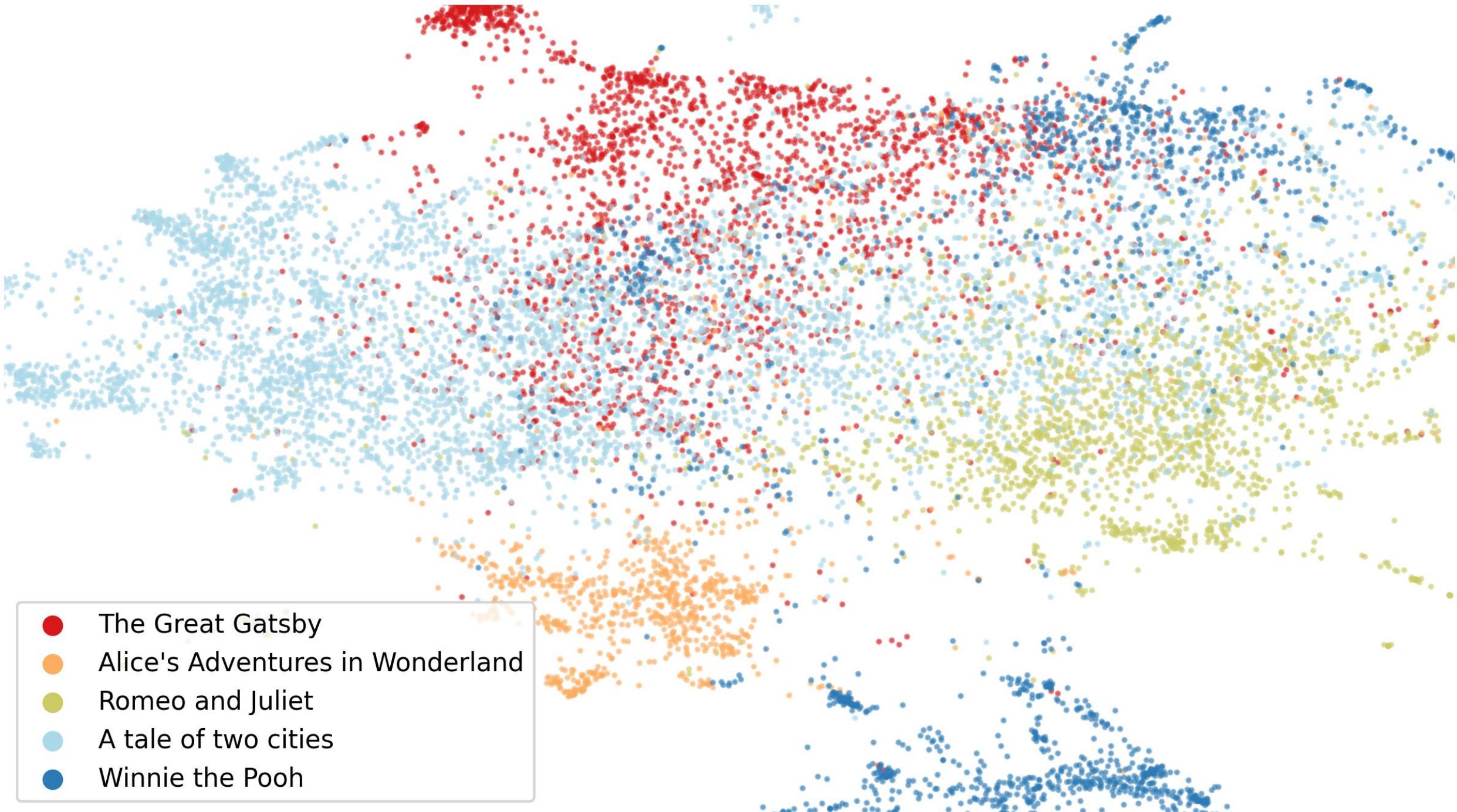
The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

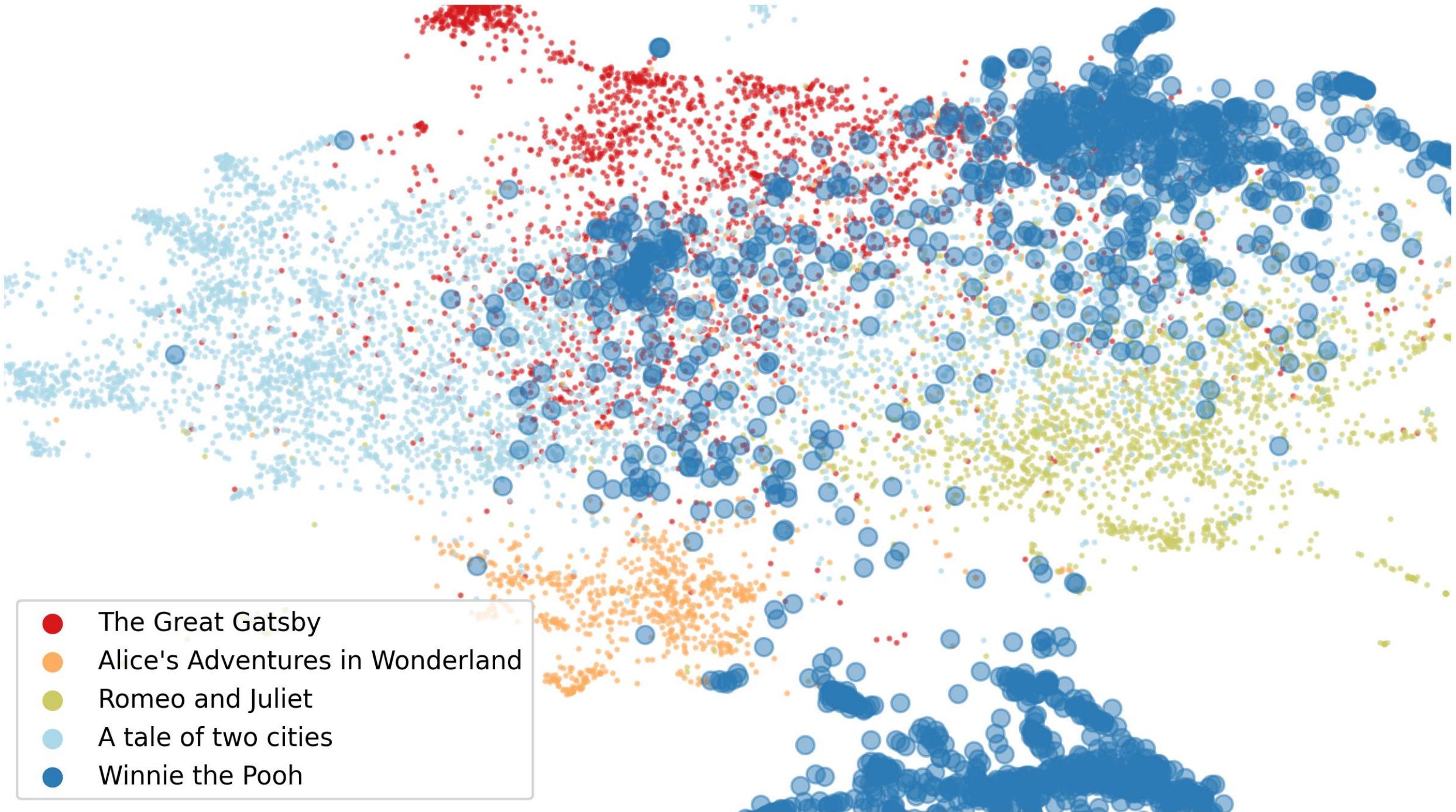
Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and book-shelves; here and there she saw maps and pictures hung upon pegs. She took down a jar from one of the shelves as she passed; it was labelled "ORANGE MARMALADE", but to her great disappointment it was empty: she did not like to drop the jar for fear of killing somebody underneath, so managed to put it into one of the cupboards as she fell past it.





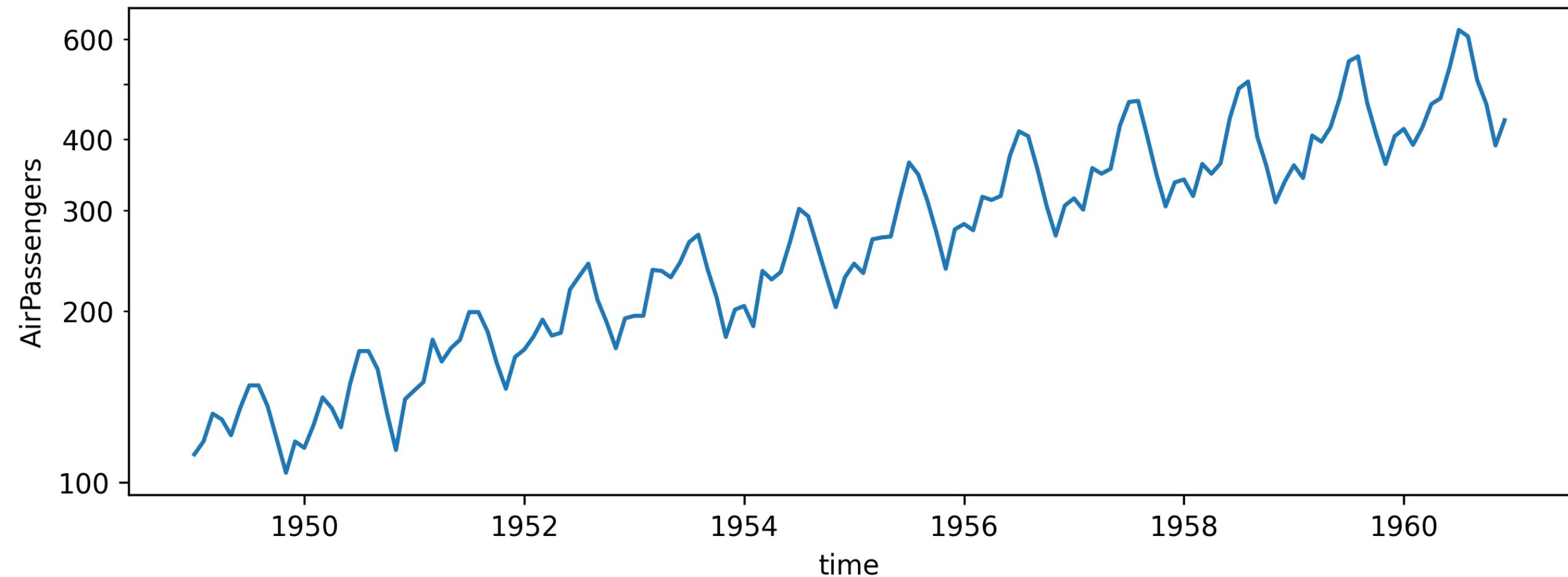
Sentences



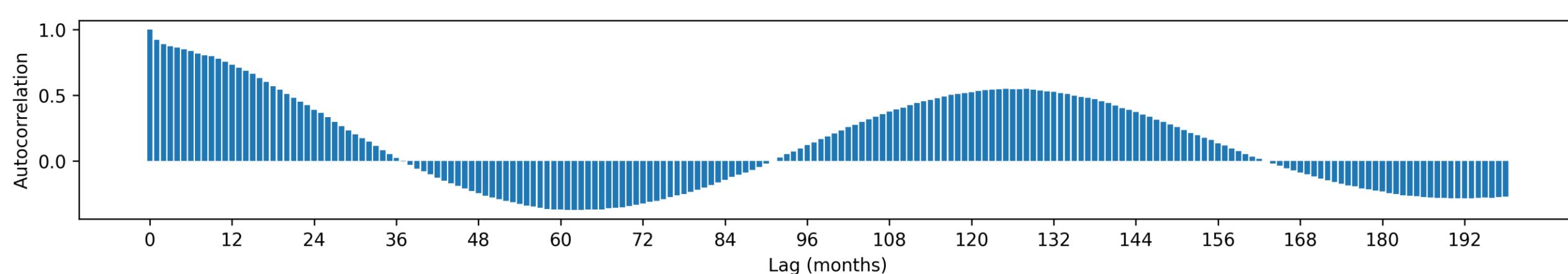
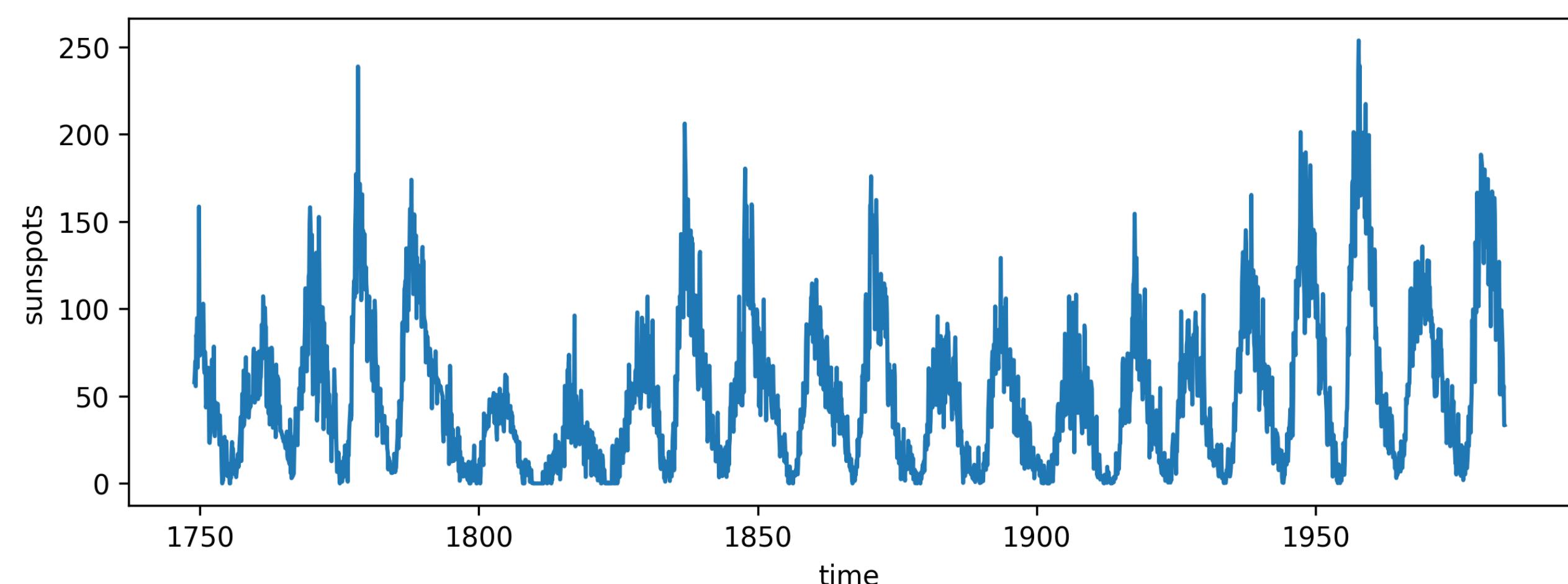


Air

passengers



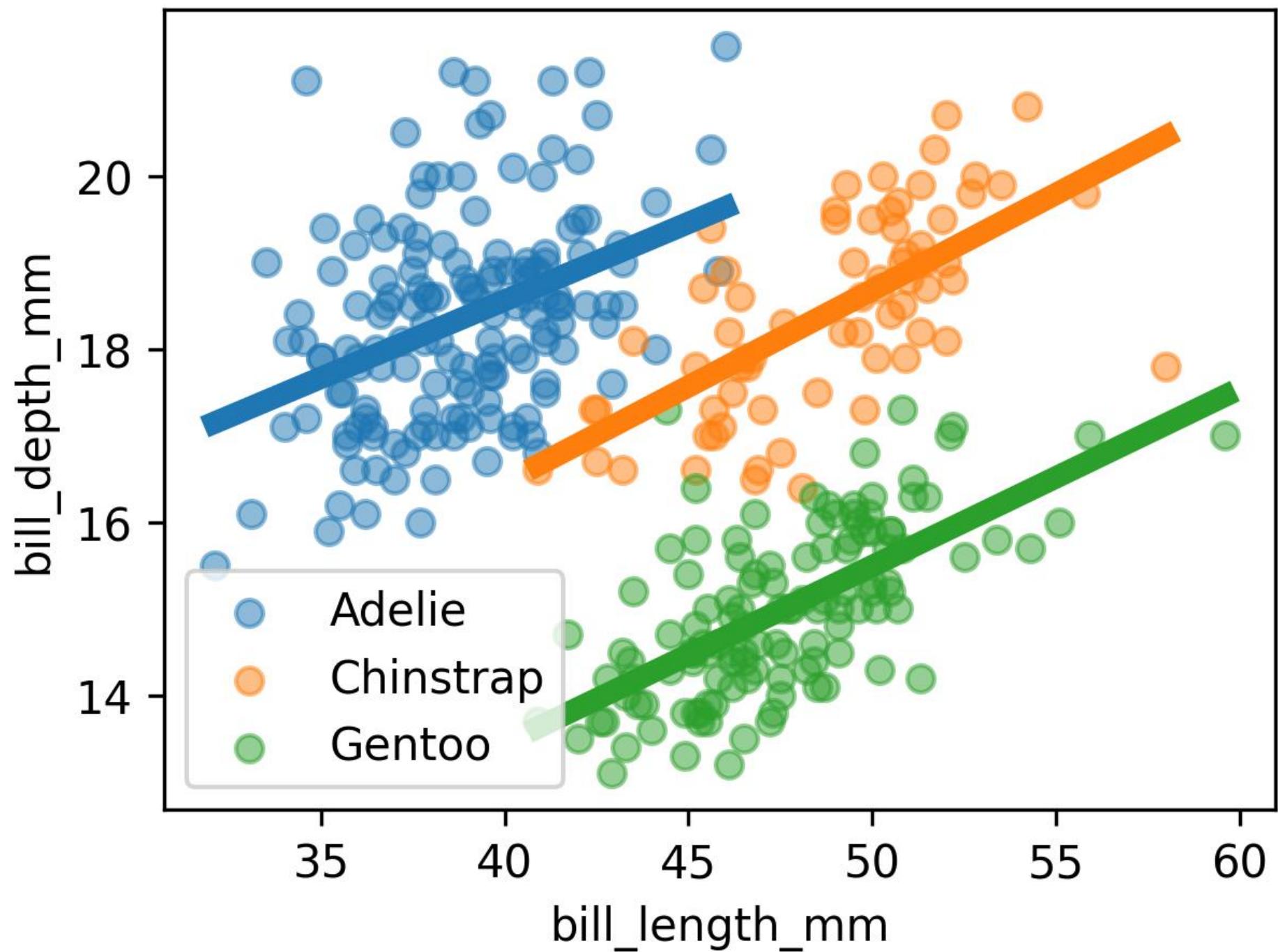
Sunspots



**Palmer
Penguins**



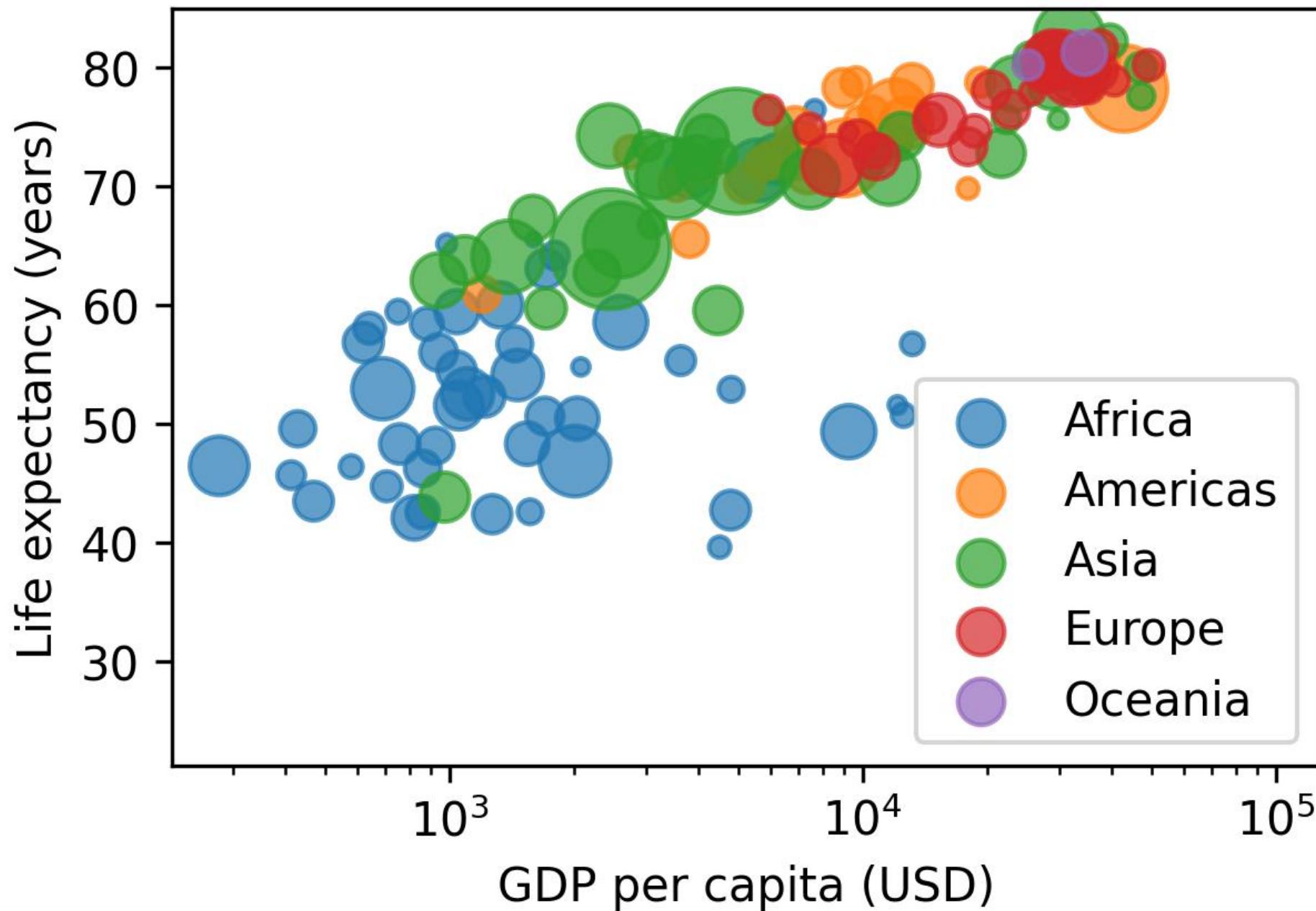
species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007

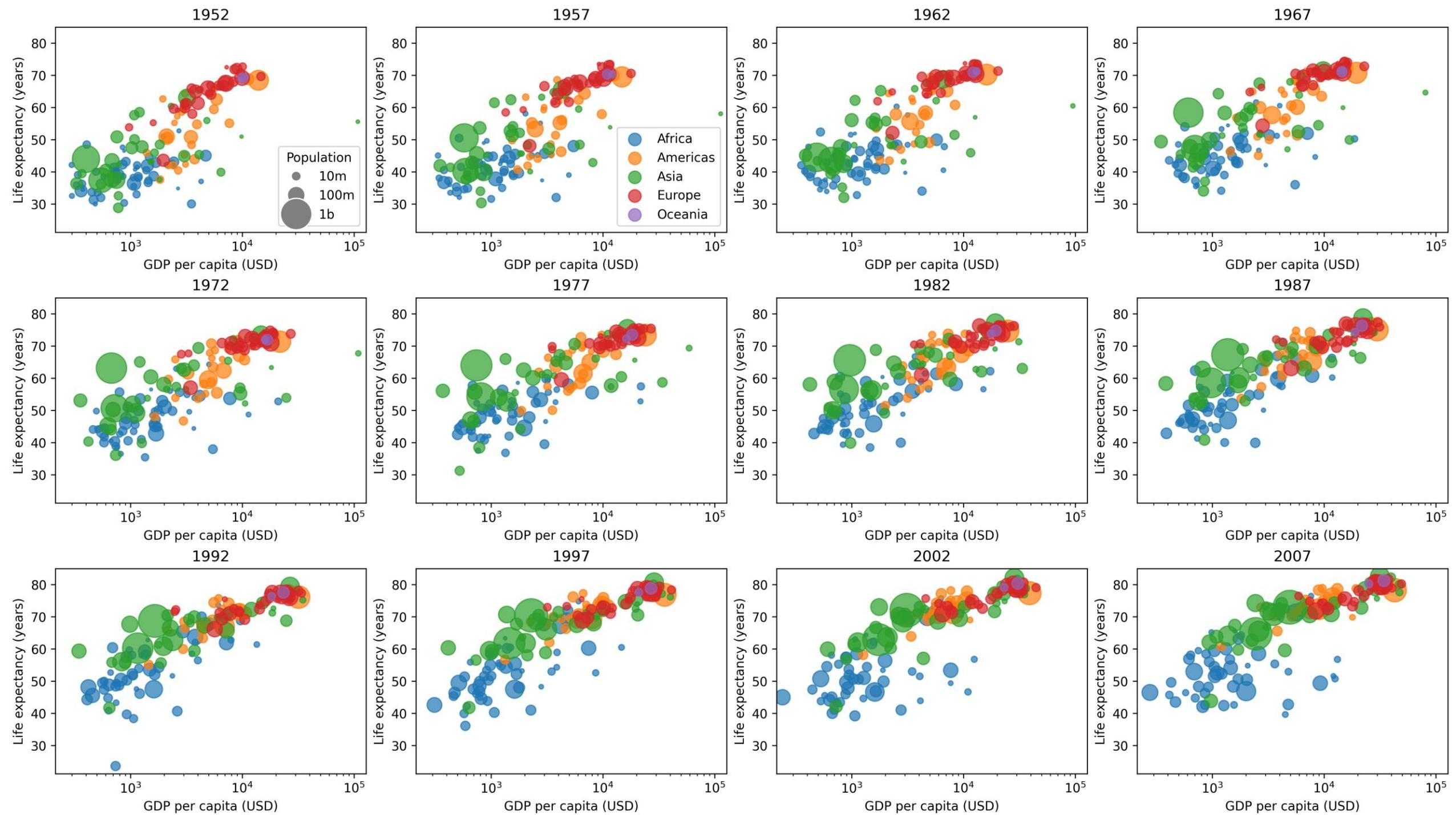


Gapminder

country	continent	year	lifeExp	pop	gdpPerCap
Afghanistan	Asia	1952	28.801	8425333	779.445314
Afghanistan	Asia	1957	30.332	9240934	820.853030
Afghanistan	Asia	1962	31.997	10267083	853.100710
Afghanistan	Asia	1967	34.020	11537966	836.197138
Afghanistan	Asia	1972	36.088	13079460	739.981106

2007





Housing

MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	price
8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...
1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

Housing Prices in Boston																																									
MSSubClass		MSZoning		LotFrontage		LotArea		Street		Alley		LotShape		LandContour		Utilities		LotConfig		...		PoolArea		PoolQC		Fence		MiscFeature		MiscVal		MoSold		YrSold		SaleType		SaleCondition		SalePrice	
60	RL	65.000000		8450	Pave	Grvl	Reg		Lvl	AllPub		Inside	...		0	Gd		MnPrv	Shed		0	2		2008	WD	Normal		208500													
20	RL	80.000000		9600	Pave	Grvl	Reg		Lvl	AllPub		FR2	...		0	Gd		MnPrv	Shed		0	5		2007	WD	Normal		181500													
60	RL	68.000000		11250	Pave	Grvl	IR1		Lvl	AllPub		Inside	...		0	Gd		MnPrv	Shed		0	9		2008	WD	Normal		223500													
70	RL	60.000000		9550	Pave	Grvl	IR1		Lvl	AllPub		Corner	...		0	Gd		MnPrv	Shed		0	2		2006	WD	Abnorml		140000													
60	RL	84.000000		14260	Pave	Grvl	IR1		Lvl	AllPub		FR2	...		0	Gd		MnPrv	Shed		0	12		2008	WD	Normal		250000													
50	RL	85.000000		14115	Pave	Grvl	IR1		Lvl	AllPub		Inside	...		0	Gd		MnPrv	Shed		700	10		2009	WD	Normal		143000													
20	RL	75.000000		10084	Pave	Grvl	Reg		Lvl	AllPub		Inside	...		0	Gd		MnPrv	Shed		0	8		2007	WD	Normal		307000													
60	RL	70.049958		10382	Pave	Grvl	IR1		Lvl	AllPub		Corner	...		0	Gd		MnPrv	Shed		350	11		2009	WD	Normal		200000													
50	RM	51.000000		6120	Pave	Grvl	Reg		Lvl	AllPub		Inside	...		0	Gd		MnPrv	Shed		0	4		2008	WD	Abnorml		129900													
190	RL	50.000000		7420	Pave	Grvl	Reg		Lvl	AllPub		Corner	...		0	Gd		MnPrv	Shed		0	1		2008	WD	Normal		118000													

MNIST



5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
8 8 3 3 3
8 9 2 3 3 3
3 3 3 3 3
3 3 3 3 3
3 3 3 3 3

10 10 10 10

٦٦٦٦٦٦٦٦

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

5 5 5 5
-5 -5 5 5
5 5 5 5
5 5 5 5
5 5 5 5
8 8 8 8
6 8 9 9
-3 -3 3 3
3 3 3 3
3 3 3 3

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

۶۶۶۶۶۶۶۶

०००००००००००००

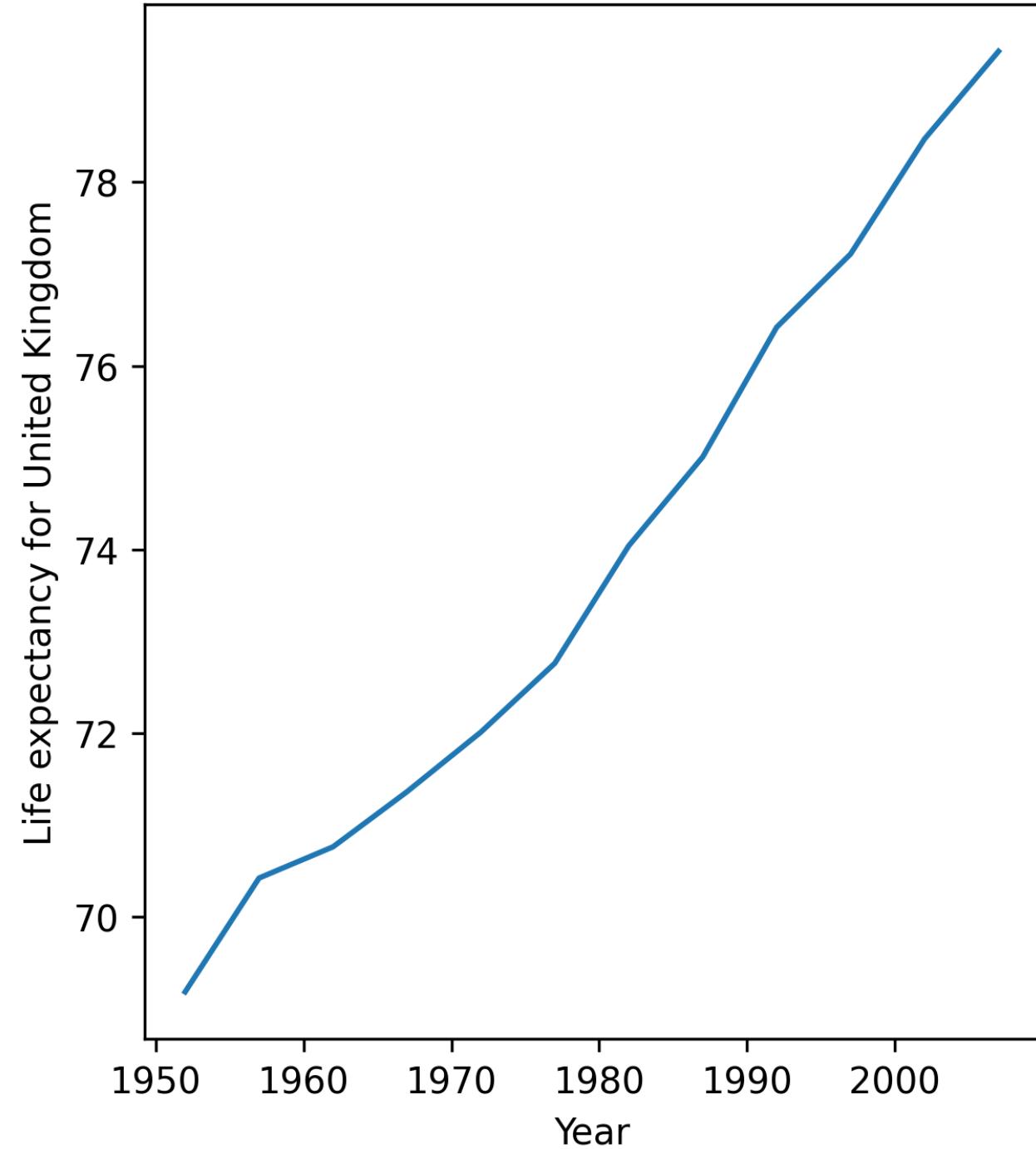
**Unused
Slides**

```
# List of all the countries available
gapminder['country'].unique()

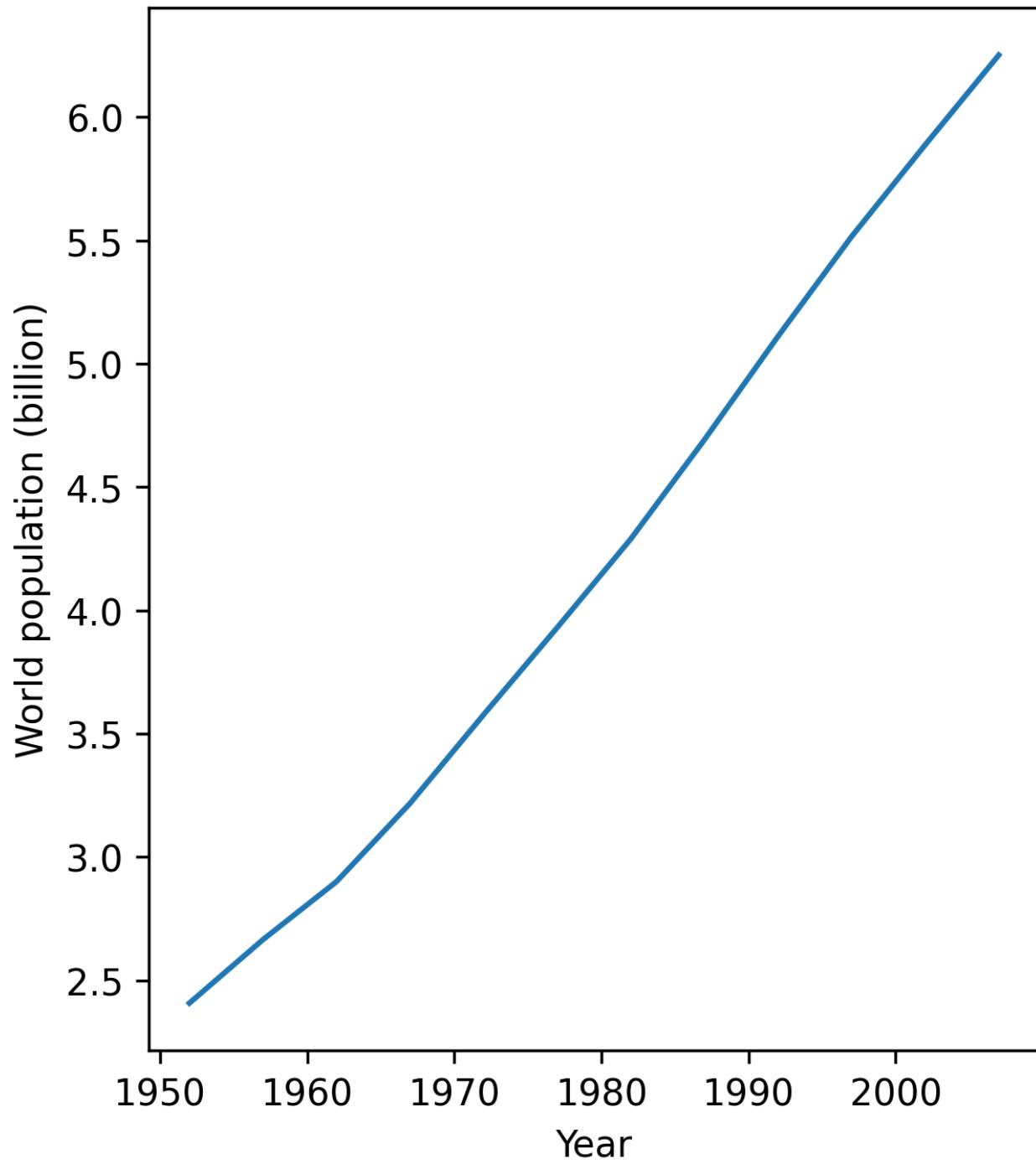
country = 'United Kingdom'

i = gapminder['country'] == country

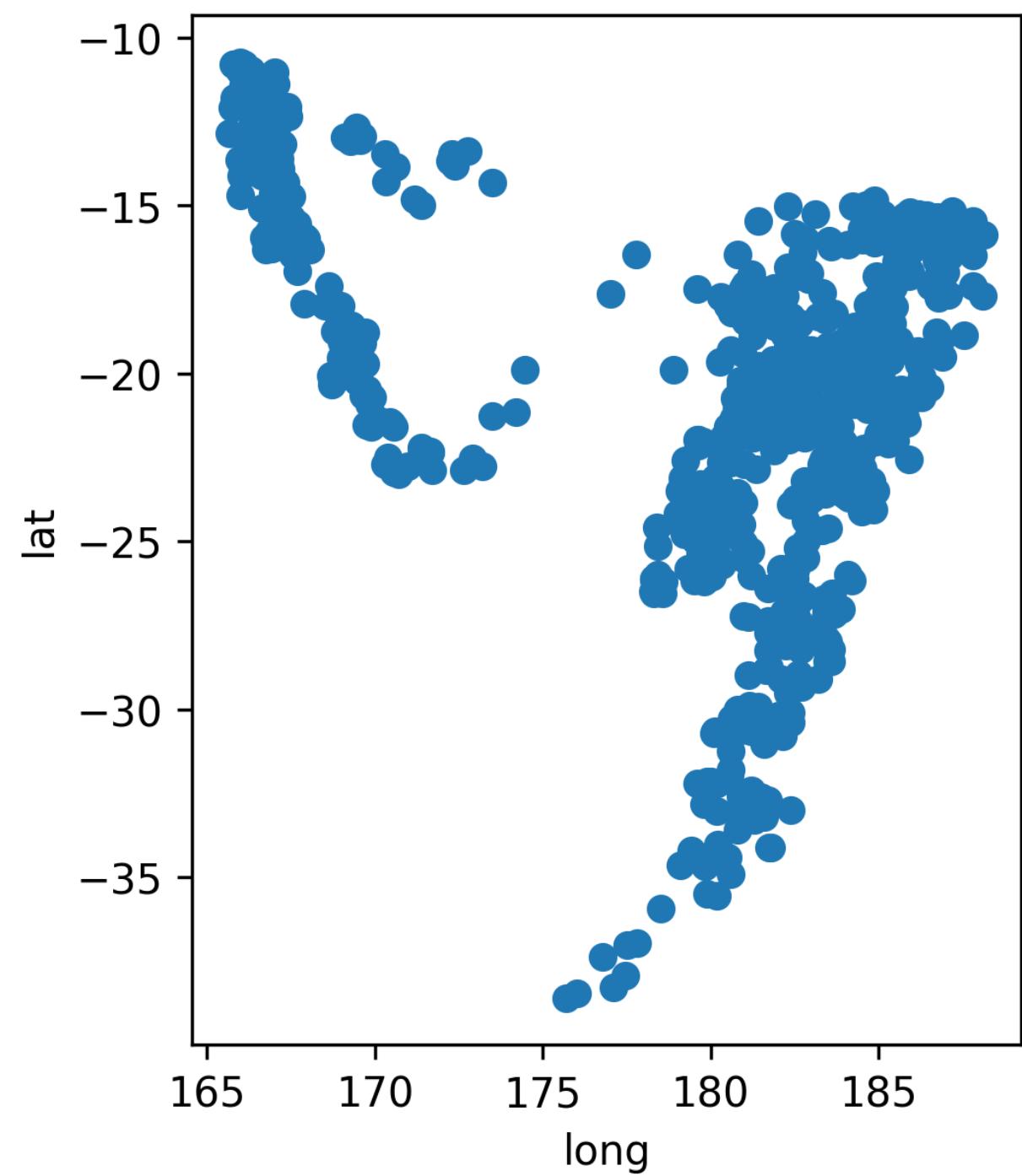
plt.plot(
    gapminder[i]['year'],
    gapminder[i]['lifeExp'],
)
```



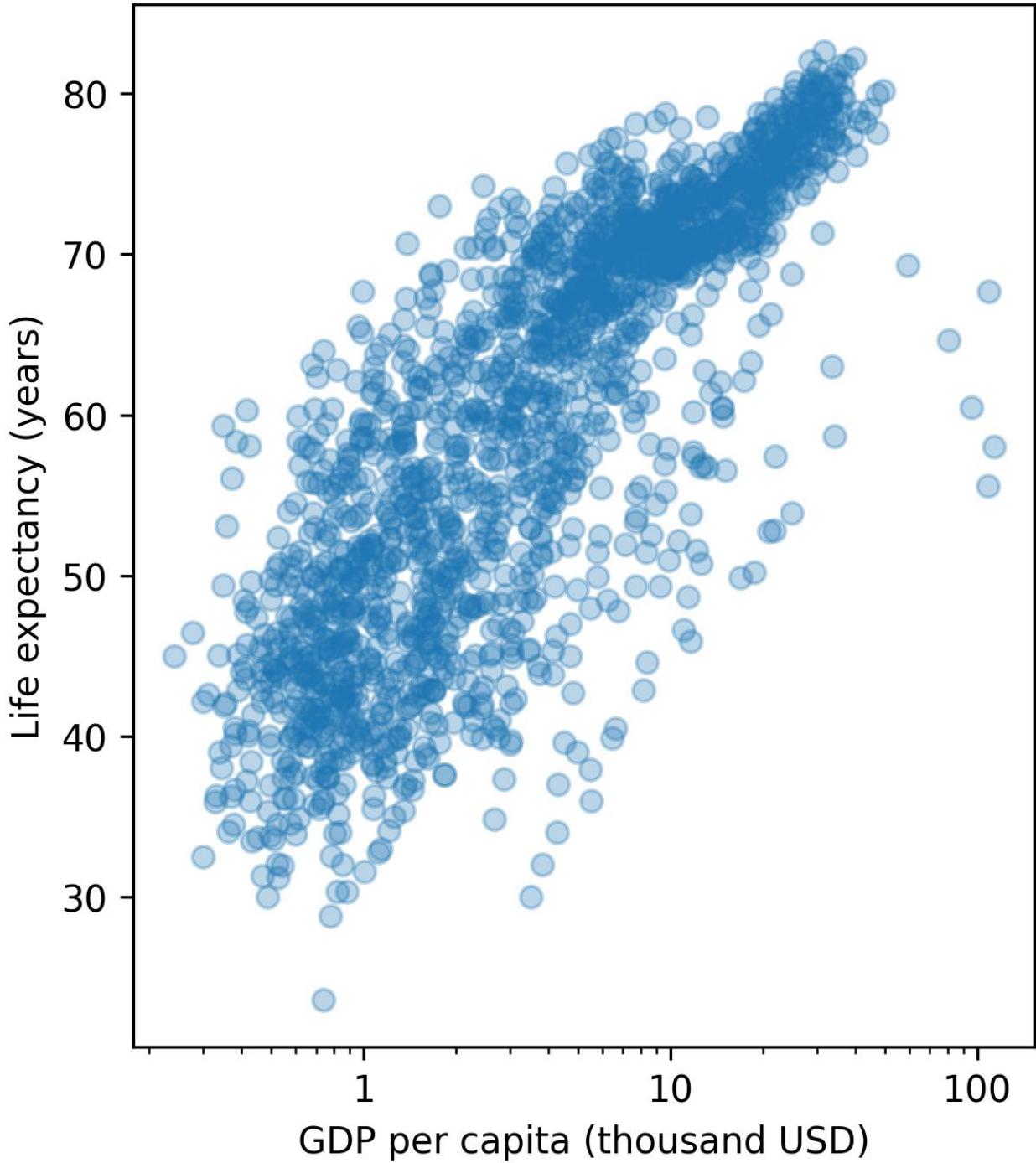
```
population = (
    gapminder.groupby('year')['pop'].sum()
)
plt.plot( population )
```



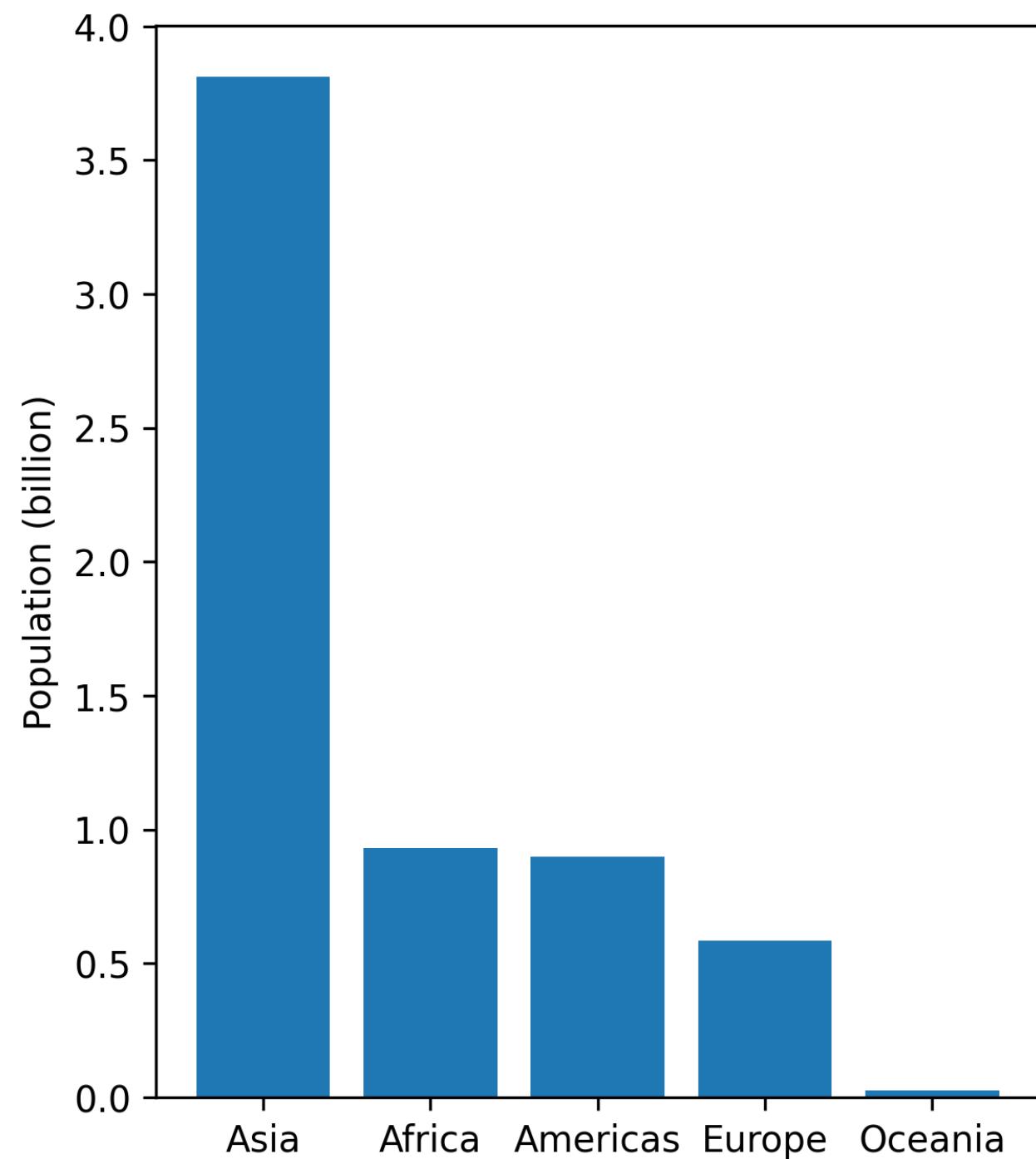
```
d.plot.scatter( x, y )
```



```
plt.scatter(  
    gapminder['gdpPercap'],  
    gapminder['lifeExp'],  
)  
plt.xscale('log')
```



```
last_date = gapminder['year'].max()
end = gapminder['year'] == last_date
population = (
    gapminder[end]
    .groupby('continent')['pop']
    .sum()
    .sort_values( ascending = False )
)
plt.bar(
    population.index,
    population.values,
)
```



```

last_date = gapminder['year'].max()
end = gapminder['year'] == last_date
population = (
    gapminder[end][['country', 'pop']]
    .sort_values('pop', ascending = False)
)
a = 20
population = pd.concat([
    population.iloc[:a],
    pd.DataFrame([
        {
            'country': 'Rest',
            'pop': population.iloc[a:]['pop'].sum()
        }
    ]),
])
population = population.iloc[::-1,:]
plt.barh(
    population['country'],
    population['pop'],
)

```

