

**Data
visualization**

Data visualization

Exploratory Data Analysis

Goals of EDA

- Find problems
- Understand
- Communicate

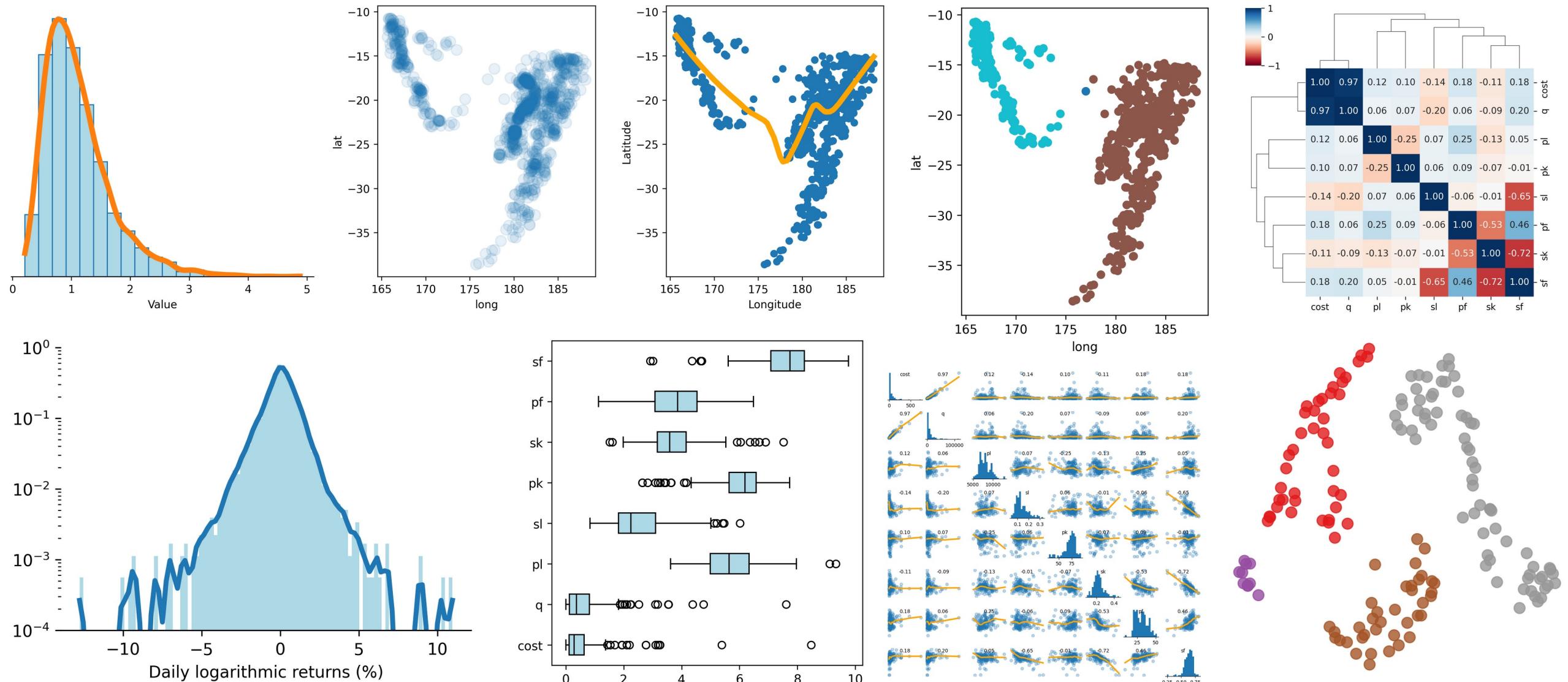
Agenda

- Univariate data
- Quantitative vs qualitative
- Bivariate data
- Multivariate data
- Unstructured data
- Potential problems
 - Numbers (“dataset features”)
 - Plots

Agenda

- Univariate data
- Quantitative vs qualitative
- Bivariate data
- Multivariate data
- Unstructured data
- Potential problems
 - Numbers (“dataset features”)
 - Plots

Agenda



Plotting

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

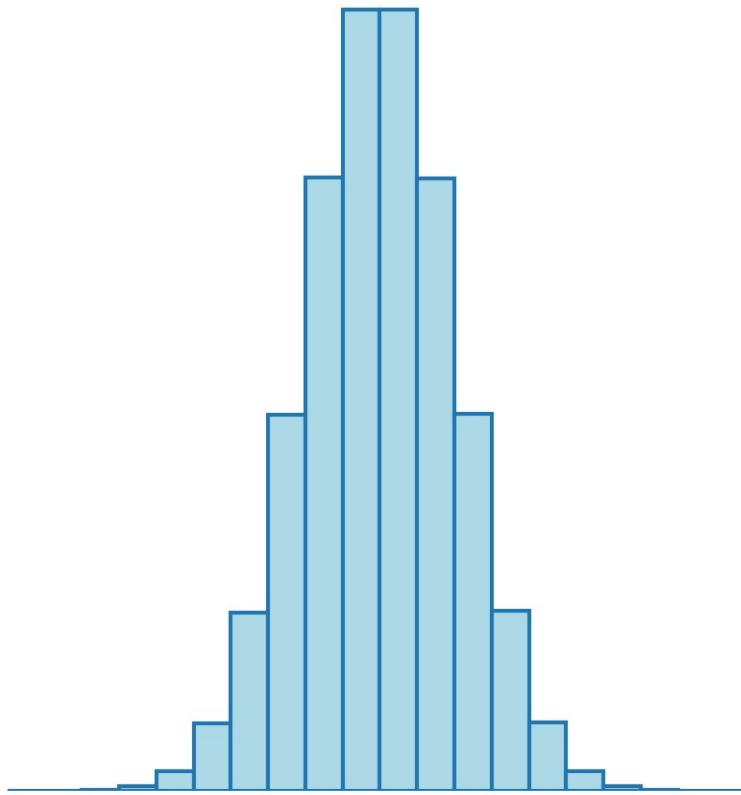
- Pandas
 - d.plot.hist
 - d.plot.scatter
 - ...
- Matplotlib
 - plt.hist
 - plt.scatter
 - ...
- Seaborn
 - sns.histplot
 - sns.scatterplot
 - ...
- Many more: plotly, etc.

```
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
```

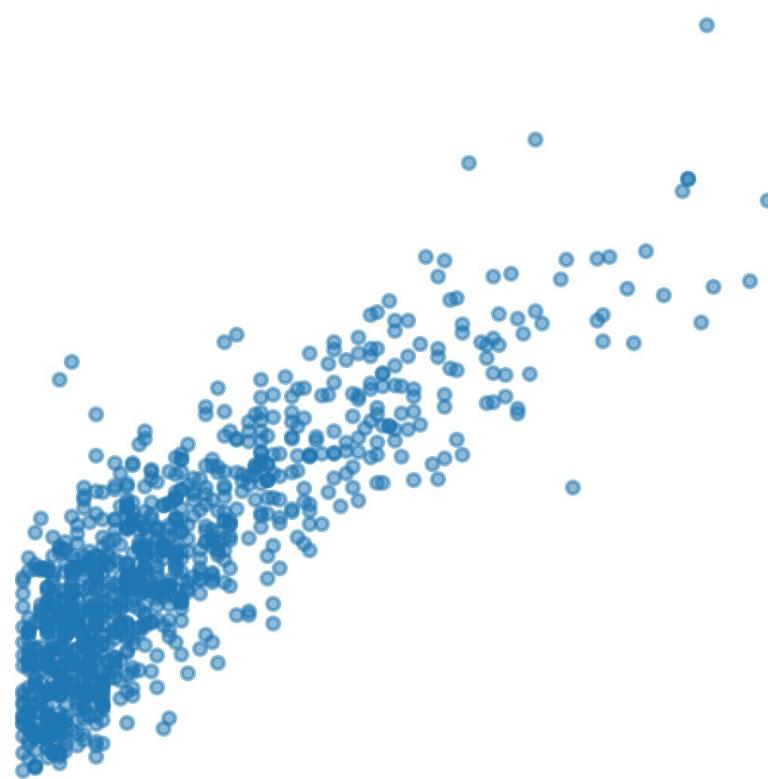
Matplotlib

histogram



`plt.hist(x)`

scatter plot



`plt.scatter(x,y)`

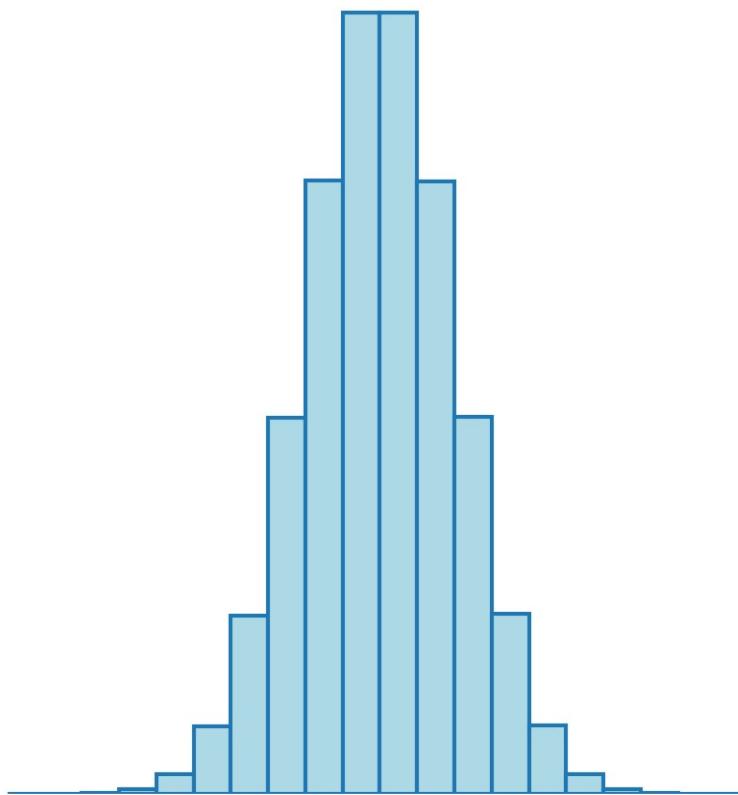
line chart



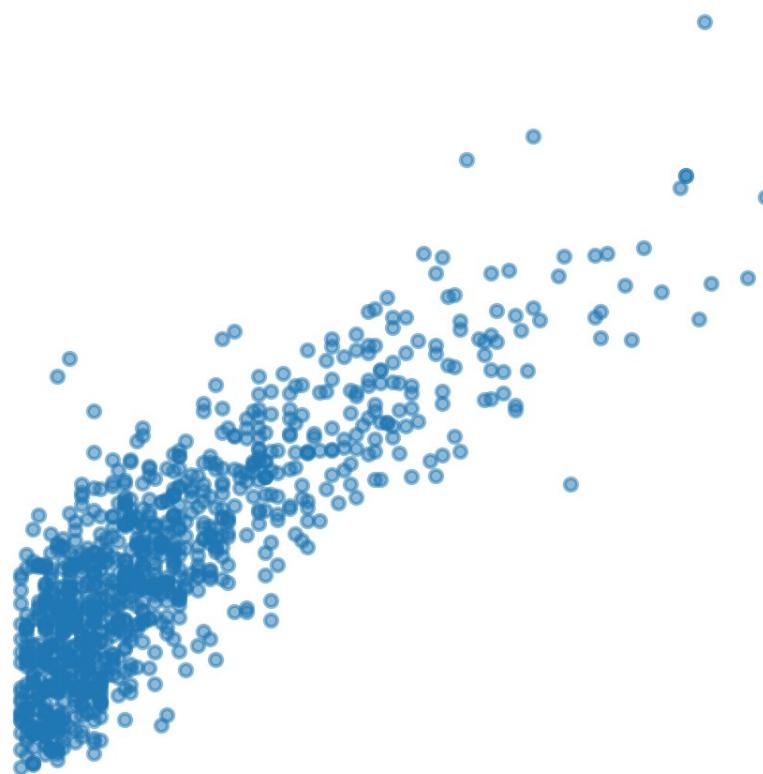
`plt.plot(x,y)`

Pandas

histogram



scatter plot

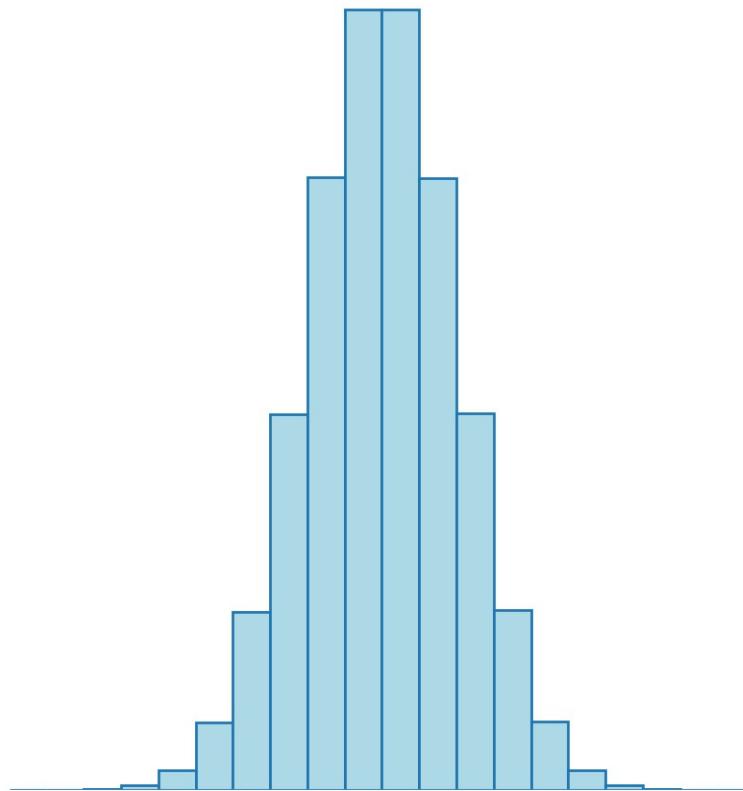


line chart

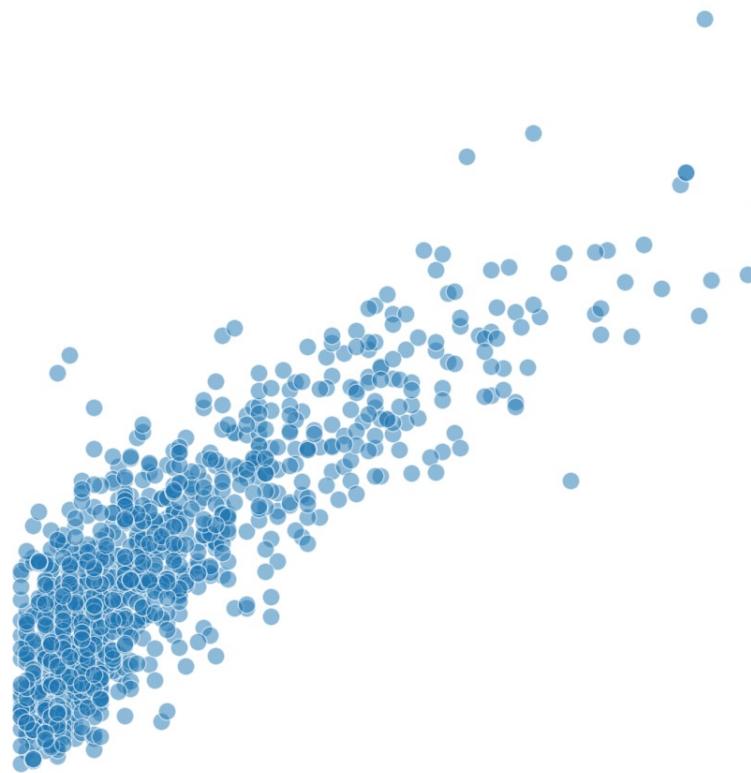


Seaborn

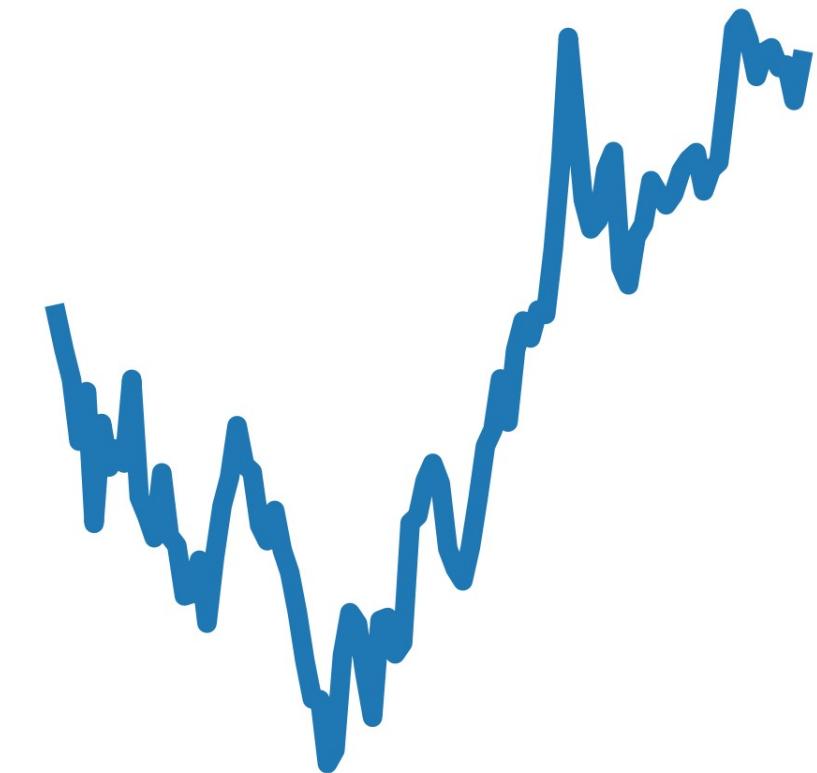
histogram



scatter plot

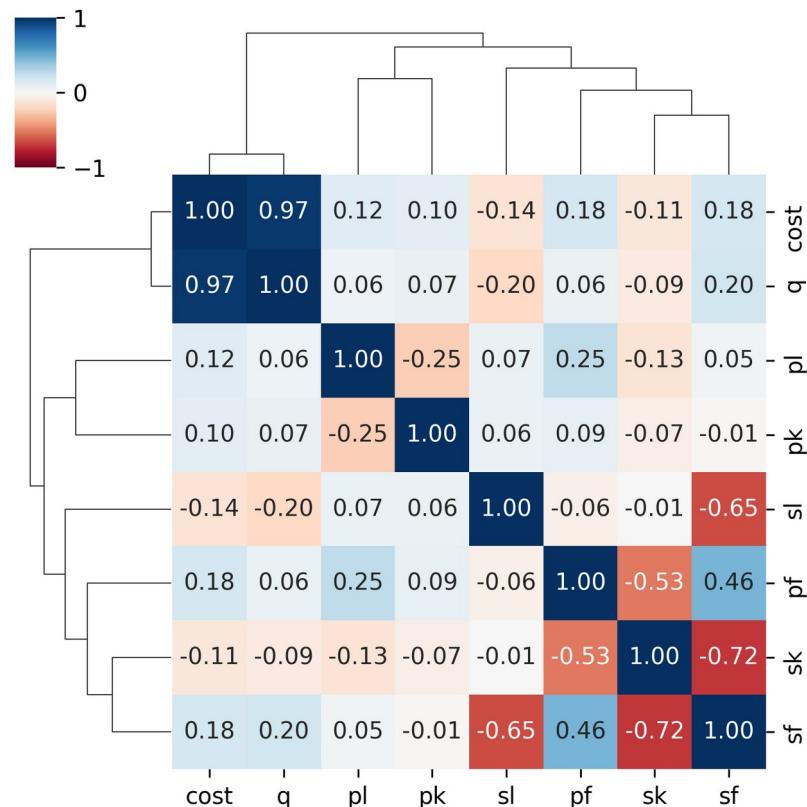


line chart



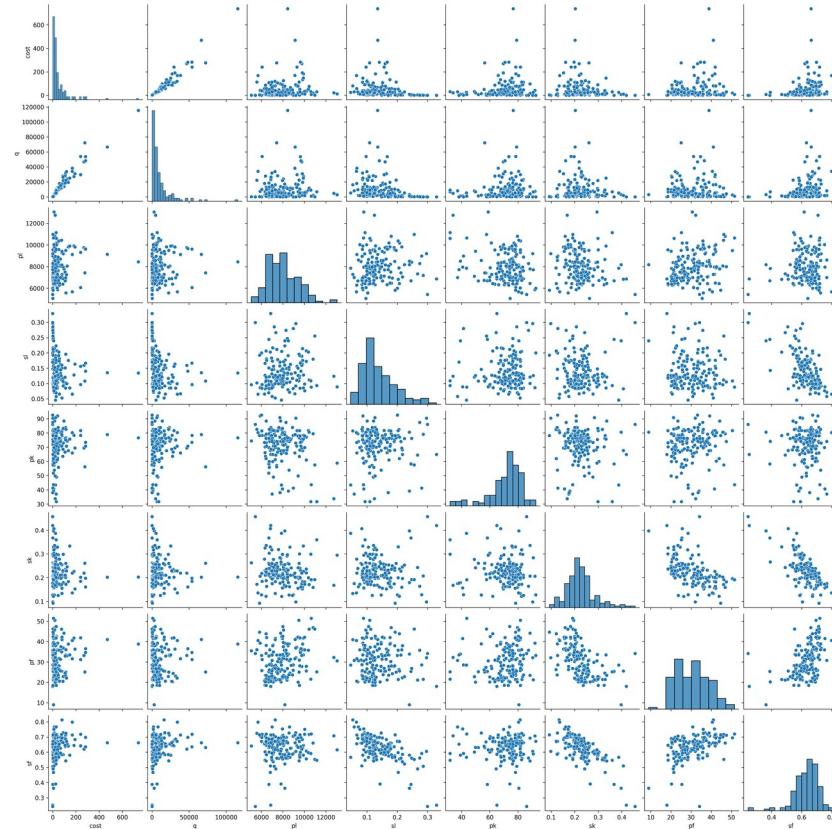
Seaborn

correlation matrix



```
sns.clustermap(d.corr(),...)
```

pair plot



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

For more complex plots

- Check the Matplotlib or Seaborn galleries
(for inspiration, or to find the name of a plot)
show the live websites
- Ask ChatGPT (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](#)

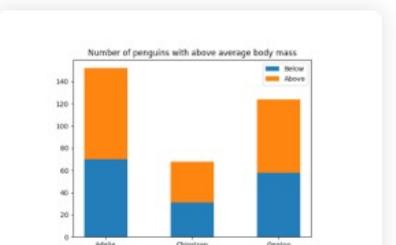
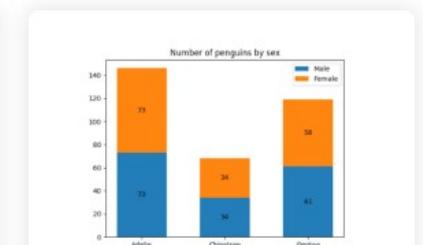
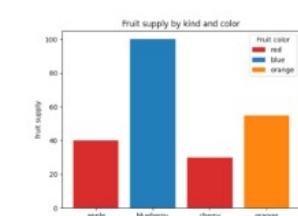
Examples

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.

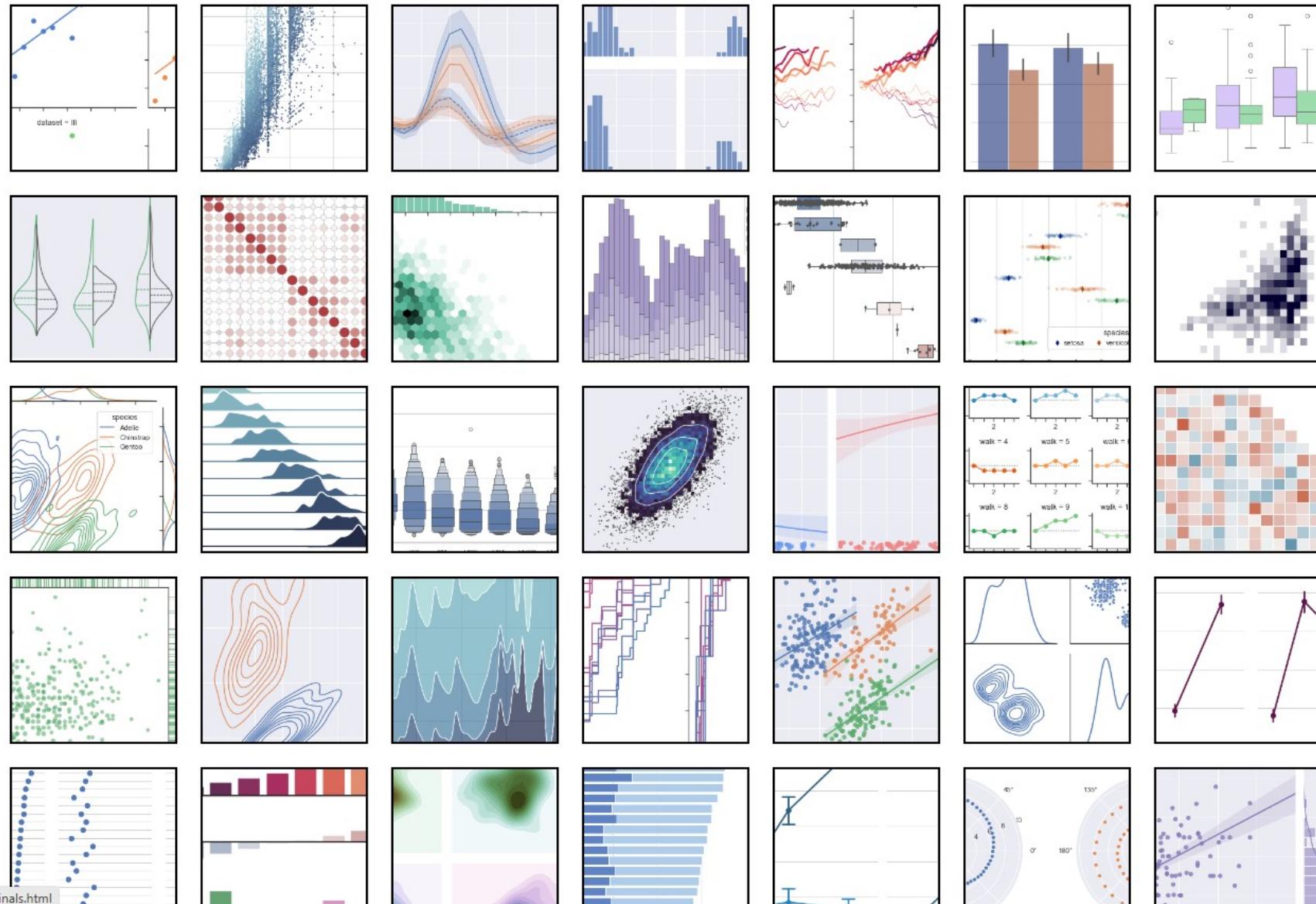
For longer tutorials, see our [tutorials page](#). You can also find [external resources](#) and a [FAQ](#) in our [user guide](#).

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'}
```

```
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()
```

```
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()
```

```
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': '^'}

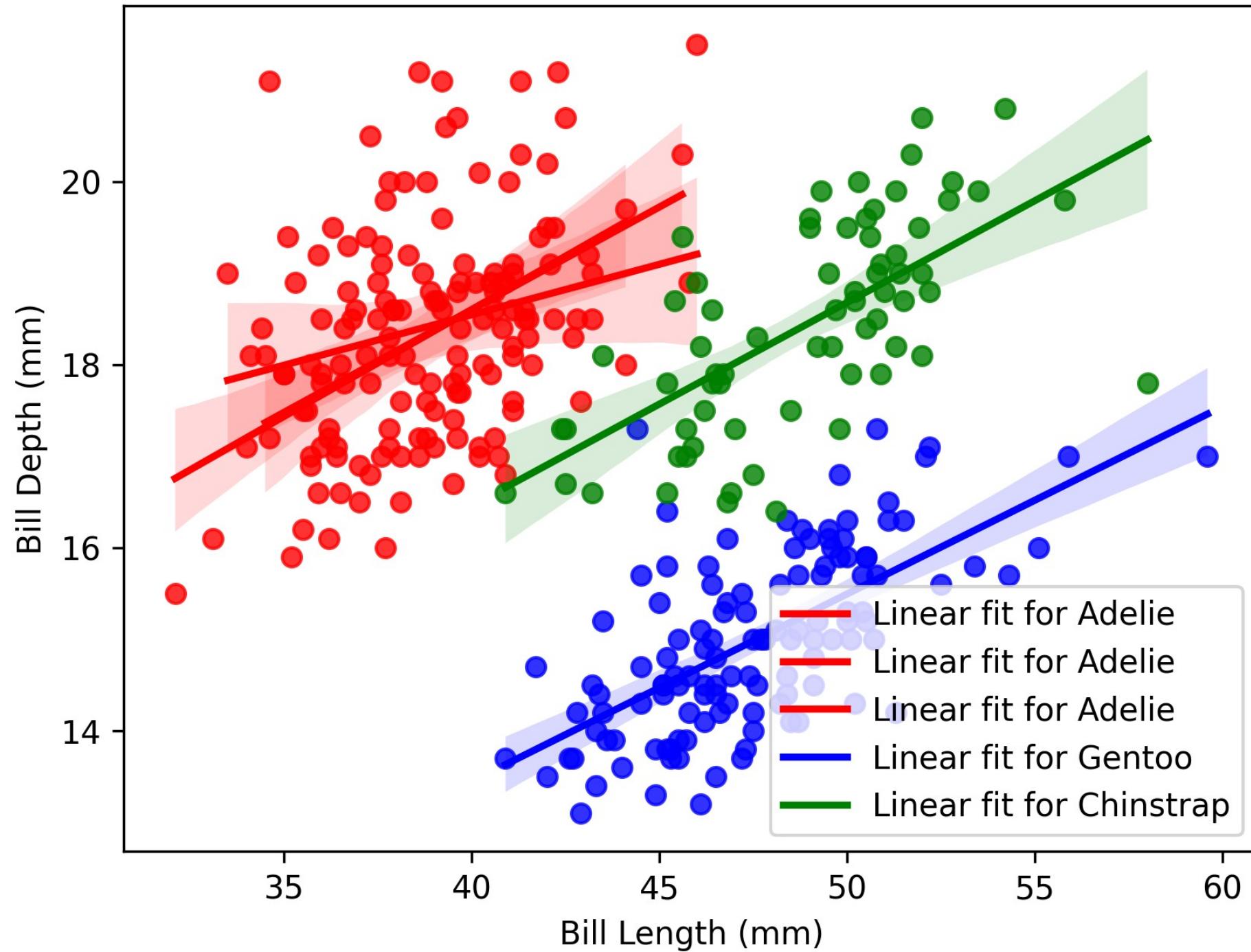
# Unique colors for each species
colors = {'Adelie': 'r', 'Chinstrap': 'g', 'Gentoo': 'b'}

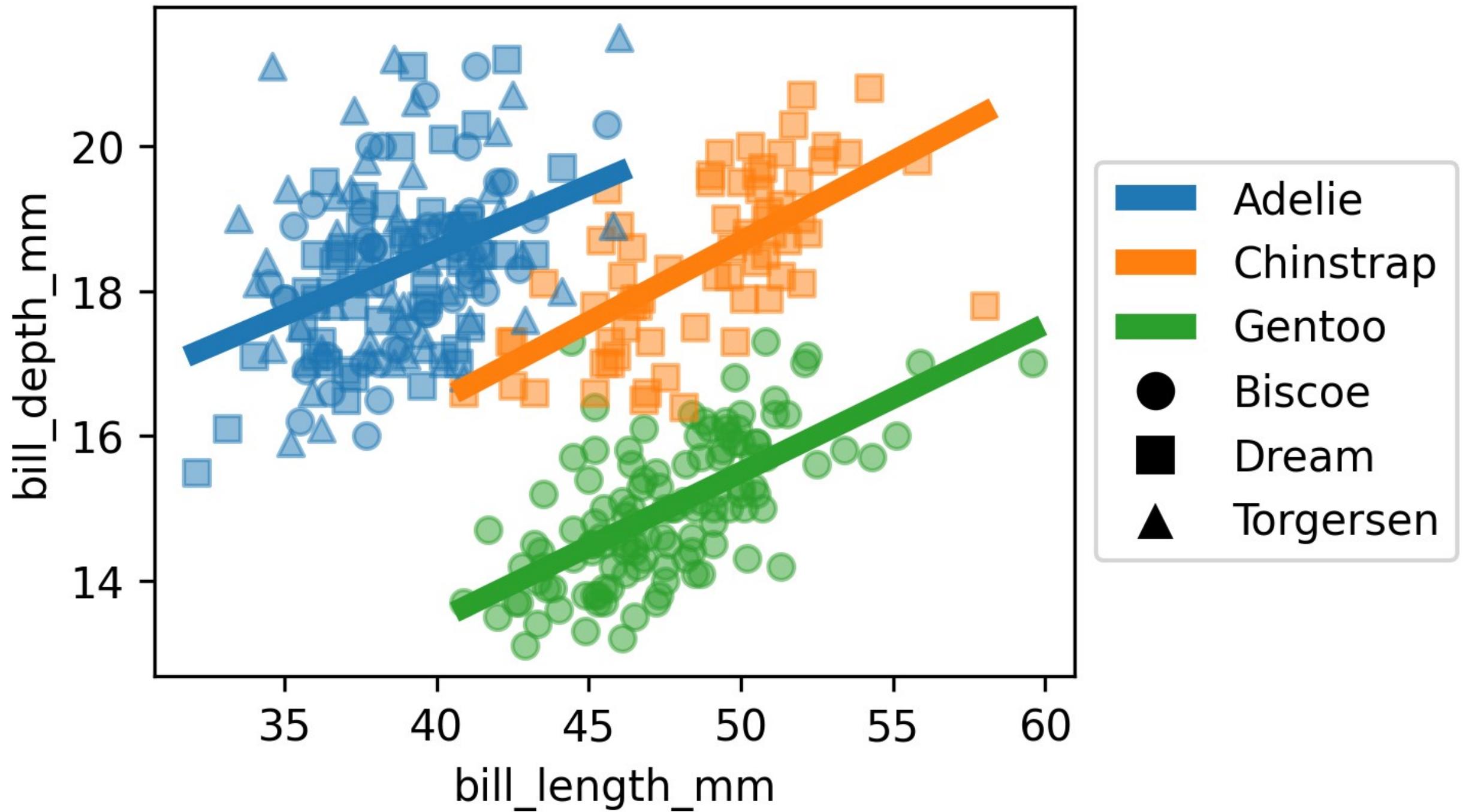
# Plot each species
for species in df['species'].unique():
    for island in df['island'].unique():
        species_island_df = df[(df['species'] == species) & (df['island'] == island)]
        sns.regplot(x='bill_length_mm', y='bill_depth_mm', data=species_island_df,
                    scatter_kws={'color': 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()
```





Univariate Data

Univariate Data

- A list of numbers, e.g.,

[0.51, 0.31, 0.03, ..., 1.11, -0.19, -0.29]

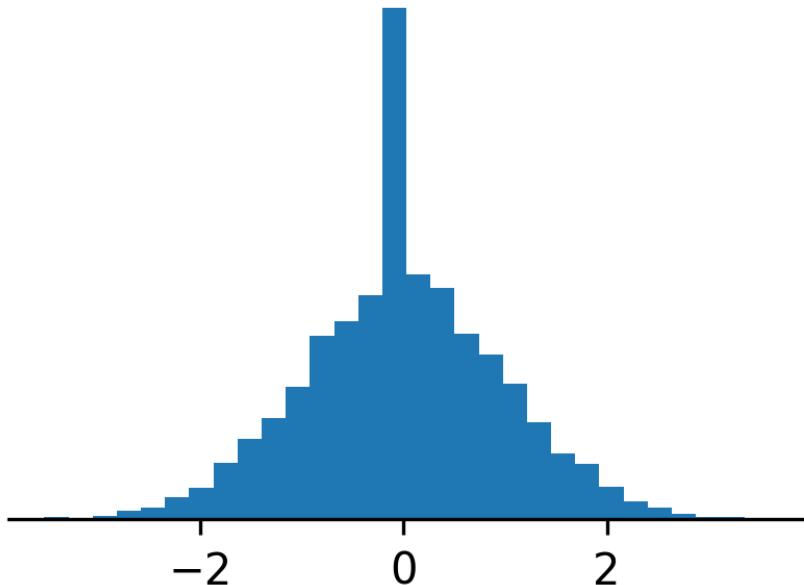
Potential problems

- Data types
- Missing values
- Magic numbers
- Zero inflation
- Outliers
- Fat tails
- Skewness
- Multimodality

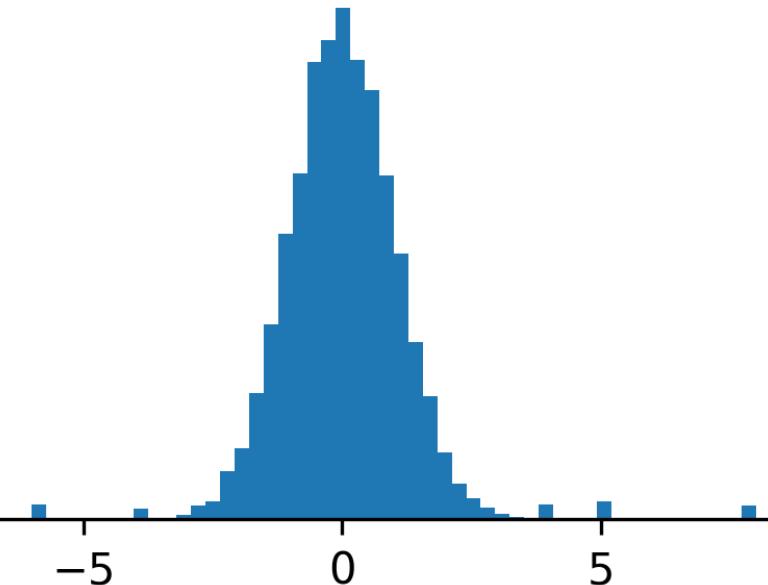
Potential Problems

- ["1", "2", "17.3", "**N/A**", ...]
- ["15,**271**", "517", "32.1", "184,**236**", ...]
- ["14**%**", "17**%**", "65**%**", "3**%**", ...]
- [17, 33, 18, 25, **nan**, 51, ...]
- [17, 33, 18, 25, **-1**, 51, ...]
- [17, 33, 18, 25, **999**, 51, ...]
- [17, 33, 18, 25, **None**, 51, ...]

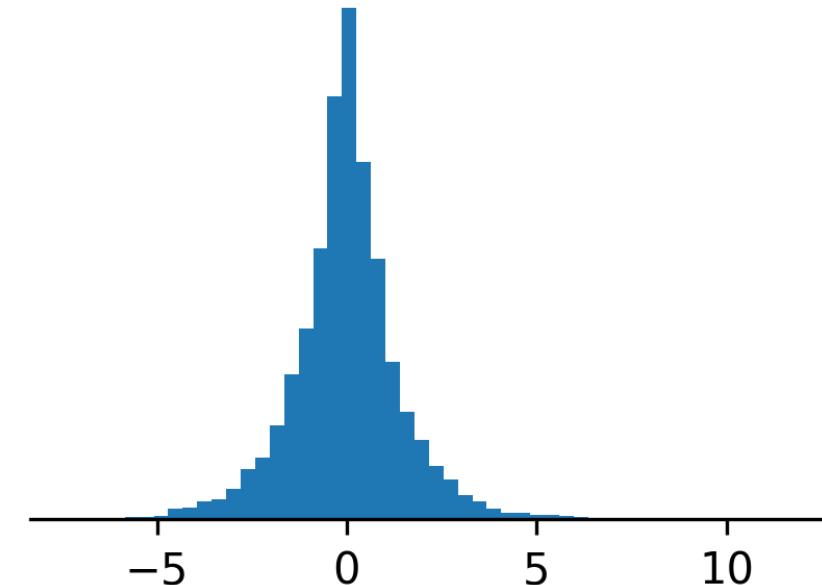
Zero inflation



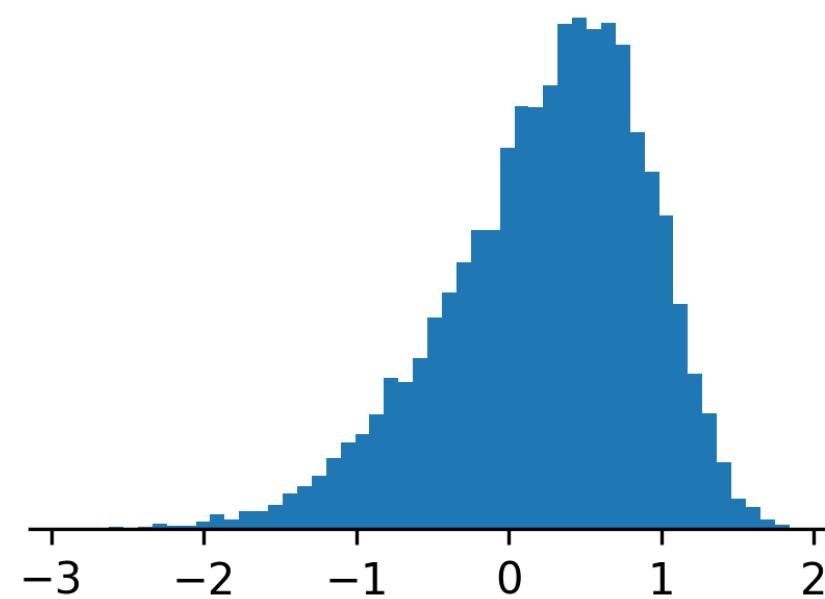
Outliers



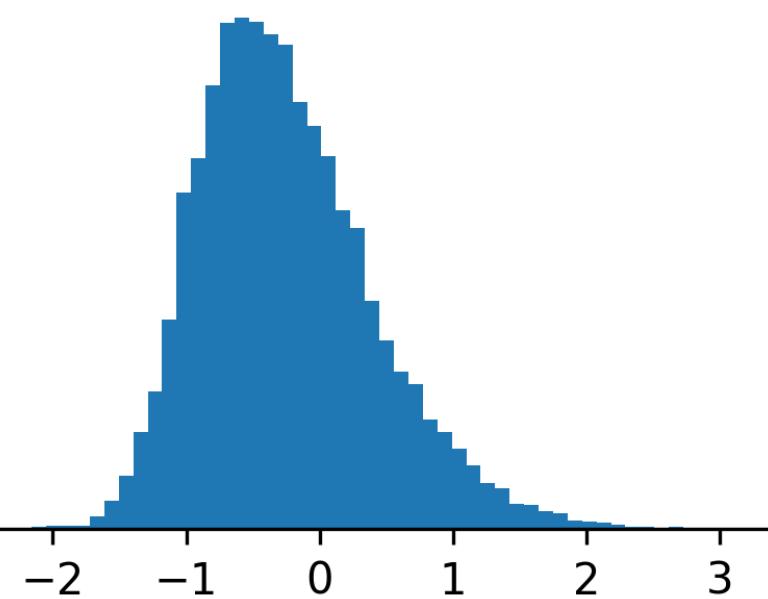
Fat tails



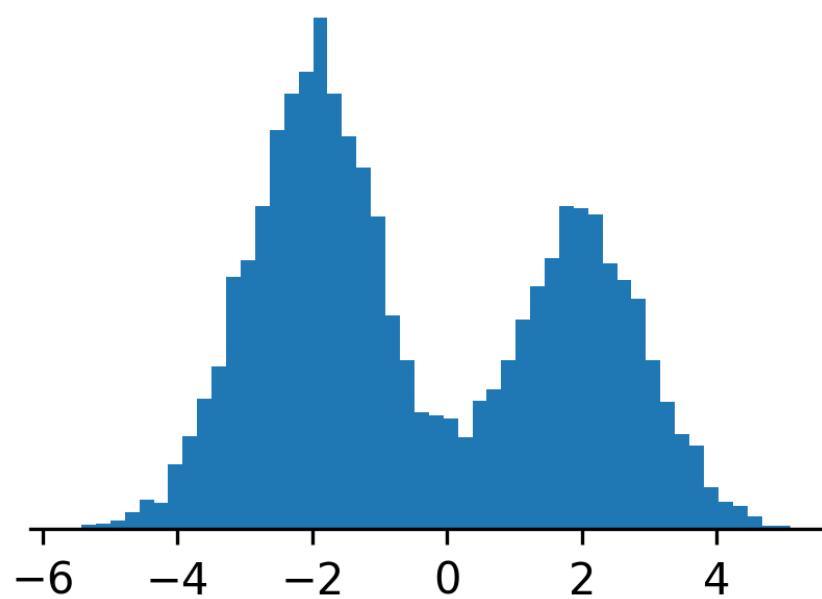
Skewness < 0



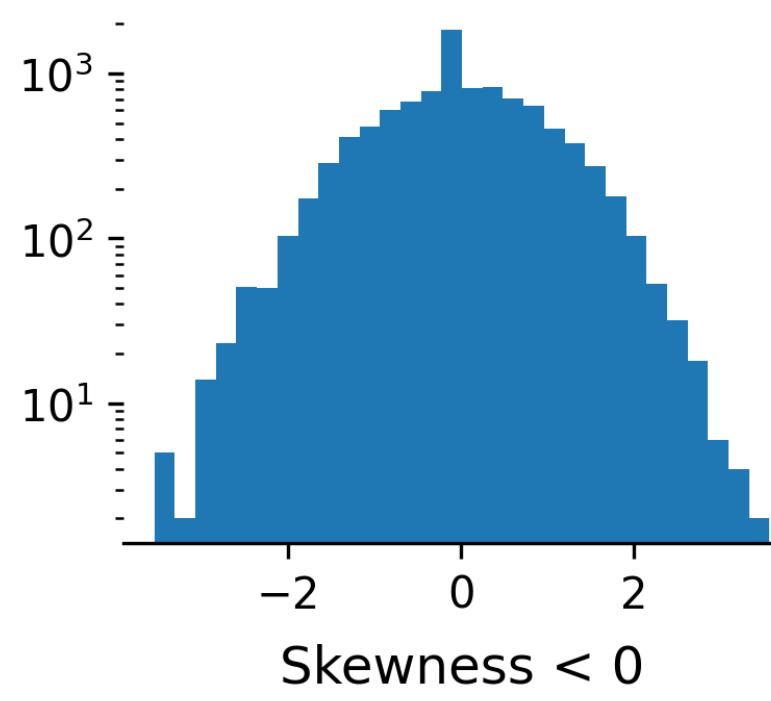
Skewness > 0



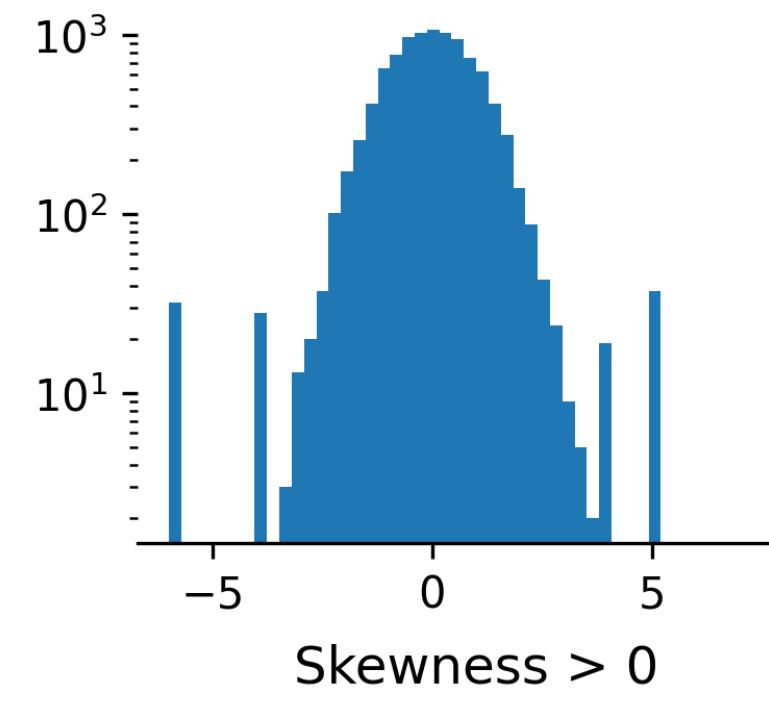
Multimodality



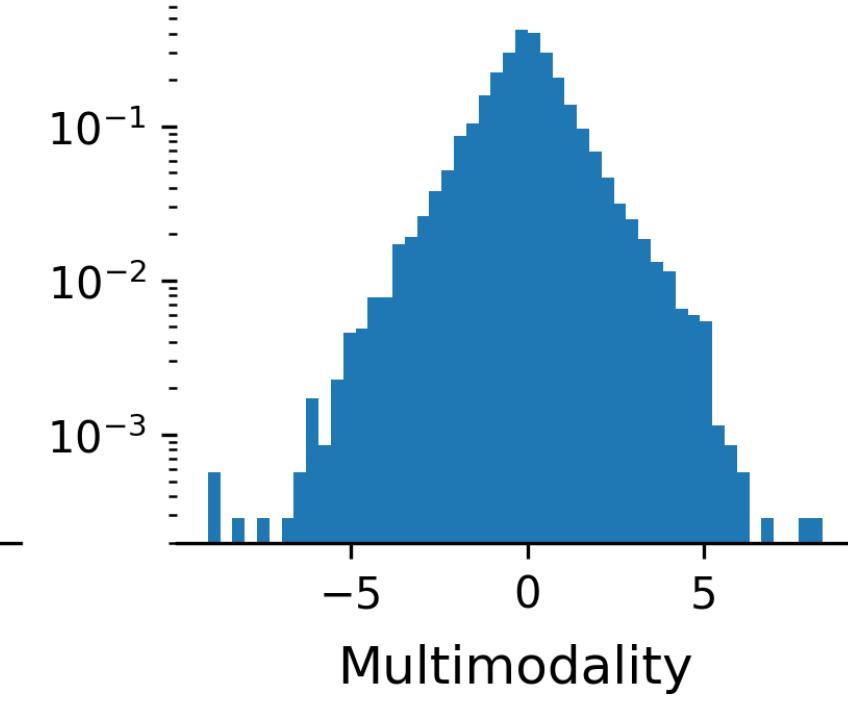
Zero inflation



Outliers



Fat tails



Skewness < 0

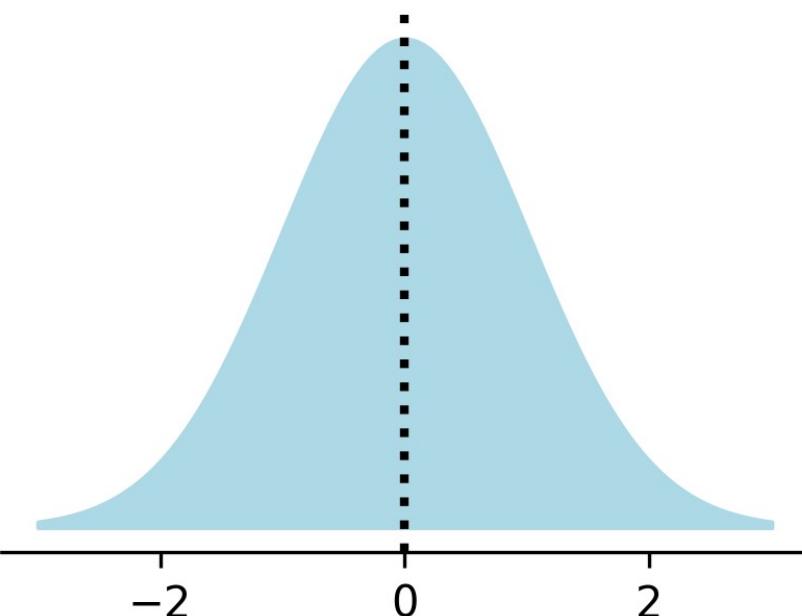
Skewness > 0

Multimodality

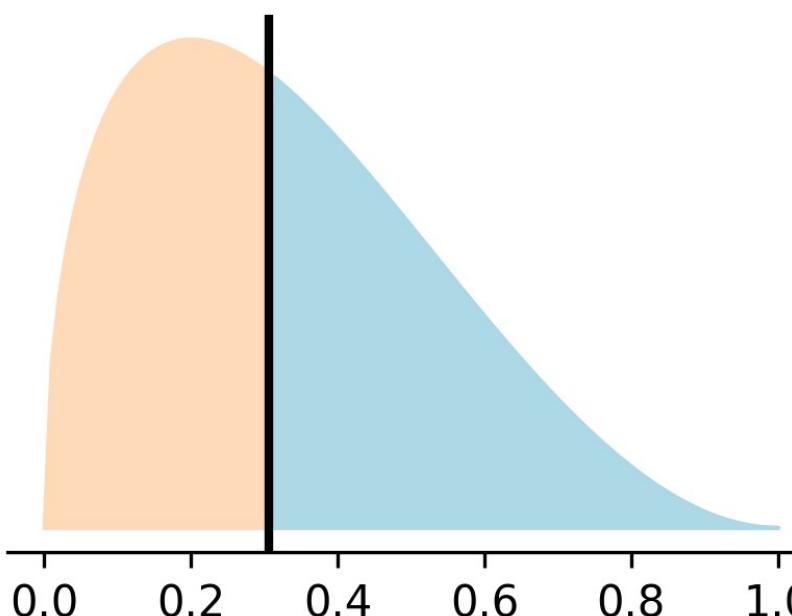
Numbers

- Number of observations
- Number of missing values
- Mode (value and number of repeats)
- Number of zeroes
- Minimum, maximum
- Median
- Standard deviation
- Quantiles
- Inter-quartile range
- Skewness
- Kurtosis

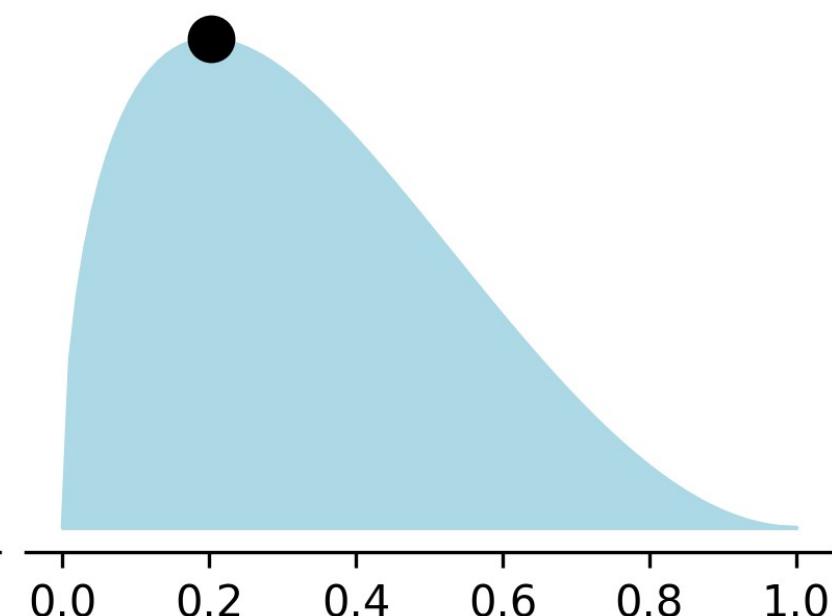
Mean



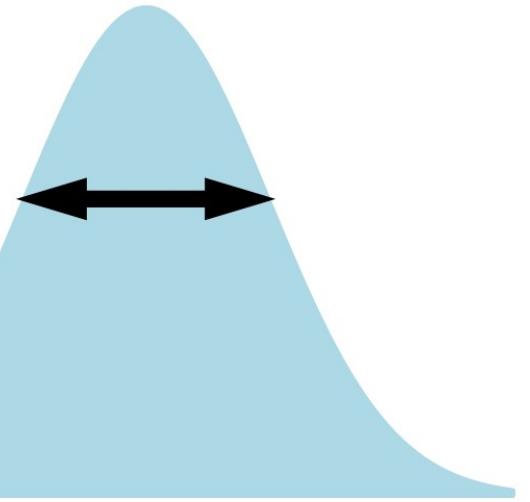
Median



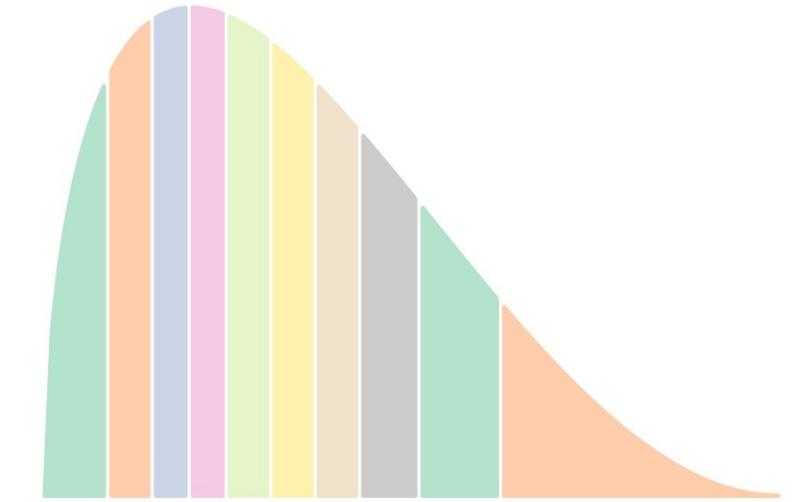
Mode



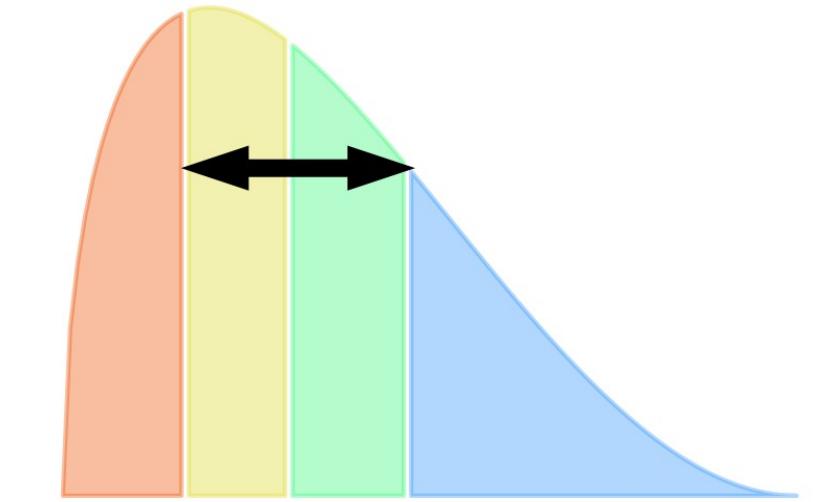
Standard Deviation



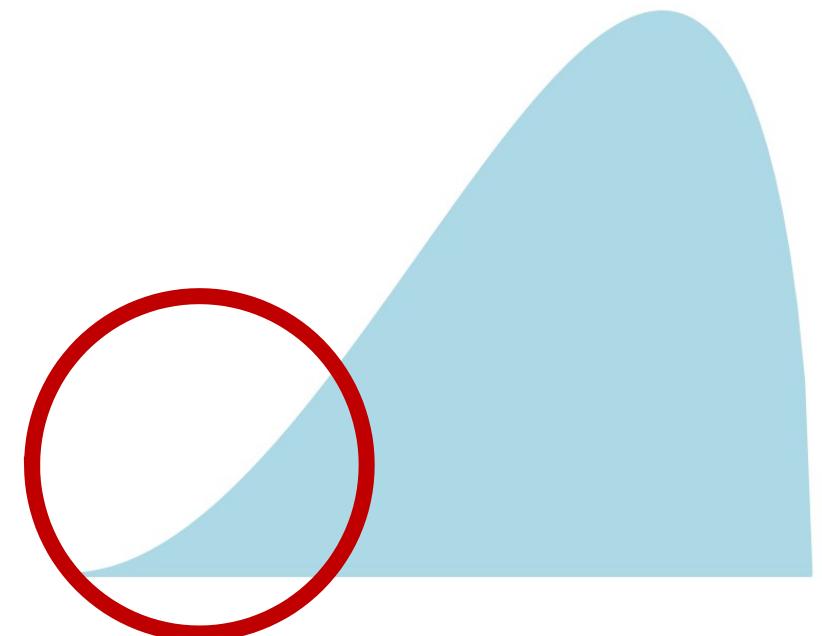
Quantiles



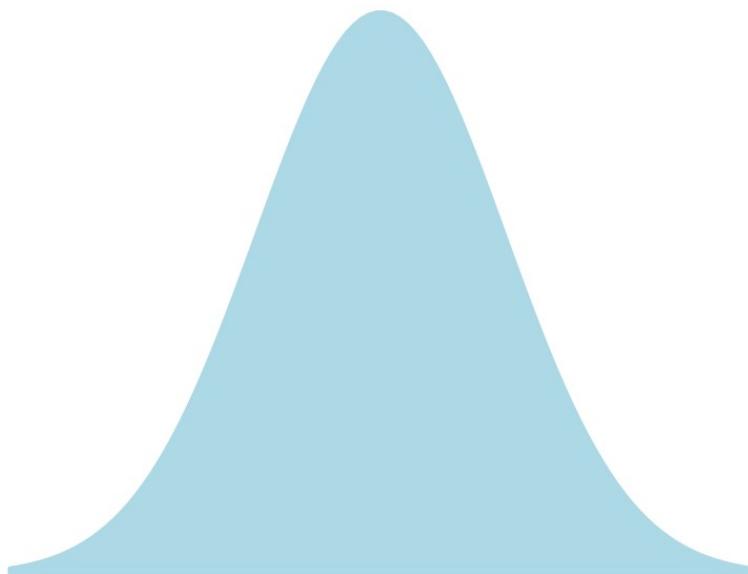
Inter-quartile range (IQR)



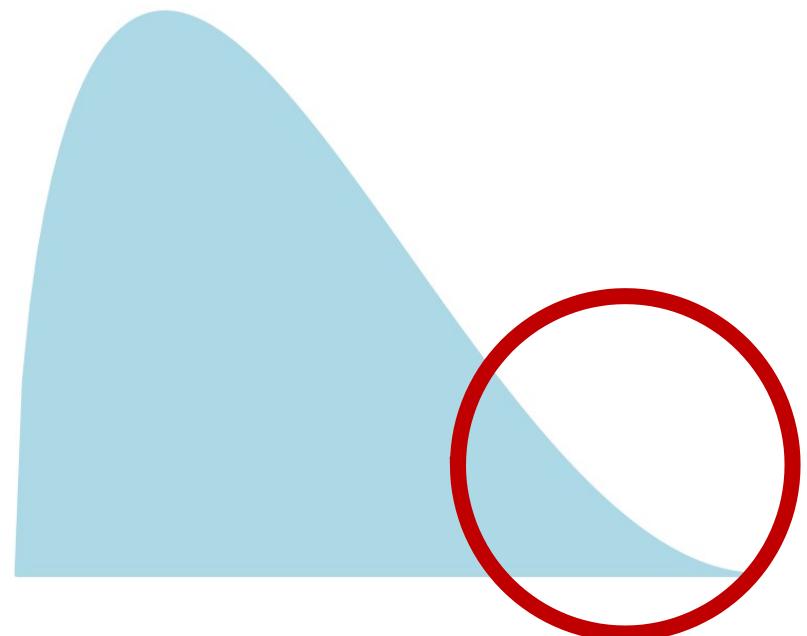
Skewness = -0.5



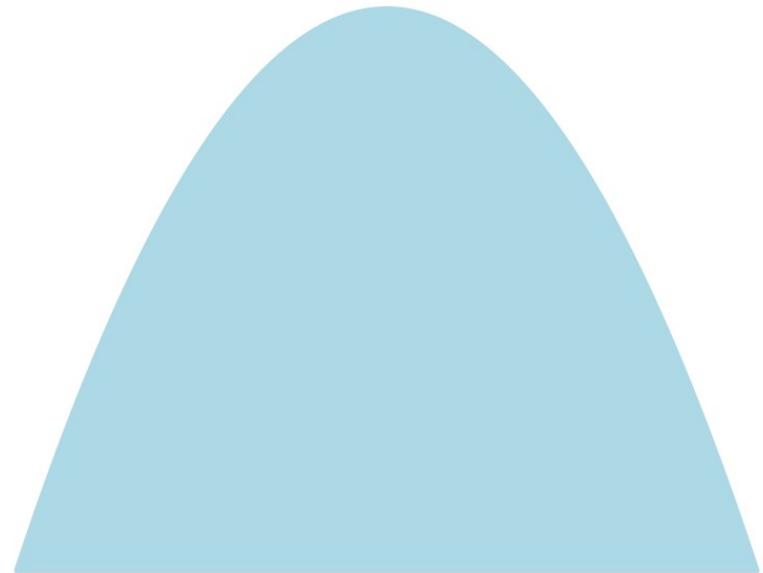
Skewness = 0



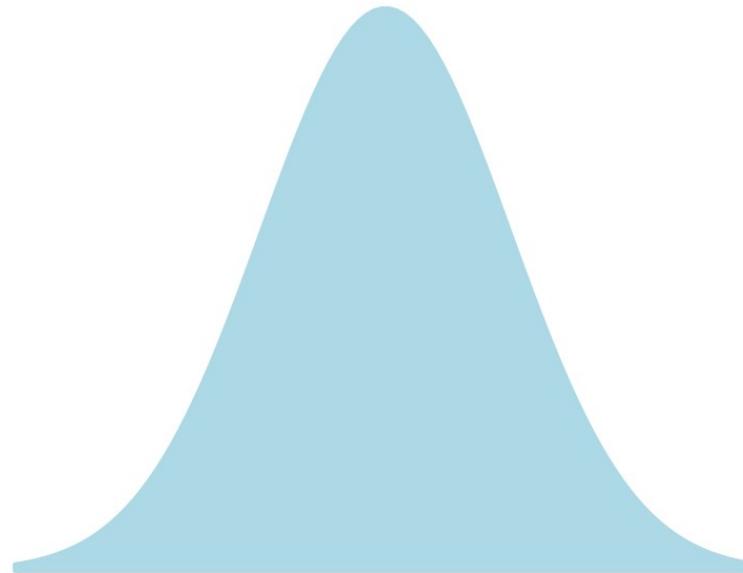
Skewness = 0.5



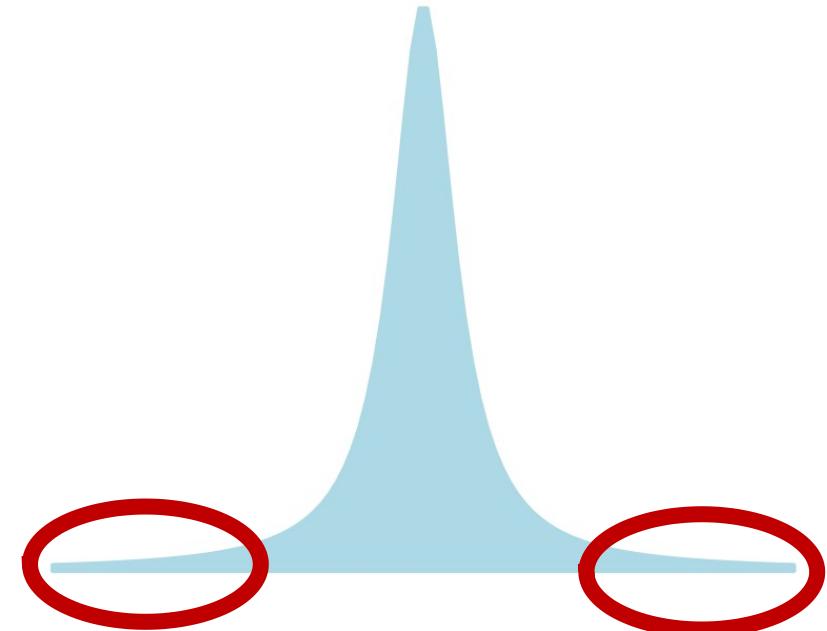
(Excess) kurtosis = -1



(Excess) kurtosis = 0



(Excess) kurtosis > 0



Numbers

```
x.describe()
```

```
count      272.000000
mean       70.897059
std        13.594974
min        43.000000
25%        58.000000
50%        76.000000
75%        82.000000
max        96.000000
Name: waiting, dtype: float64
```

Numbers

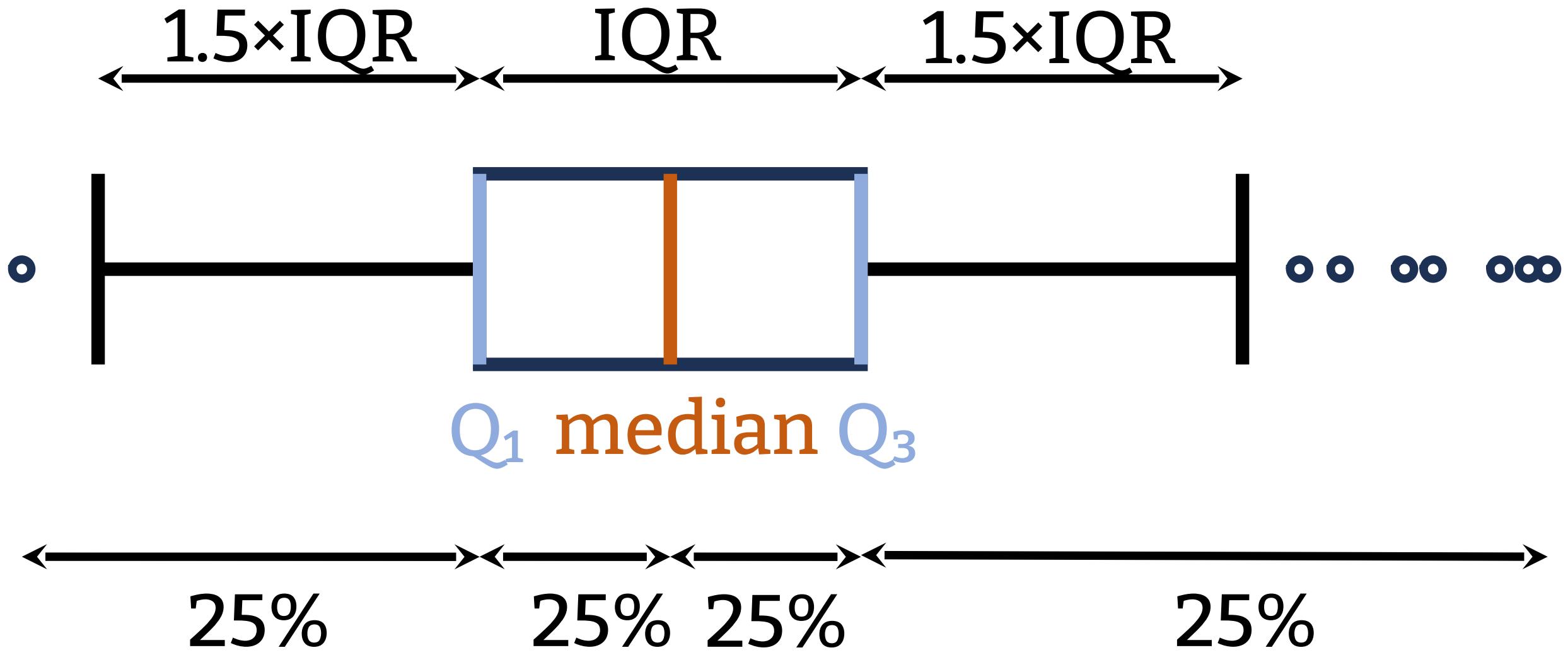
```
{  
    'dtype': str(x.dtype),  
    'observations': len(x),  
    'missing': x.isnull().sum(),  
    'mode': list(x.mode()),  
    'mode occurrences': (x == x.mode()[0]).sum(),  
    'zeroes': (x == 0).sum(),  
    'mean': x.mean(),  
    'median': x.median(),  
    'standard deviation': x.std(),  
    'inter-quartile range': x.quantile([.25,.75]).diff().iloc[-1],  
    'skewness': x.skew(),  
    'kurtosis': x.kurt(),  
    'quantiles': list(x.quantile([.25,.5,.75])),  
    'min': x.min(),  
    'max': x.max(),  
}
```

Numbers

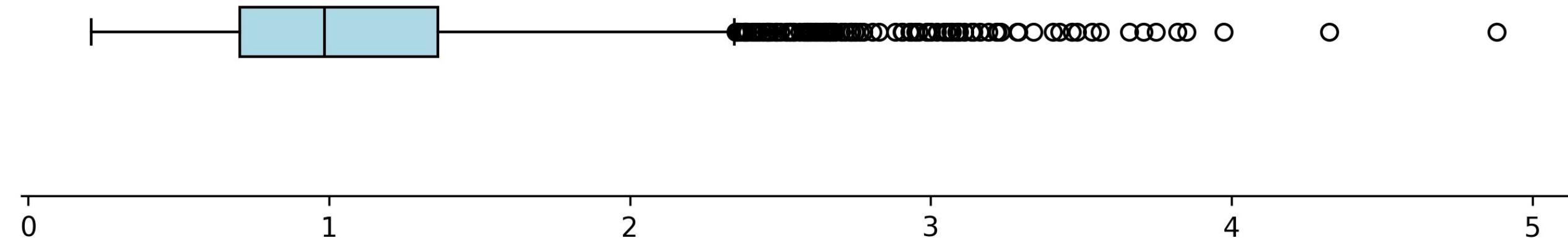
```
{  
    'dtype': str(x.dtype),  
    'observations': len(x),  
    'missing': x.isnull().sum(),  
    'mode': list(x.mode()),  
    'mode occurrences': (x == x.mode()[0]).sum(),  
    'zeroes': (x == 0).sum(),  
    'mean': x.mean(),  
    'median': x.median(),  
    'standard deviation': x.std(),  
    'inter-quartile range': x.quantile([.25,.75]).diff().iloc[-1],  
    'skewness': x.skew(),  
    'kurtosis': x.kurt(),  
    'quantiles': list(x.quantile(.25,.5,.75))),  
    'min': x.min(),  
    'max': x.max(),  
}  
  
{  
    'dtype': 'int64',  
    'observations': 272,  
    'missing': 0,  
    'mode': [78],  
    'mode occurrences': 15,  
    'zeroes': 0,  
    'mean': 70.90,  
    'median': 76.0,  
    'standard deviation': 13.59,  
    'inter-quartile range': 24.0,  
    'skewness': -0.42,  
    'kurtosis': -1.14,  
    'quantiles': [58.0, 76.0, 82.0],  
    'min': 43,  
    'max': 96,  
}
```

Plots

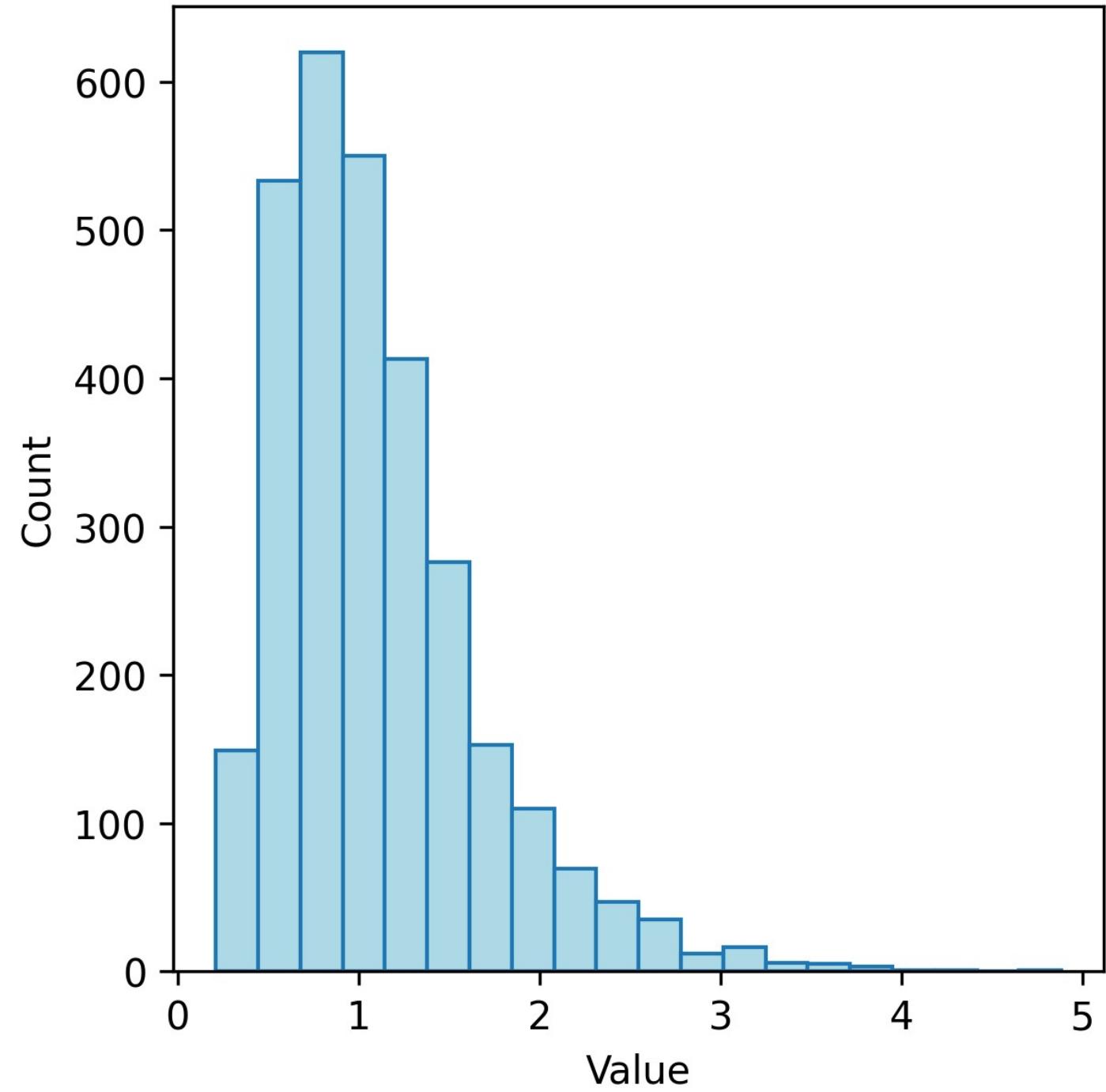
- Boxplot
- Histogram
- Density, density with a logarithmic scale
- Quantile-quantile plot



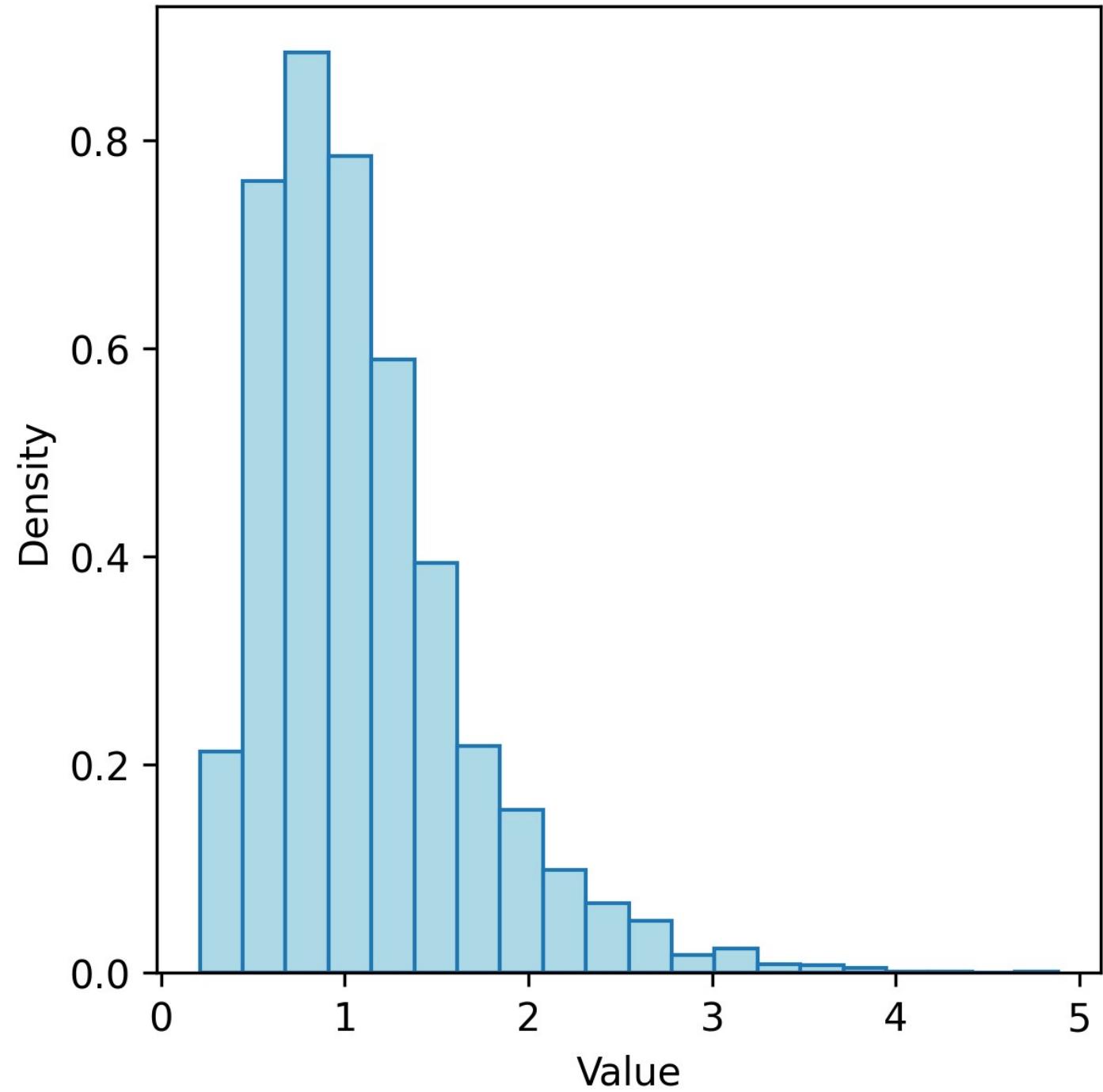
```
plt.boxplot(x)
```



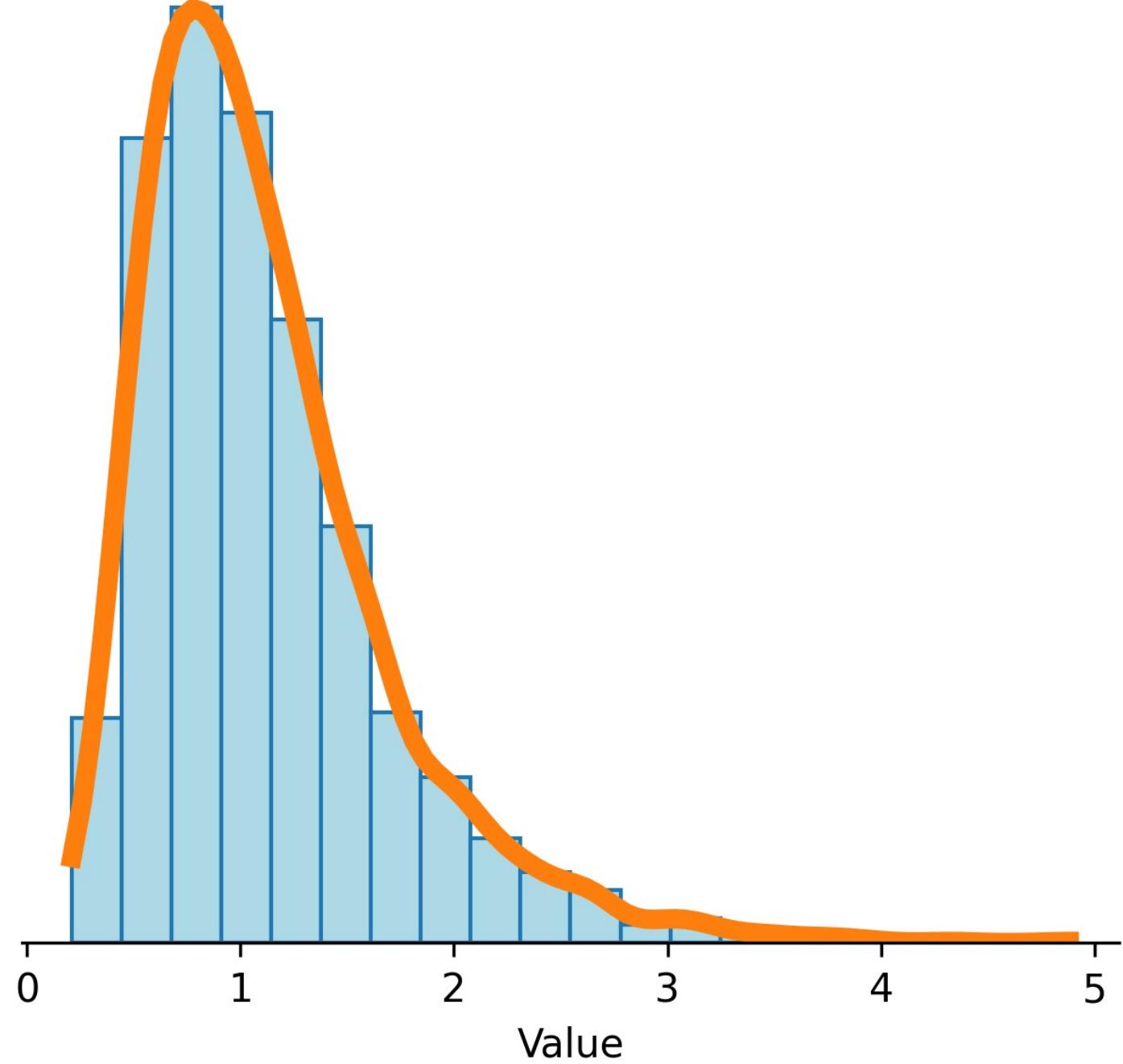
```
plt.hist(x)
```



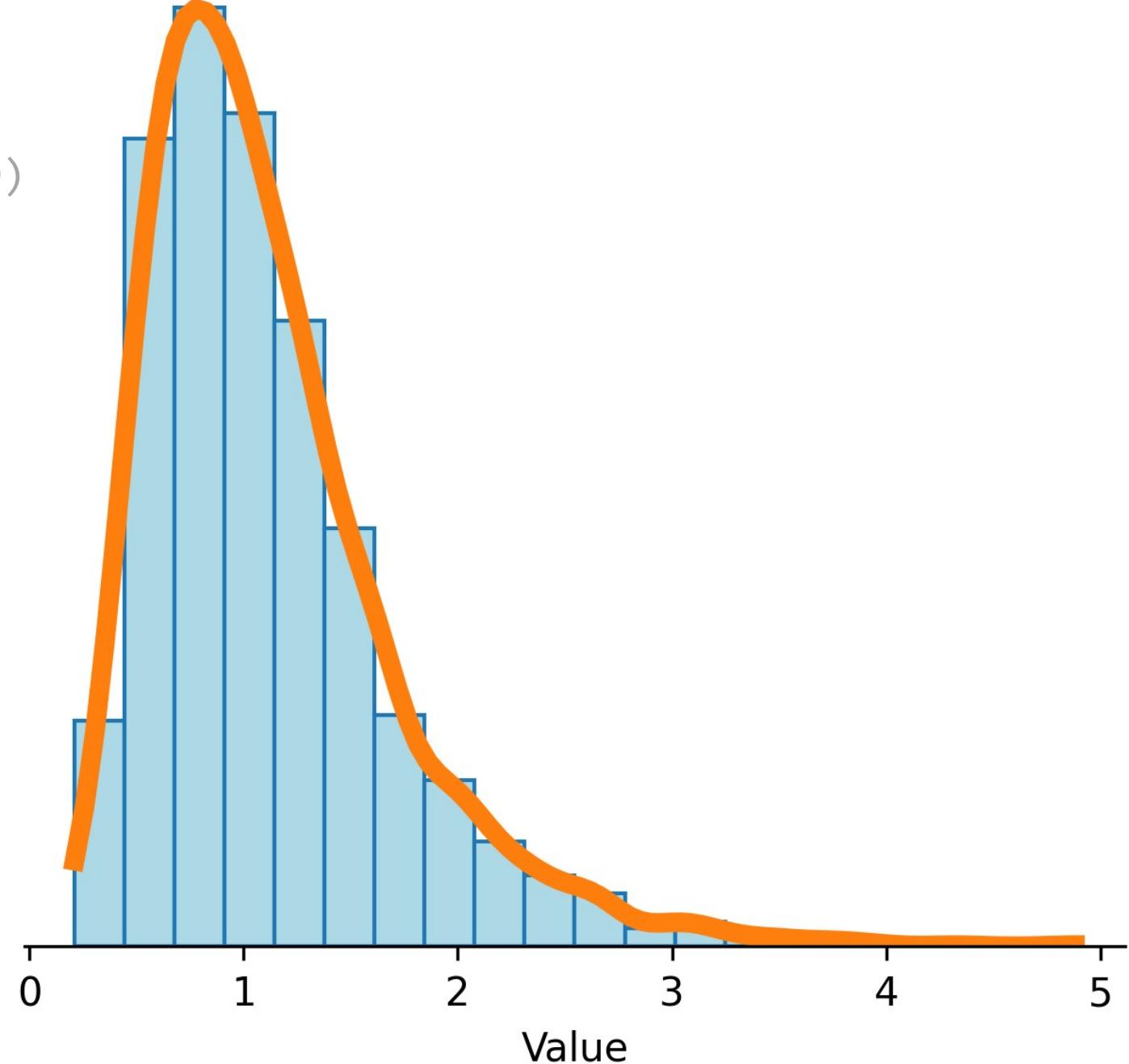
```
plt.hist(x, density=True)
```



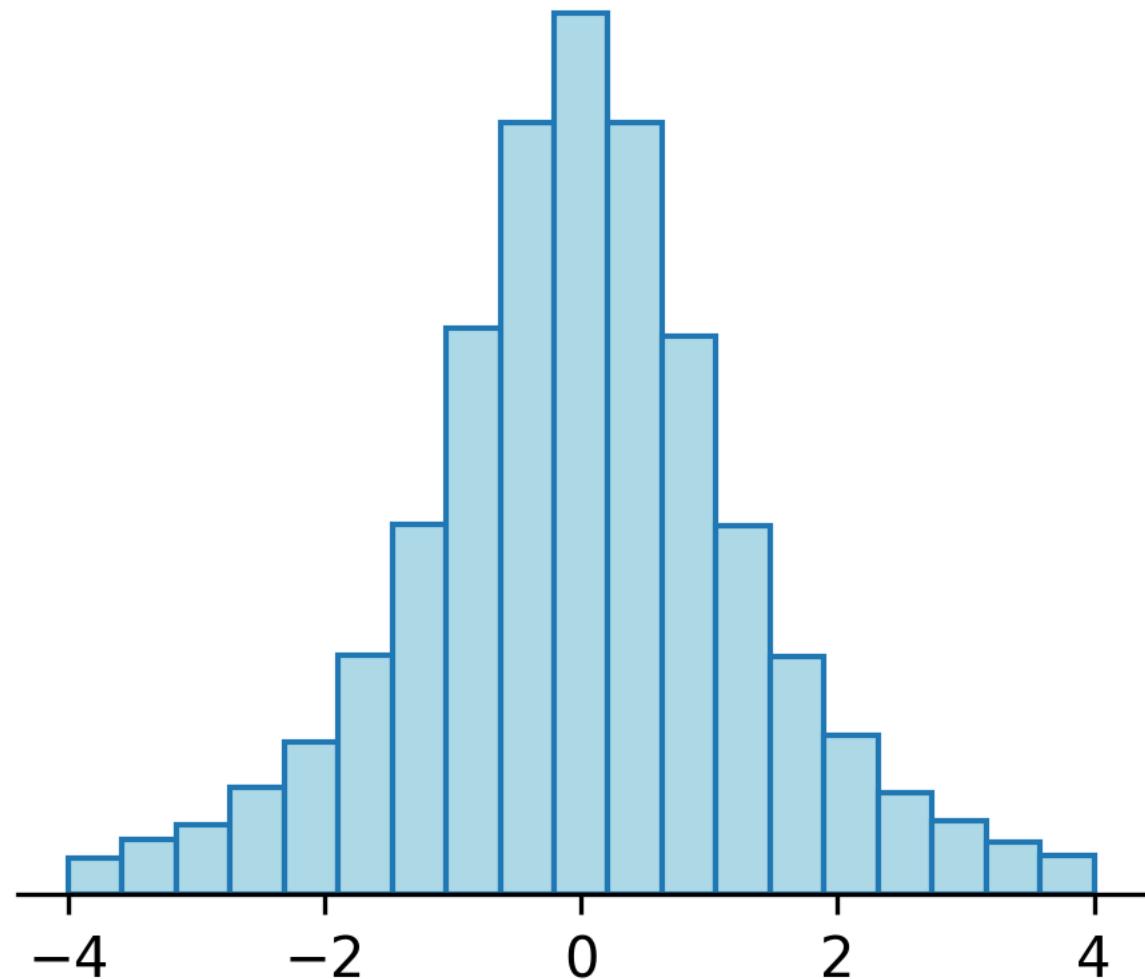
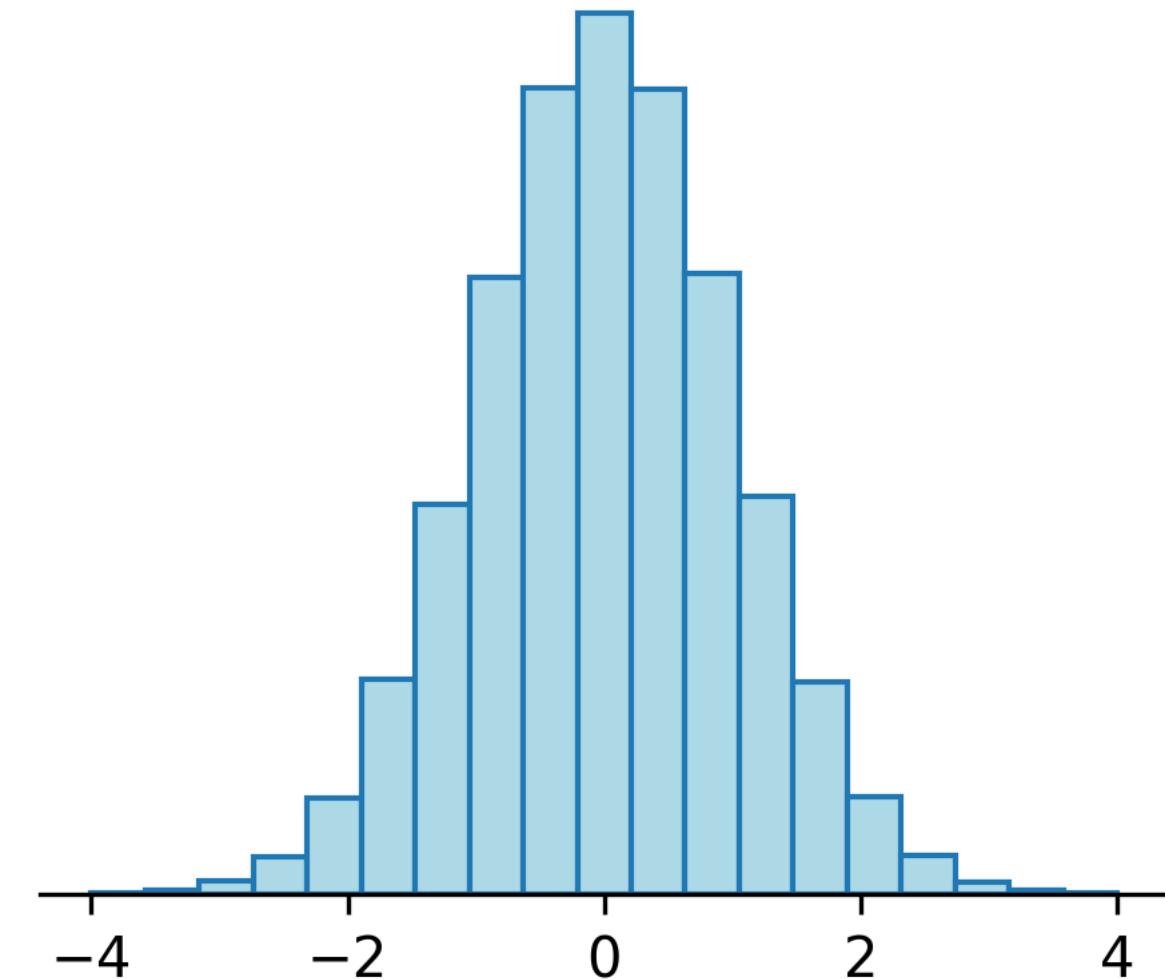
```
x.plot.hist( density = True )  
x.plot.density()
```



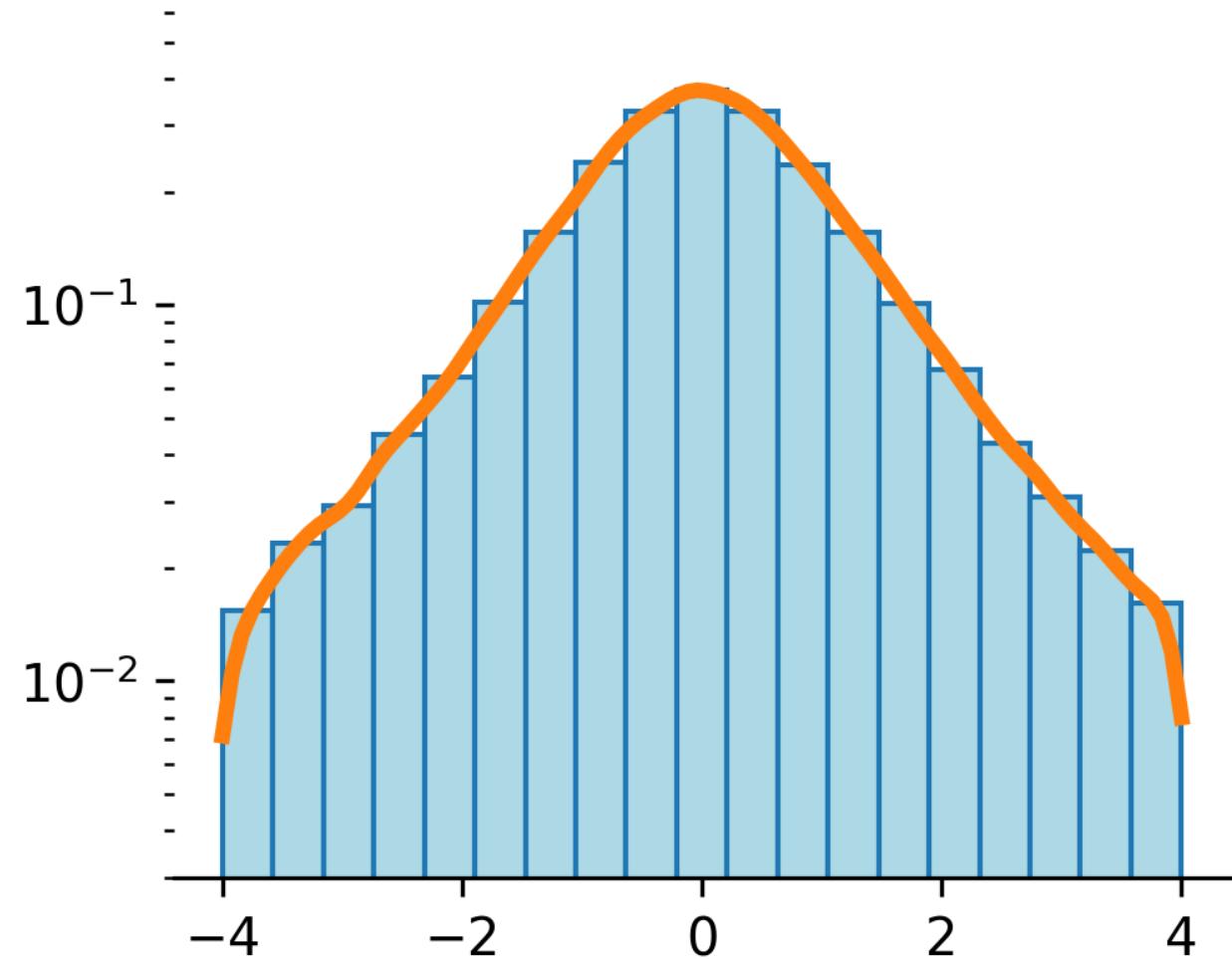
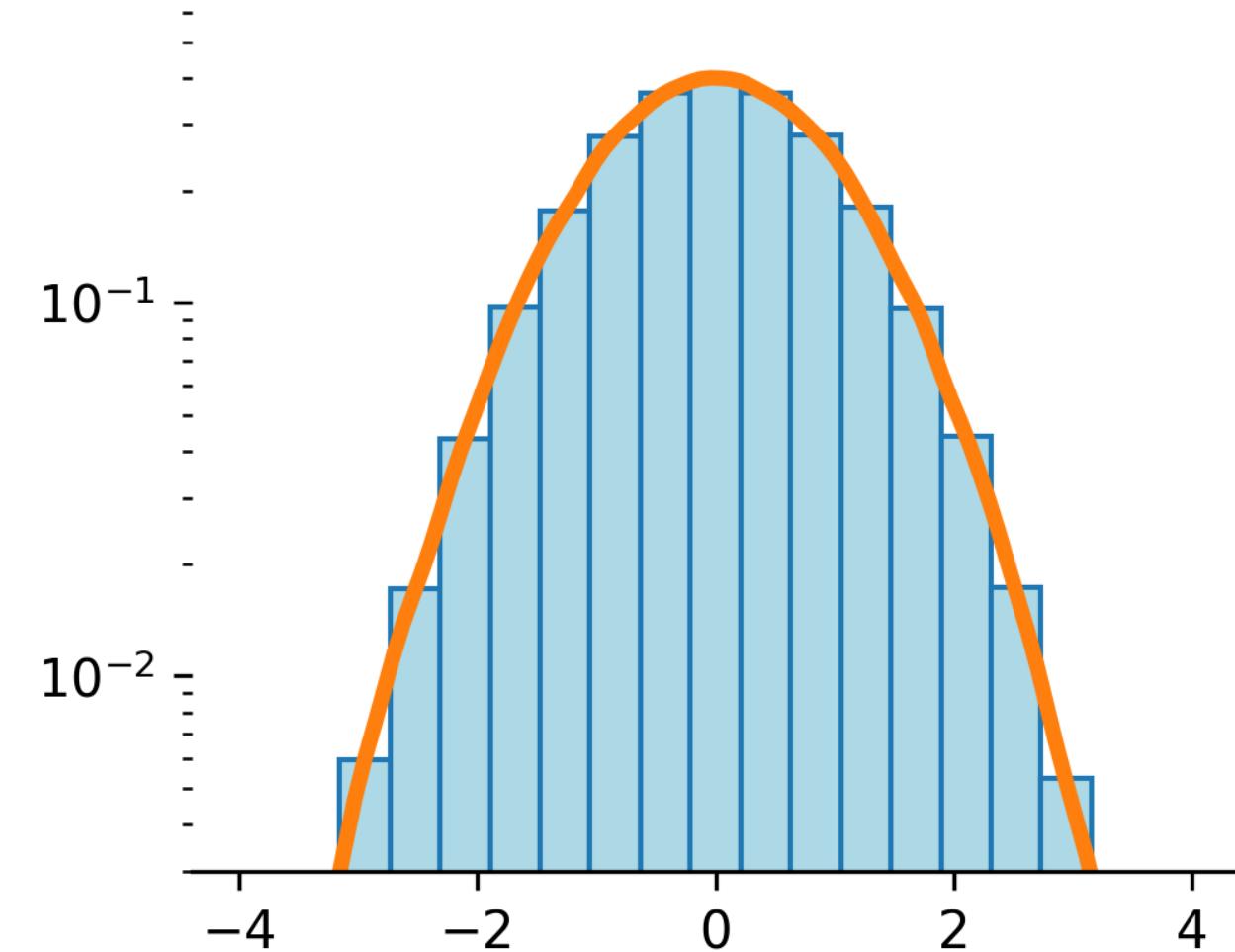
```
plt.hist( x, density = True )
xs = np.linspace(x.min(), x.max(), 100)
de = scipy.stats.gaussian_kde(x)
ys = de(xs)
plt.plot( xs, ys, linewidth = 5 )
```



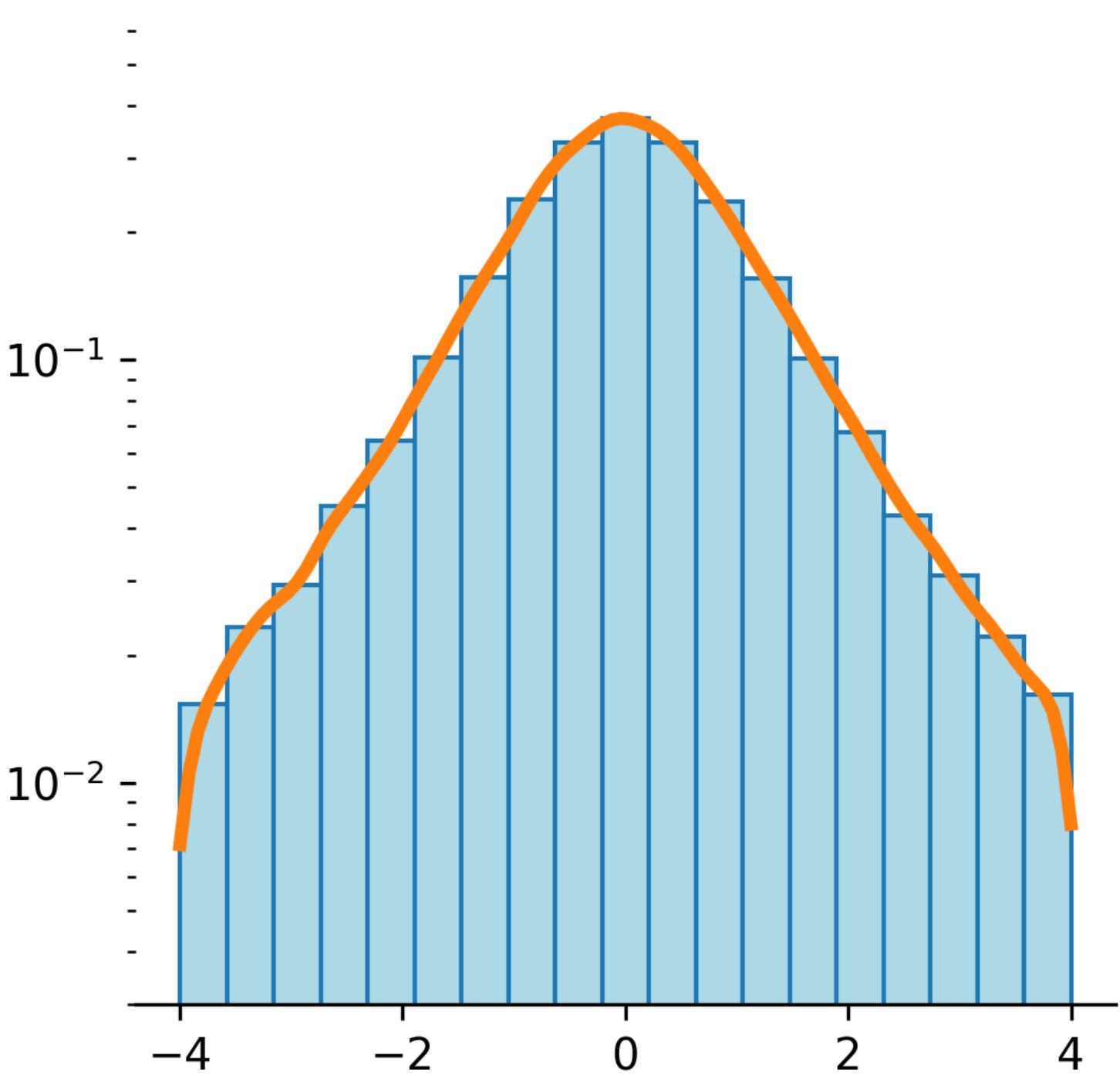
Can you spot the difference?



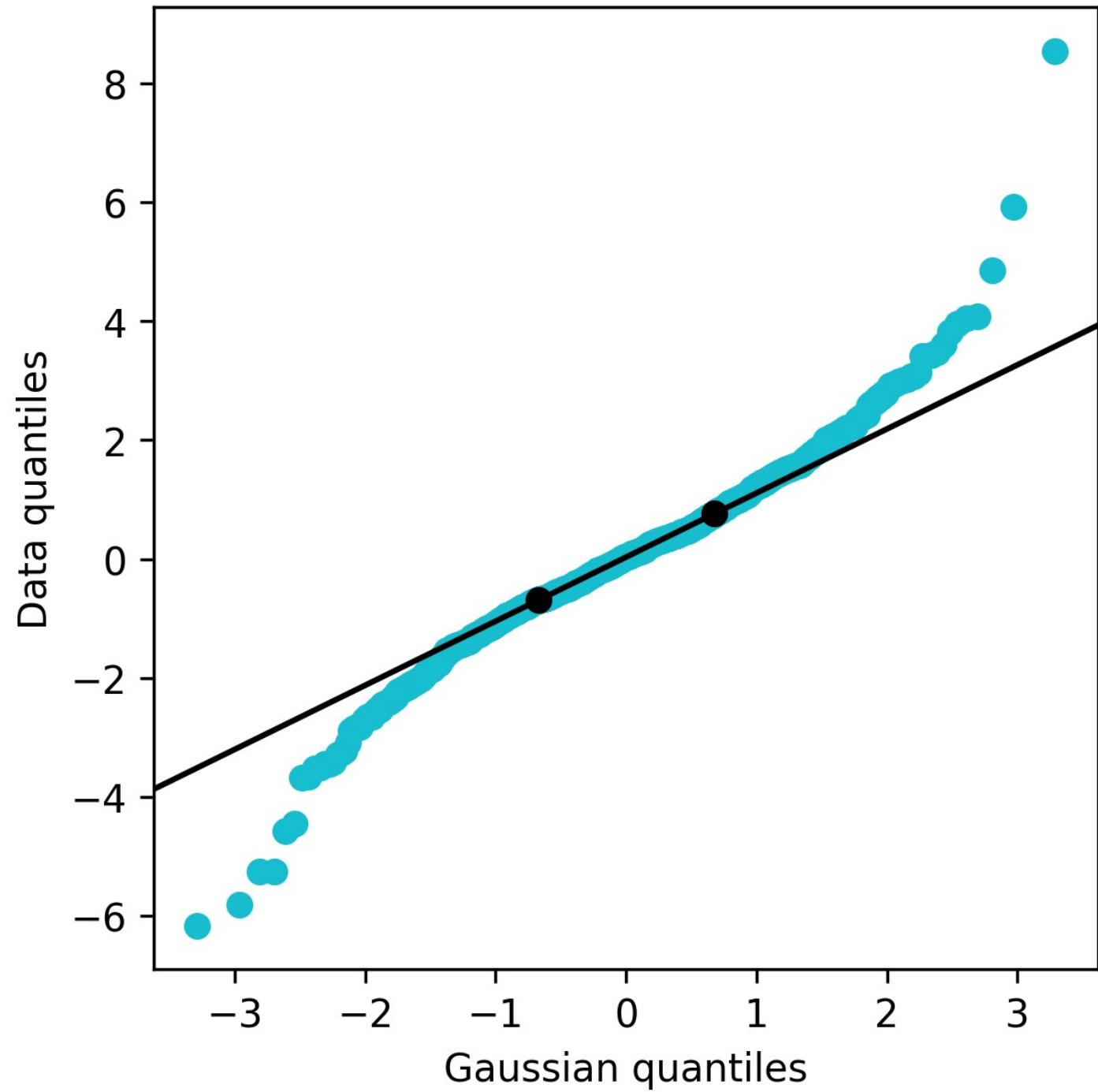
Can you spot the difference?



```
x = pd.Series(x)
x.plot.hist(density=True)
x.plot.density()
plt.yscale('log')
plt.ylim(1e-3,1)
```



```
n = len(x)  
D = scipy.stats.norm()  
ps = np.linspace(0, 1, 2*n+1)[1::2]  
ys = sorted(x)  
xs = D.ppf(ps)  
plt.scatter(xs, ys )
```

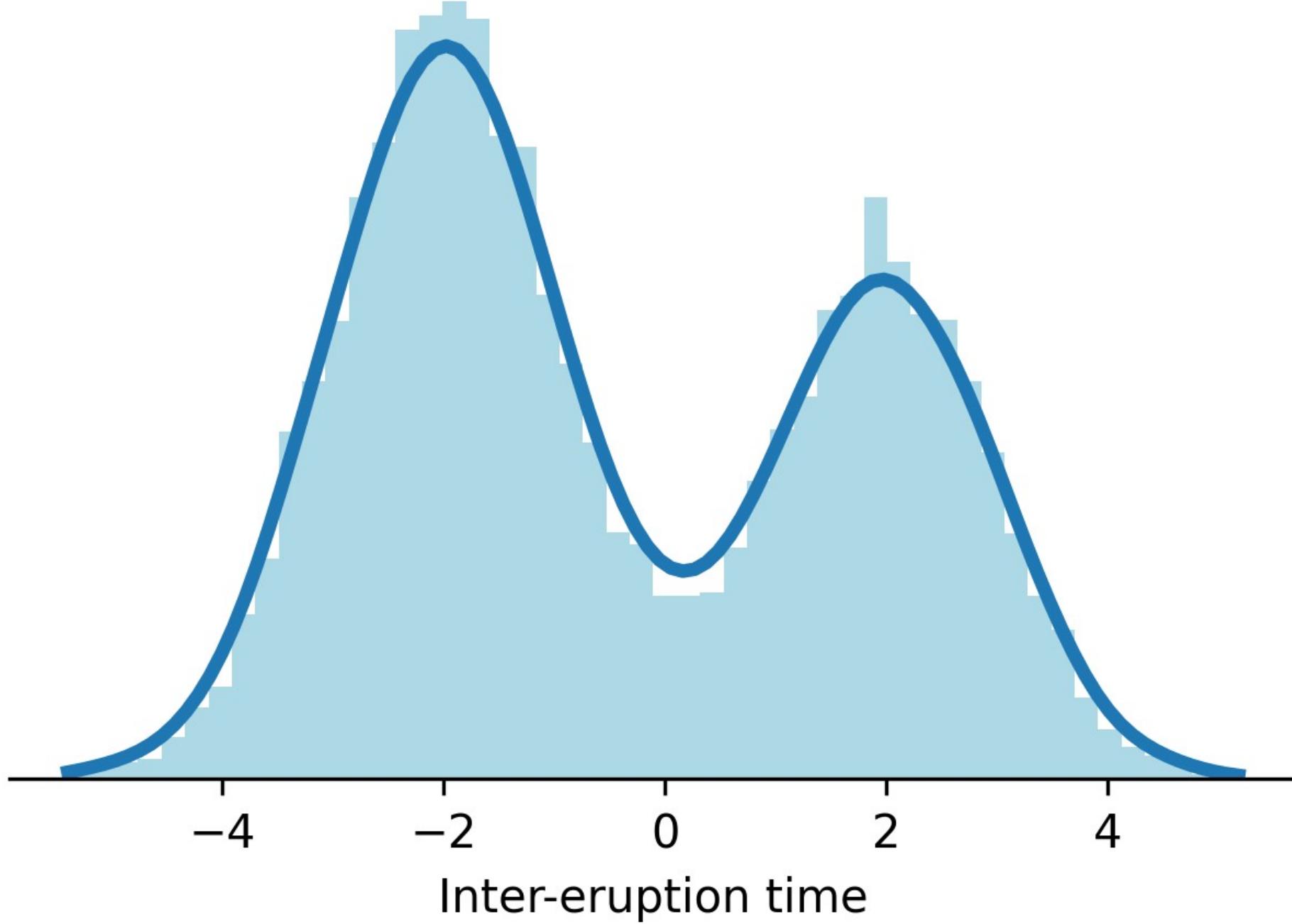


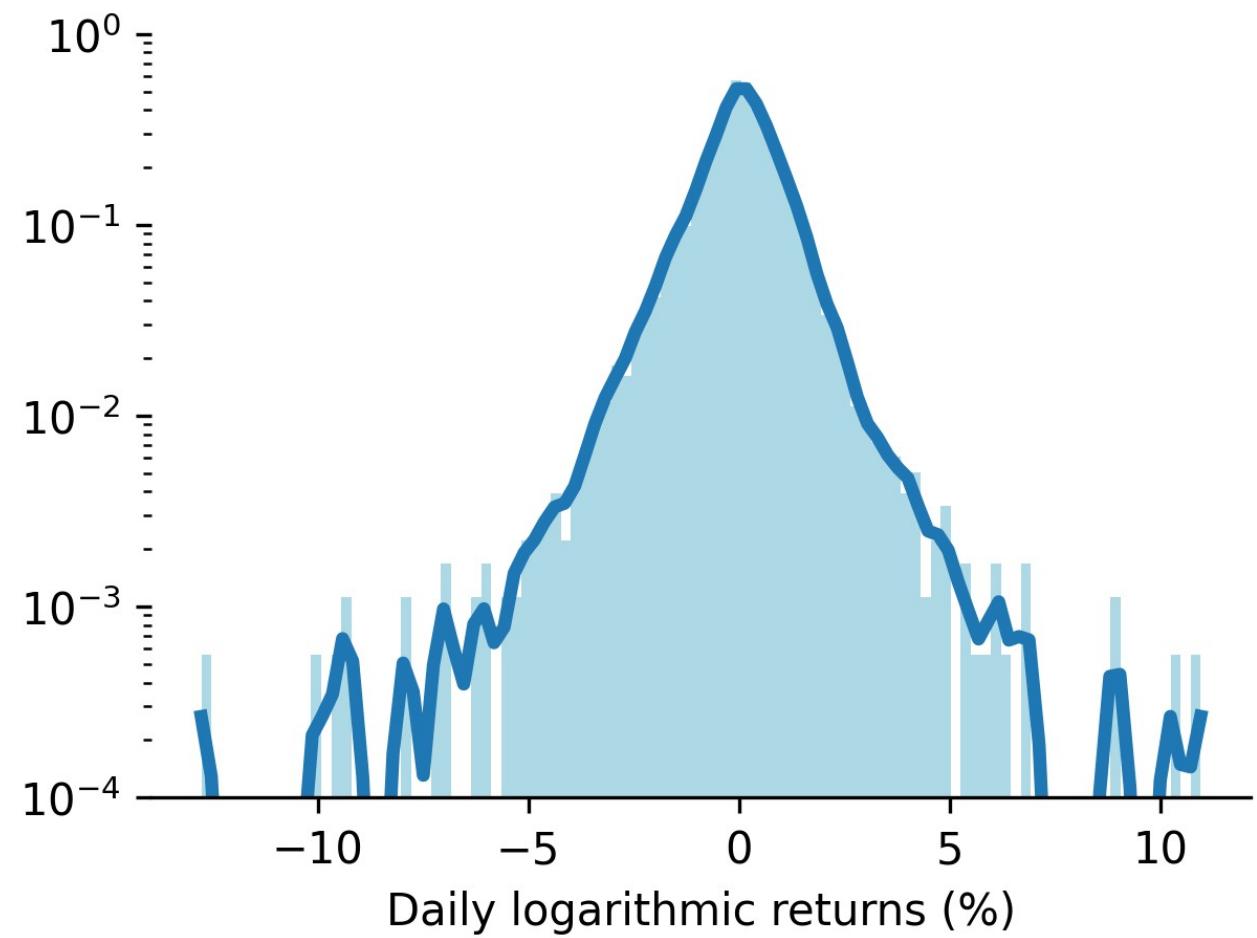
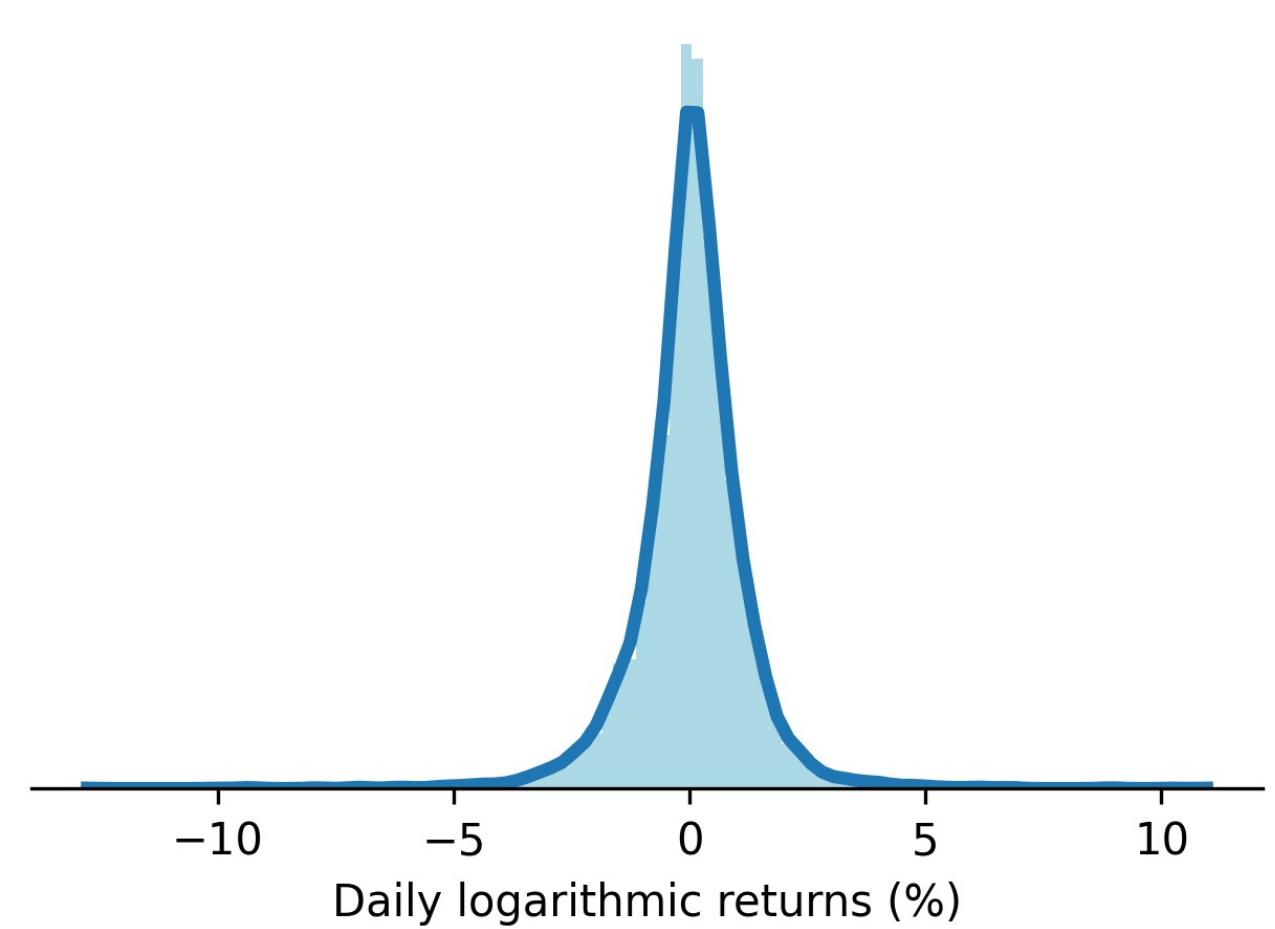
Exercise

Exercise

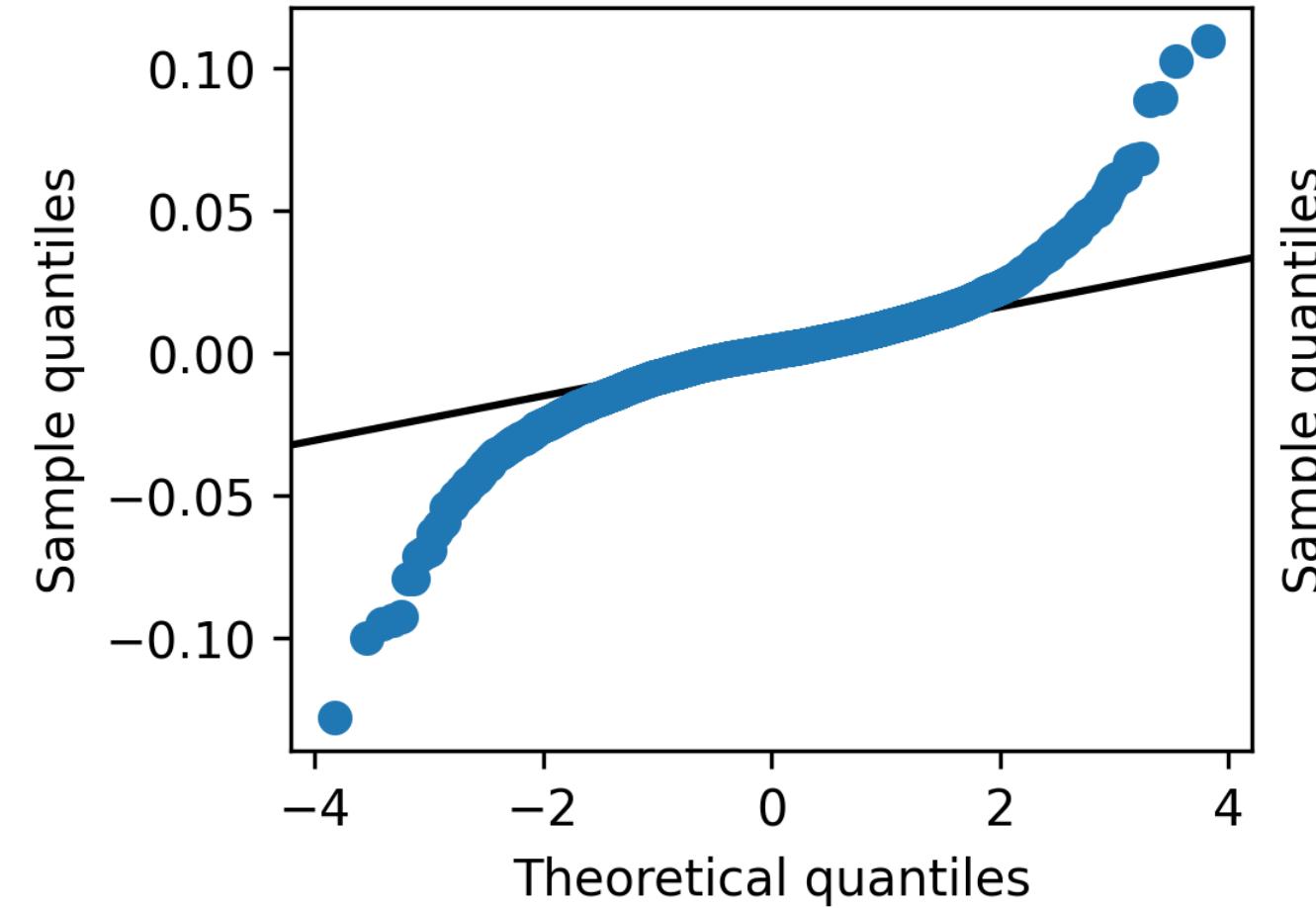
For each dataset:

- Compute the numeric summary
- Plot a boxplot, a histogram (with a linear and a logarithmic scale), a density
- Which of the problems mentioned above are present?

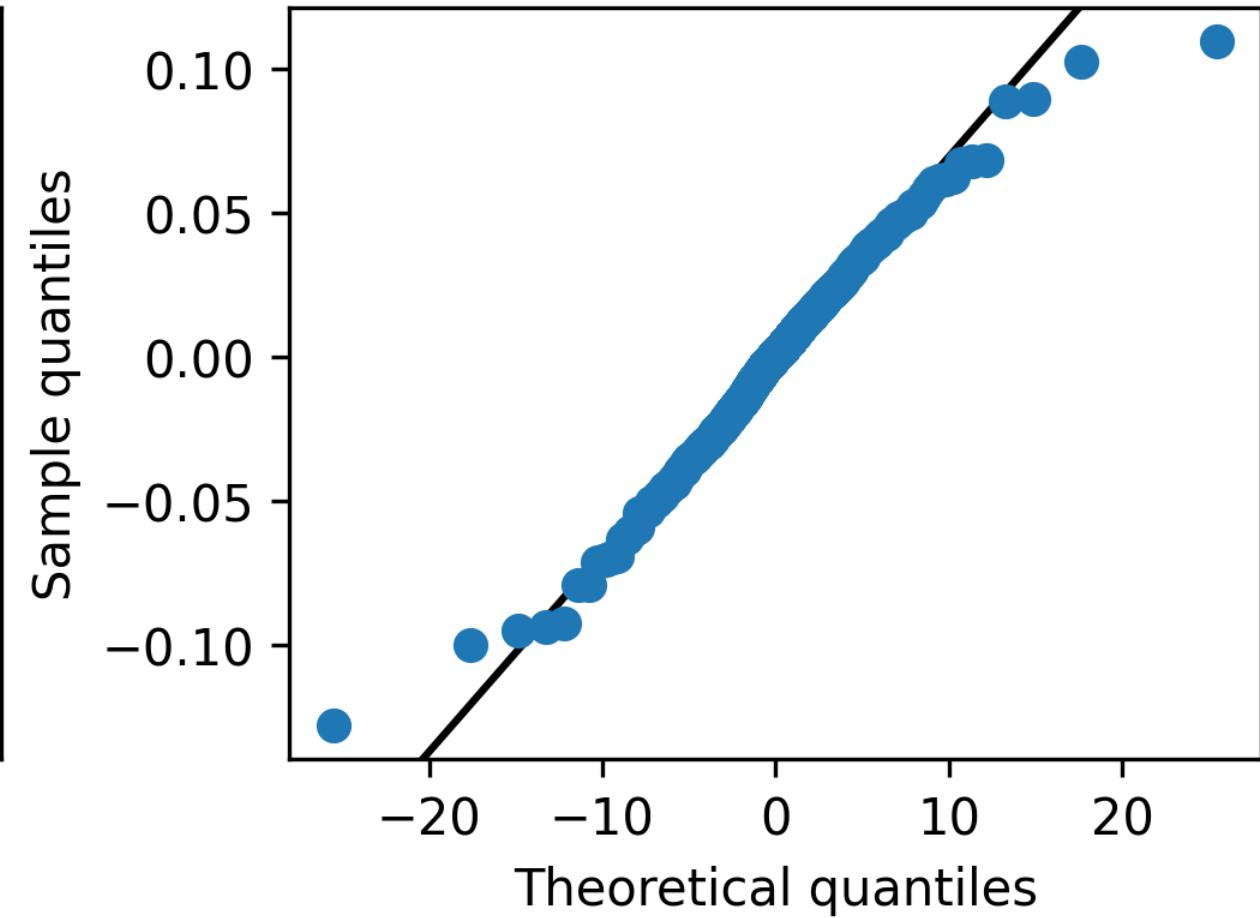




Comparison with
a Gaussian distribution



Comparison with
a Student T($df=4$) distribution



Bivariate Data

Bivariate data

- Two lists of numbers;
a 2-column array
of numbers

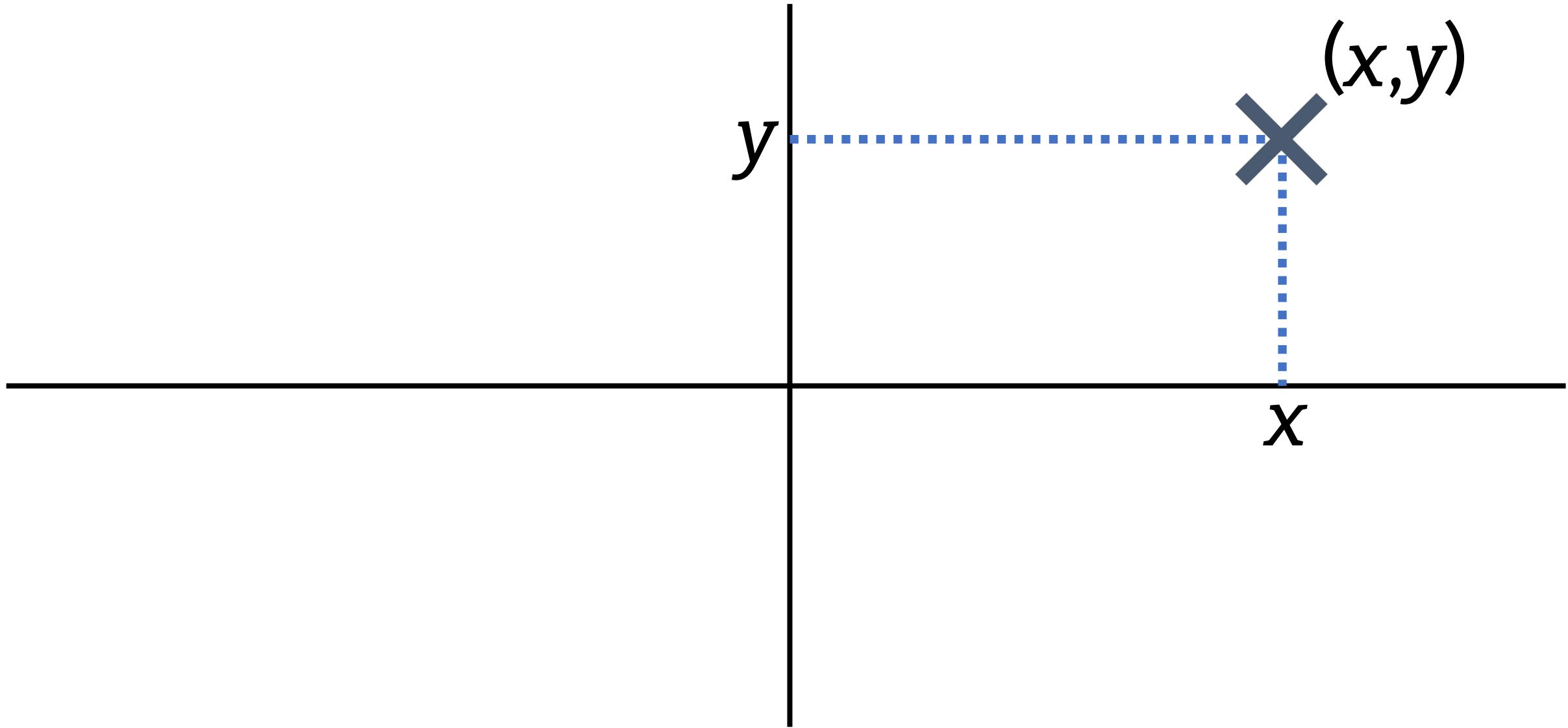
long	lat
181.62	-20.42
181.03	-20.62
184.10	-26.00
181.66	-17.97
181.96	-20.42
...	...

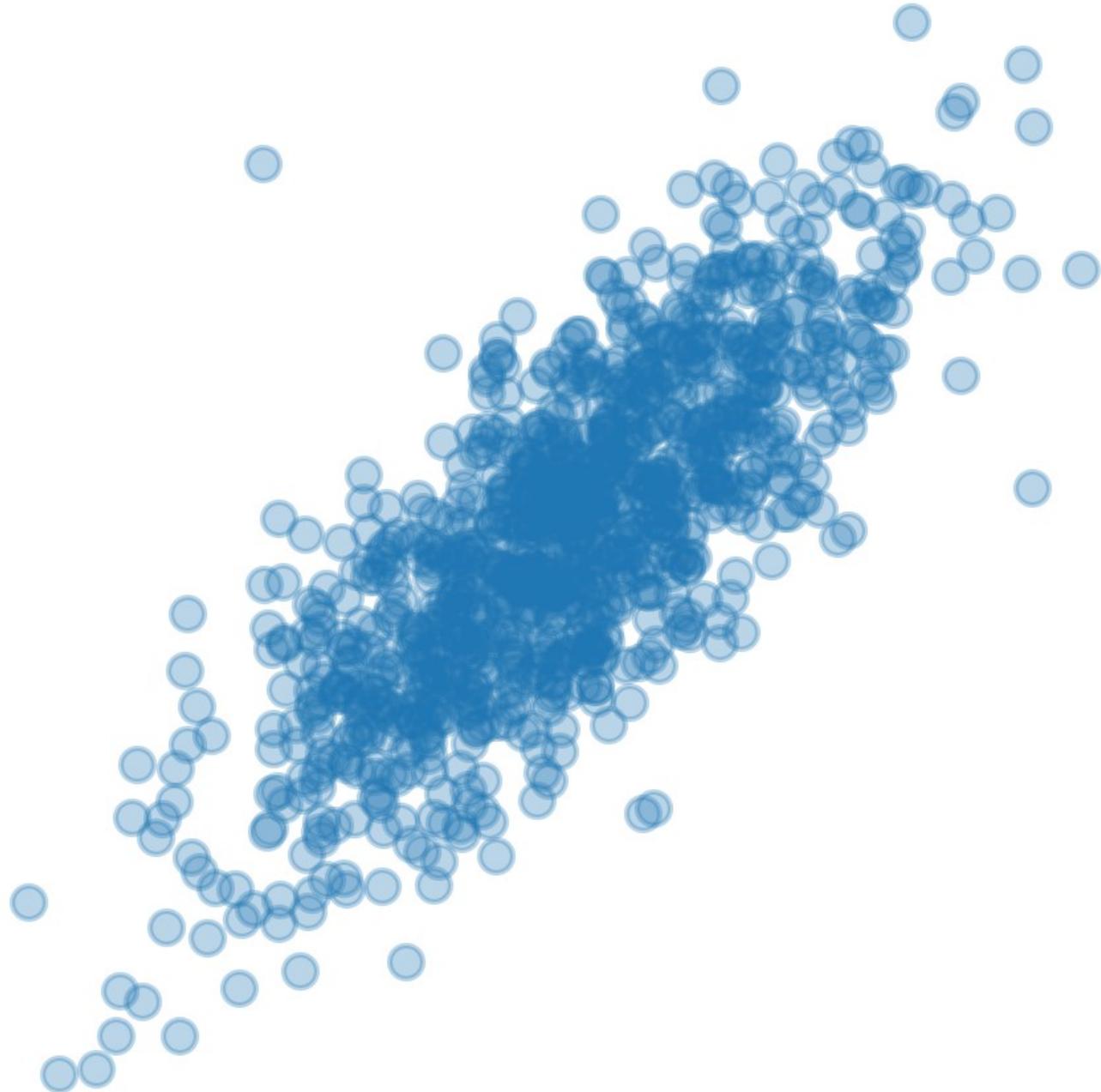
Bivariate data

- Two lists of numbers;
a 2-column array
of numbers

Lot Area (sq.ft.)	Sale Price (USD)
8450	208500
9600	181500
11250	223500
9550	140000
14260	250000
...	...

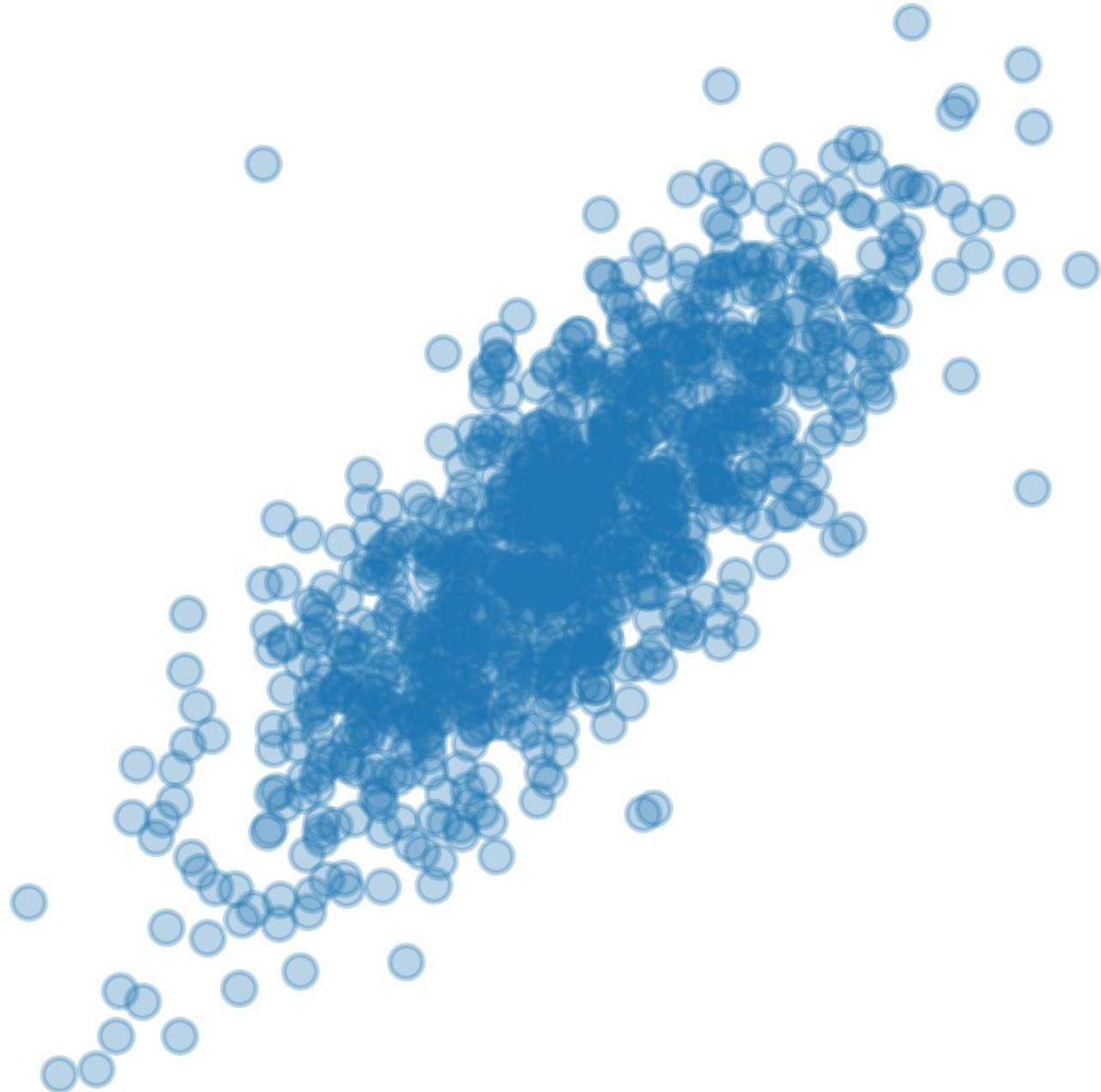
Coordinates

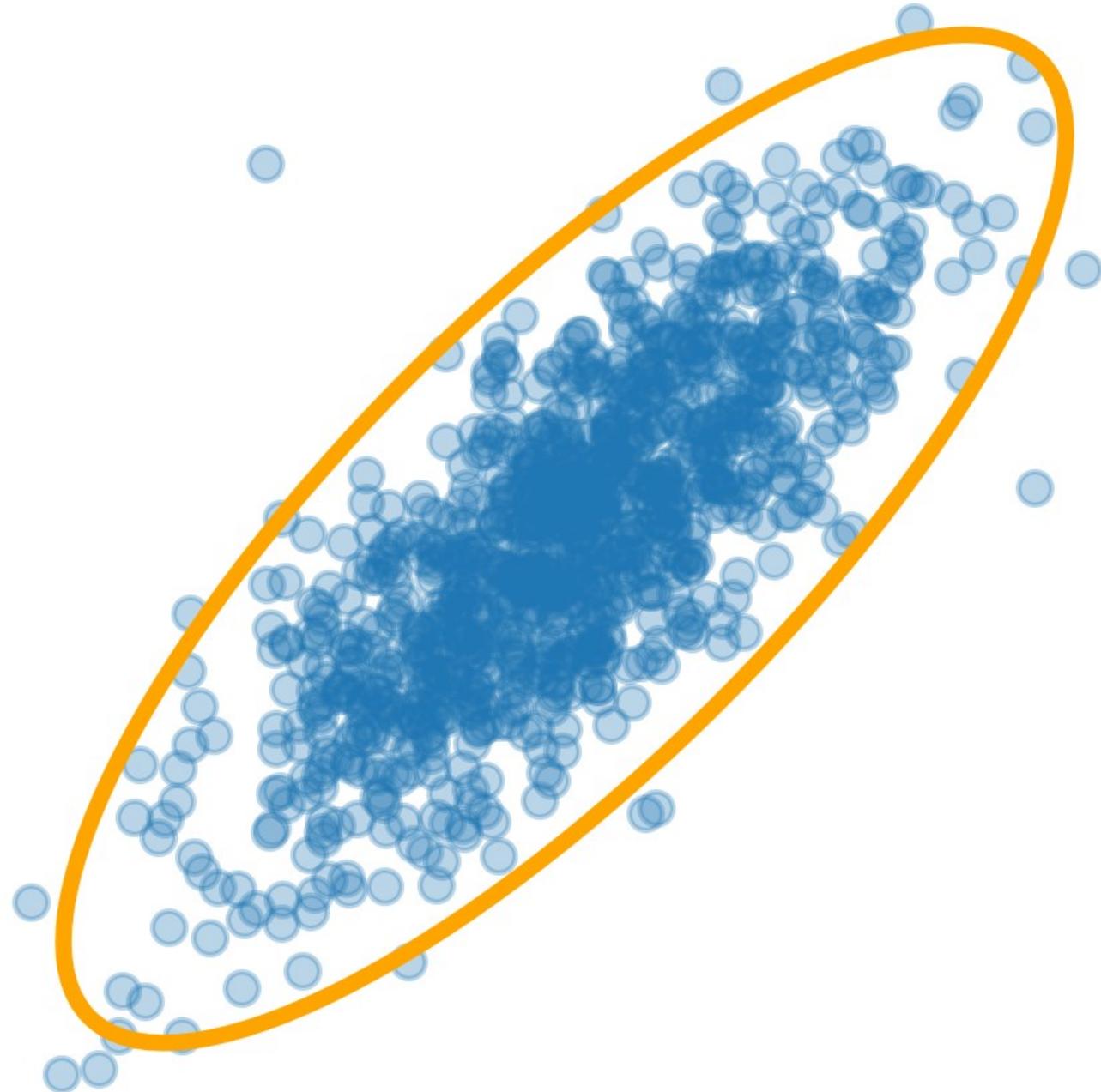


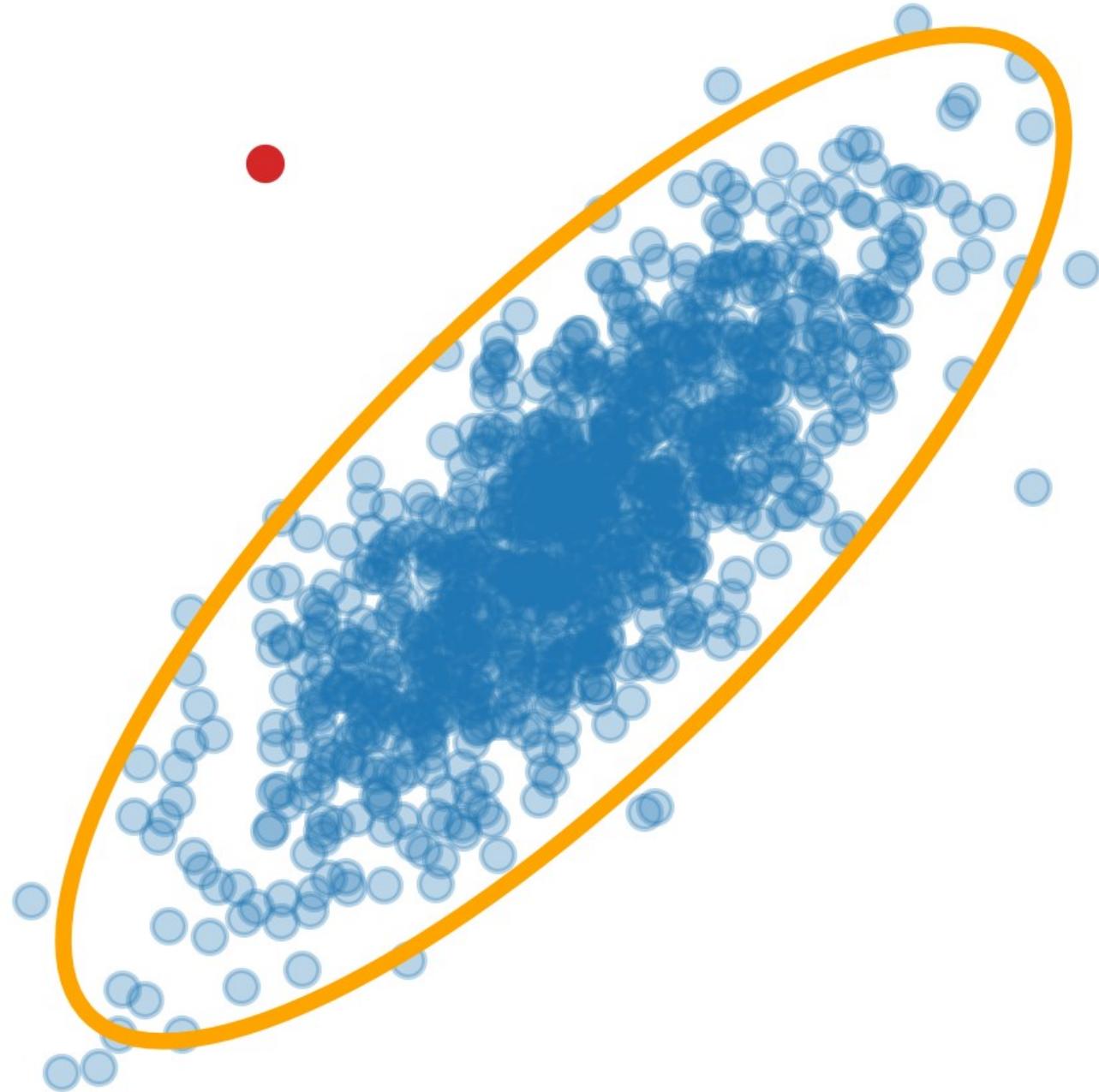


Potential Problems

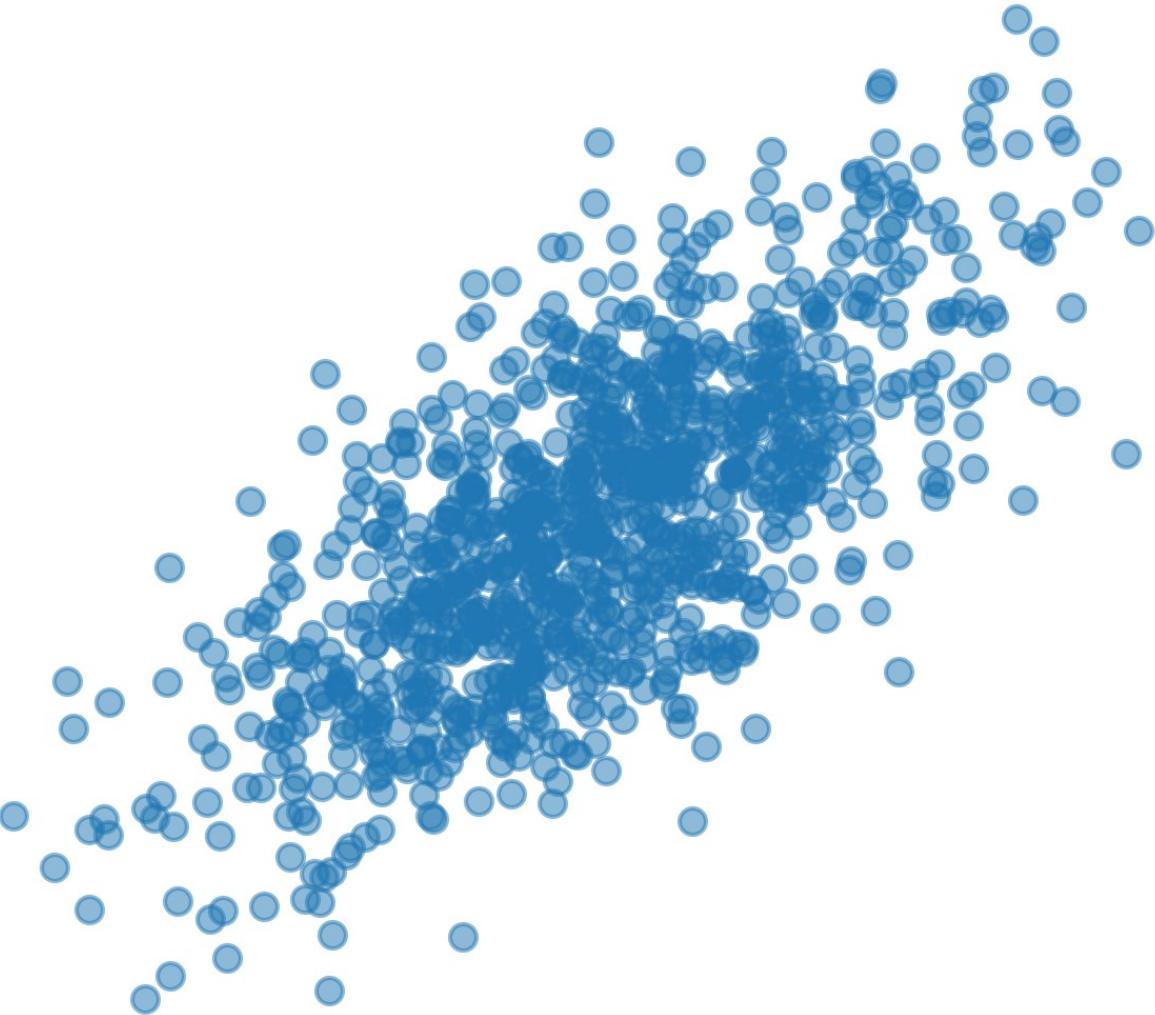
- Same as with univariate data
- (Bivariate) outliers
- Non-linearities
- Heteroskedasticity
- Missing values



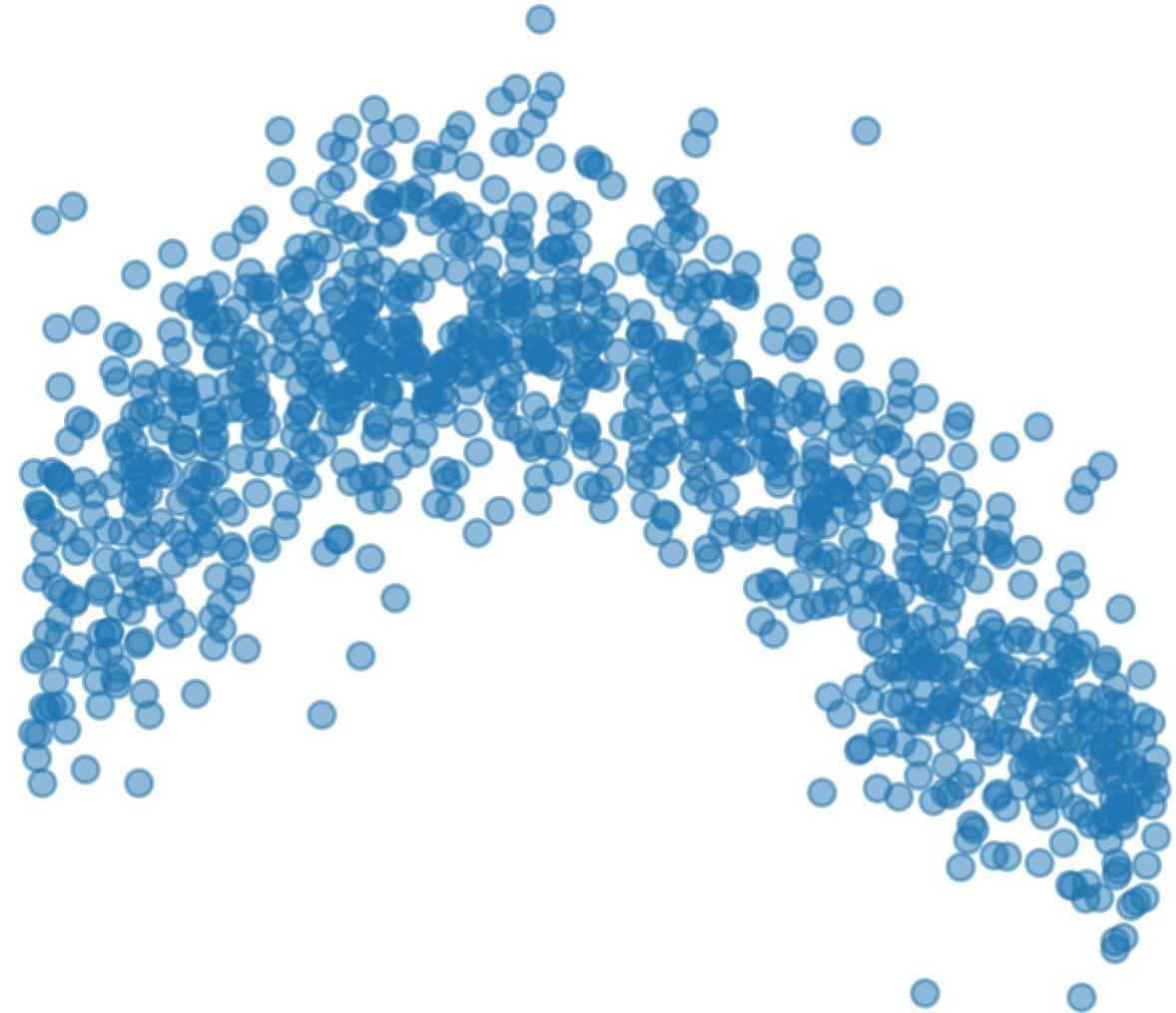




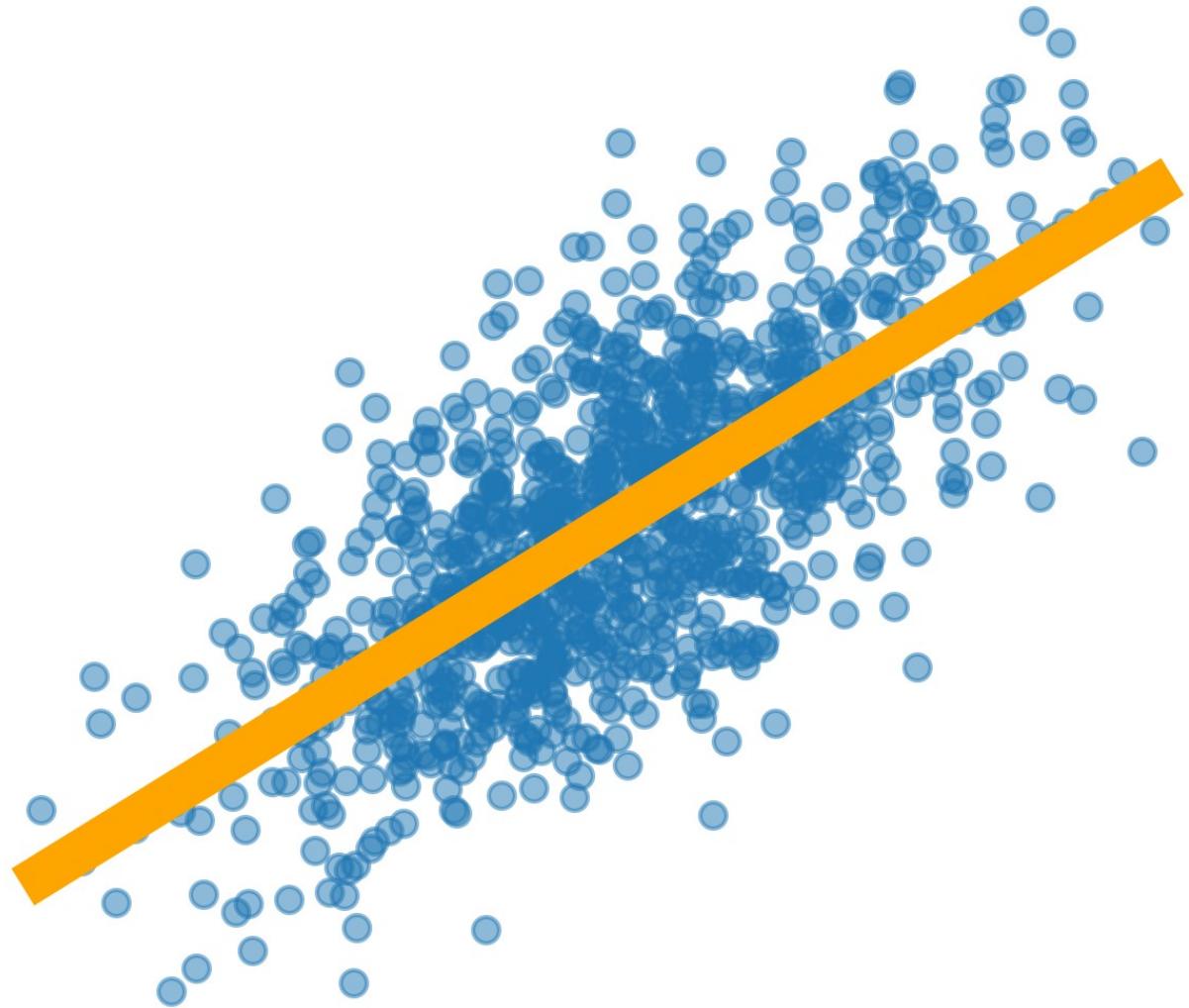
Linear



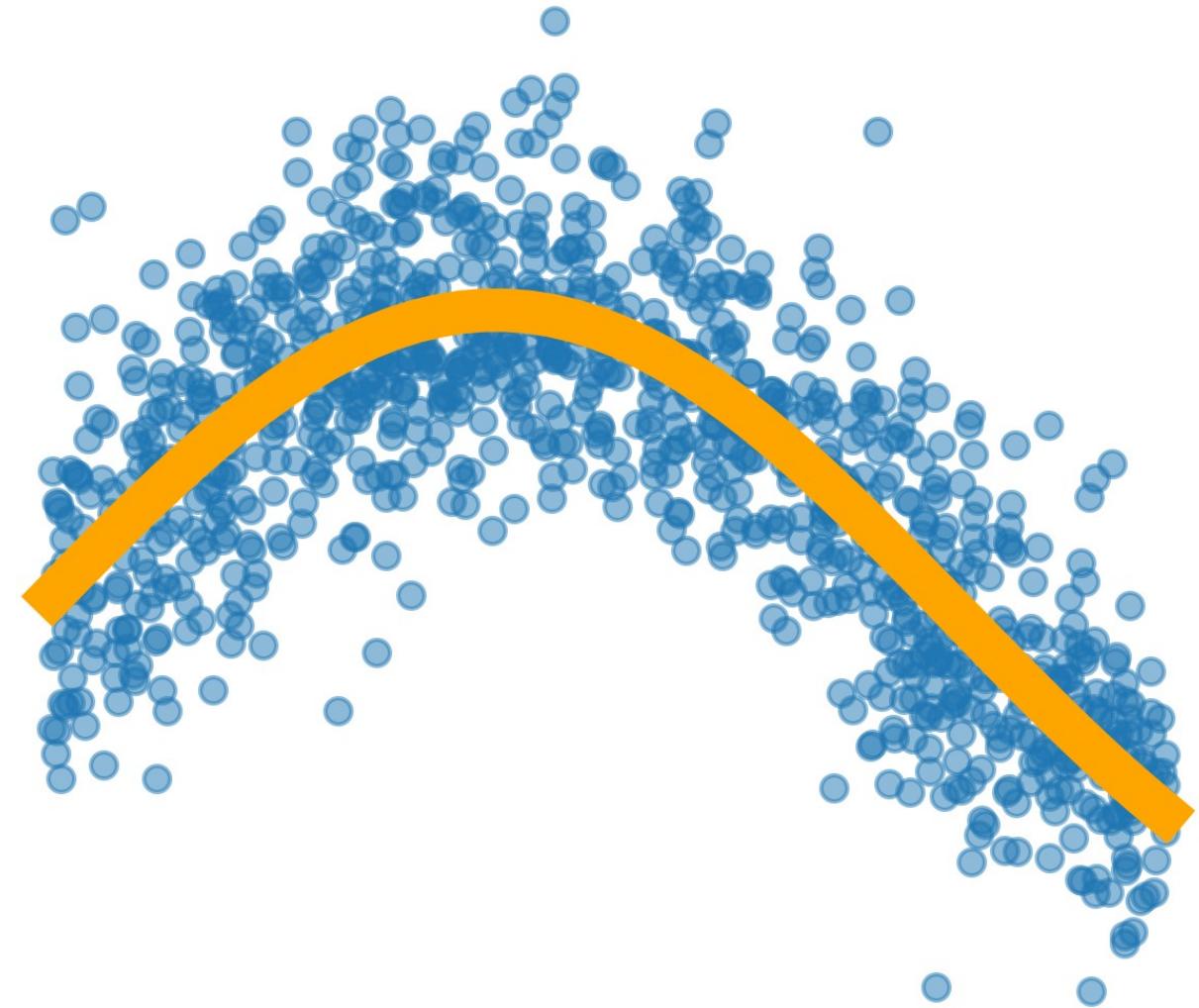
Non-Linear

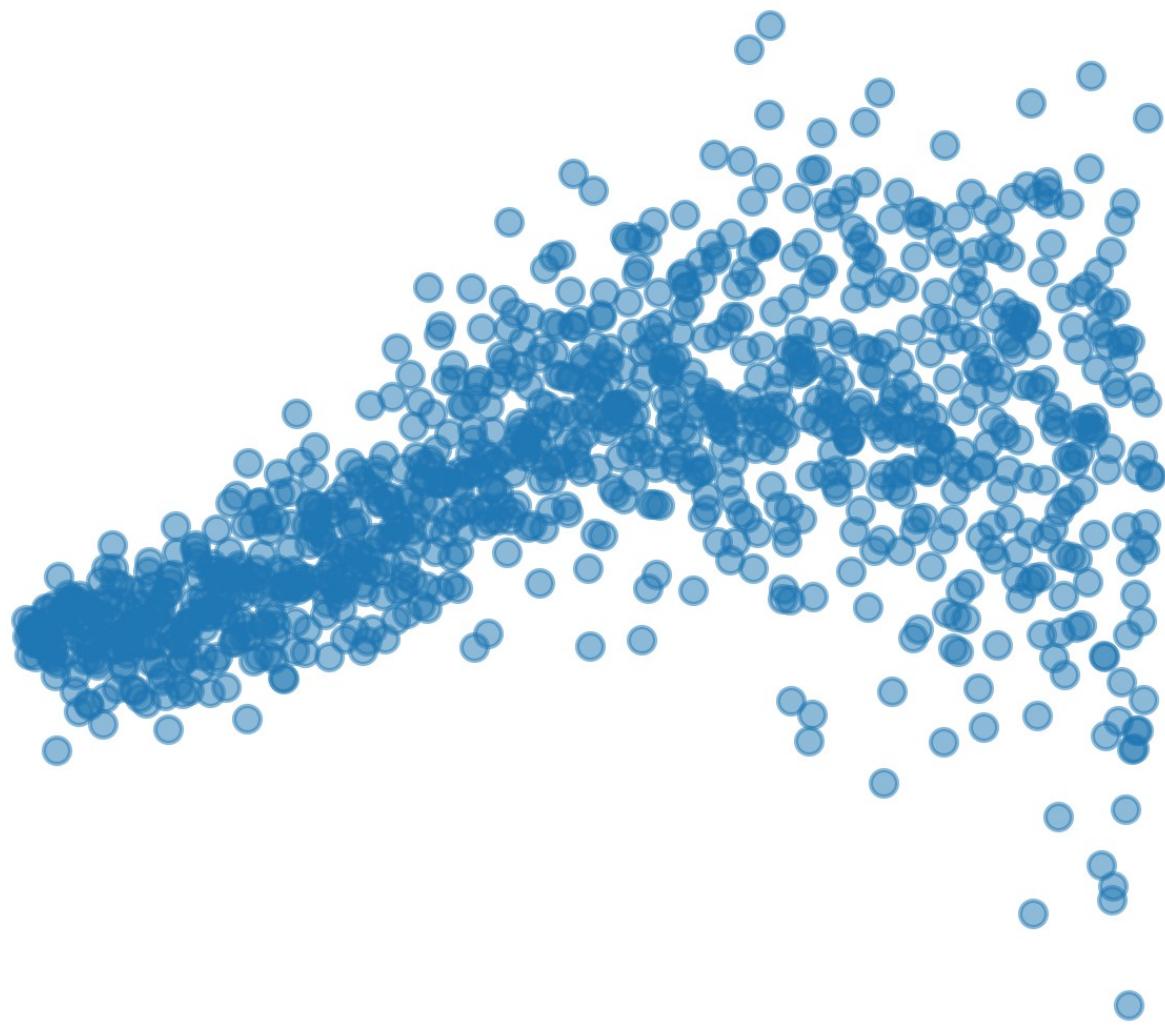
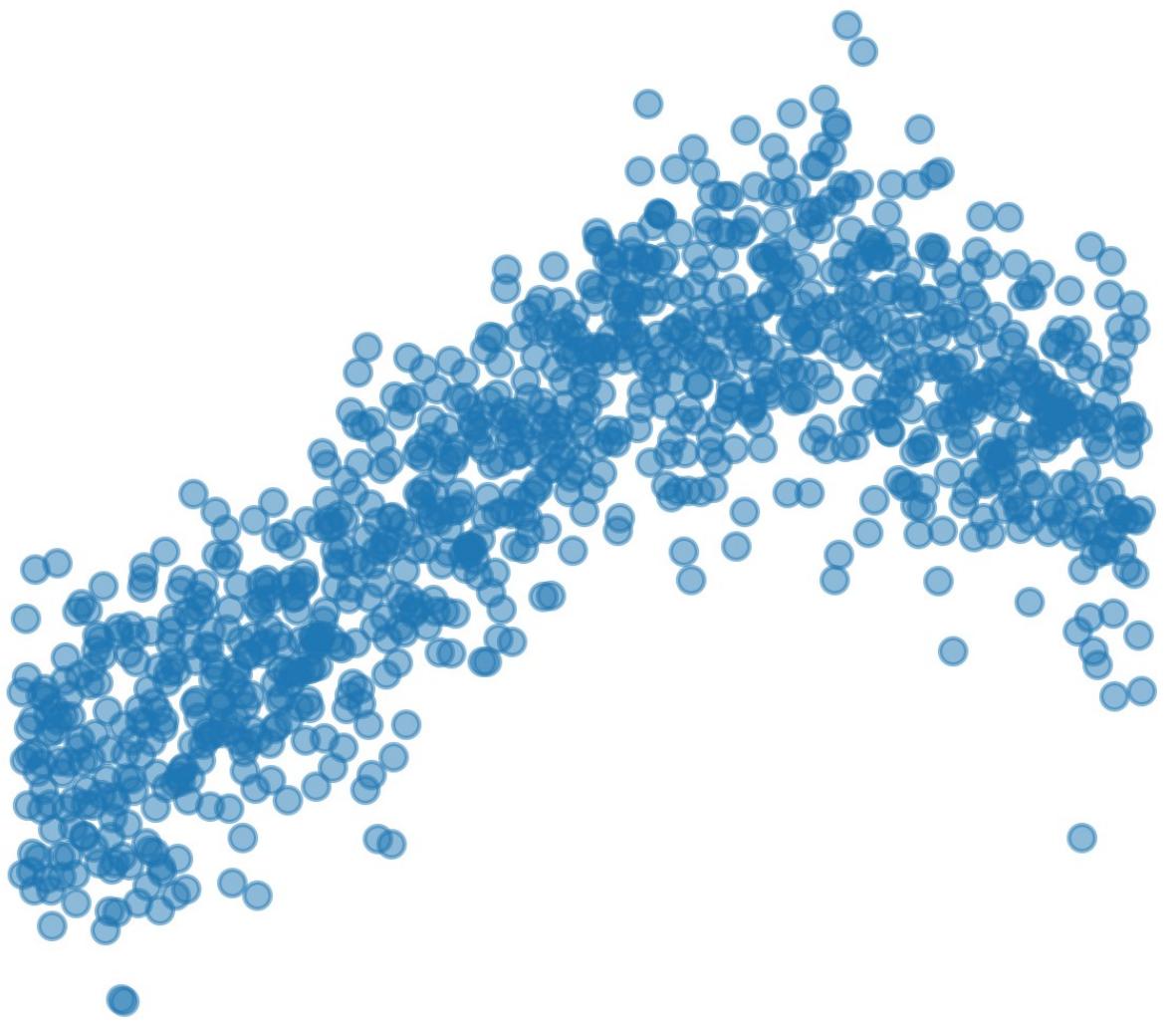


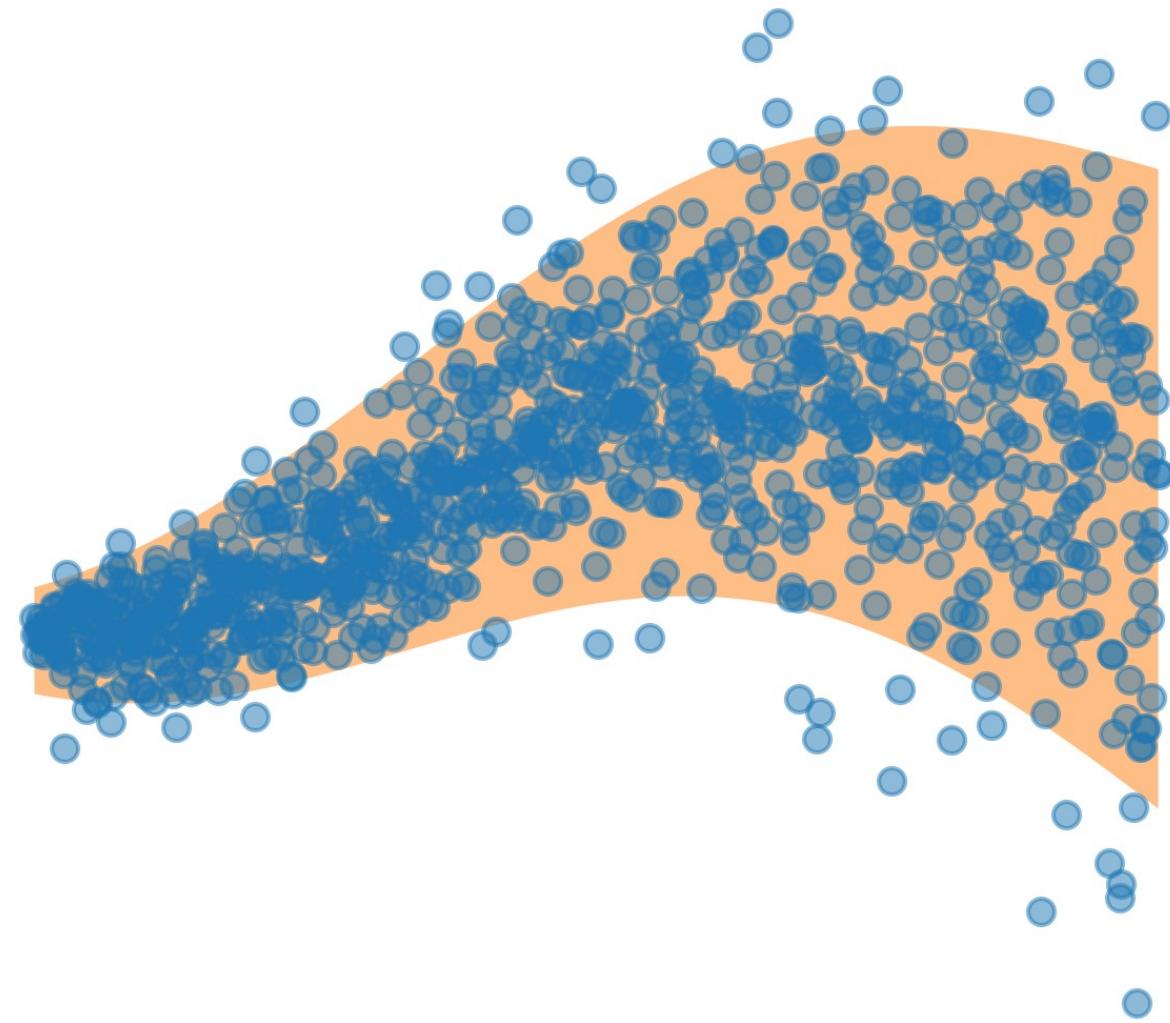
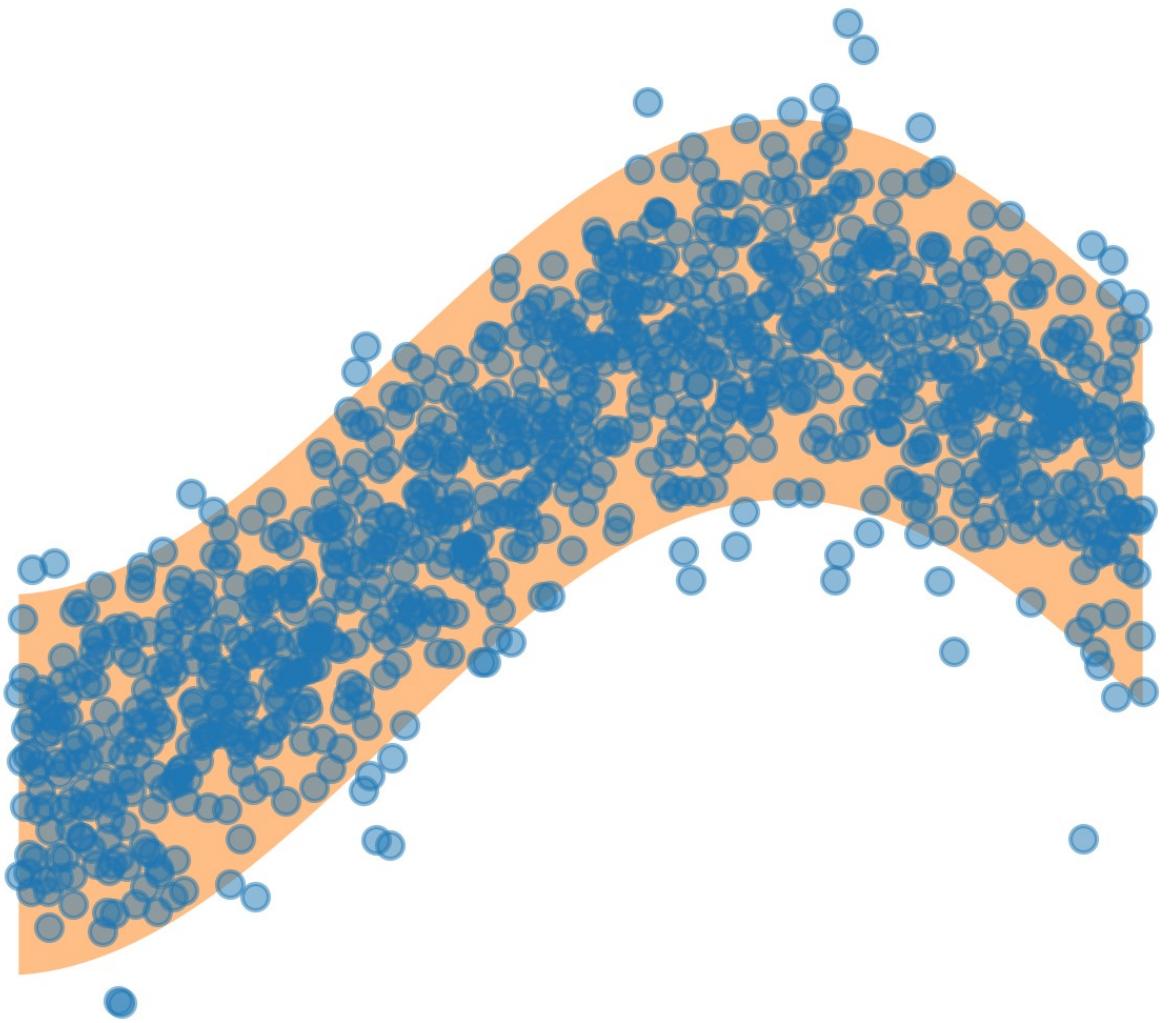
Linear



Non-Linear







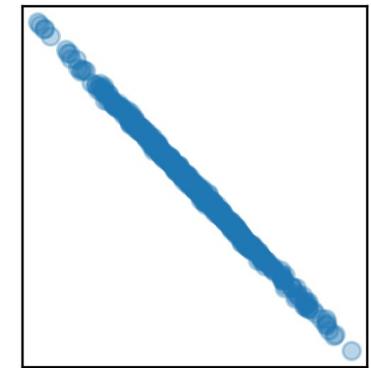
Numbers

- Same as for univariate data
- Correlation

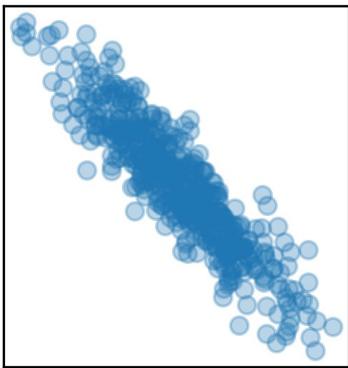
```
d.corr().iloc[0,1]
```

-0.36454403688646975

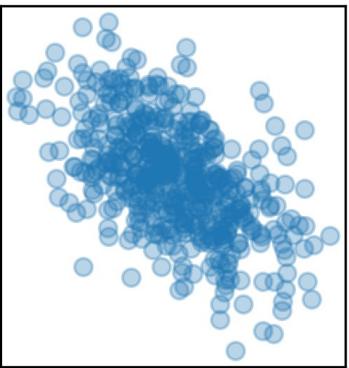
$\rho=-1.0$



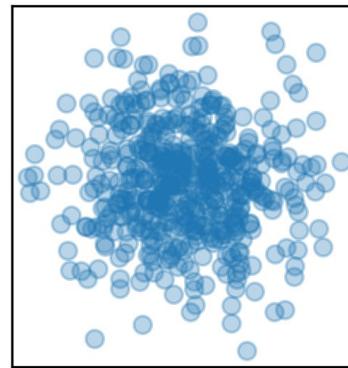
$\rho=-0.9$



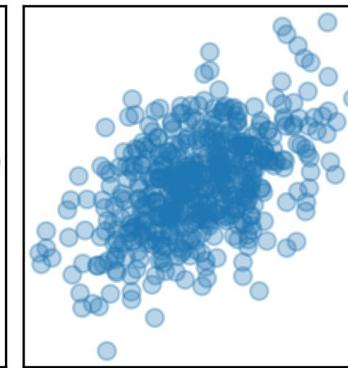
$\rho=-0.5$



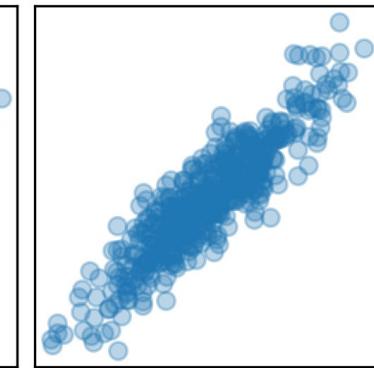
$\rho=0.0$



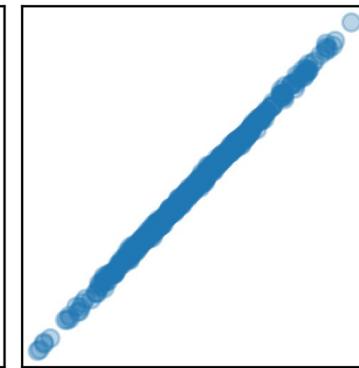
$\rho=0.5$

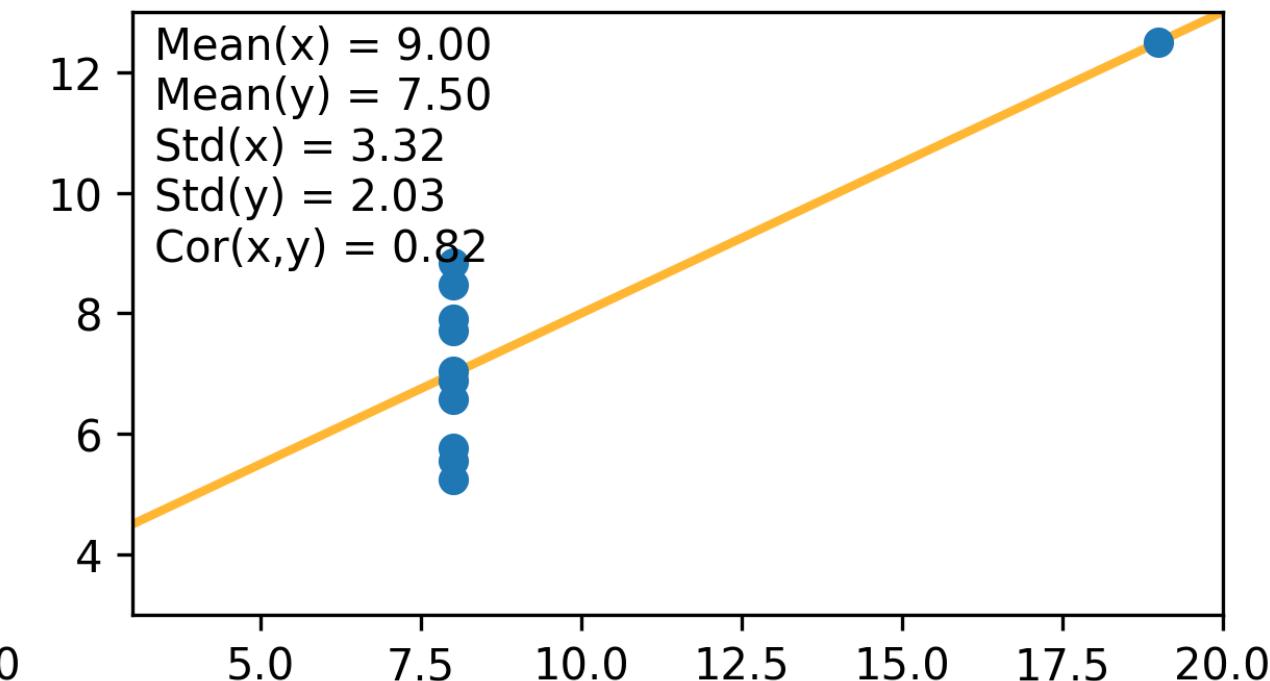
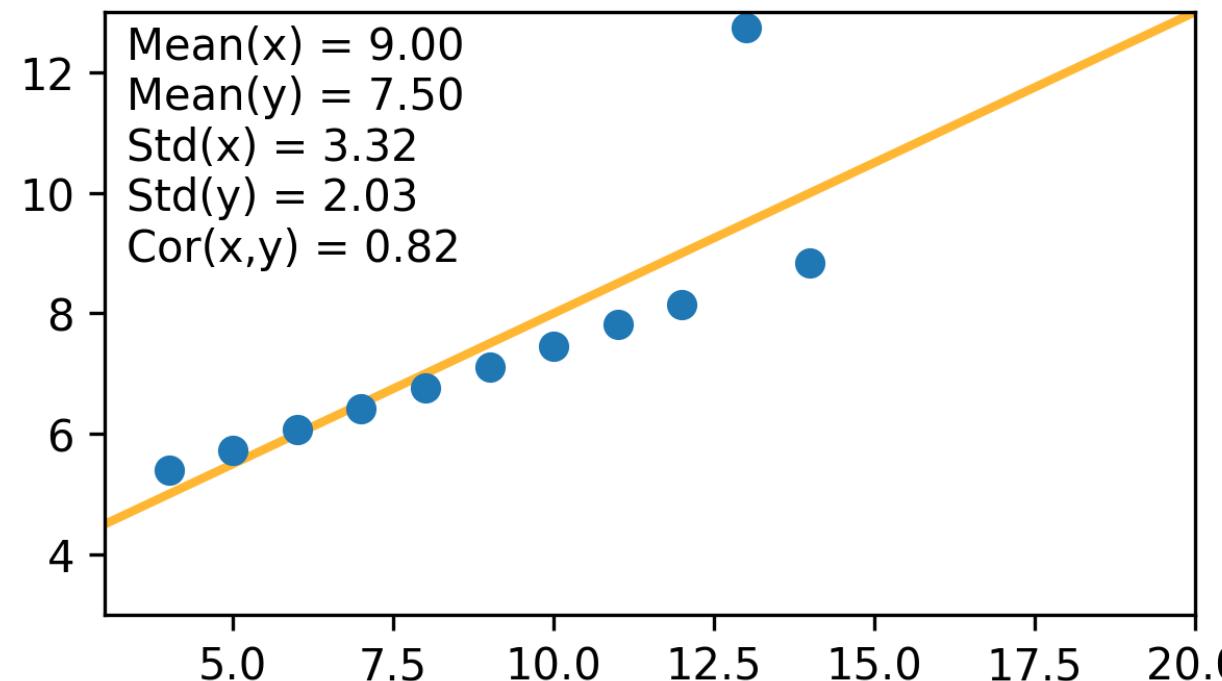
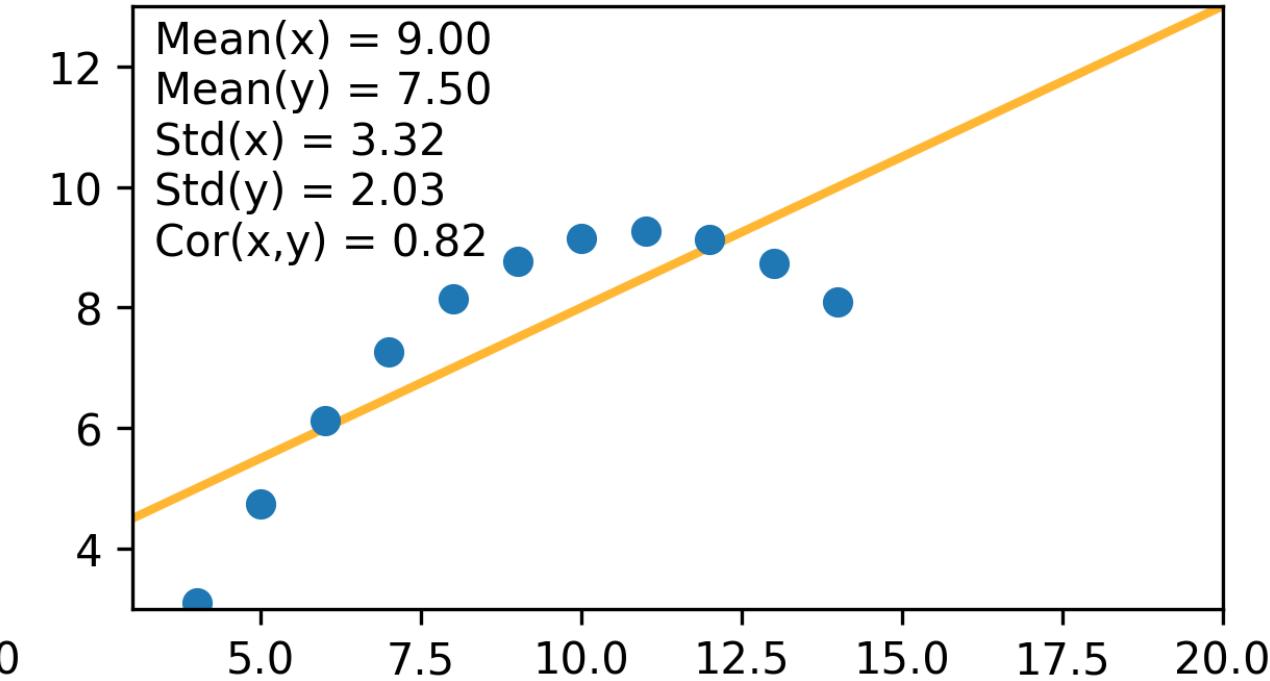
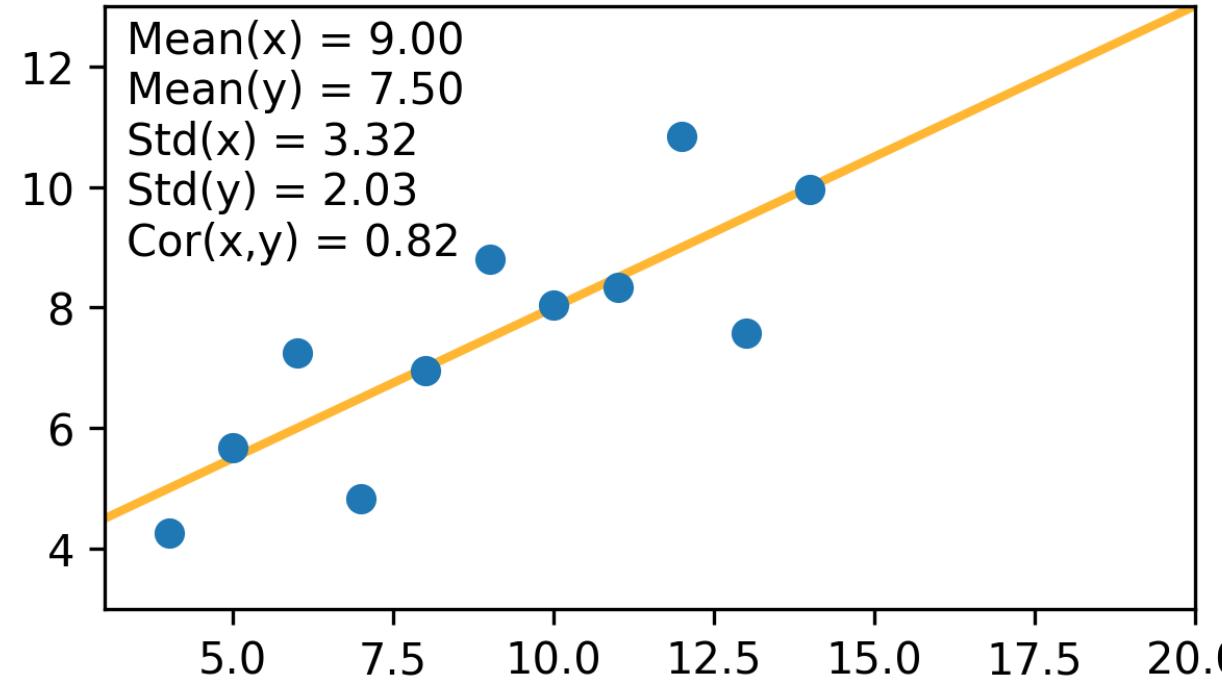


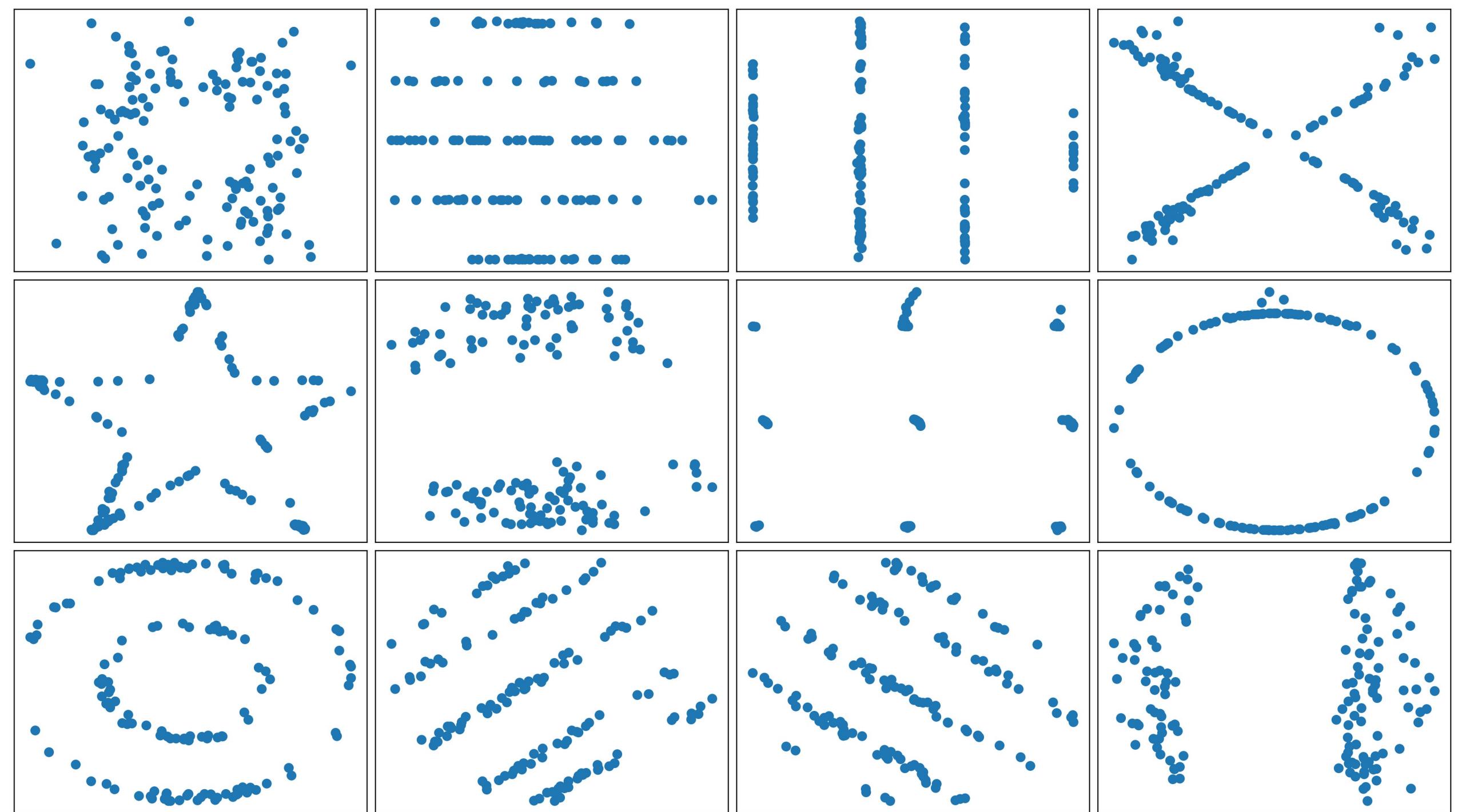
$\rho=0.9$

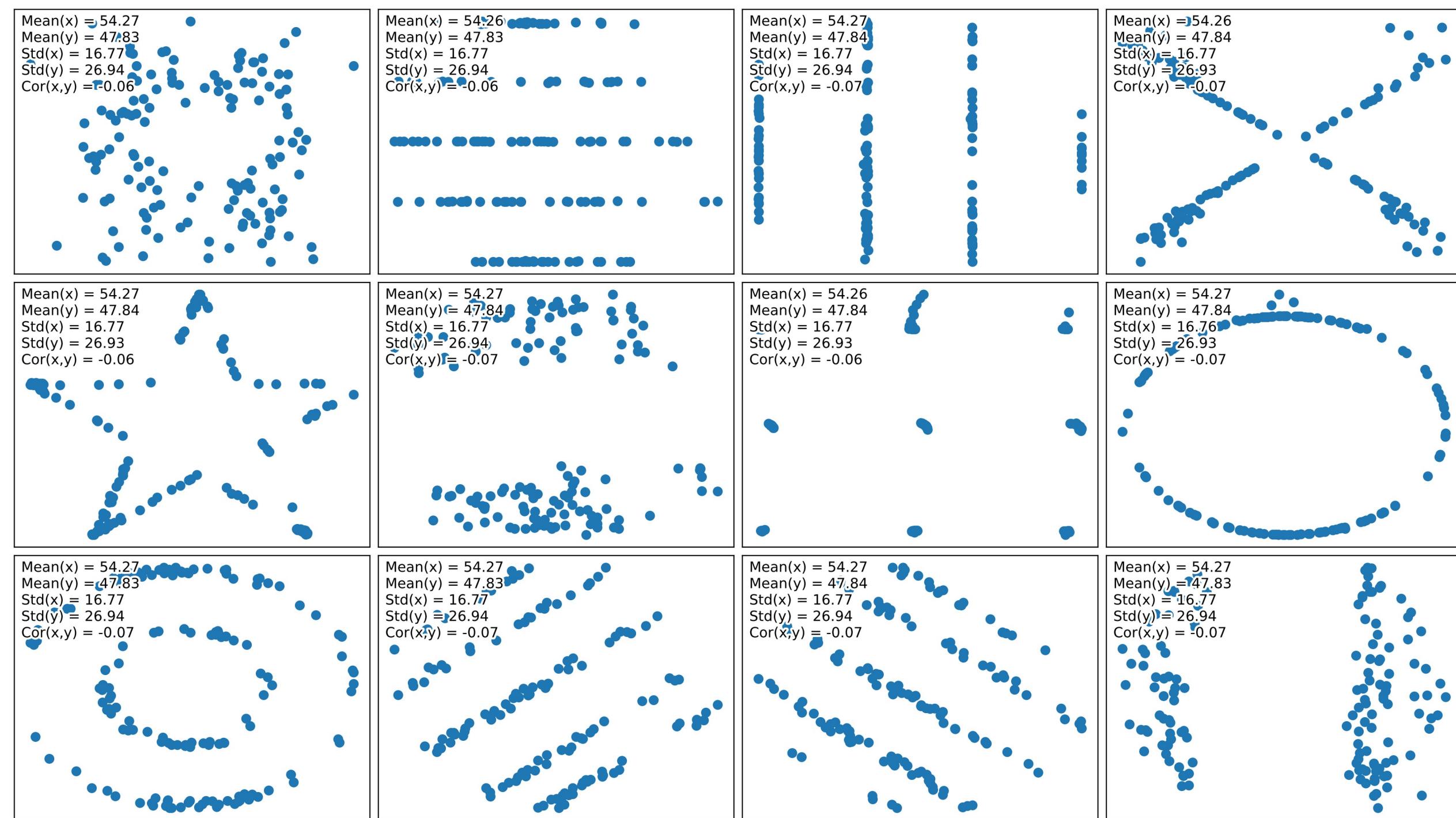


$\rho=1.0$





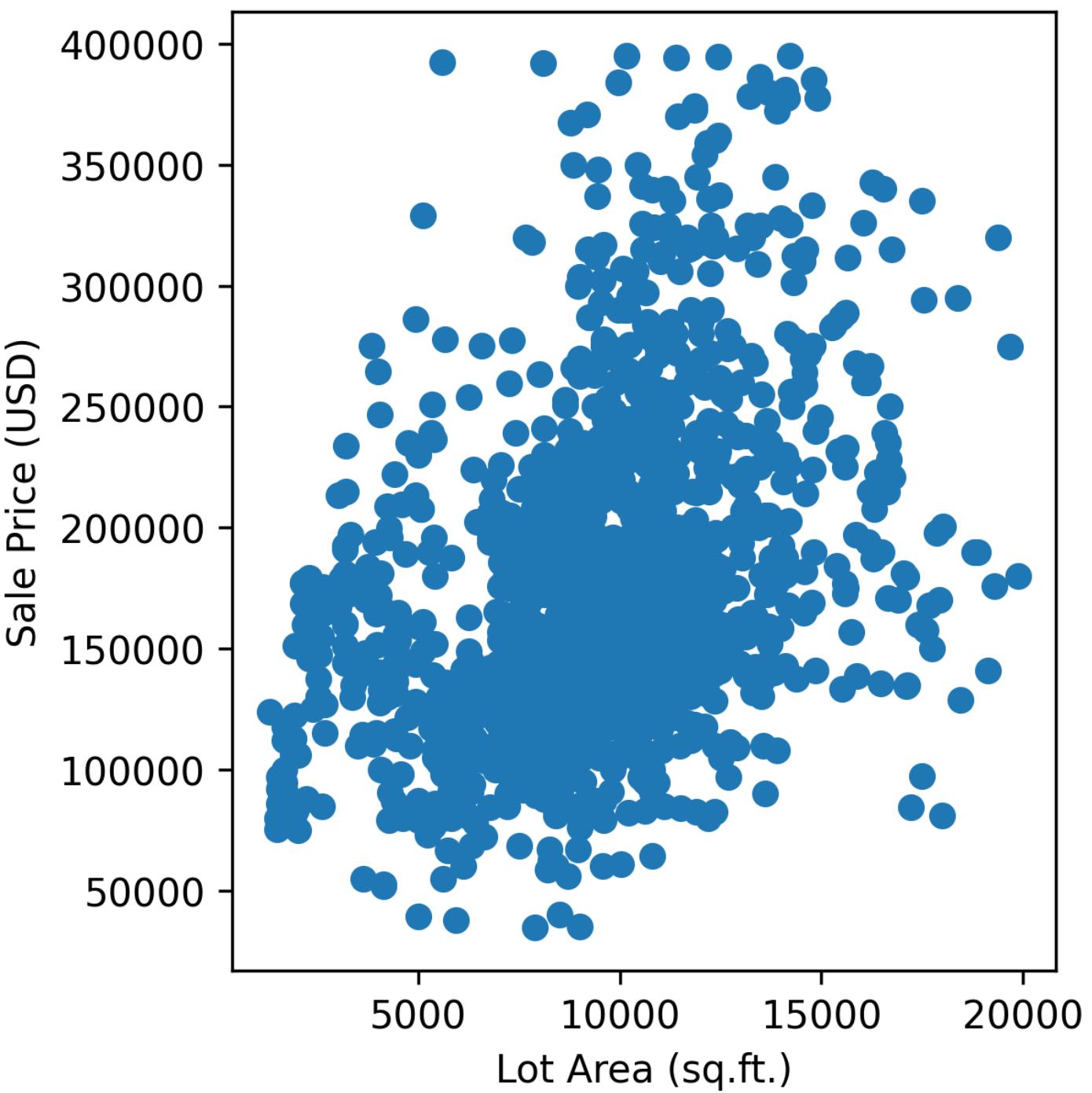




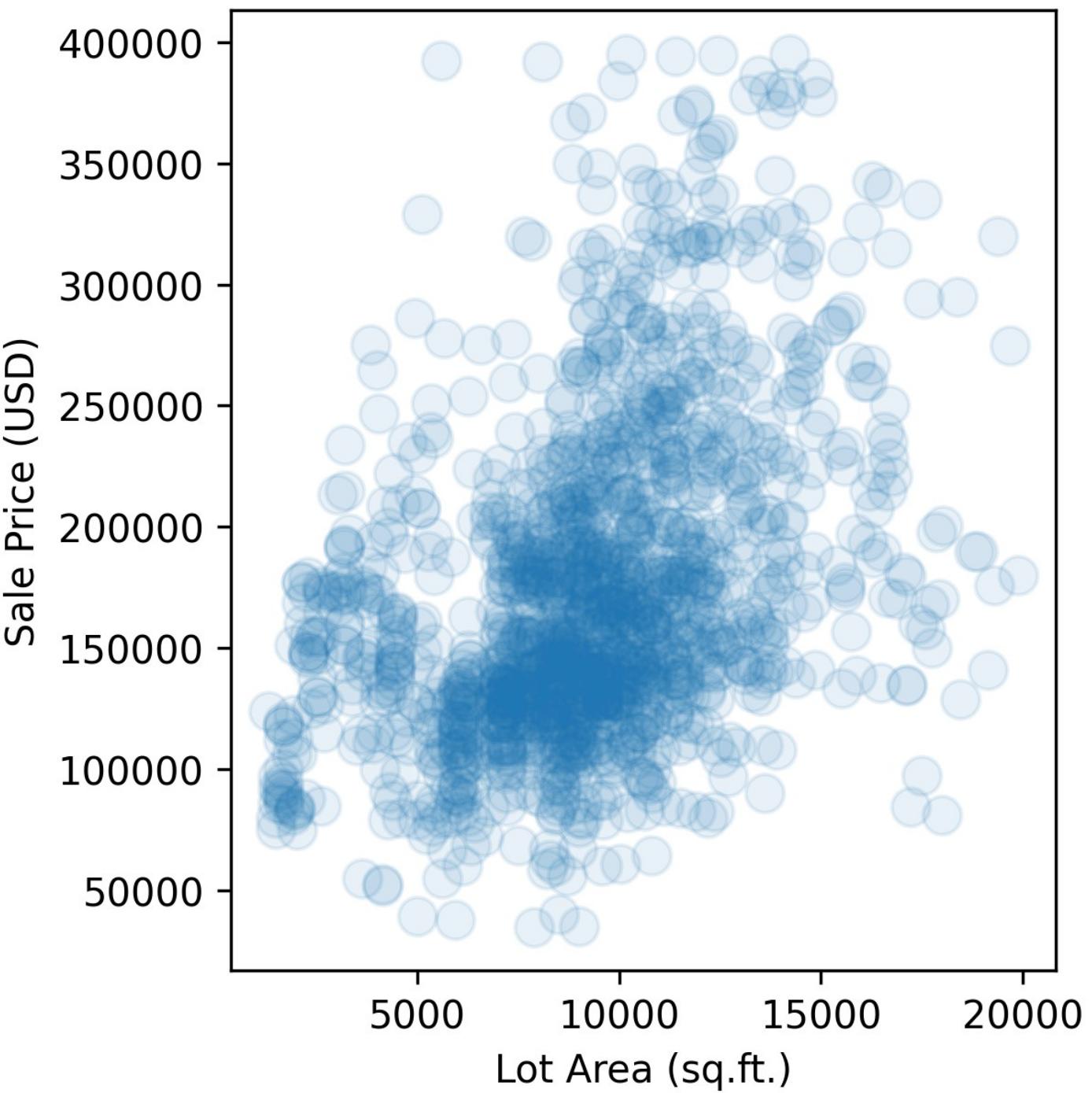
Plots

- Scatterplot (and variants: hexbin, regression line, etc.)
- Clustering (k-means, hdbscan)
- Density estimation

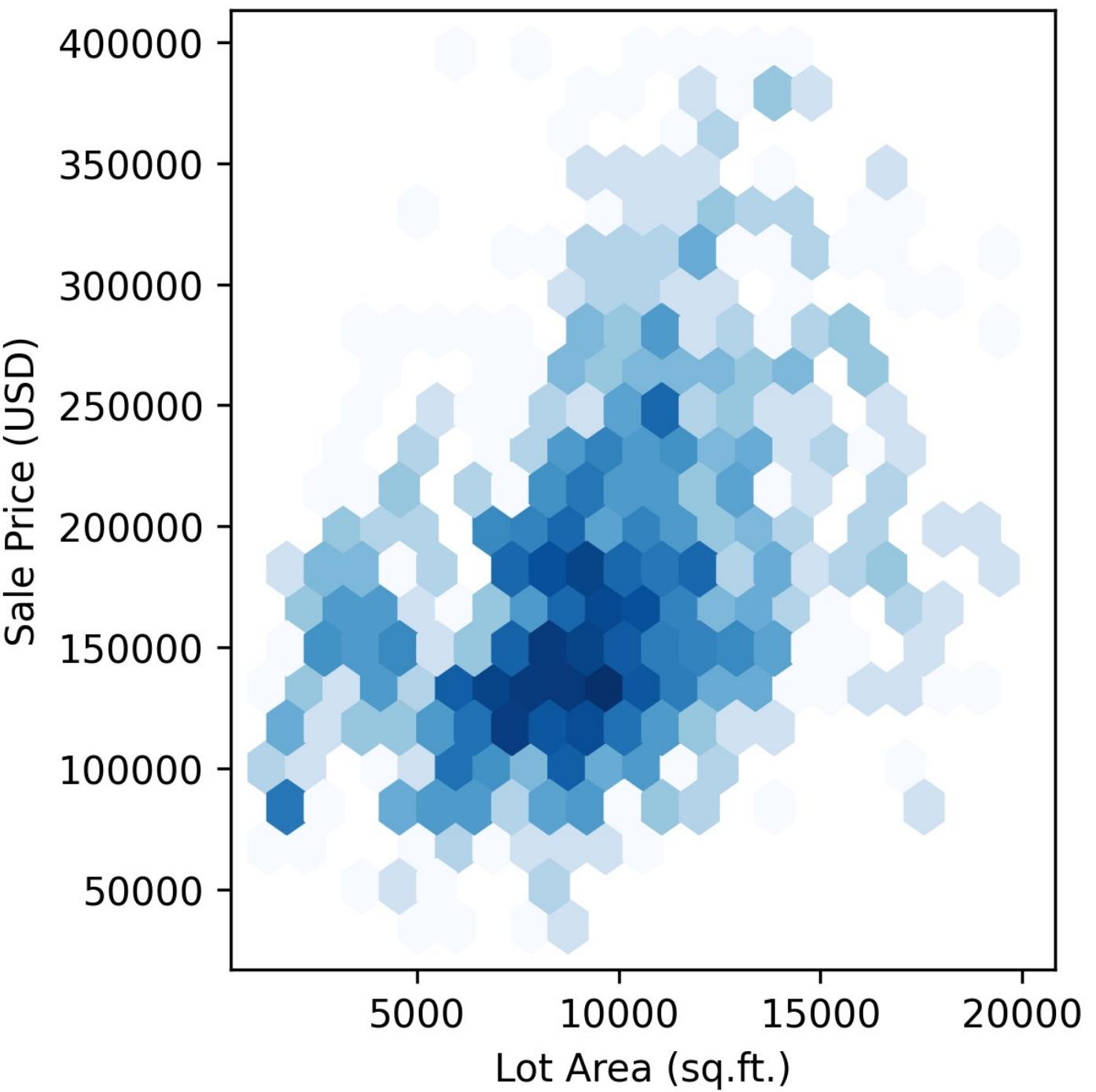
```
plt.scatter( x, y )
```



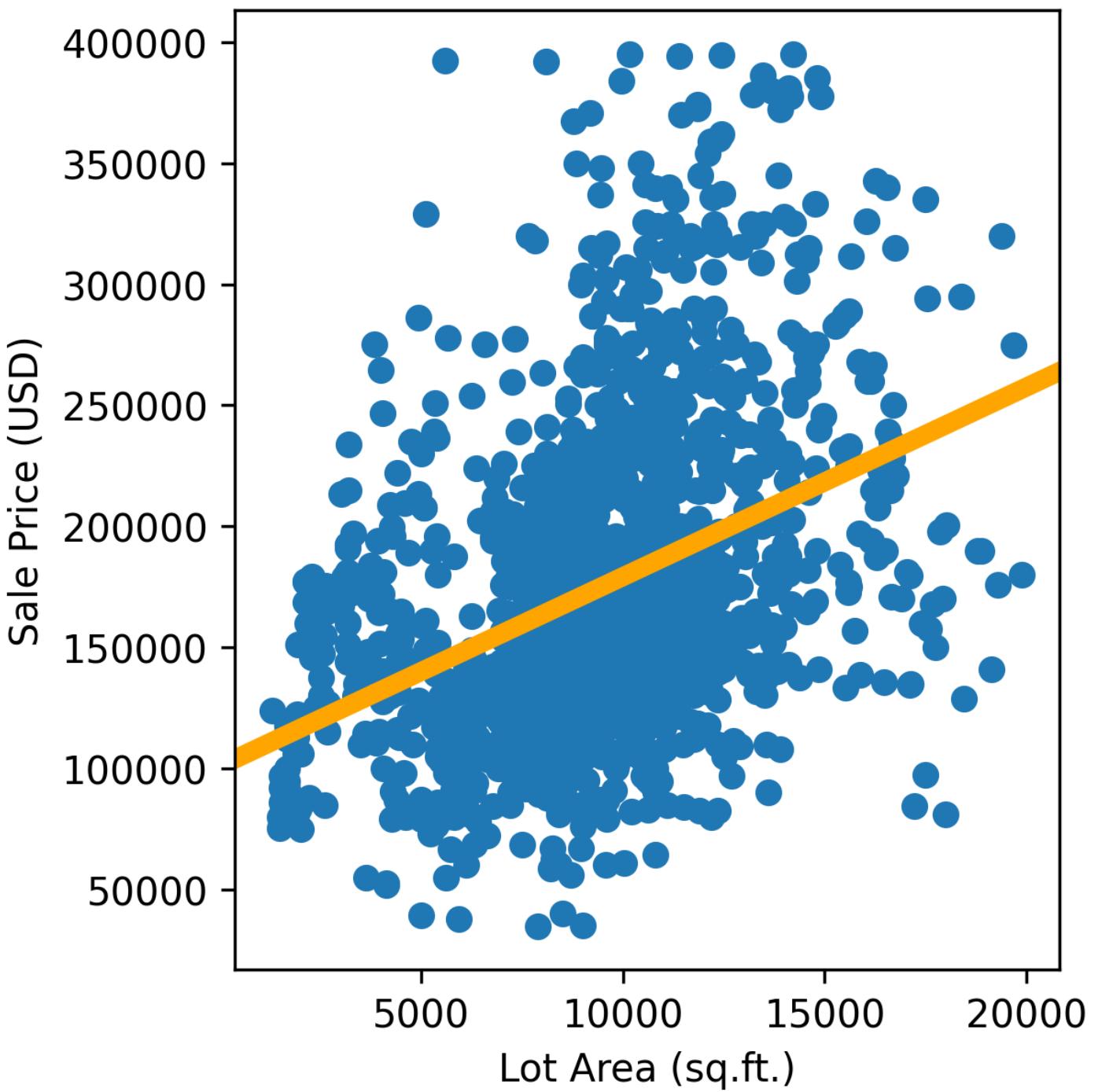
```
plt.scatter(  
    x, y,  
    alpha = .1,  
    s = 100,  
)
```



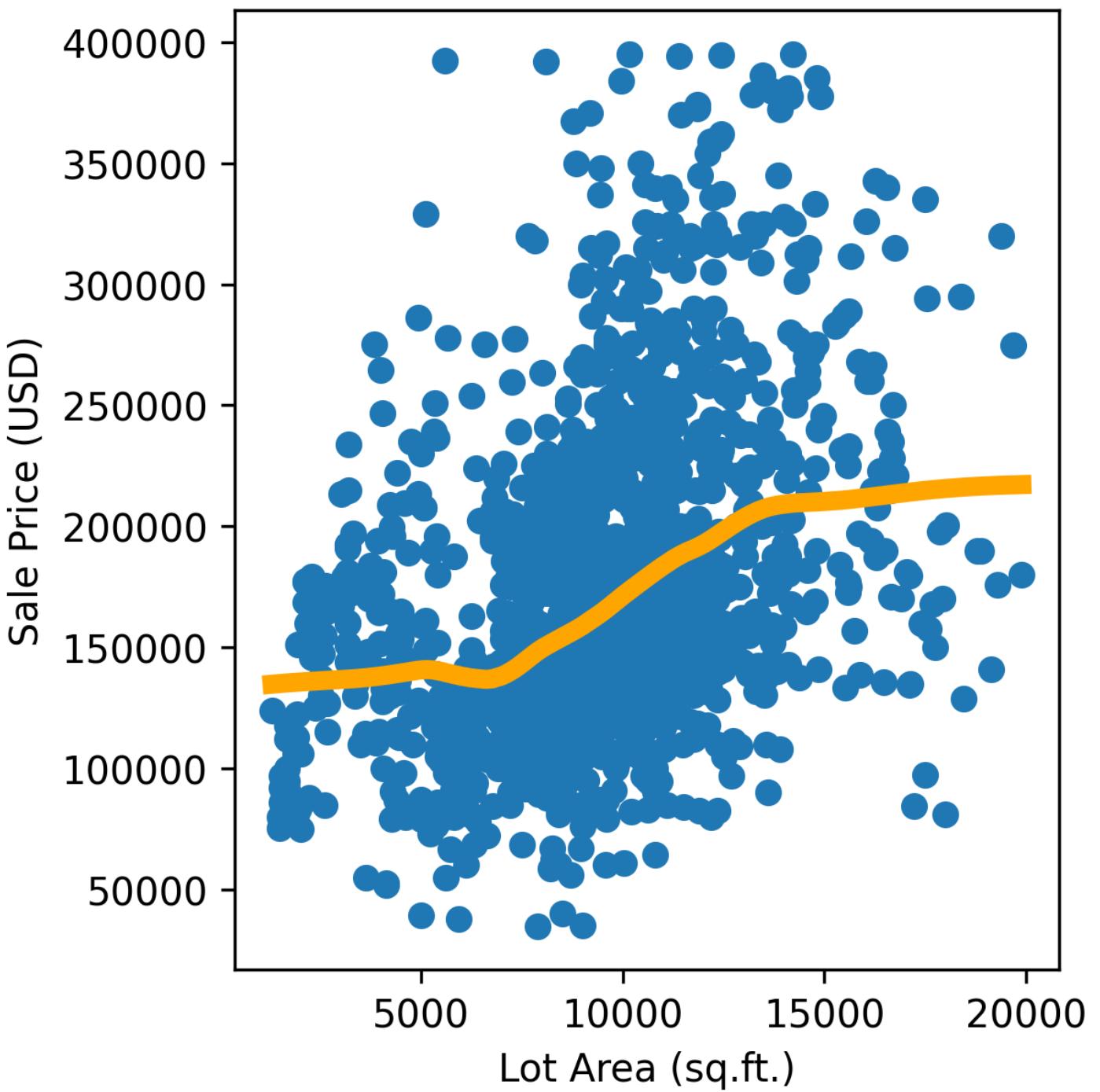
```
plt.hexbin(  
    x, y,  
    cmap = 'Blues',  
    gridsize = 20,  
    bins = 'log',  
)
```

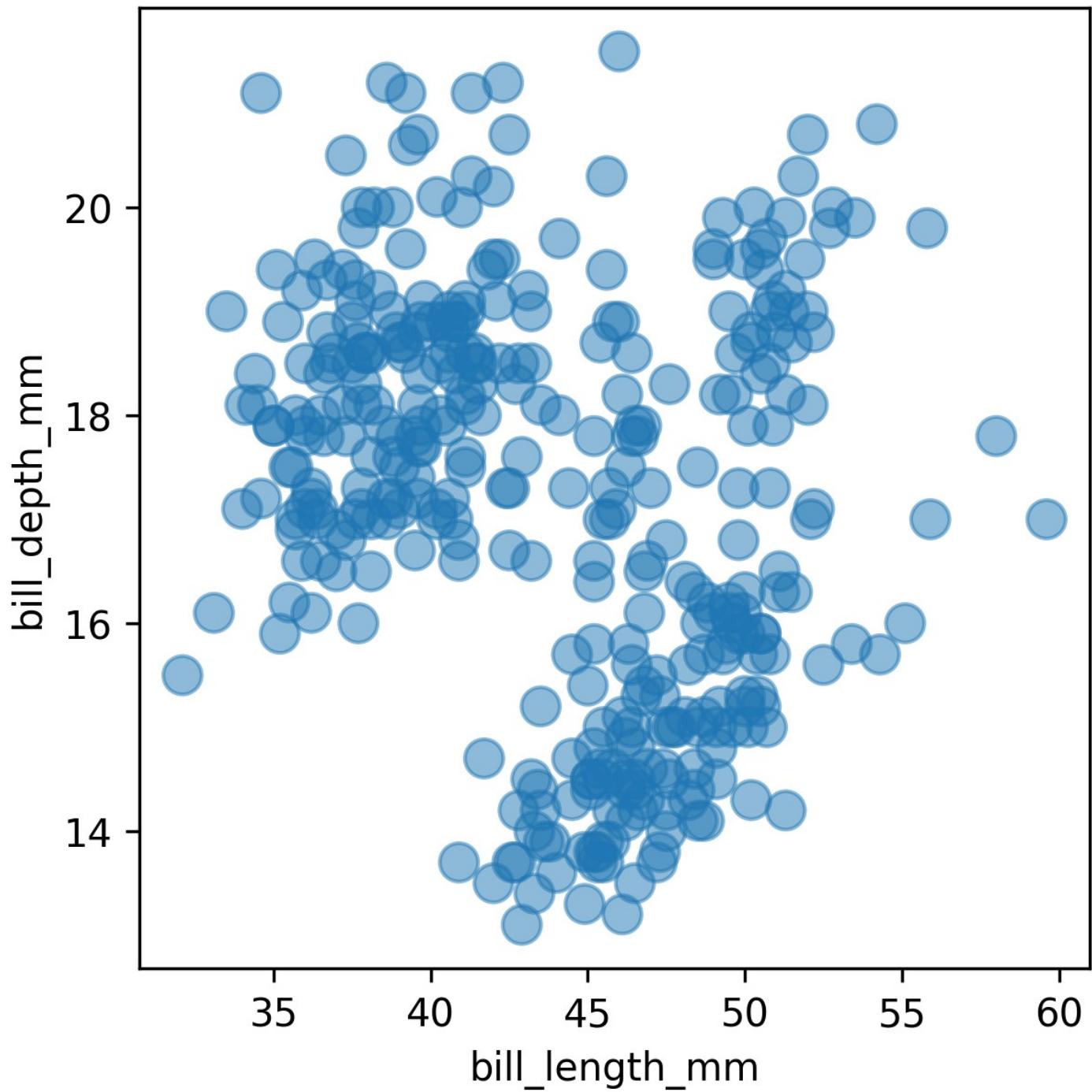


```
sns.regplot(x=x, y=y)
```

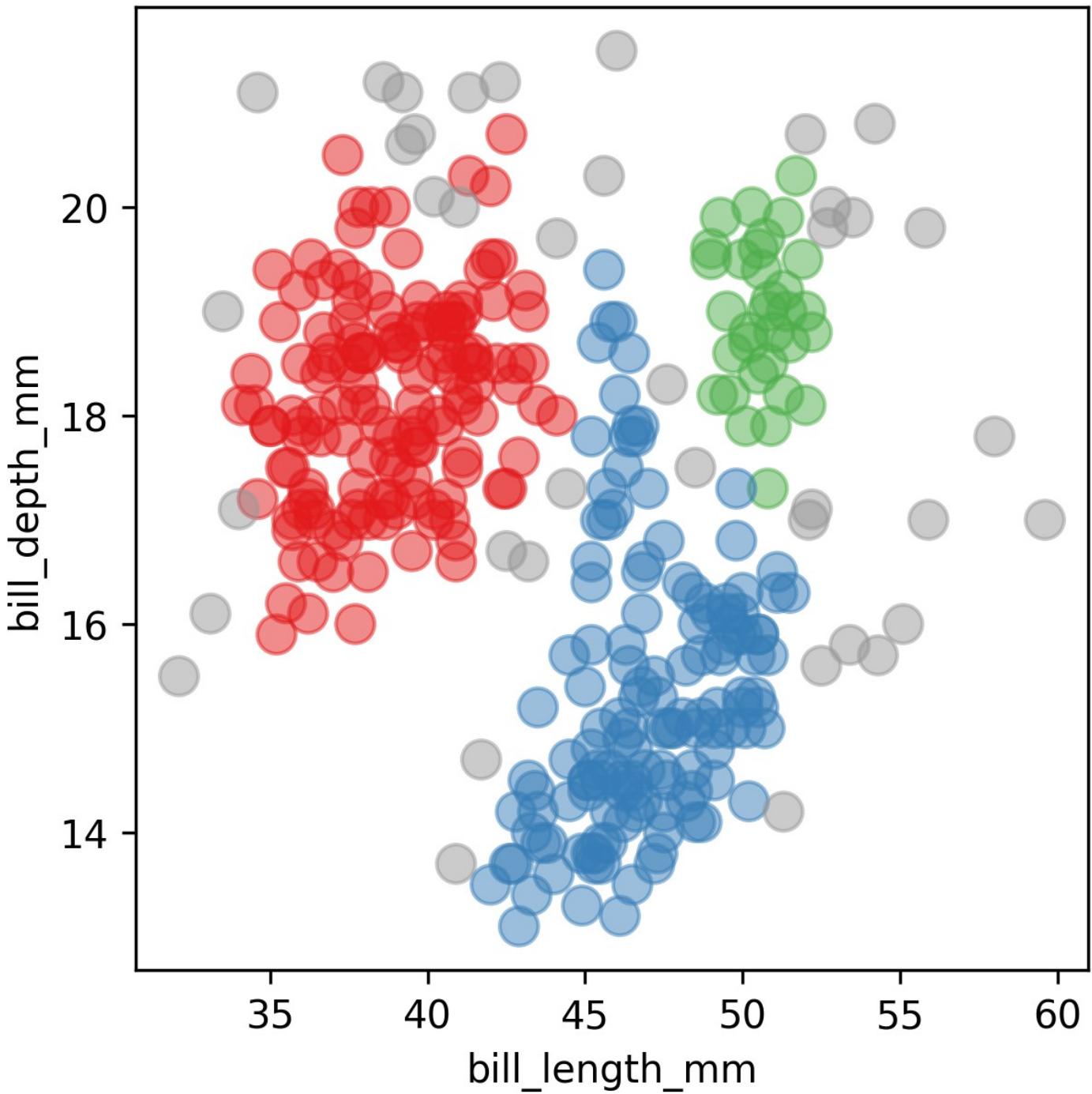


```
sns.regplot(x=x, y=y, lowess=True)
```

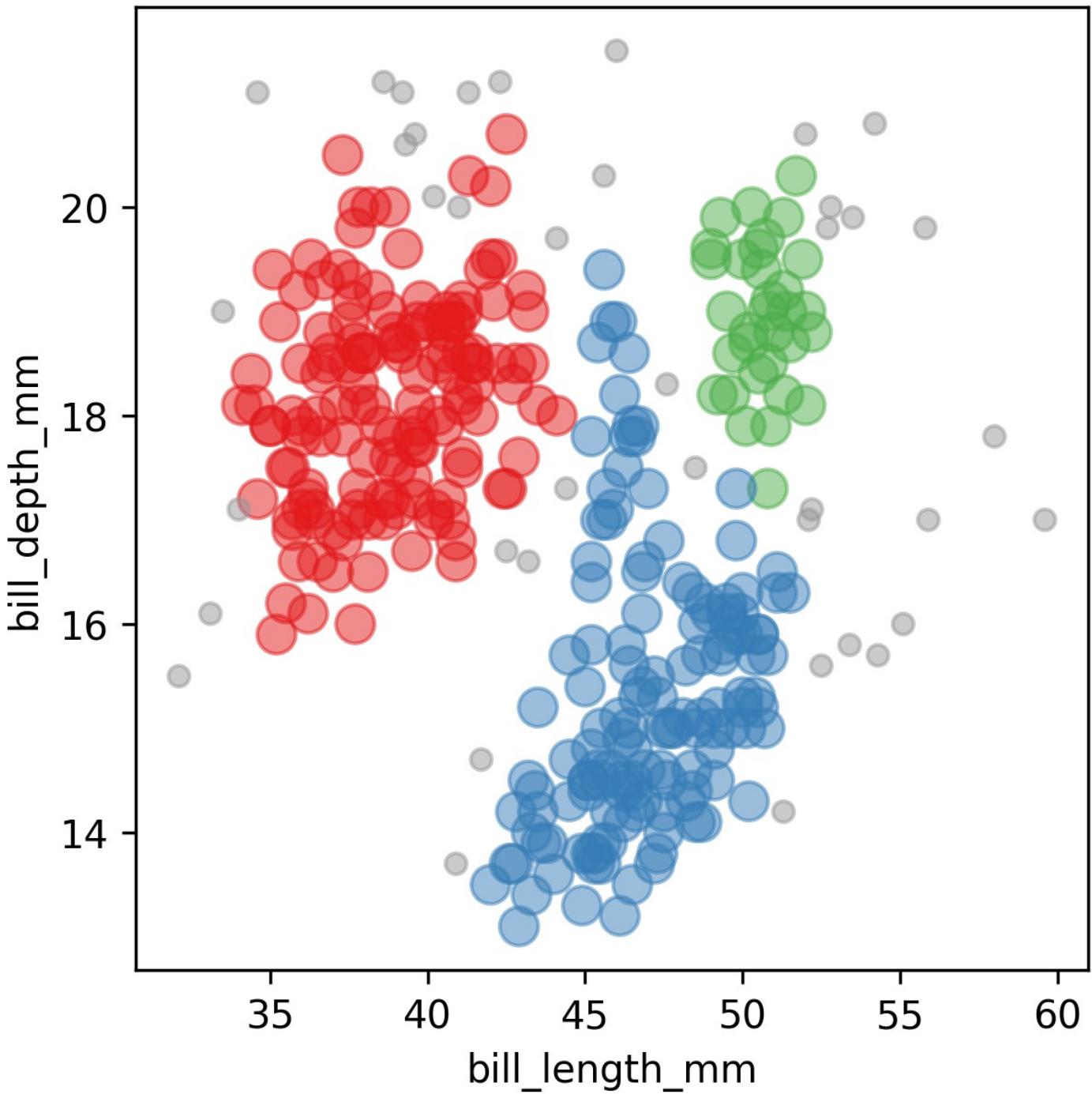




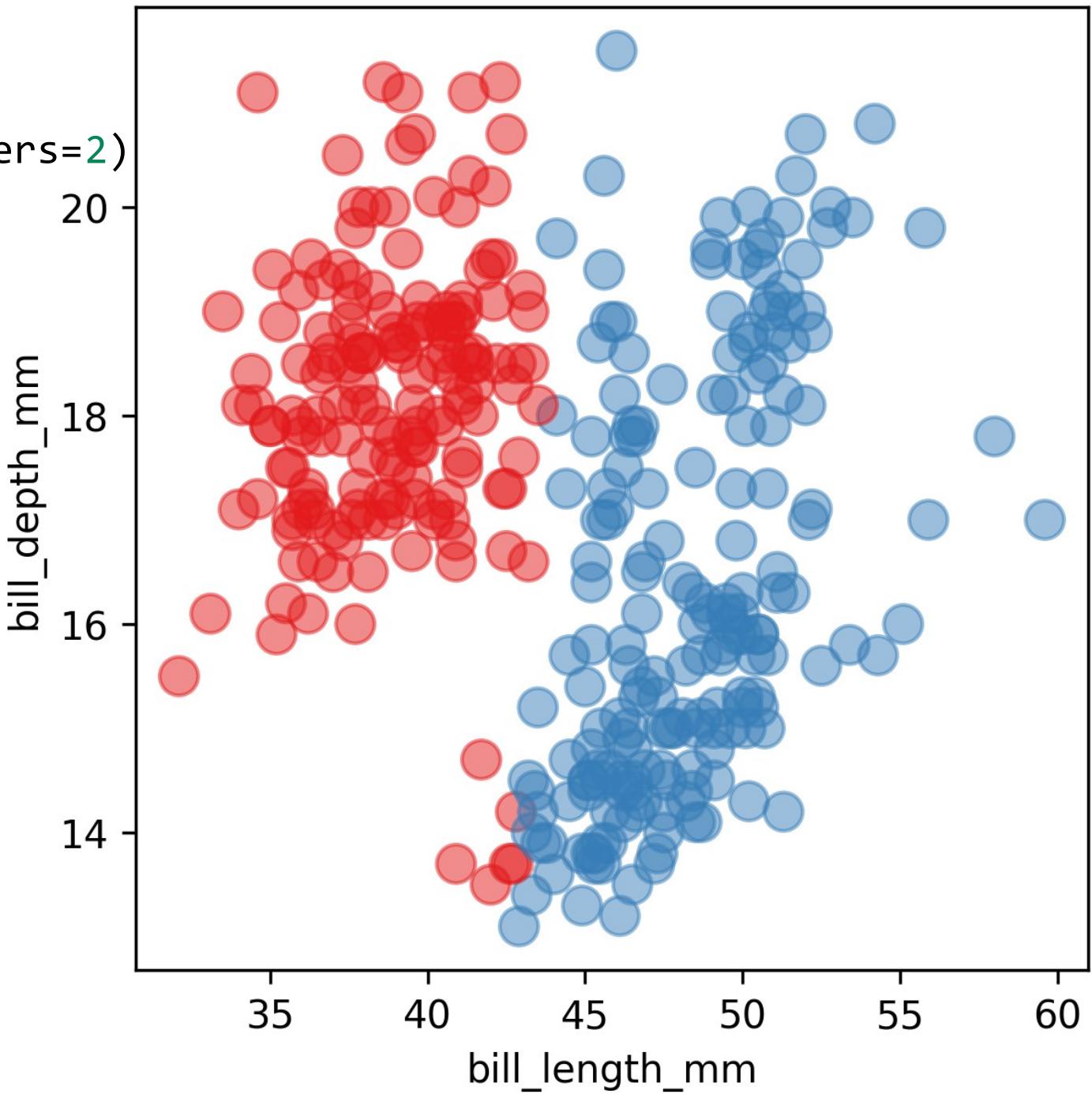
```
model = sklearn.cluster.DBSCAN()  
c = model.fit_predict(d)  
plt.scatter( x, y, c=c )
```



```
model = sklearn.cluster.DBSCAN()  
c = model.fit_predict(d)  
plt.scatter( x, y, c=c )
```



```
model = sklearn.cluster.KMeans(n_clusters=2)  
c = model.fit_predict(d)  
plt.scatter( x, y, c=c )
```



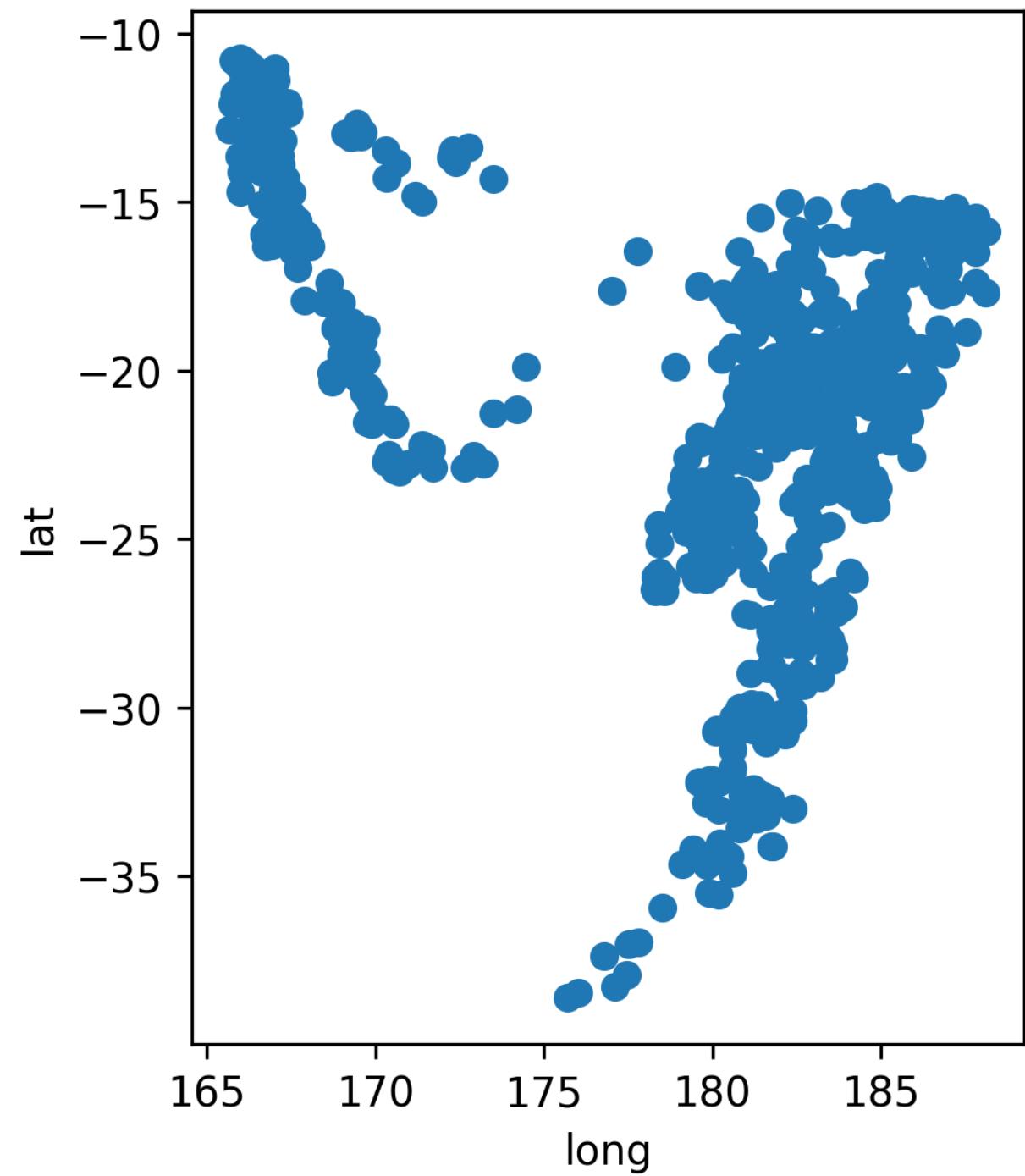
Exercise

Exercise

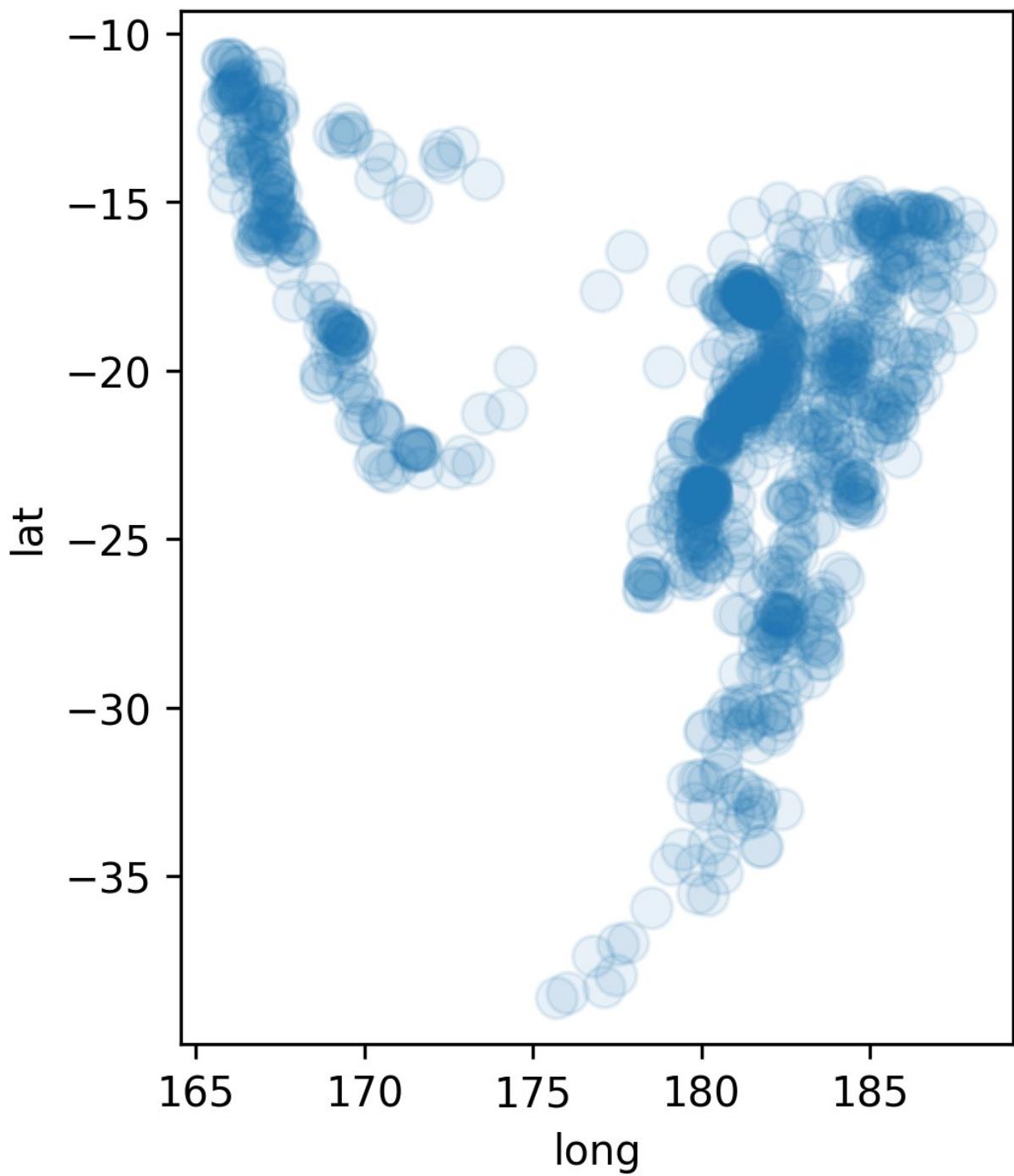
For each dataset:

- Compute the numeric univariate and bivariate summaries
- Plot (univariate) histograms and scatterplots
- Are there clusters?
- Which of the problems mentioned above are present?

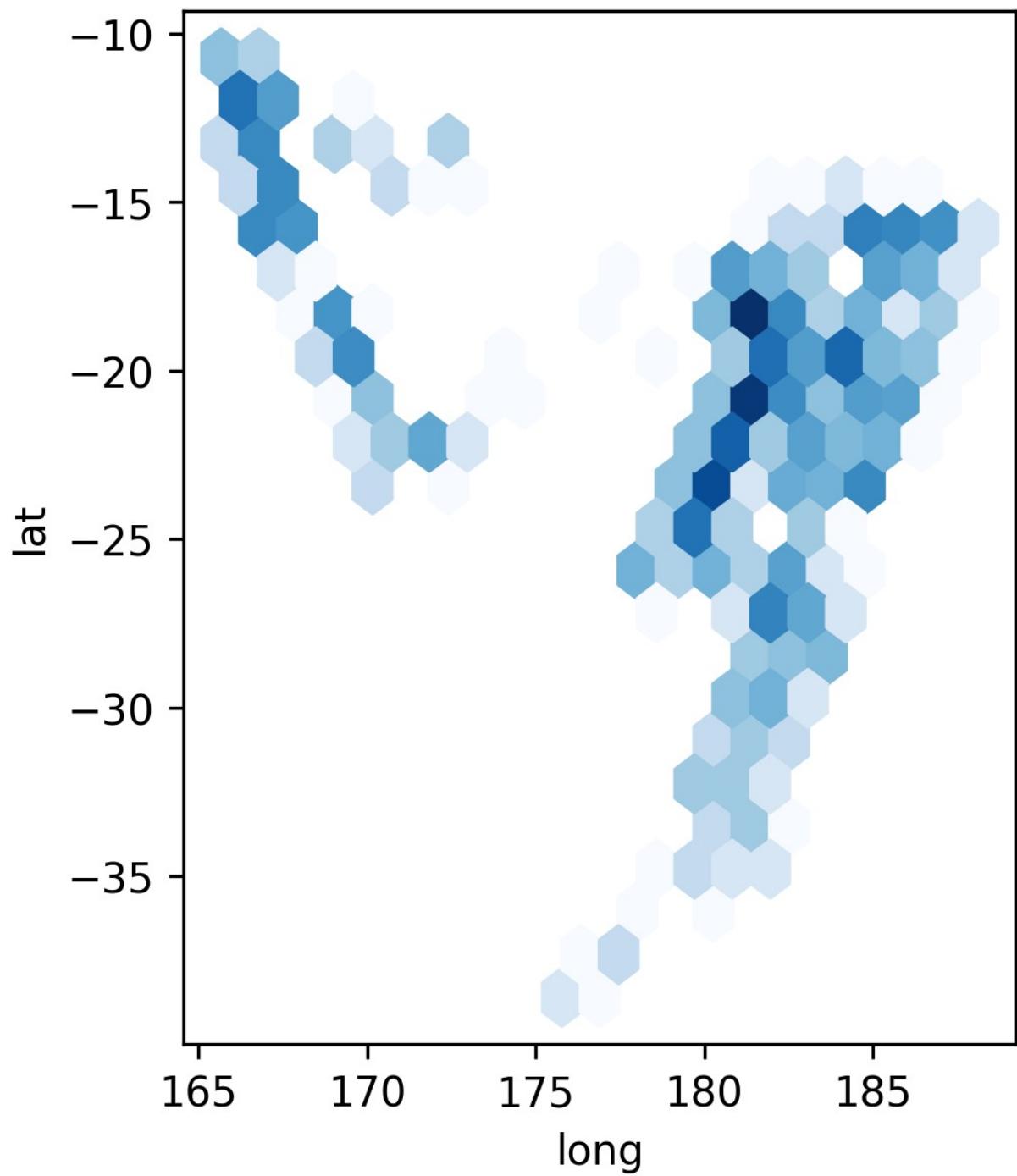
```
plt.scatter( x, y )
```



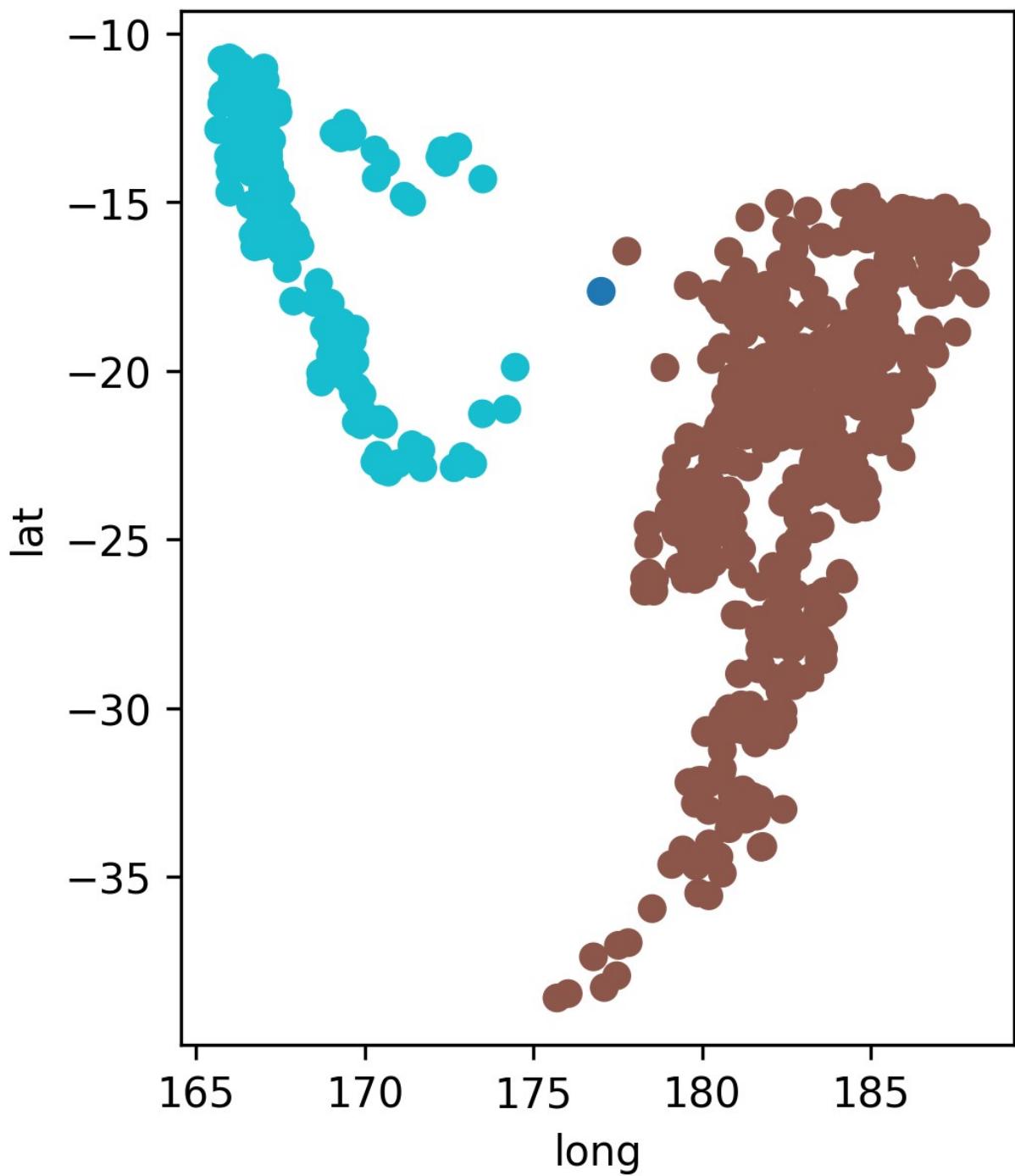
```
plt.scatter(  
    x, y,  
    alpha = .1,  
    s = 100,  
)
```



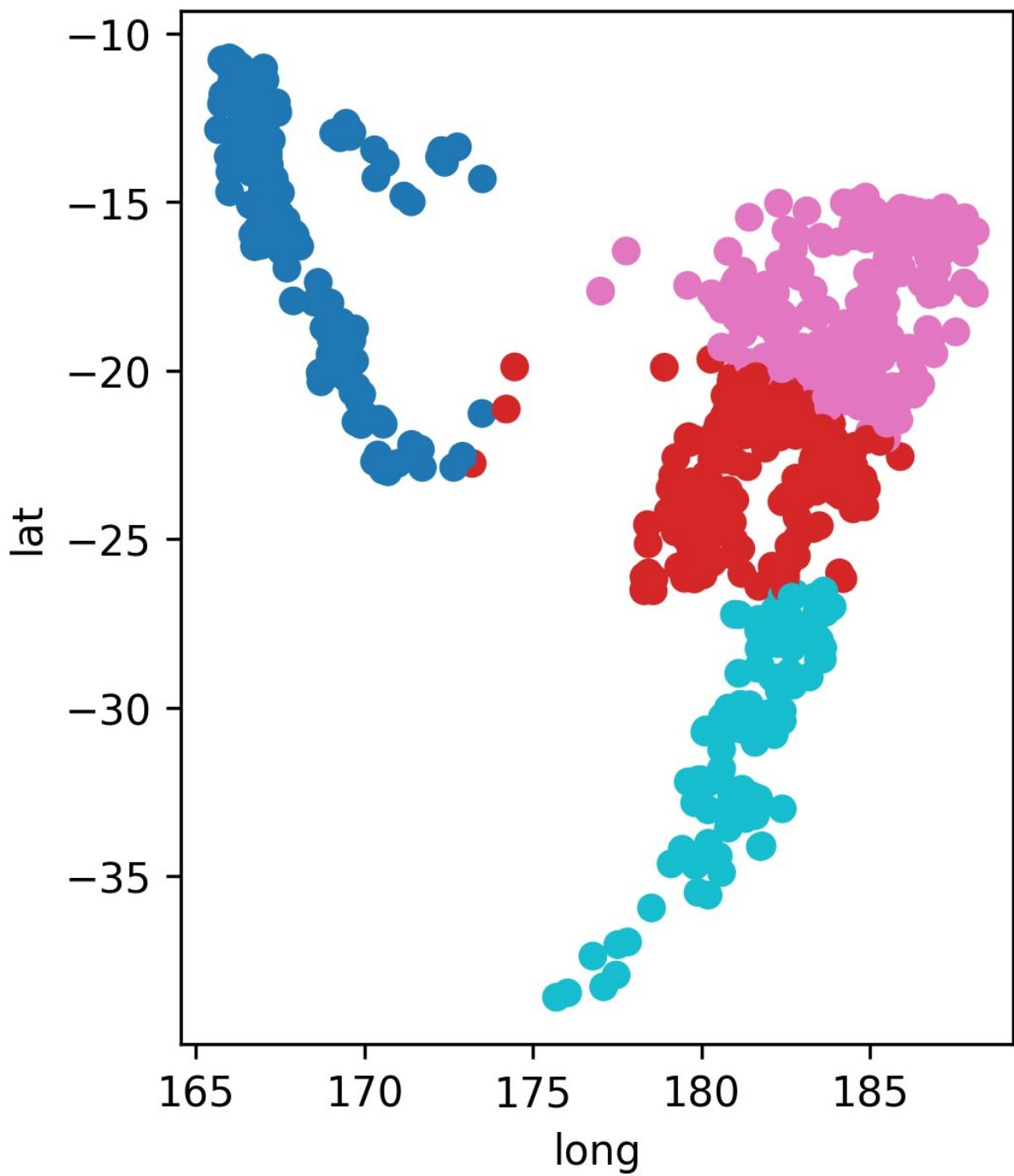
```
plt.hexbin(  
    x, y,  
    cmap = 'Blues',  
    gridsize = 20,  
    bins = 'log',  
)
```



```
model = sklearn.cluster.DBSCAN()  
c = model.fit_predict(d)  
plt.scatter( x, y, c=c )
```



```
model = sklearn.cluster.KMeans(n_clusters=4)
c = model.fit_predict(d)
plt.scatter( x, y, c=c )
```



**Qualitative
Data**

Qualitative/Categorical Data

- List of strings, e.g.,

```
['SI2', 'SI1', 'VS1', ..., 'SI1', 'SI2', 'SI2']
```

Potential Problems

- Unbalanced data
- Many values
- Long tail
- Missing values

Numbers

- Number of observations
- Number of missing values
- Number of different values
- Mode

Numbers

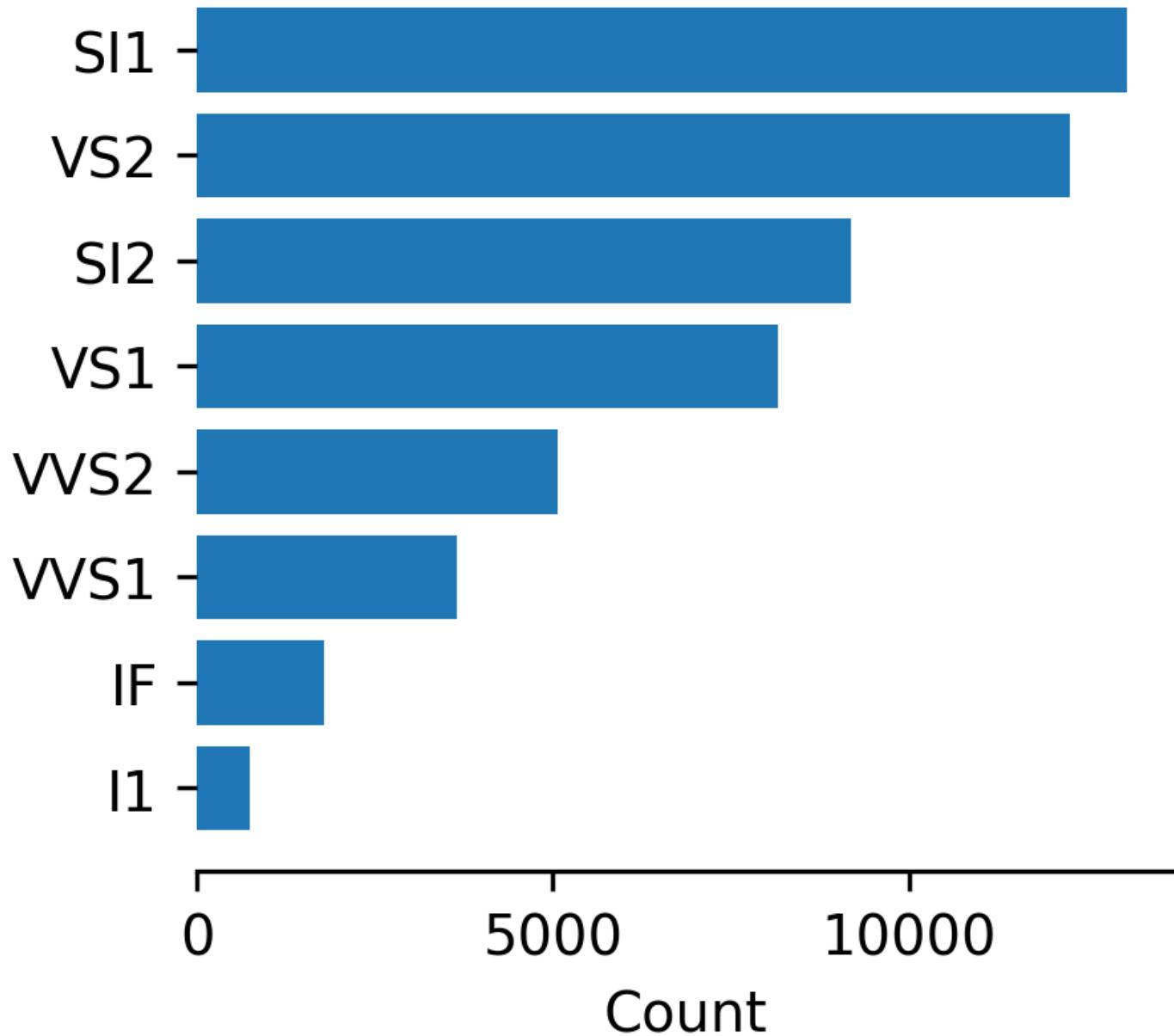
```
{  
    'observations': len(x),  
    'missing': x.isnull().sum(),  
    'Unique values': len(np.unique(x.values)),  
    'Mode': Counter(x).most_common(1)[0][0],  
    'Mode count': Counter(x).most_common(1)[0][1],  
}
```

Plots

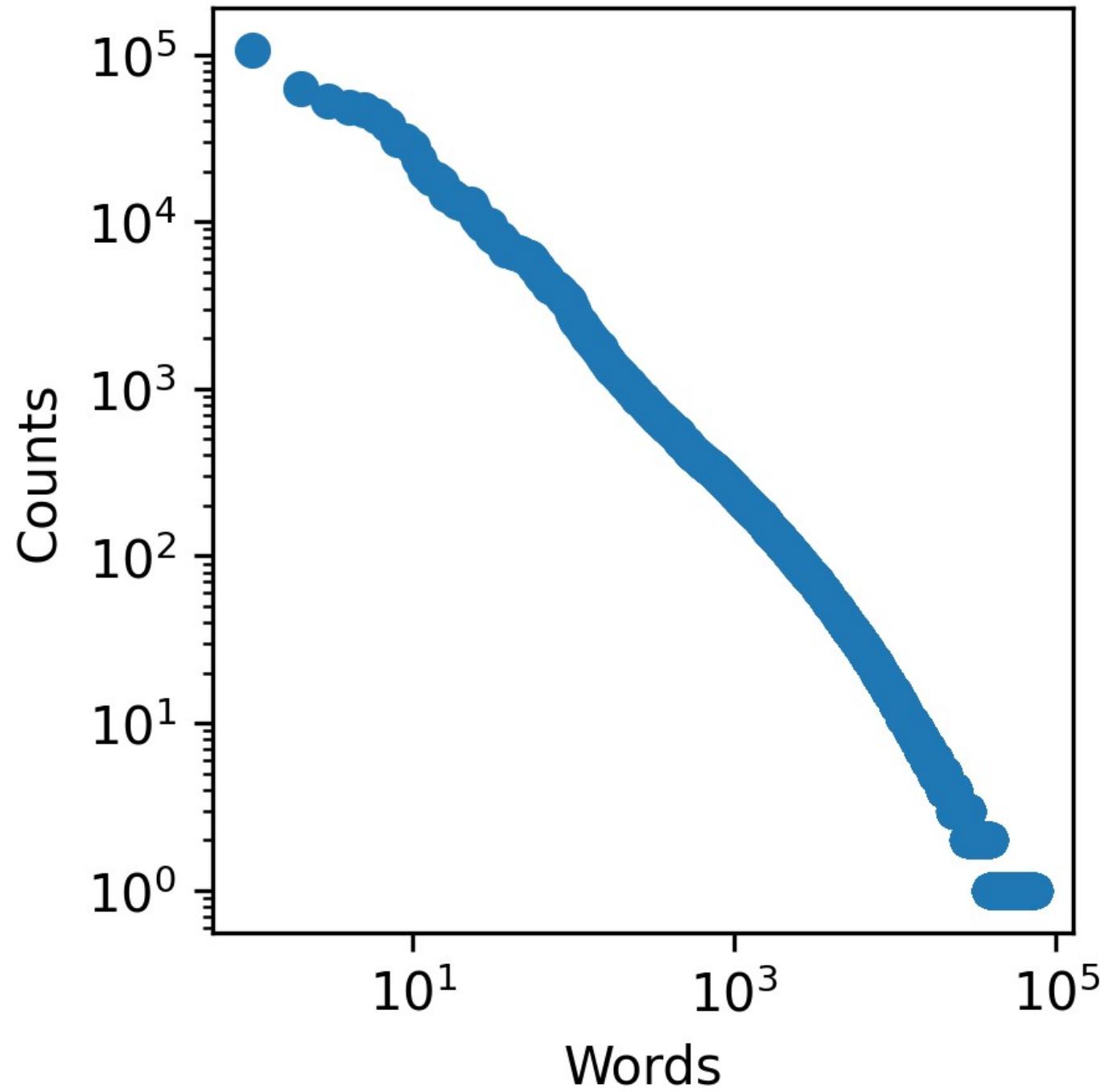
- Barplot
- Barplot on a log-log scale, if there are many values

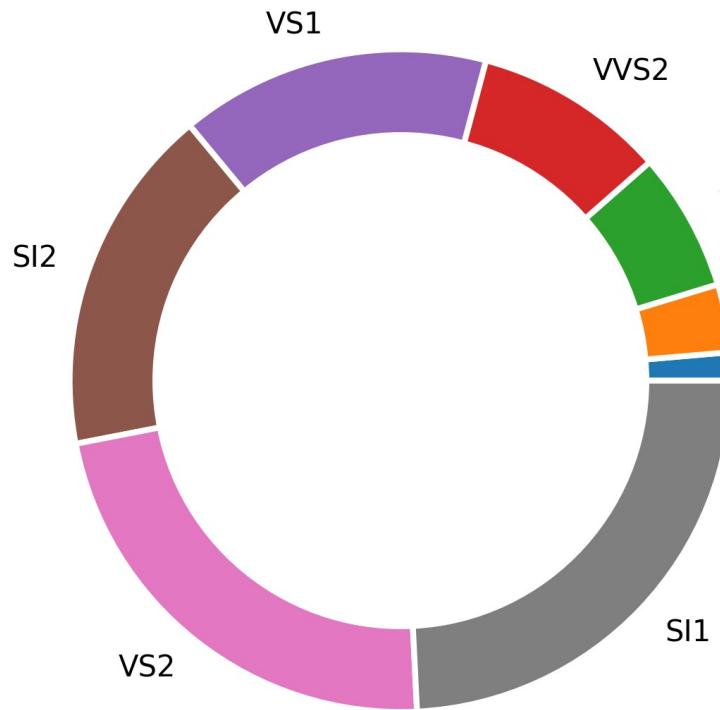
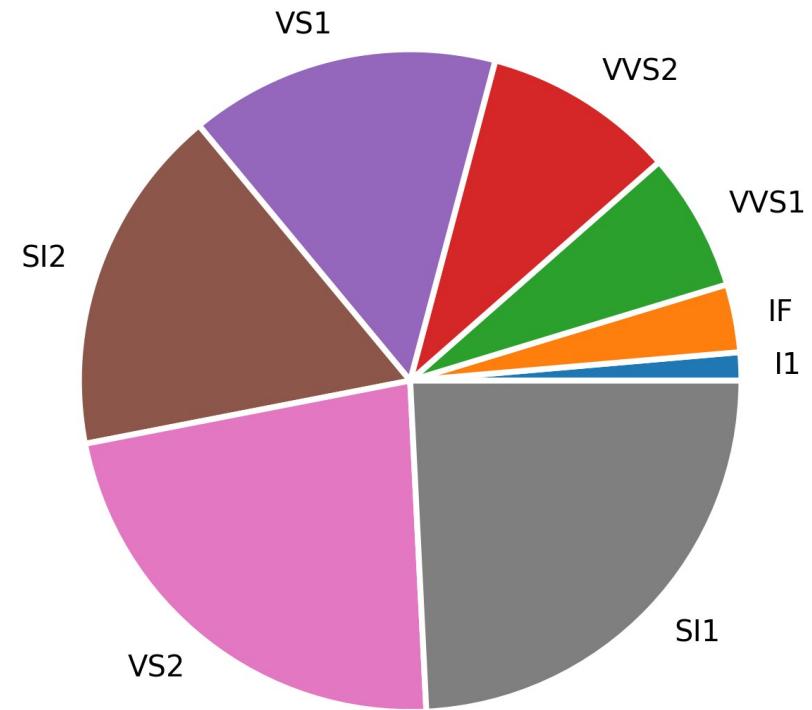
Clarity

```
c = Counter(x)
c = pd.DataFrame( {
    'clarity': c.keys(),
    'count': c.values(),
} ).sort_values('count')
plt.barh( c['clarity'], c['count'] )
```



```
c = Counter(x)
c = pd.DataFrame( {
    'word': c.keys(),
    'count': c.values(),
} ).sort_values(
    'count',
    ascending = False,
)
plt.scatter(
    1+np.arange(c.shape[0]),
    c['count'],
)
ax.xscale('log')
ax.yscale('log')
```





Exercise

Multivariate Data

Multivariate Data

- A table of numbers, with one “variable” per column.

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

Potential Problems

- Same as before
- Different scales or units

Numbers

- Univariate and bivariate summaries

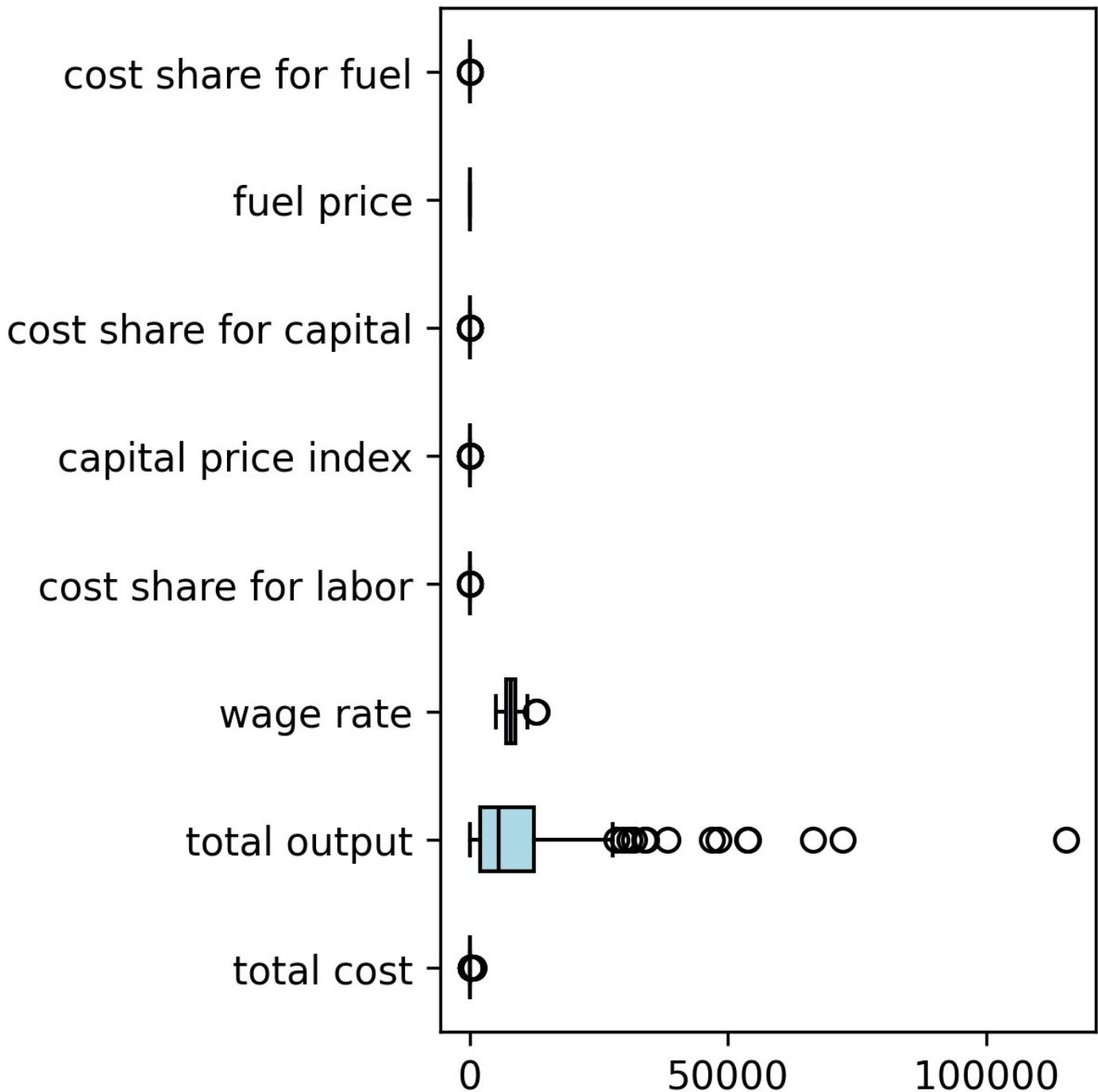
```
d.describe()
```

	cost	q	pl	s1	pk
count	158.000000	158.000000	158.000000	158.000000	158.000000
mean	53.269962	10469.410759	8001.863228	0.138972	71.421063
std	87.059326	15187.520802	1398.837478	0.054735	11.977489
min	0.130400	4.000000	5063.490000	0.045900	31.725000
25%	10.221050	1971.000000	6975.177500	0.099725	67.605000
50%	25.545400	5645.500000	7890.185000	0.123100	74.120000
75%	55.315900	12365.750000	8855.230000	0.169800	78.794250
max	737.408800	115500.000000	13044.000000	0.329100	92.650000

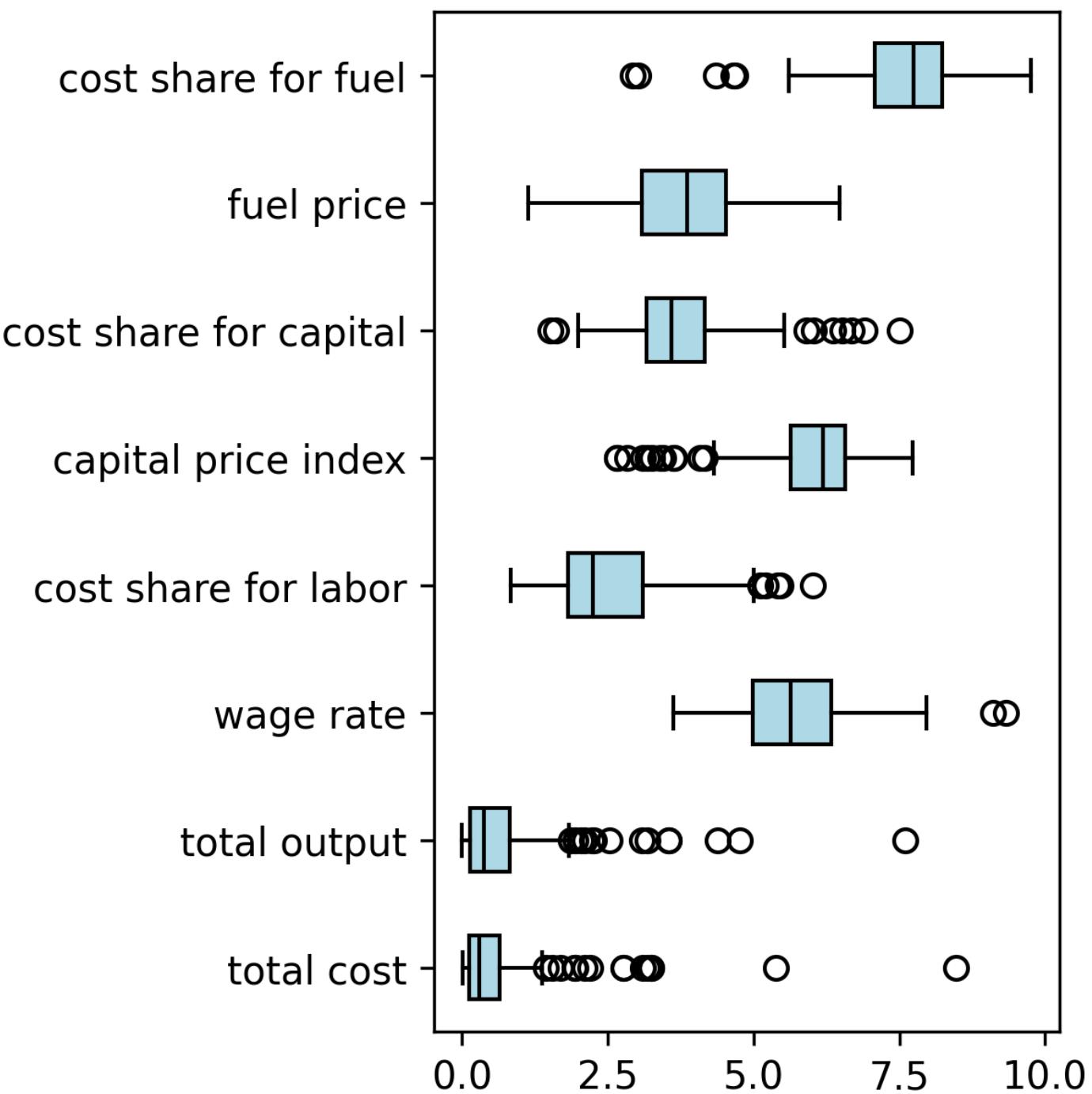
Plots

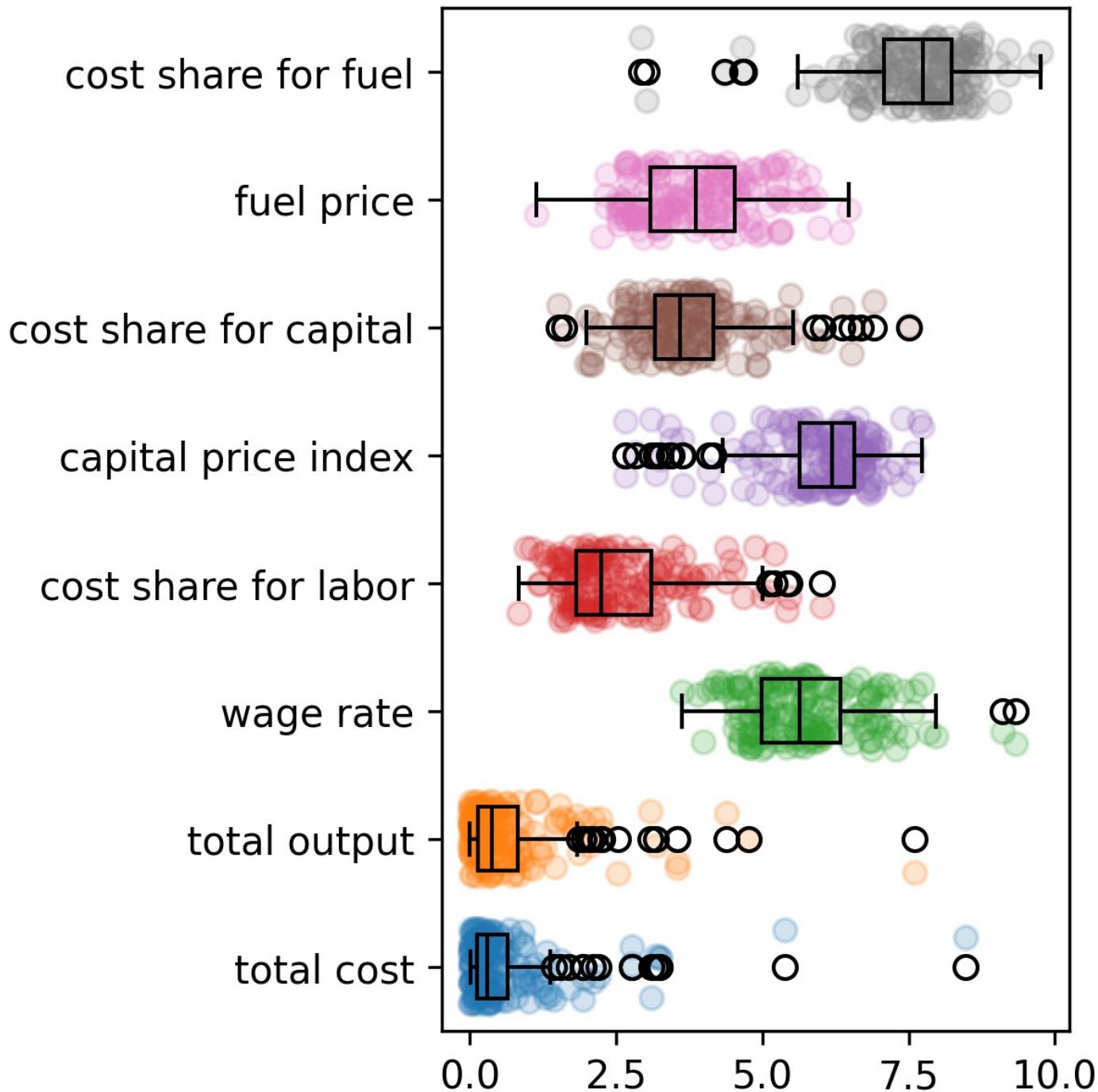
- Boxplot
- Pairs plots
- Correlation matrix
- Dimension reduction: PCA, UMAP
- Clustering: k-means, DBScan

```
plt.boxplot(d, vert=False)
```

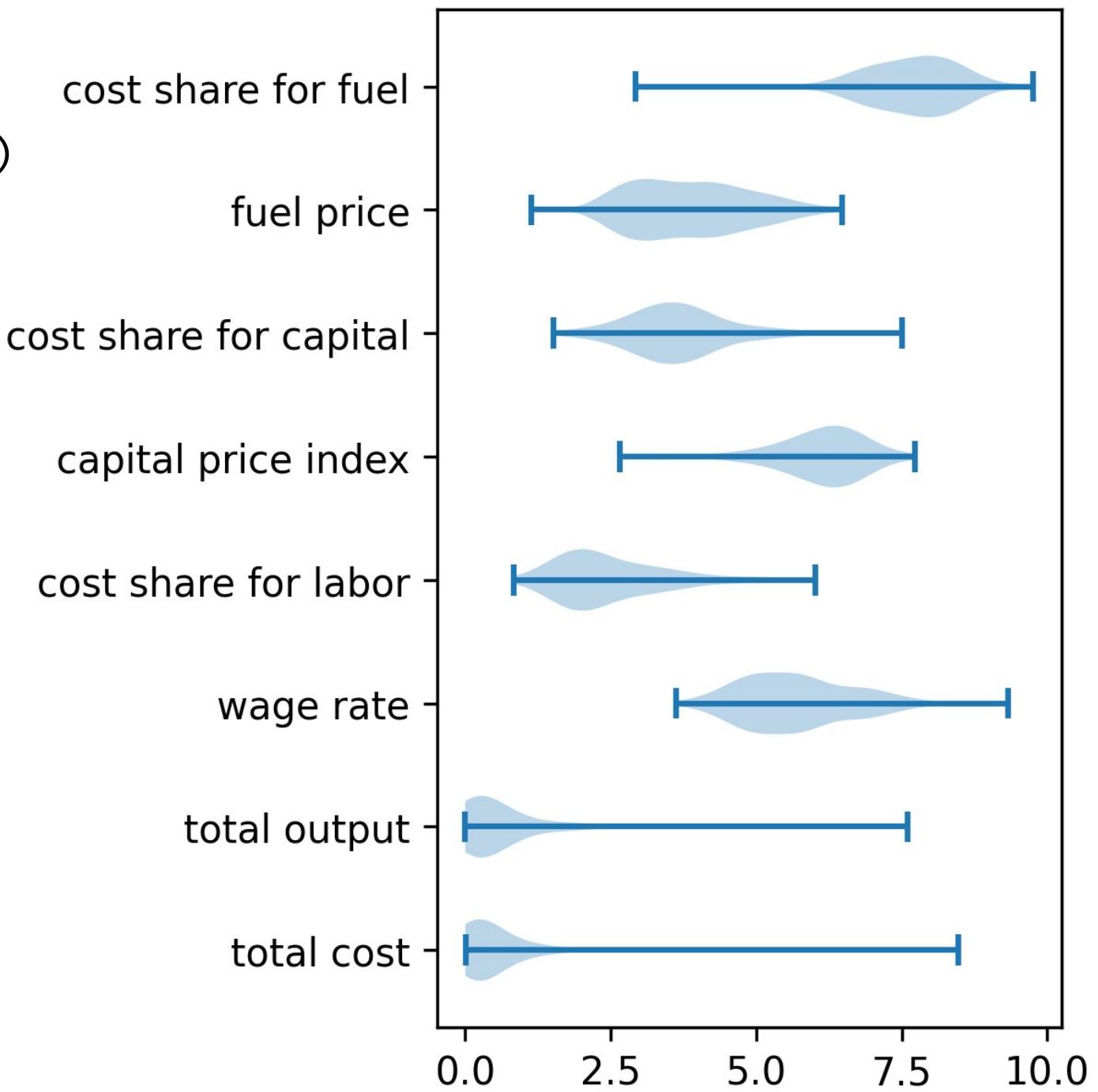


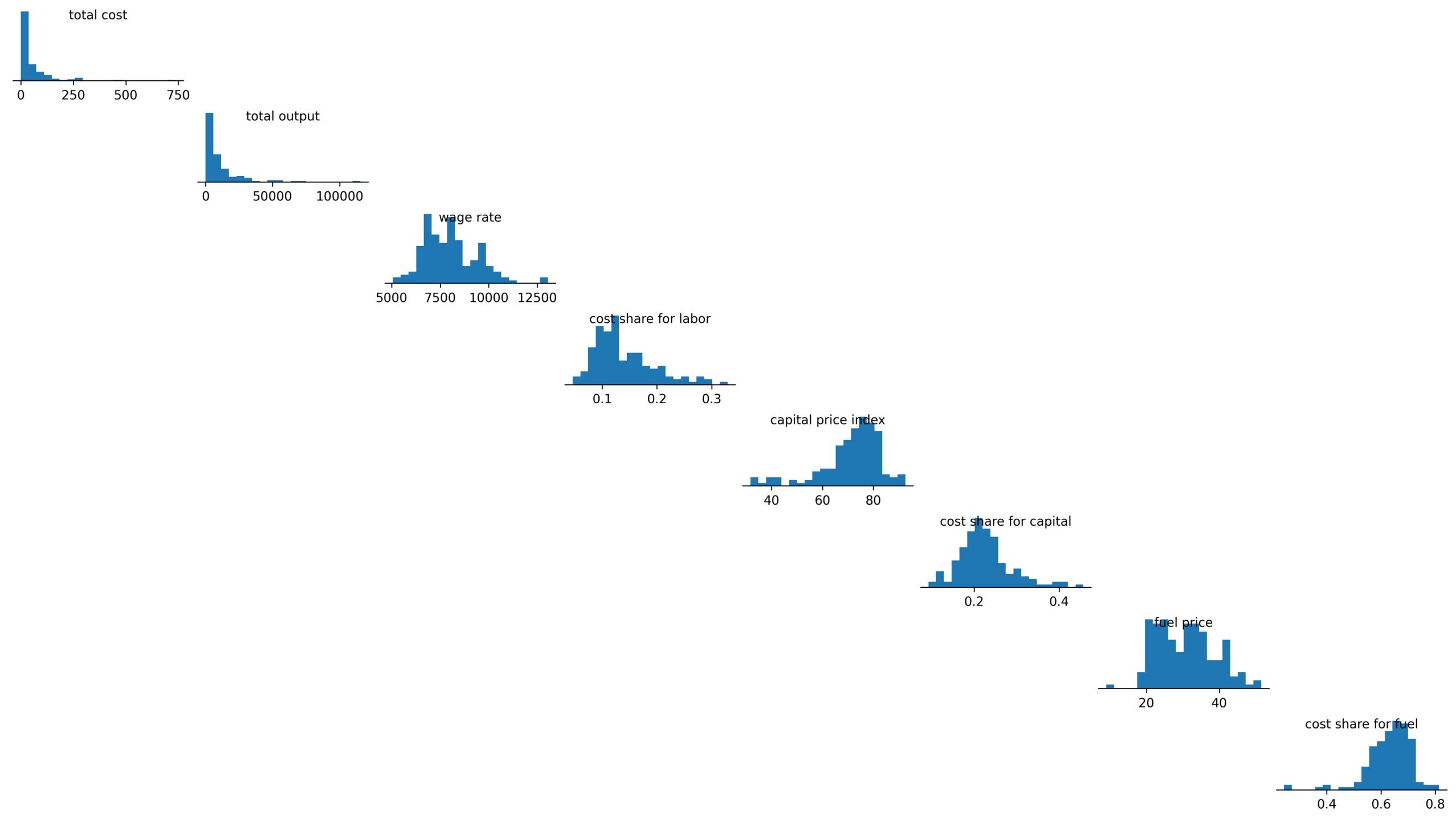
```
plt.boxplot(d / d.std(), vert=False)
```



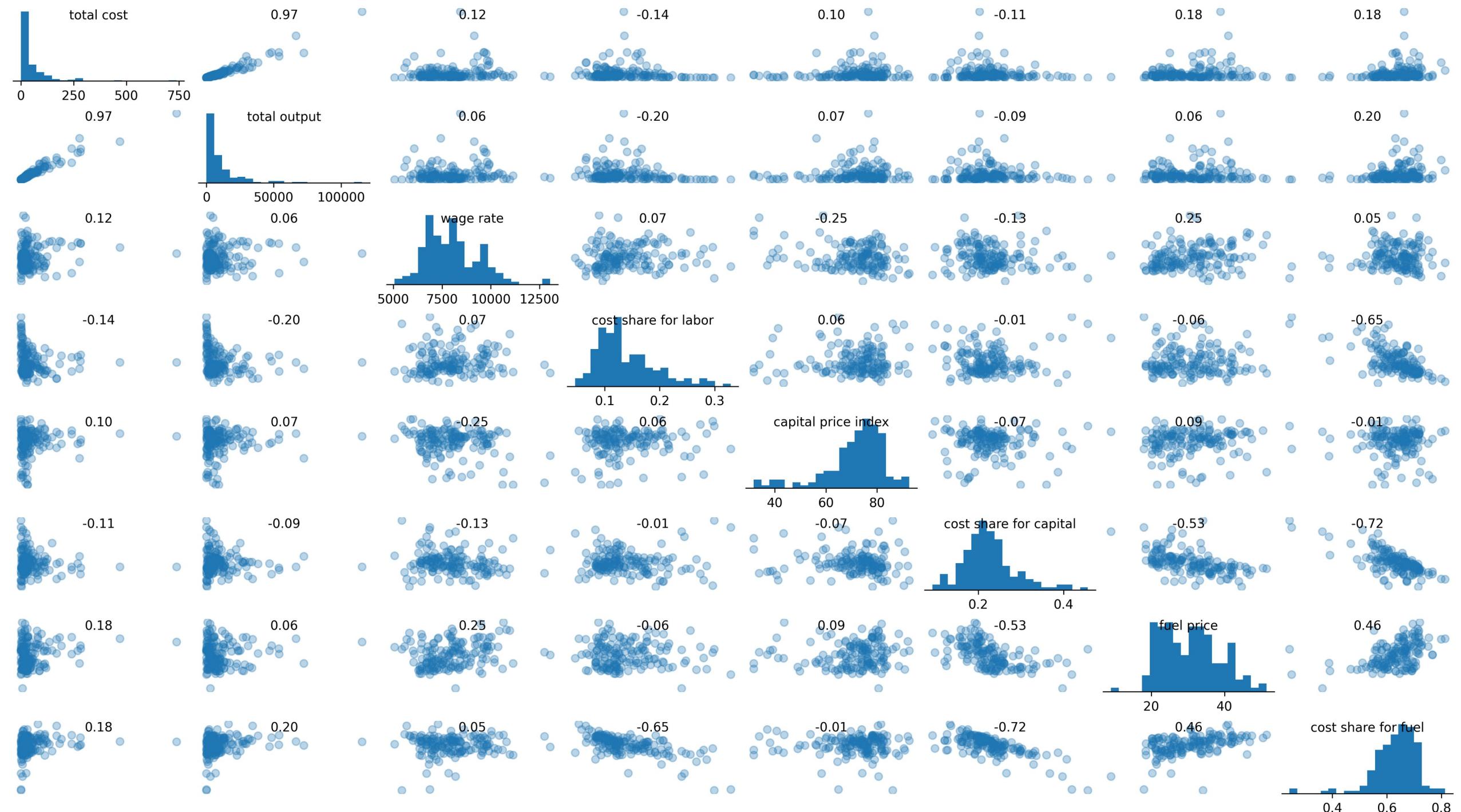


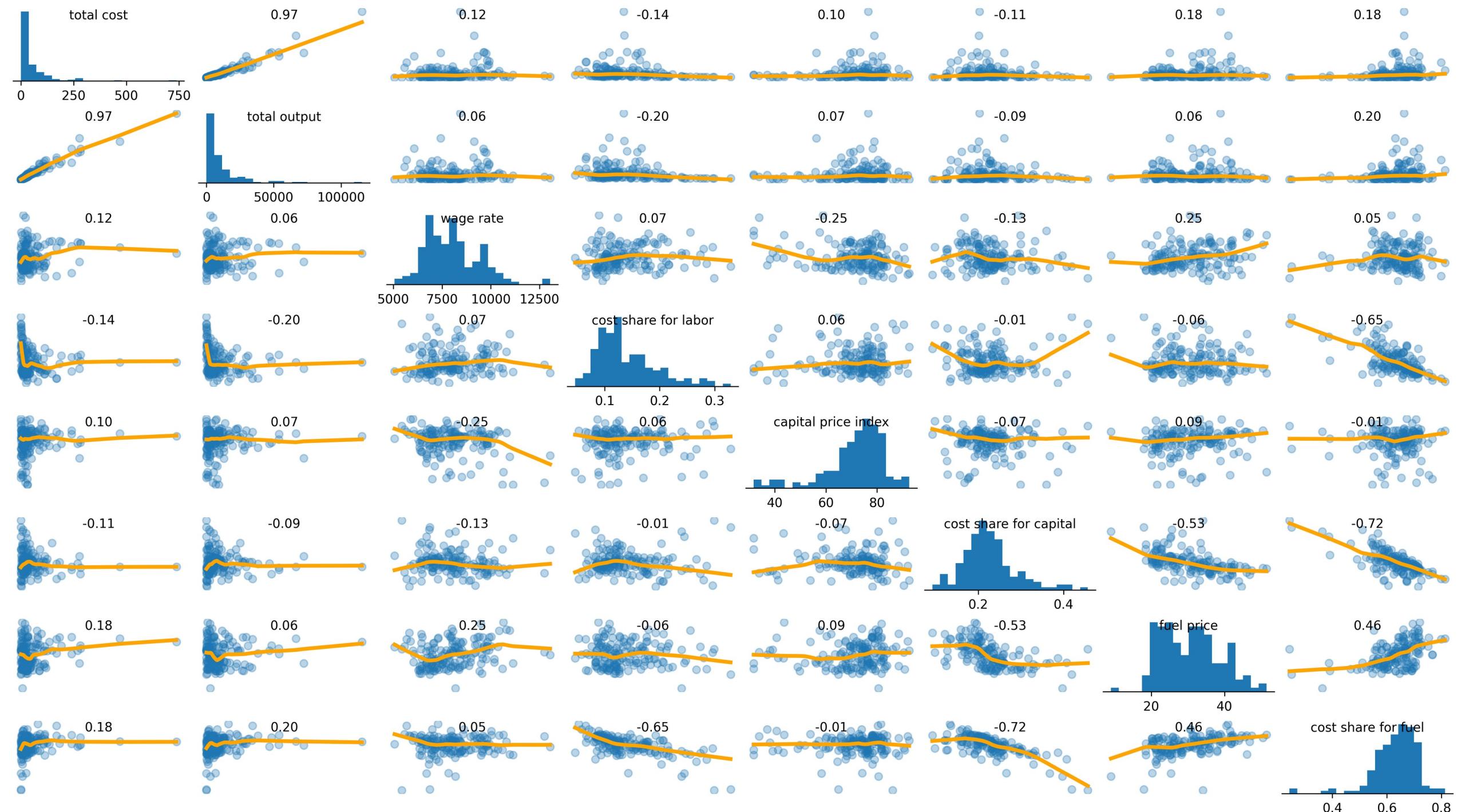
```
plt.violinplot(d / d.std(), vert=False)
```



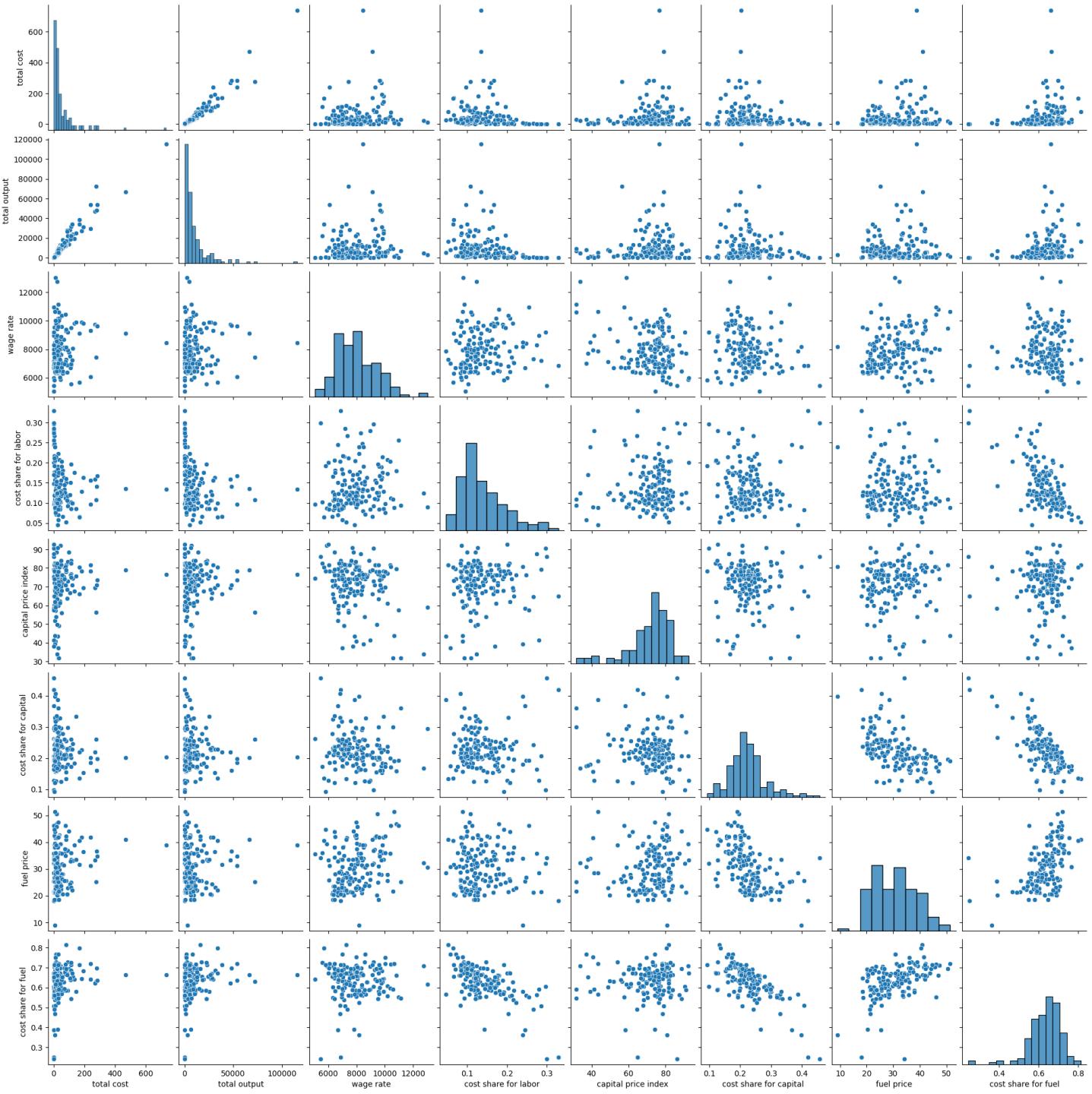








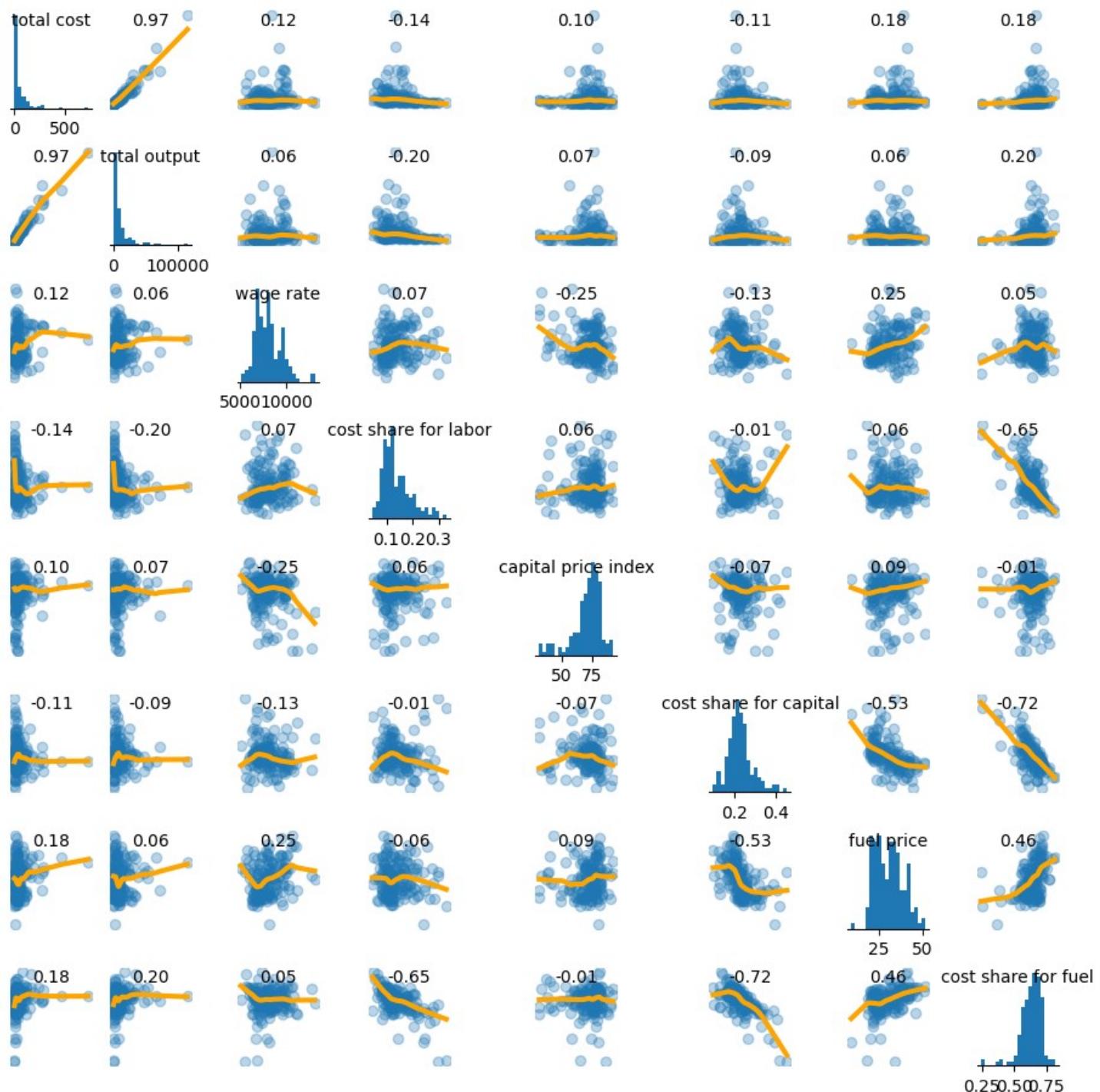
```
sns.pairplot(d)
```



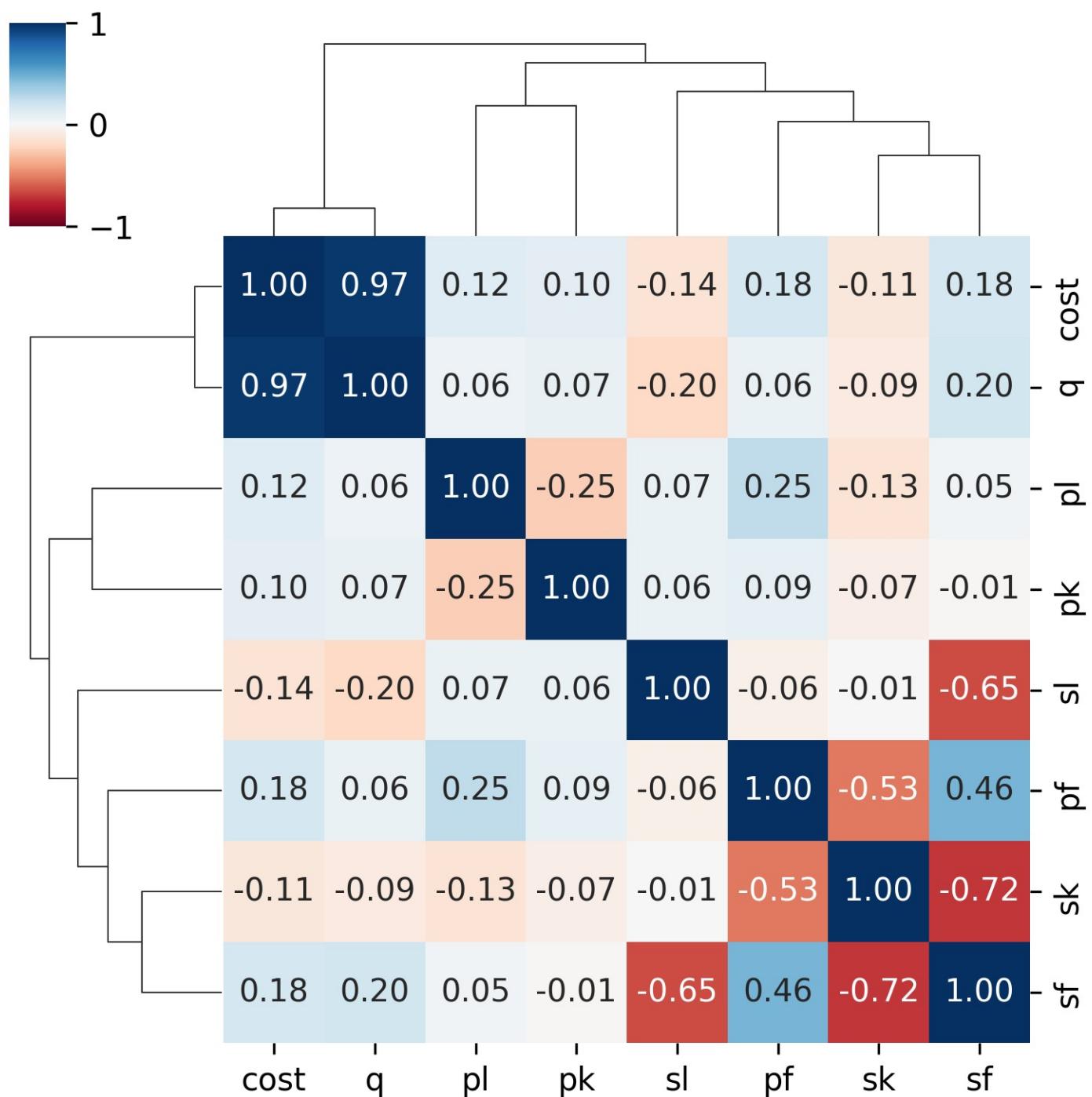
```

k = d.shape[1]
fig, axs = plt.subplots(
    k, k,
    figsize = (16,9),
    layout = 'constrained',
)
for i, id1 in enumerate( d.columns ):
    for j, id2 in enumerate( d.columns ):
        ax = axs[i,j]
        if i == j:
            ax.hist( d[id1], bins = 20, density = True )
            ax.text(.5, .99, id1,
                    ha = 'center', va = 'top',
                    transform = ax.transAxes)
            for side in ['left', 'right', 'top']:
                ax.spines[side].set_visible(False)
            ax.set_yticks([])
        if i != j:
            ax.scatter( d[id2], d[id1], alpha = .3 )
            ax.text(
                .5, .99,
                f"np.corrcoef( d[{id1}], d[{id2}] )[0,1]:.2f}",
                ha = 'center', va = 'top',
                transform = ax.transAxes,
            )
            ys = statsmodels.nonparametric.smoothers_lowess(
                d[id1], d[id2],
                frac = .5,
            )
            ax.plot(
                ys[:,0], ys[:,1],
                color = 'orange', linewidth = 3,
            )
            ax.axis('off')
plt.show()

```

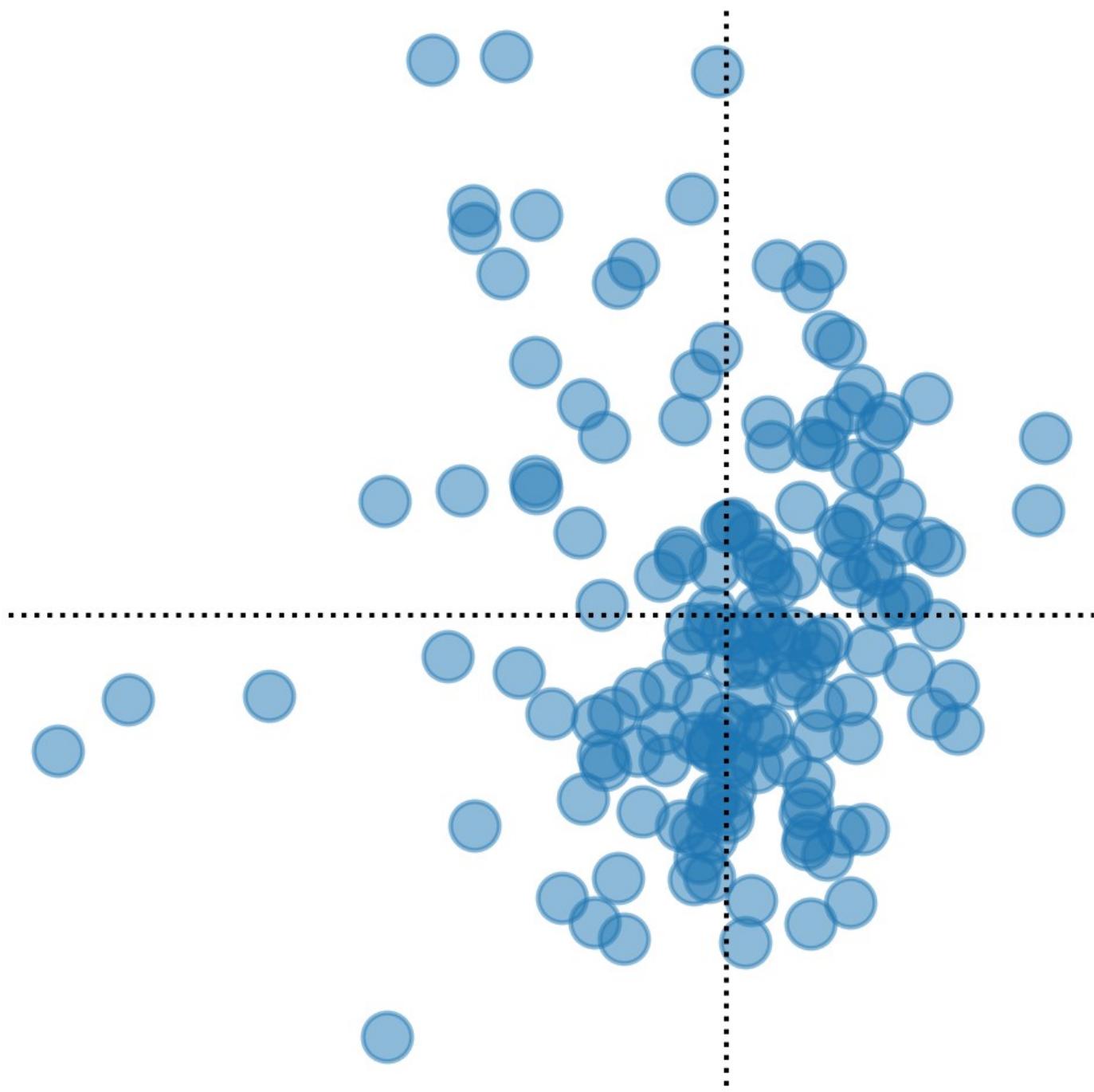


```
sns.clustermap(  
    d.corr(),  
    vmin = -1,  
    vmax = +1,  
    cmap = 'RdBu',  
    figsize = (5, 5),  
    annot = True,  
    fmt = ".2f",  
)
```



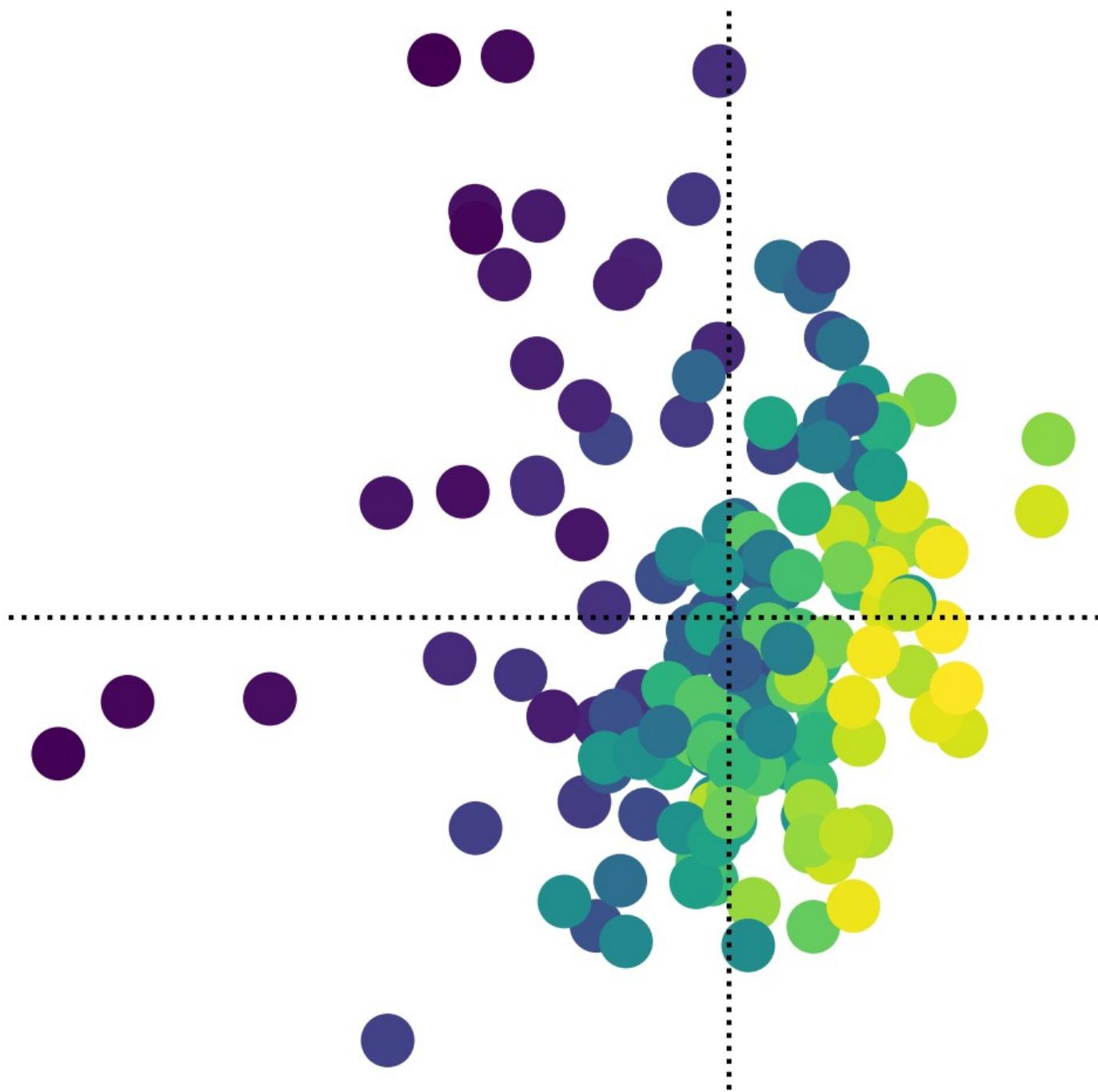
```
x = sklearn.decomposition.PCA(2).  
    fit_transform(dd)
```

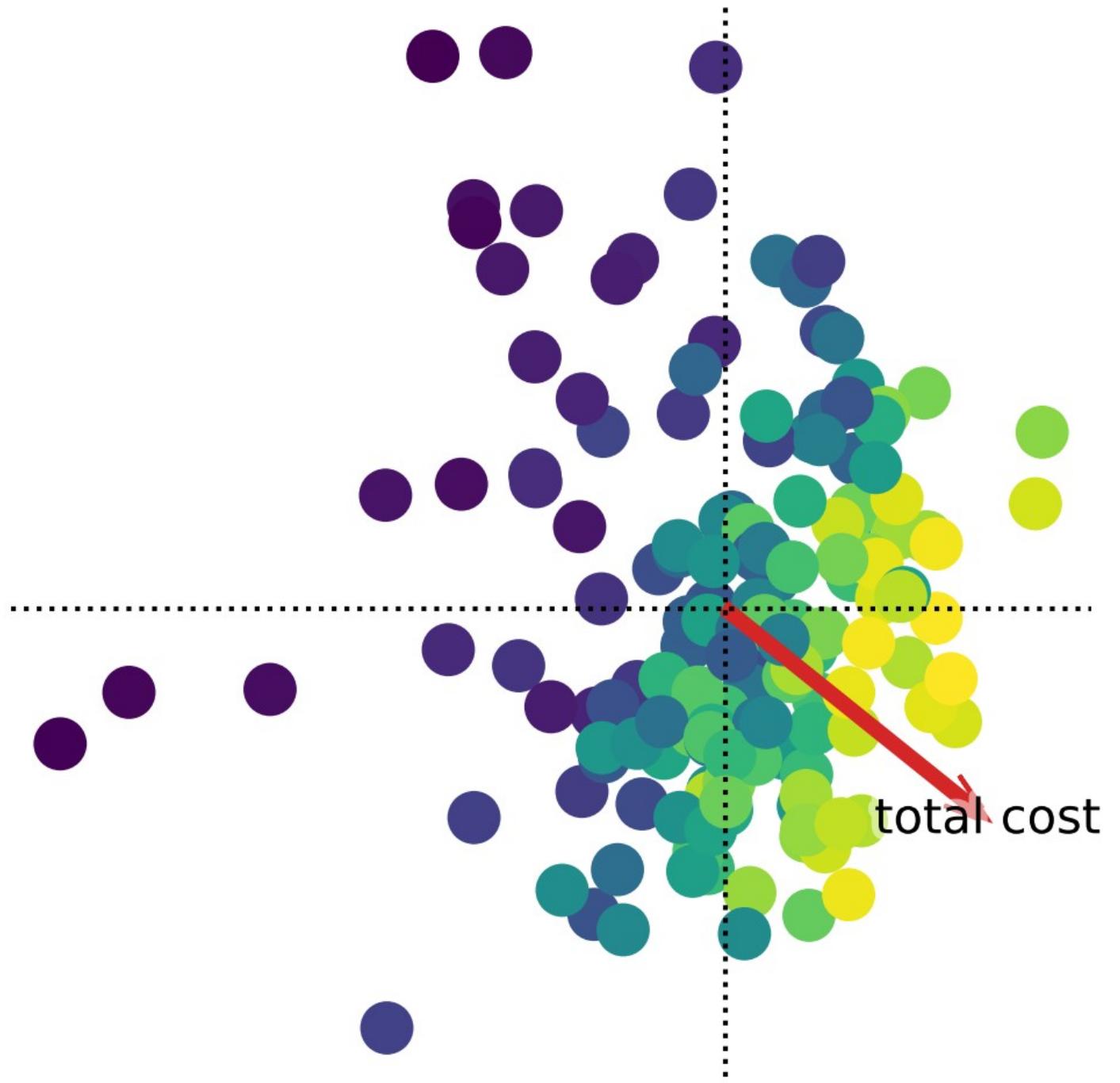
```
plt.scatter( X[:,0], X[:,1] )
```

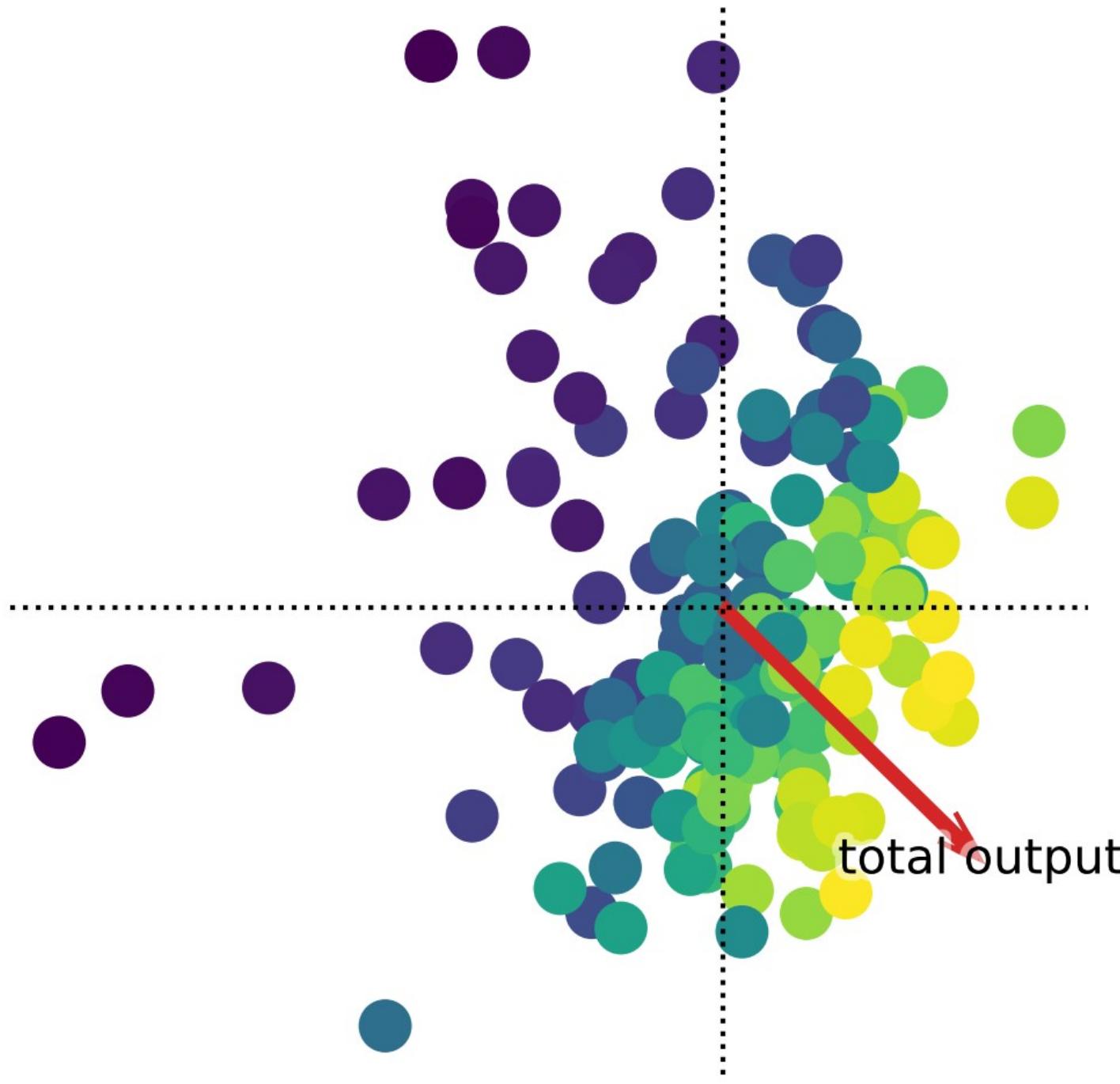


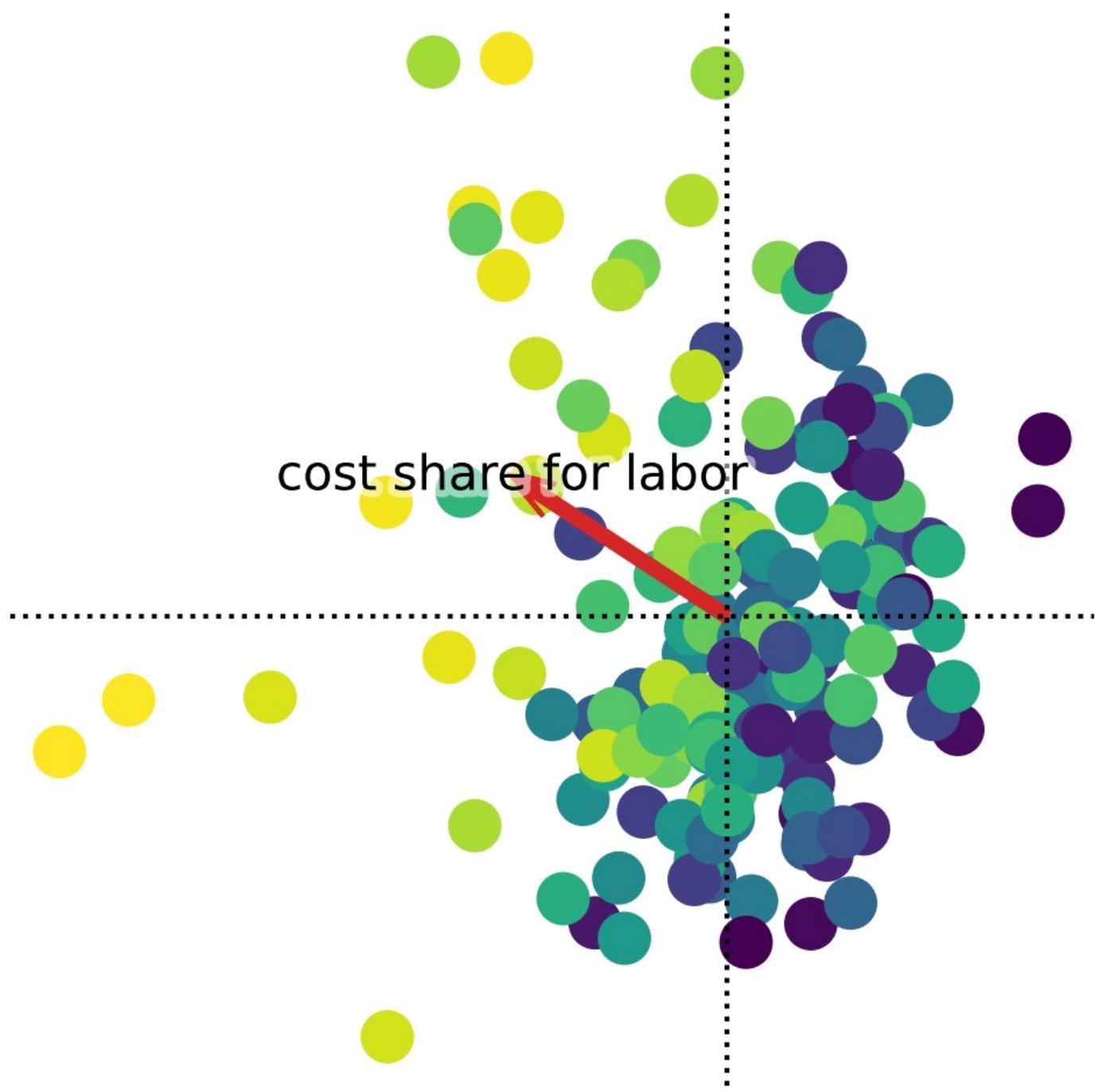
```
x = sklearn.decomposition.PCA(2).  
    fit_transform(d)
```

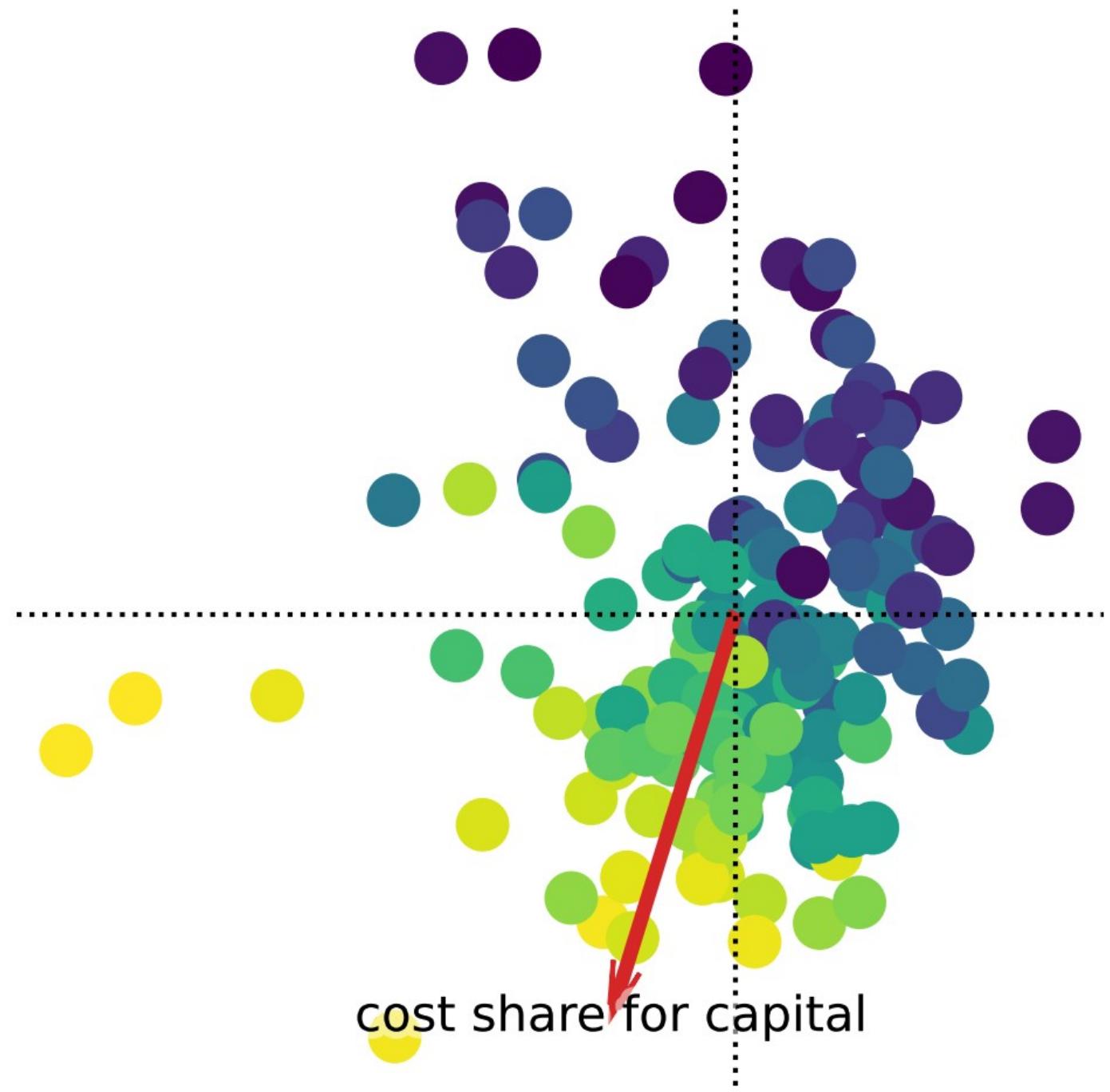
```
plt.scatter(  
    X[:,0], X[:,1],  
    c = d[column],  
)
```

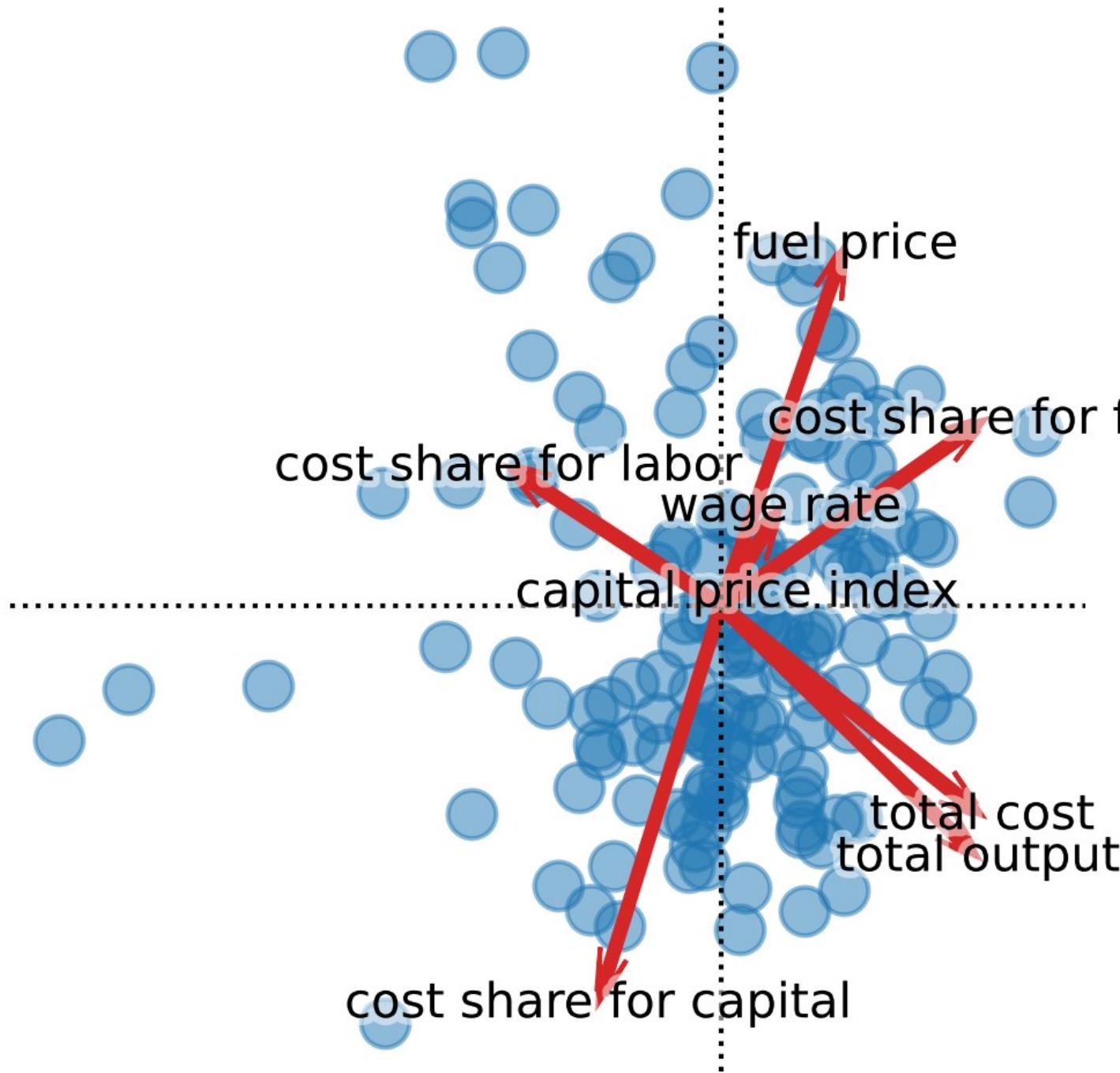












```
X = sklearn.decomposition.PCA(2).  
    fit_transform(d)
```

```
fig, axs = plt.subplots( 3, 3 )
```

```
for i, column in enumerate(d.columns):
```

```
    ax = axs.flatten()[i]
```

```
    ax.scatter(
```

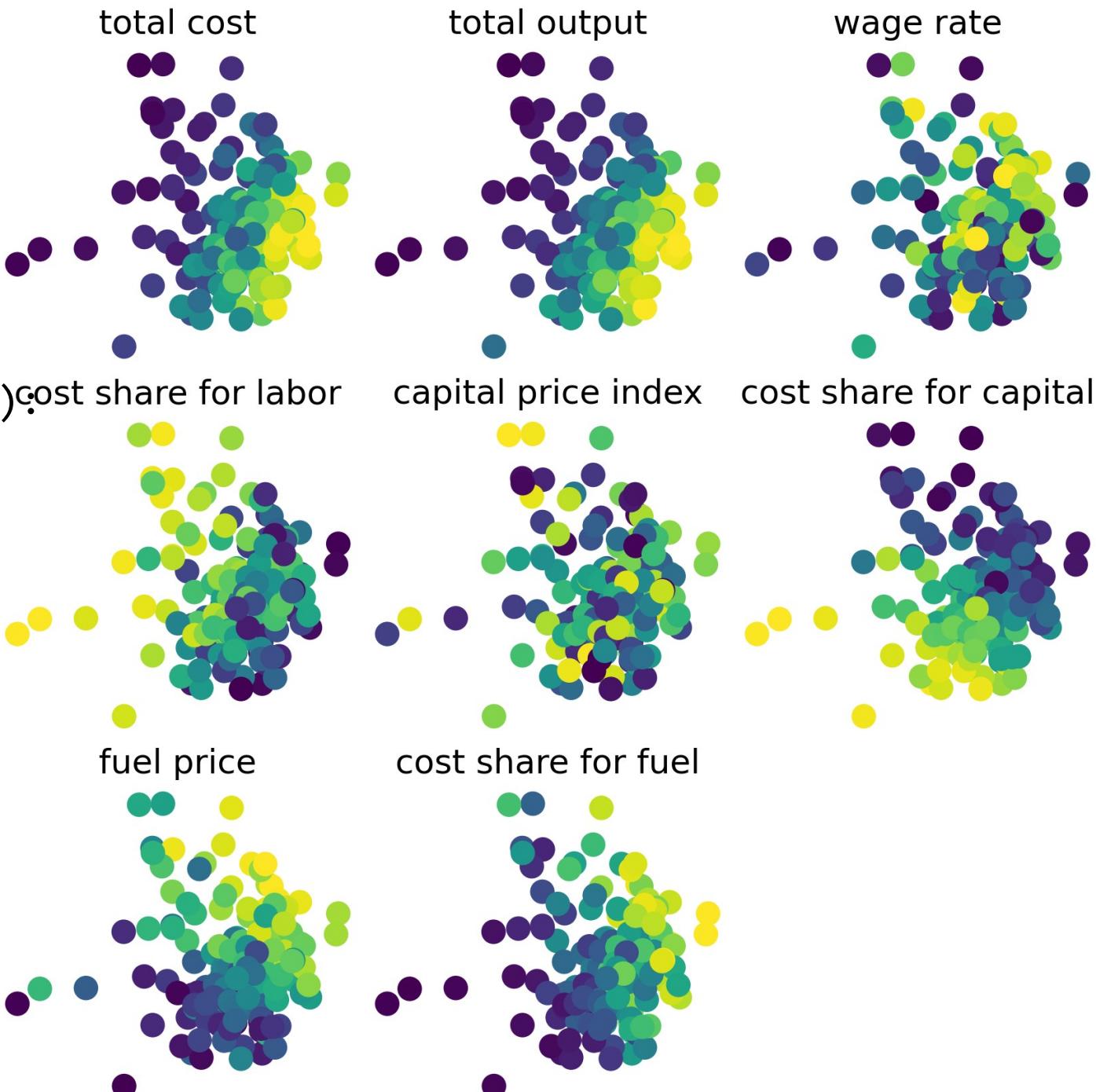
```
        X[:,0], X[:,1],  
        c = d[column],
```

```
)
```

```
    ax.axis('off')
```

```
    ax.set_title(column)
```

```
plt.show()
```



```
X = sklearn.decomposition.PCA(2).  
    fit_transform(d)
```

```
fig, axs = plt.subplots( 3, 3 )
```

```
for i, column in enumerate(d.columns):
```

```
    ax = axs.flatten()[i]
```

```
    ax.scatter(
```

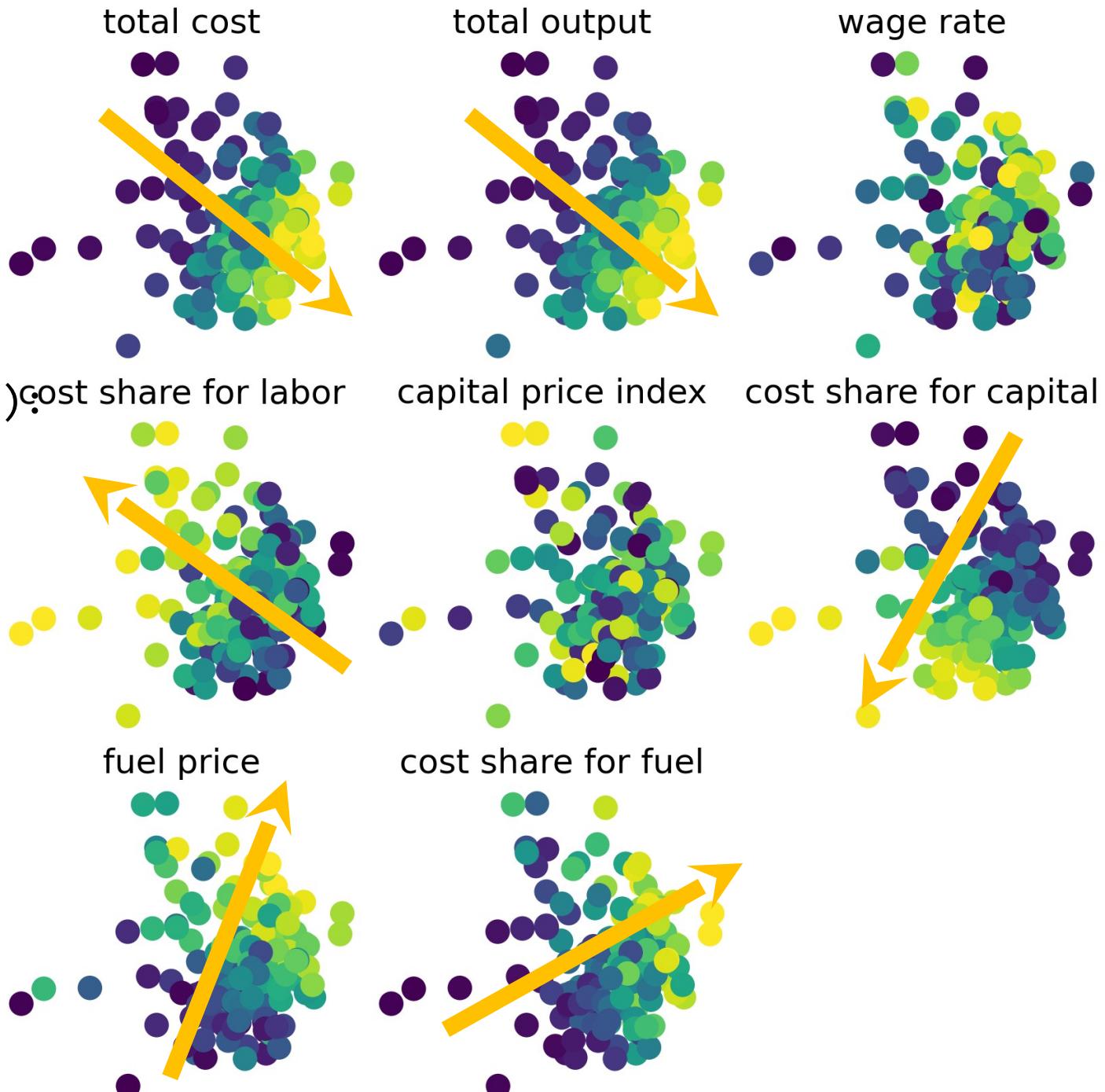
```
        X[:,0], X[:,1],  
        c = d[column],
```

```
)
```

```
    ax.axis('off')
```

```
    ax.set_title(column)
```

```
plt.show()
```



```
x = UMAP().fit_transform(d)  
plt.scatter( X[:,0], X[:,1] )
```



```
X = UMAP().fit_transform(d)
```

```
fig, axs = plt.subplots( 3, 3 )
```

```
for i, column in enumerate(d.columns):
```

```
    ax = axs.flatten()[i]
```

```
    ax.scatter(
```

```
        X[:,0], X[:,1],  
        c = d[column],
```

```
)
```

```
    ax.axis('off')
```

```
    ax.set_title(column)
```

```
plt.show()
```

total cost



total output



wage rate



cost share for labor



capital price index



cost share for capital



fuel price



cost share for fuel



```
X = UMAP().fit_transform(d)
```

```
fig, axs = plt.subplots( 3, 3 )
```

```
for i, column in enumerate(d.columns):
```

```
    ax = axs.flatten()[i]
```

```
    ax.scatter(
```

```
        X[:,0], X[:,1],  
        c = d[column],
```

```
)
```

```
    ax.axis('off')
```

```
    ax.set_title(column)
```

```
plt.show()
```

total cost



total output



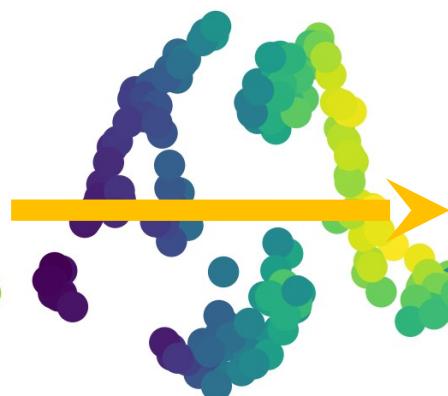
wage rate



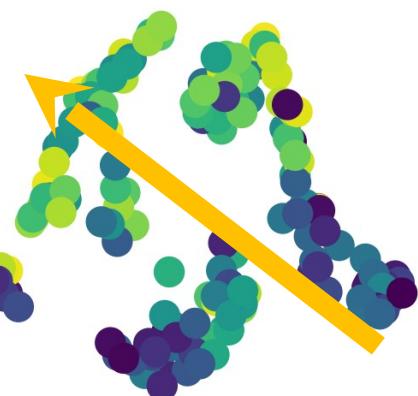
cost share for labor



capital price index



cost share for capital



fuel price



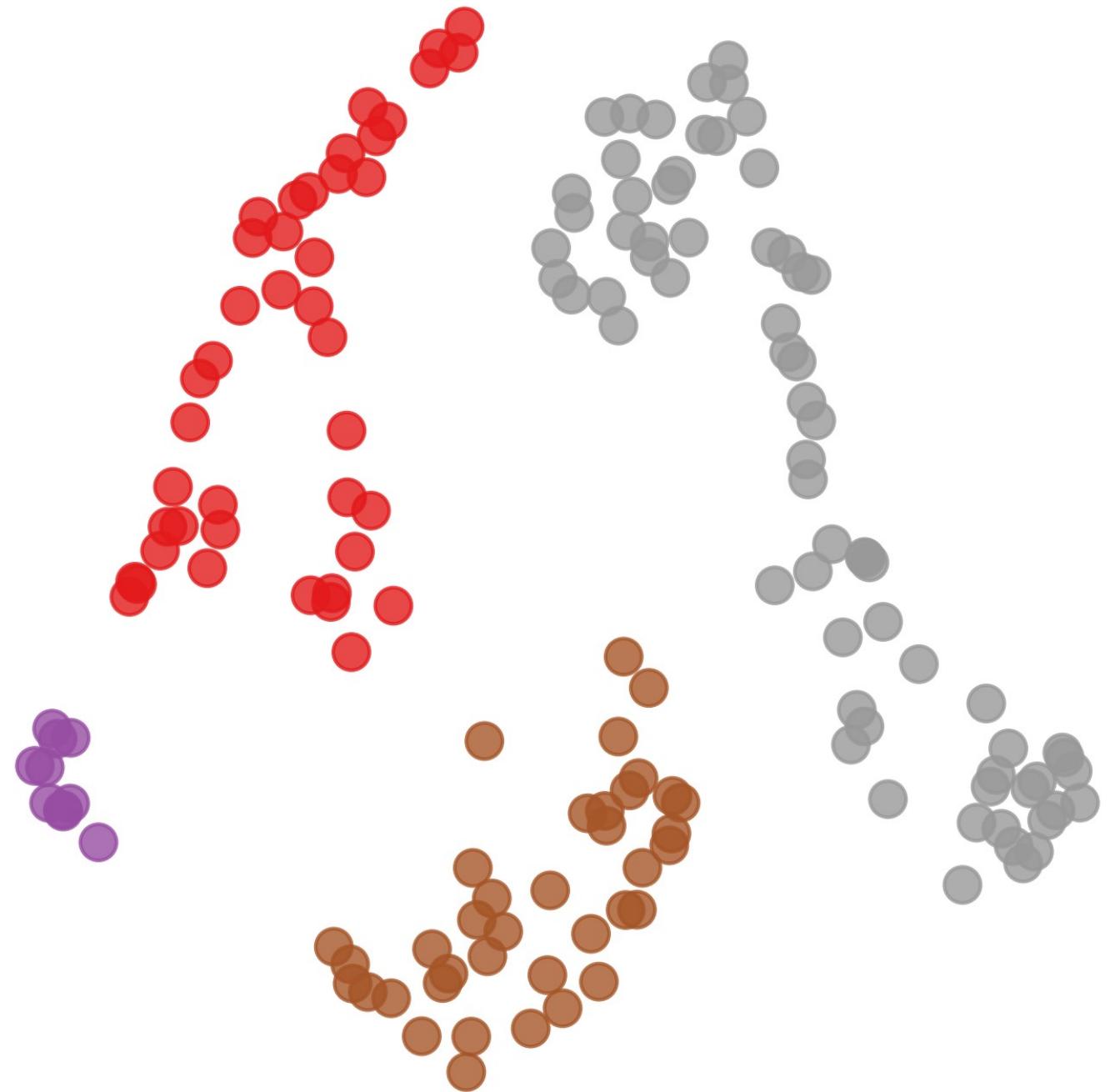
cost share for fuel



```
x = UMAP().fit_transform(d)

model = sklearn.cluster.DBSCAN()
c = model.fit(X).labels_

plt.scatter( X[:,0], X[:,1], c=c )
```

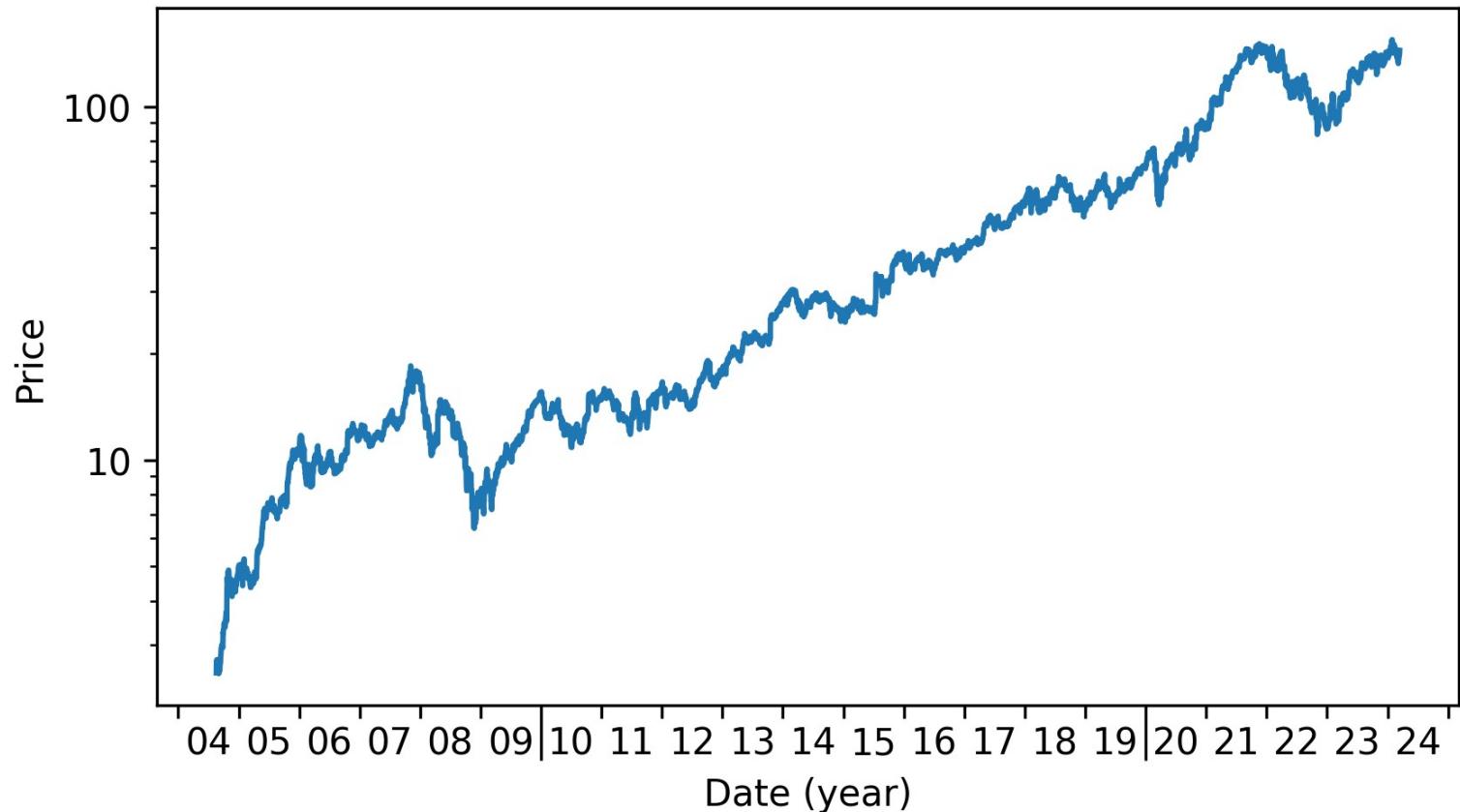


Time Series

Time Series

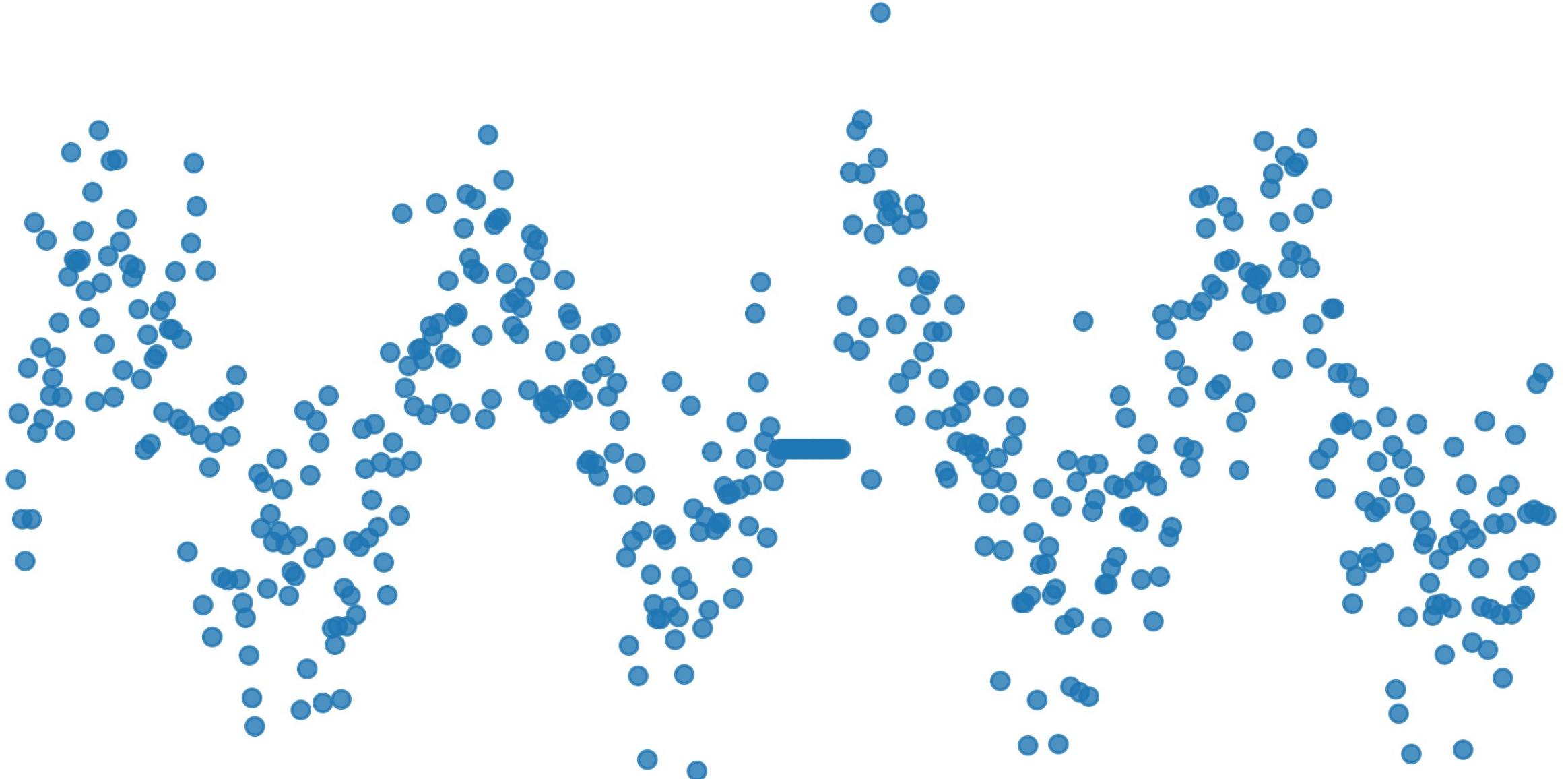
- Bivariate data, where one of the variables is time.

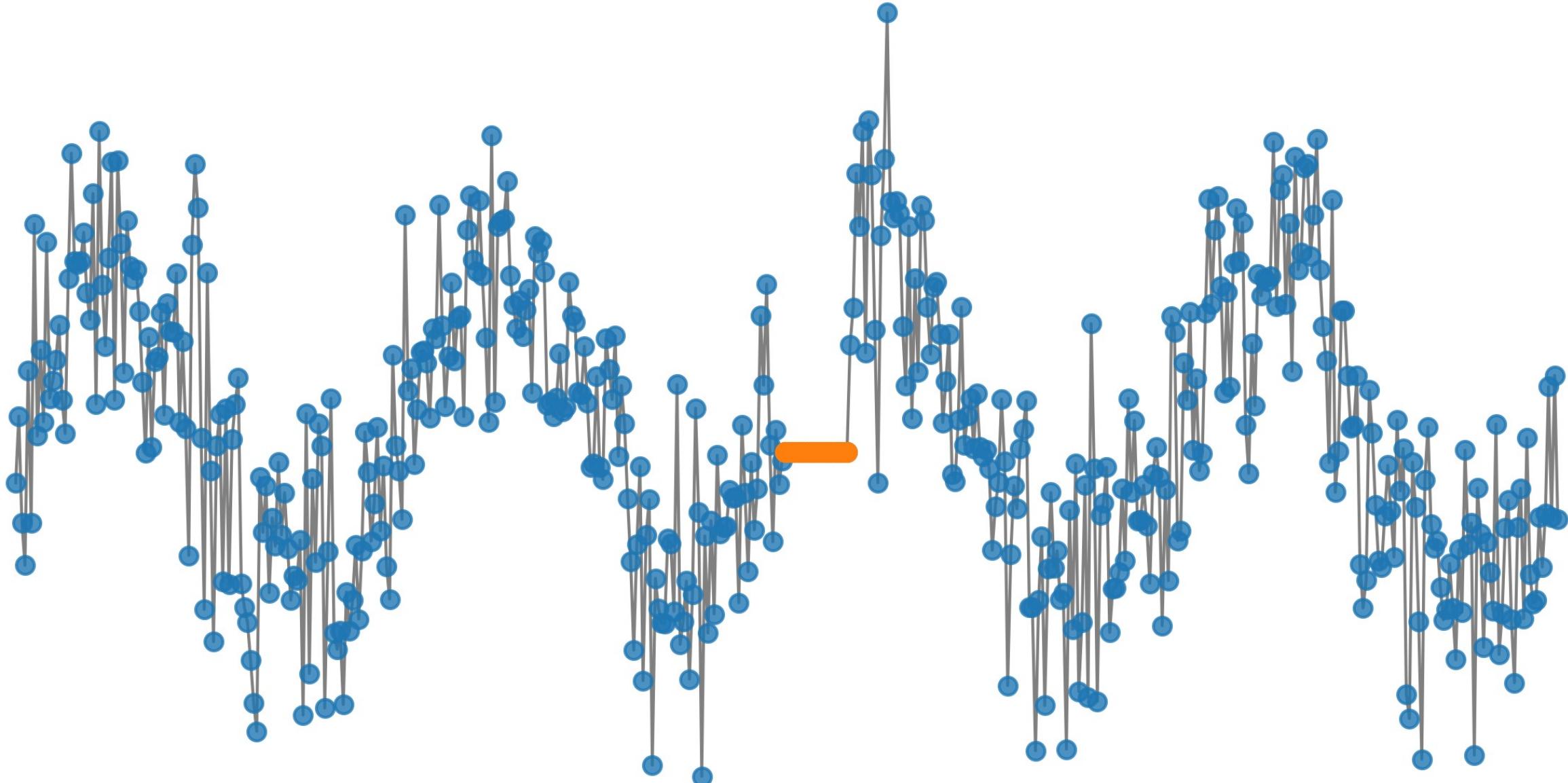
Date	Close
2004-08-19	2.50
2004-08-20	2.70
2004-08-23	2.72
2004-08-24	2.61
2004-08-25	2.64
2004-08-26	2.69

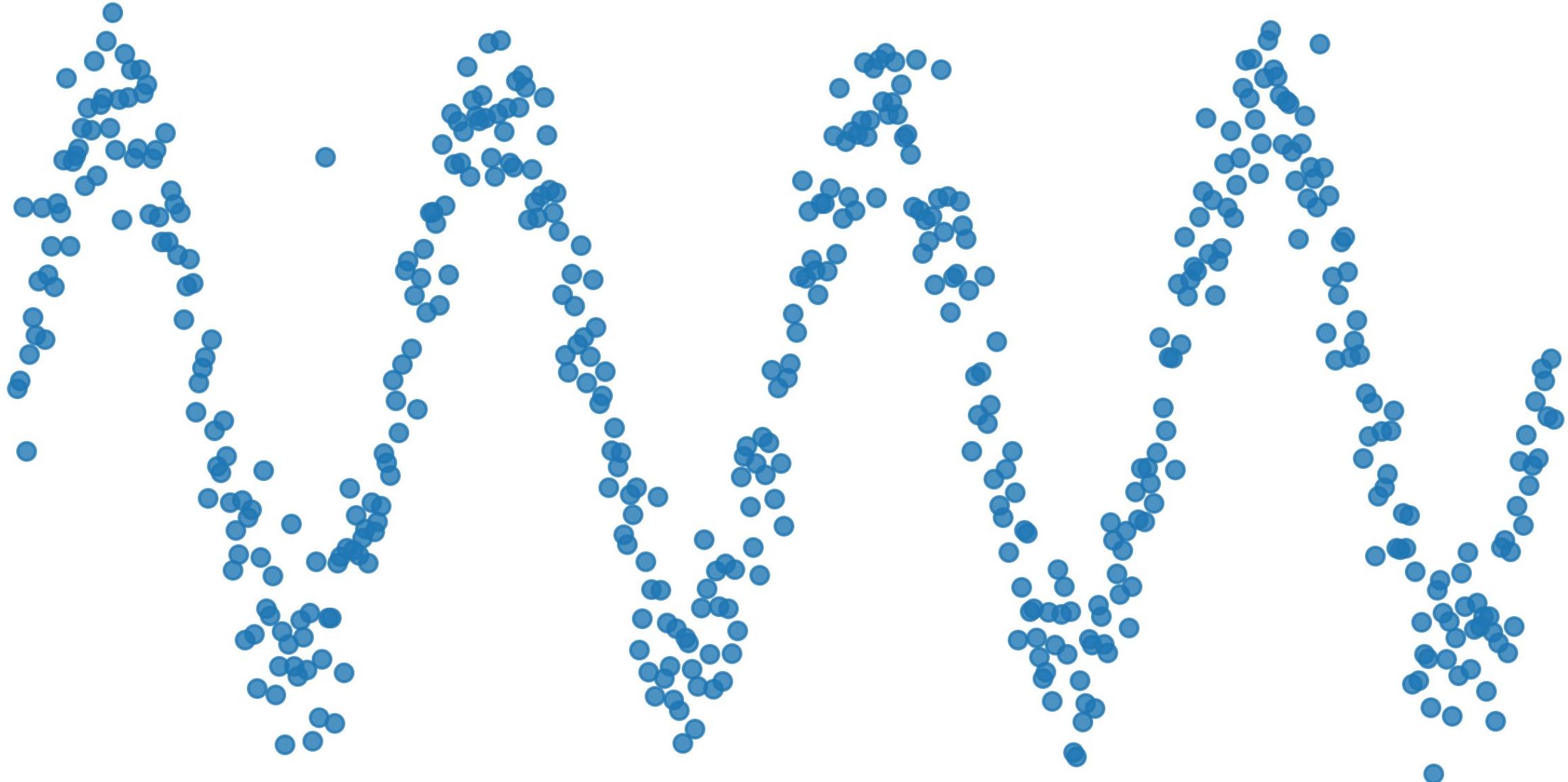


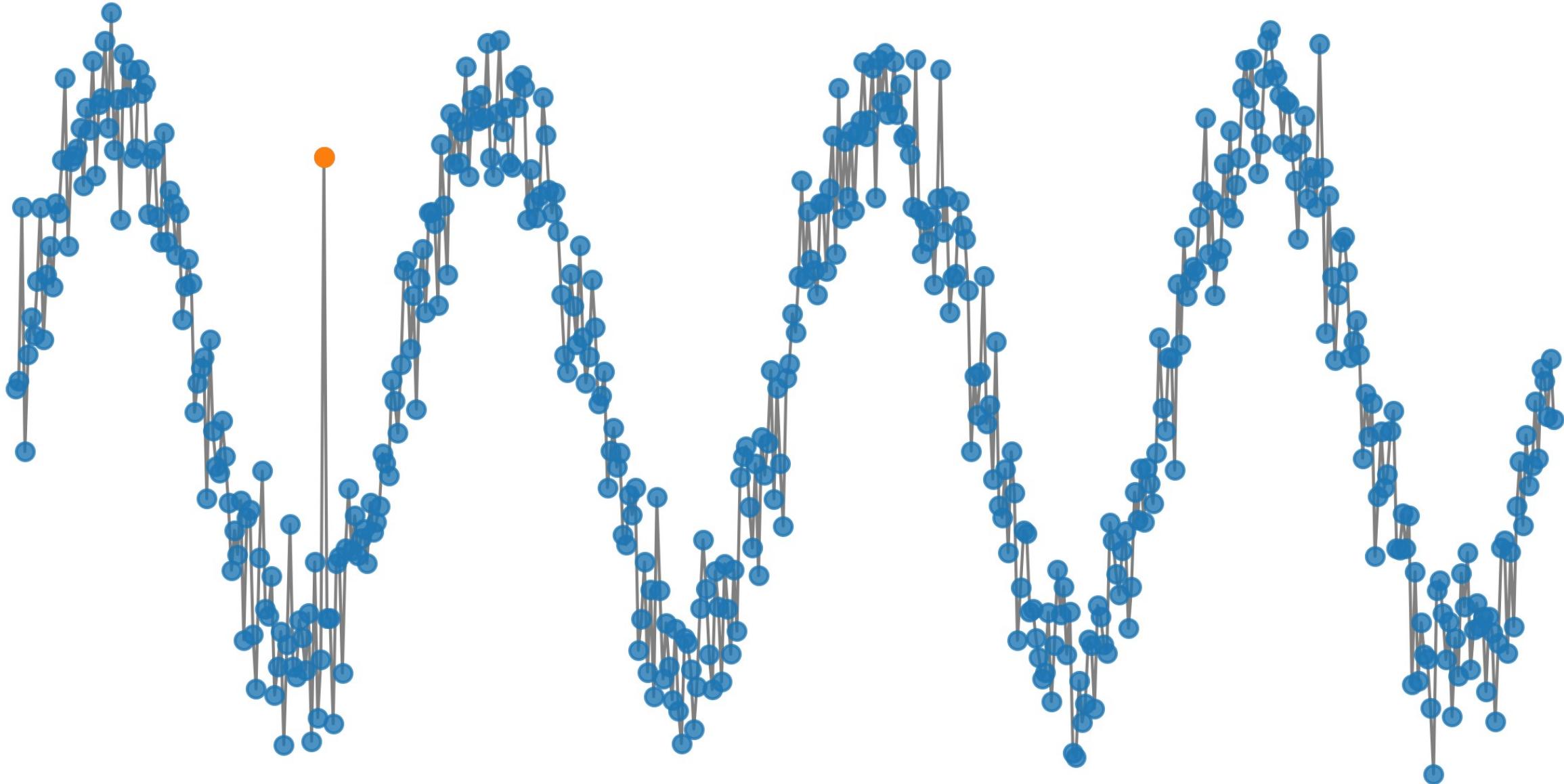
Problems

- Same as for univariate data
- Suspicious values: repetitions, anomalies, jumps
- Non-stationarity
- Irregular sampling

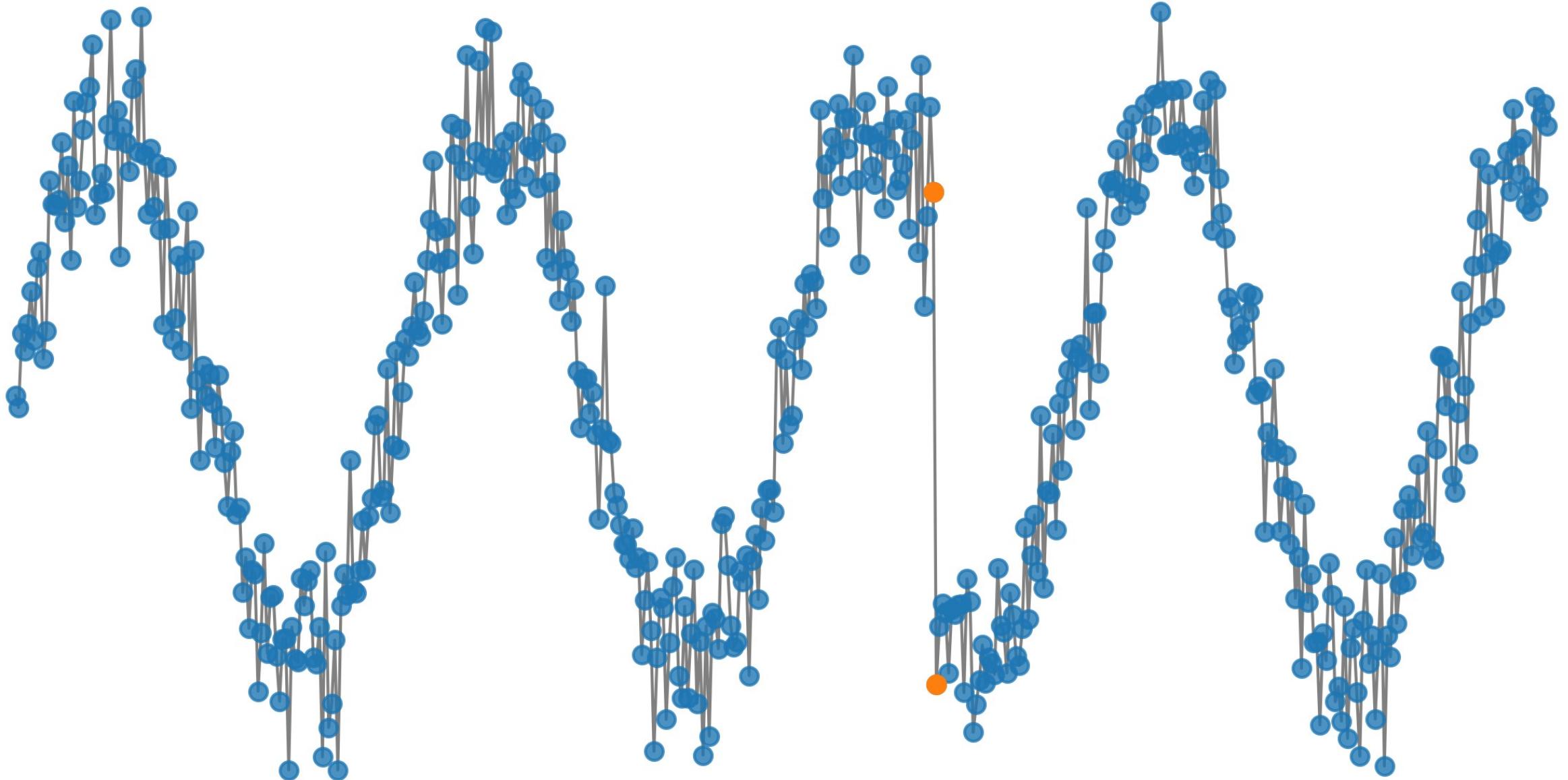


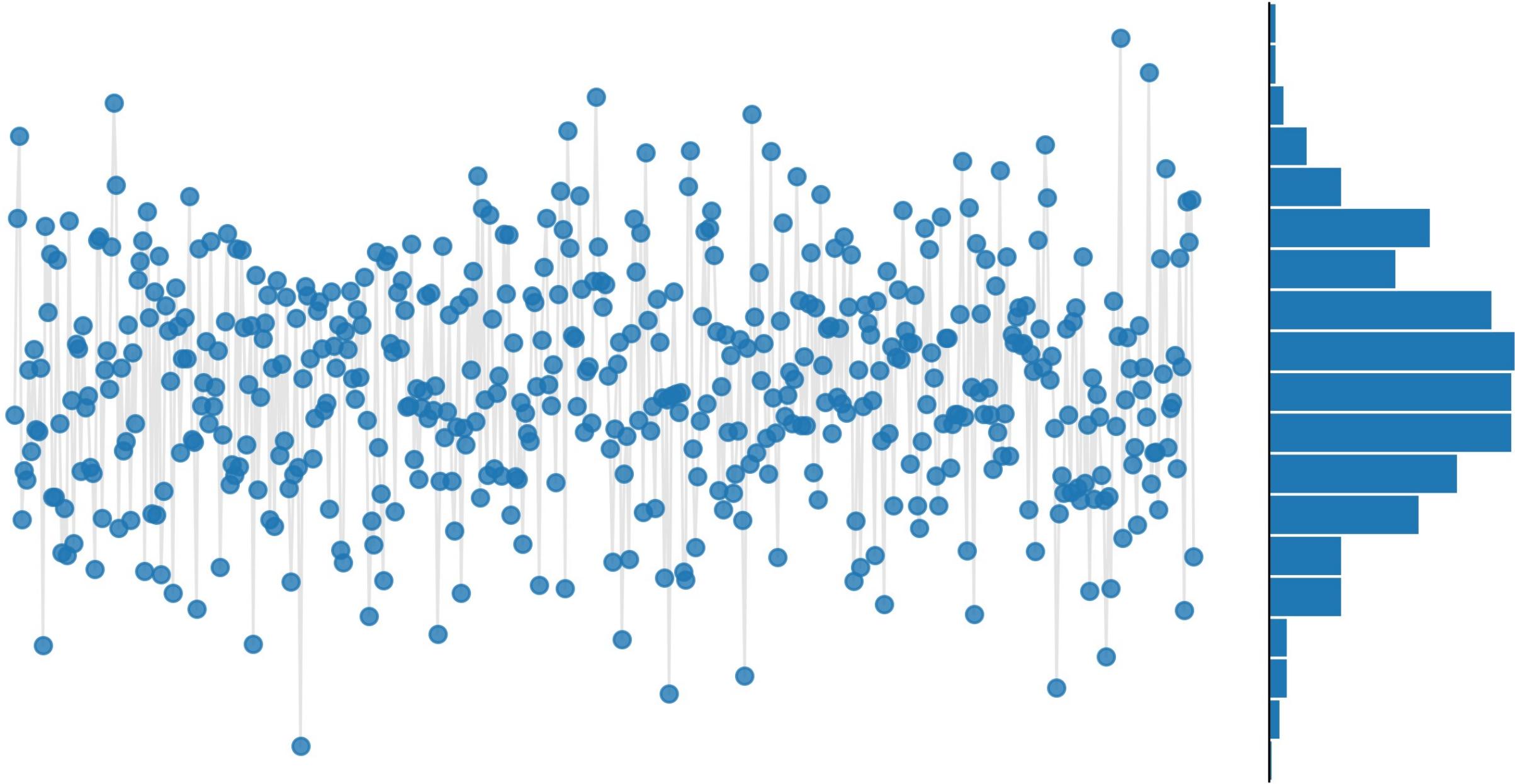


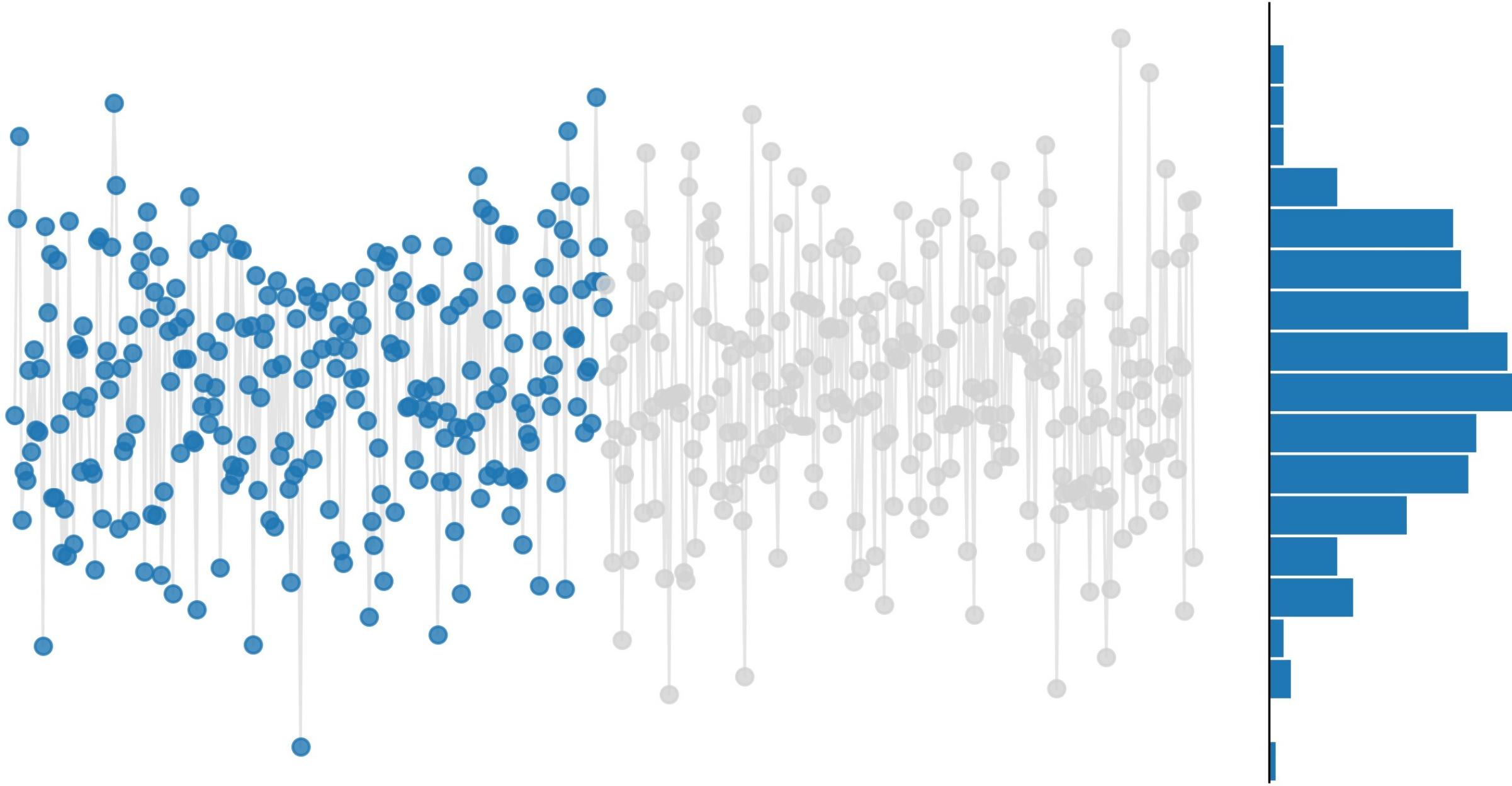


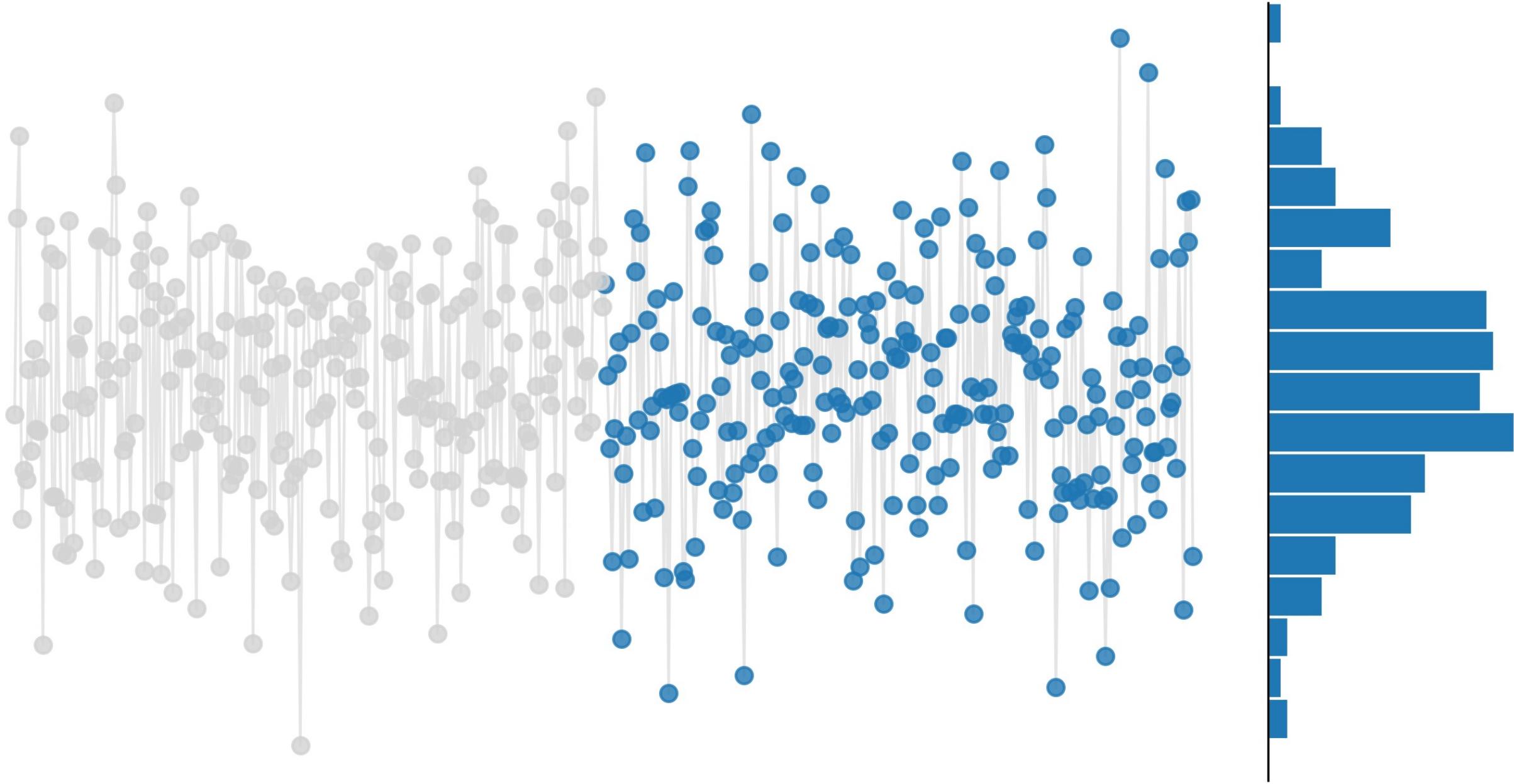


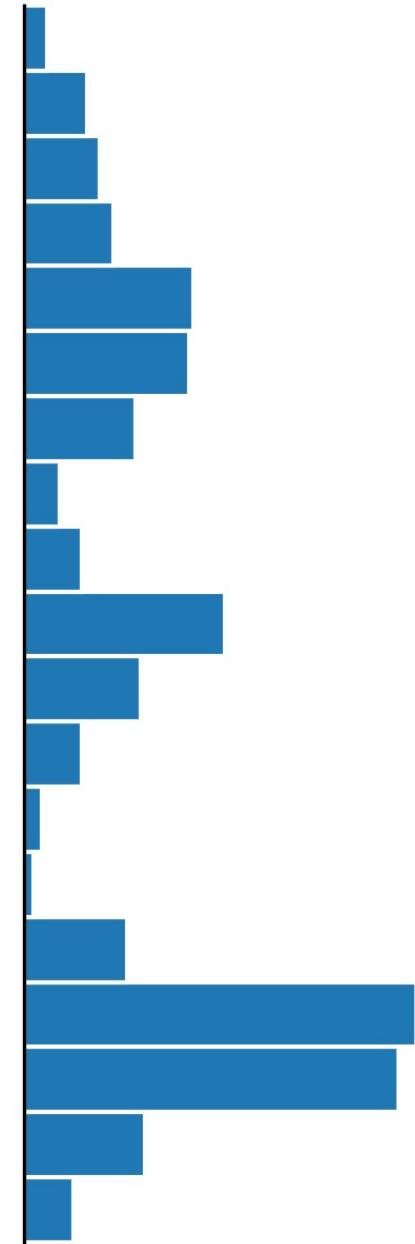
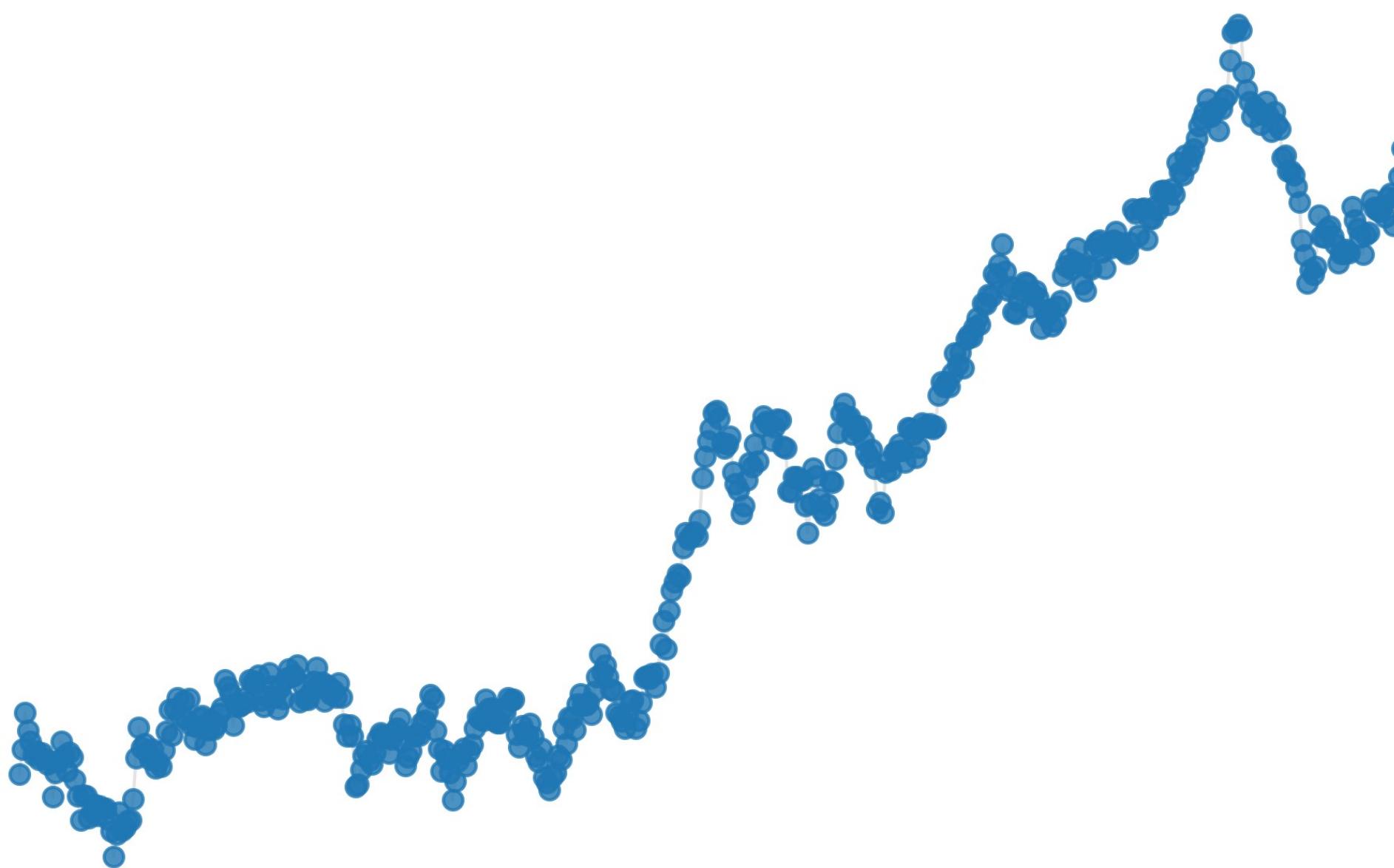


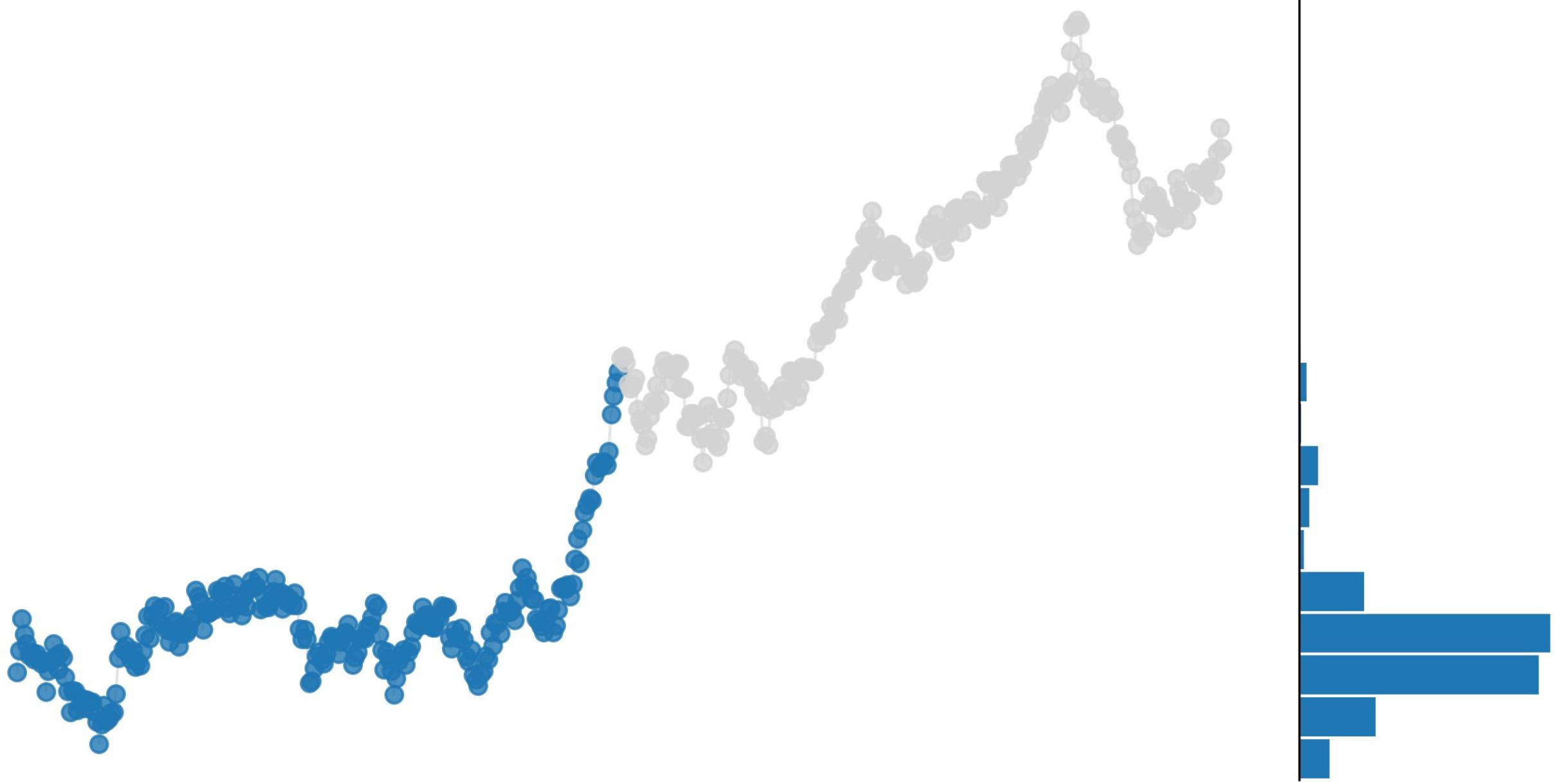


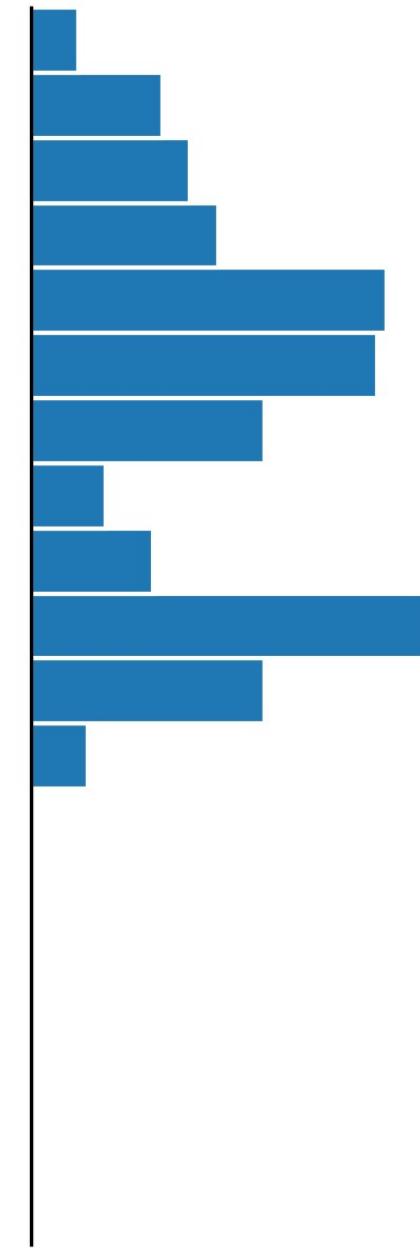
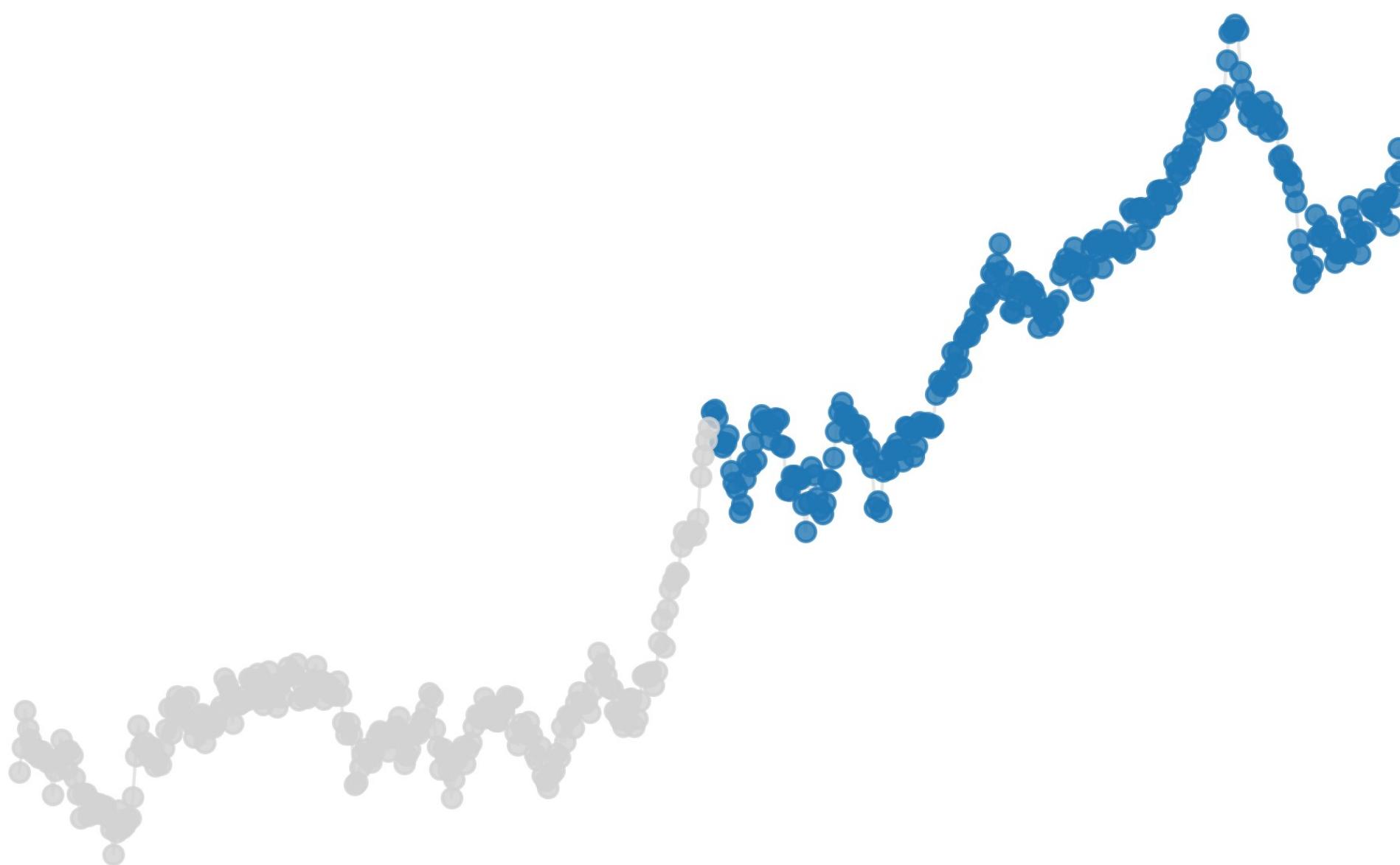


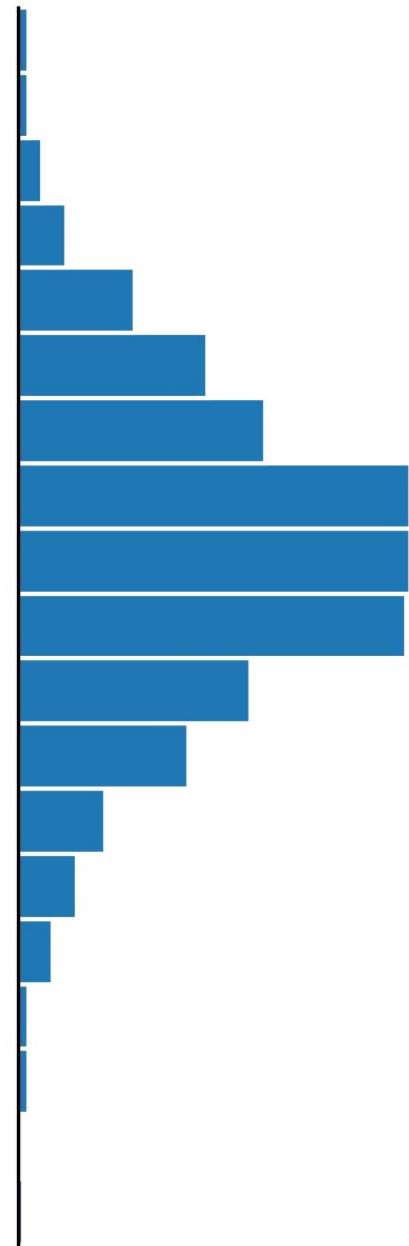
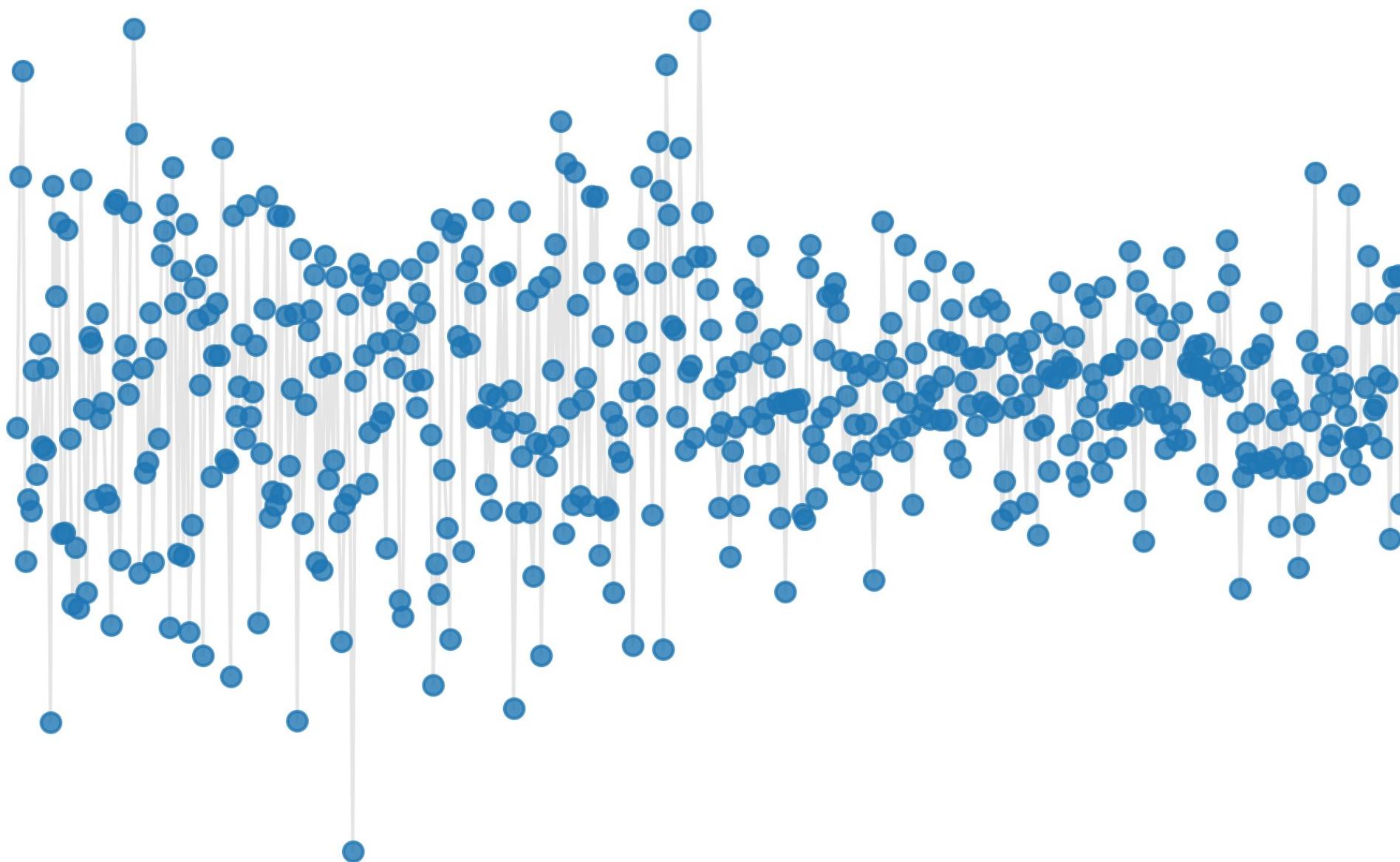


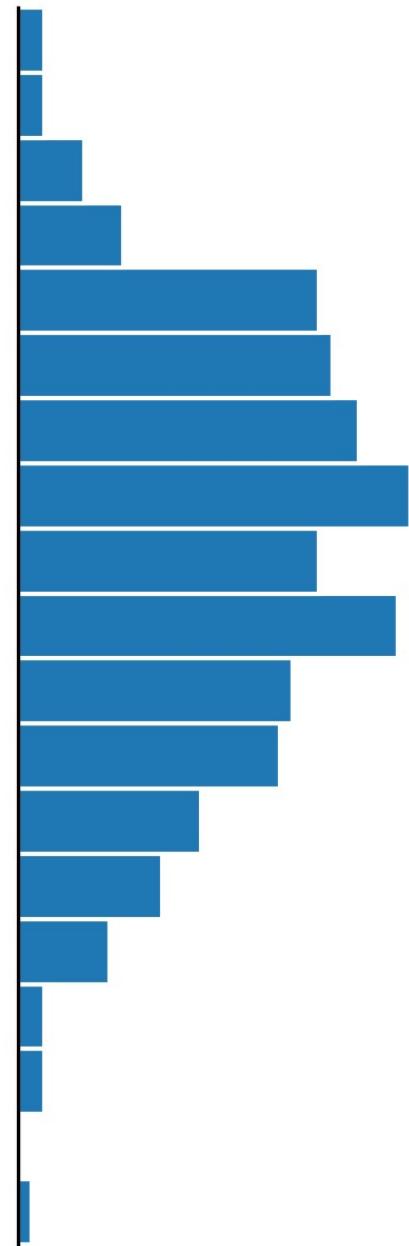
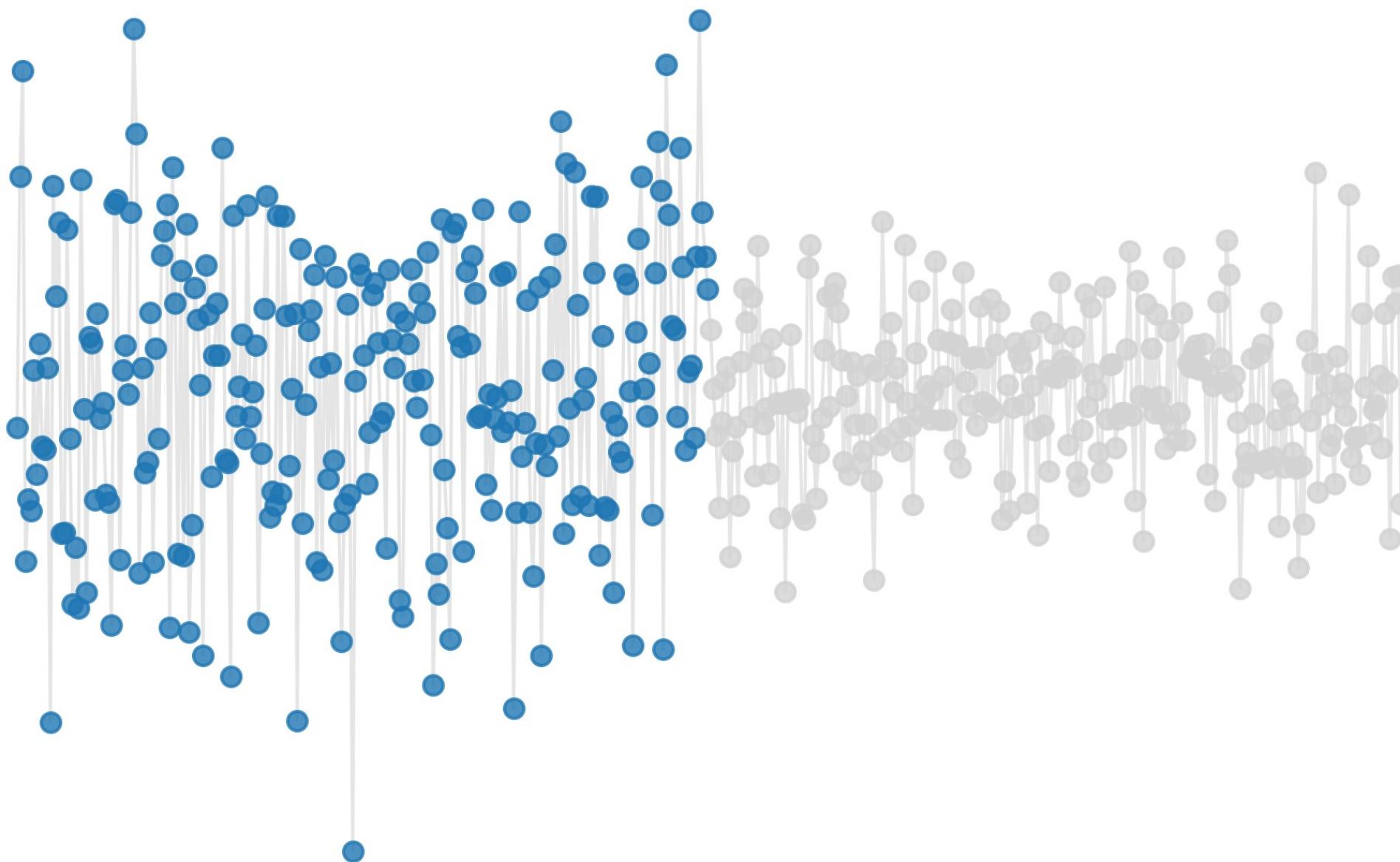


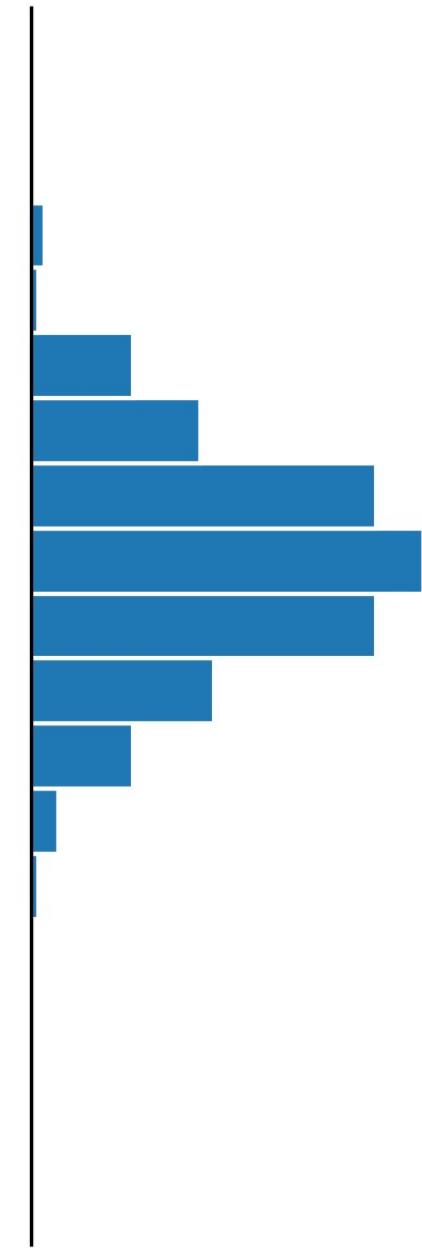
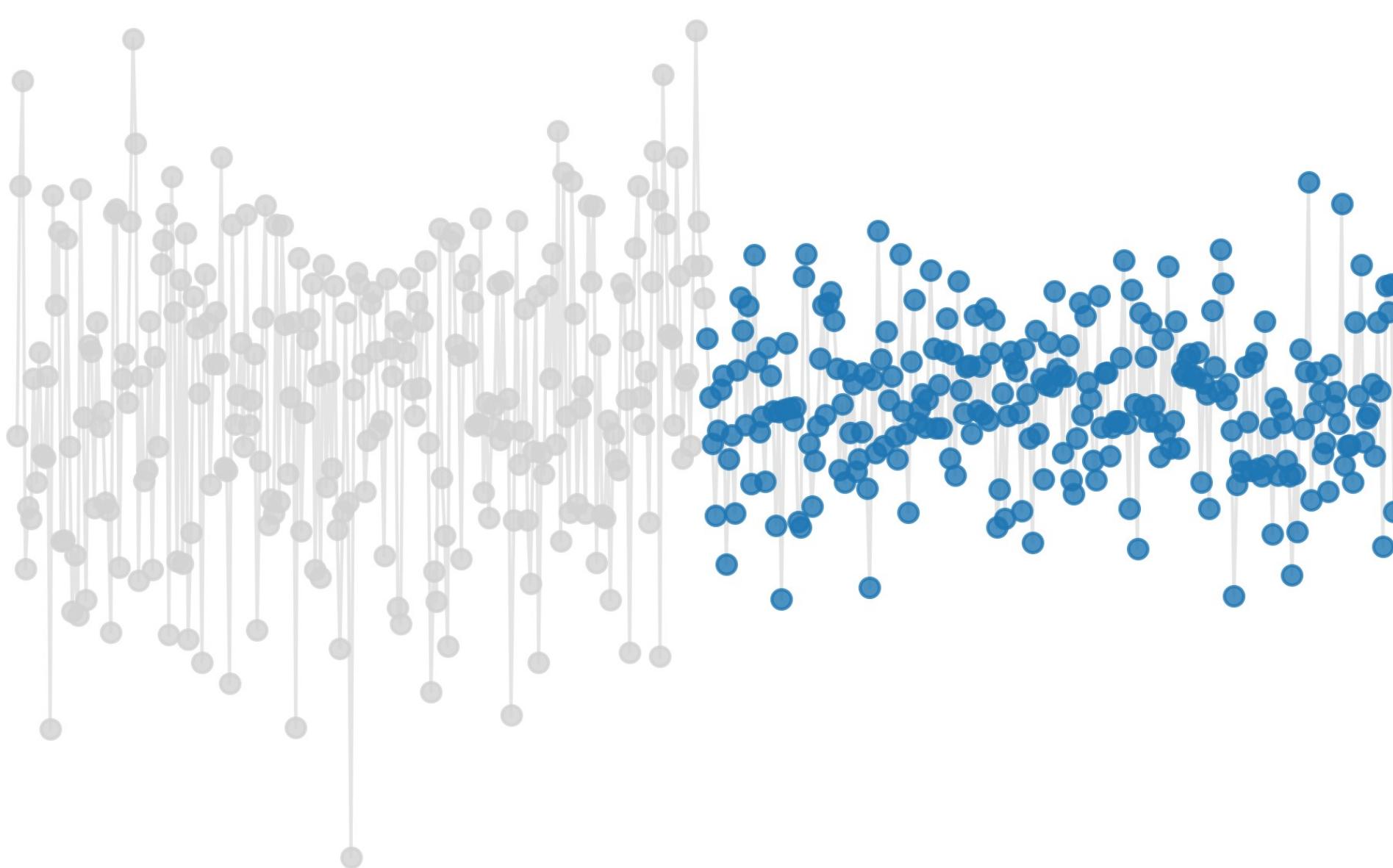












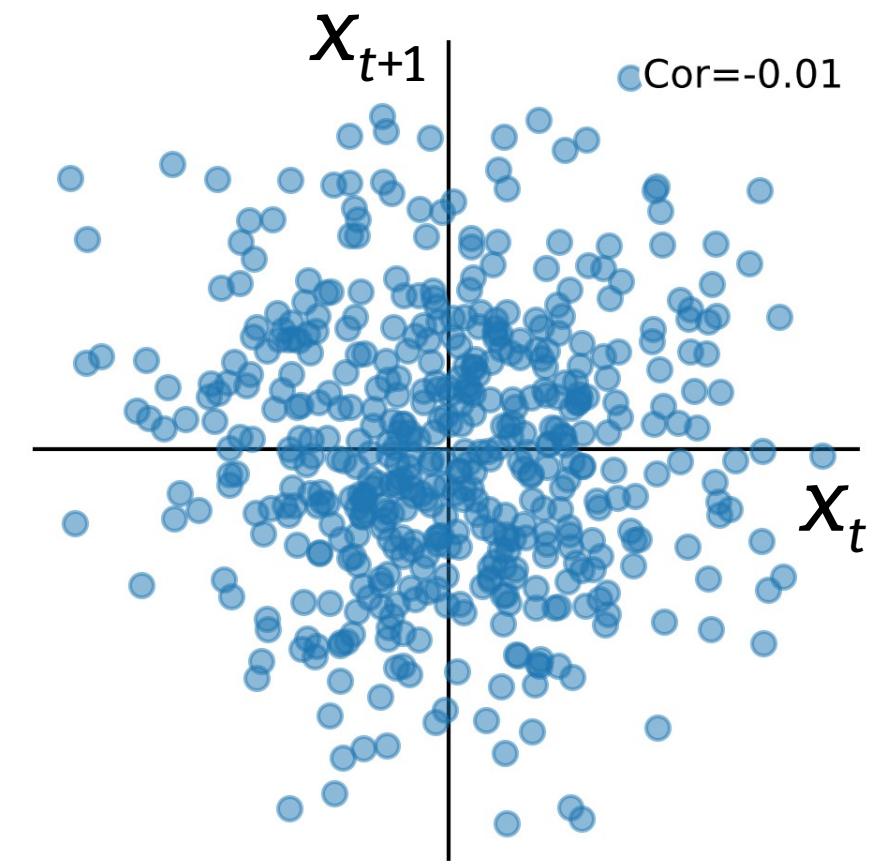
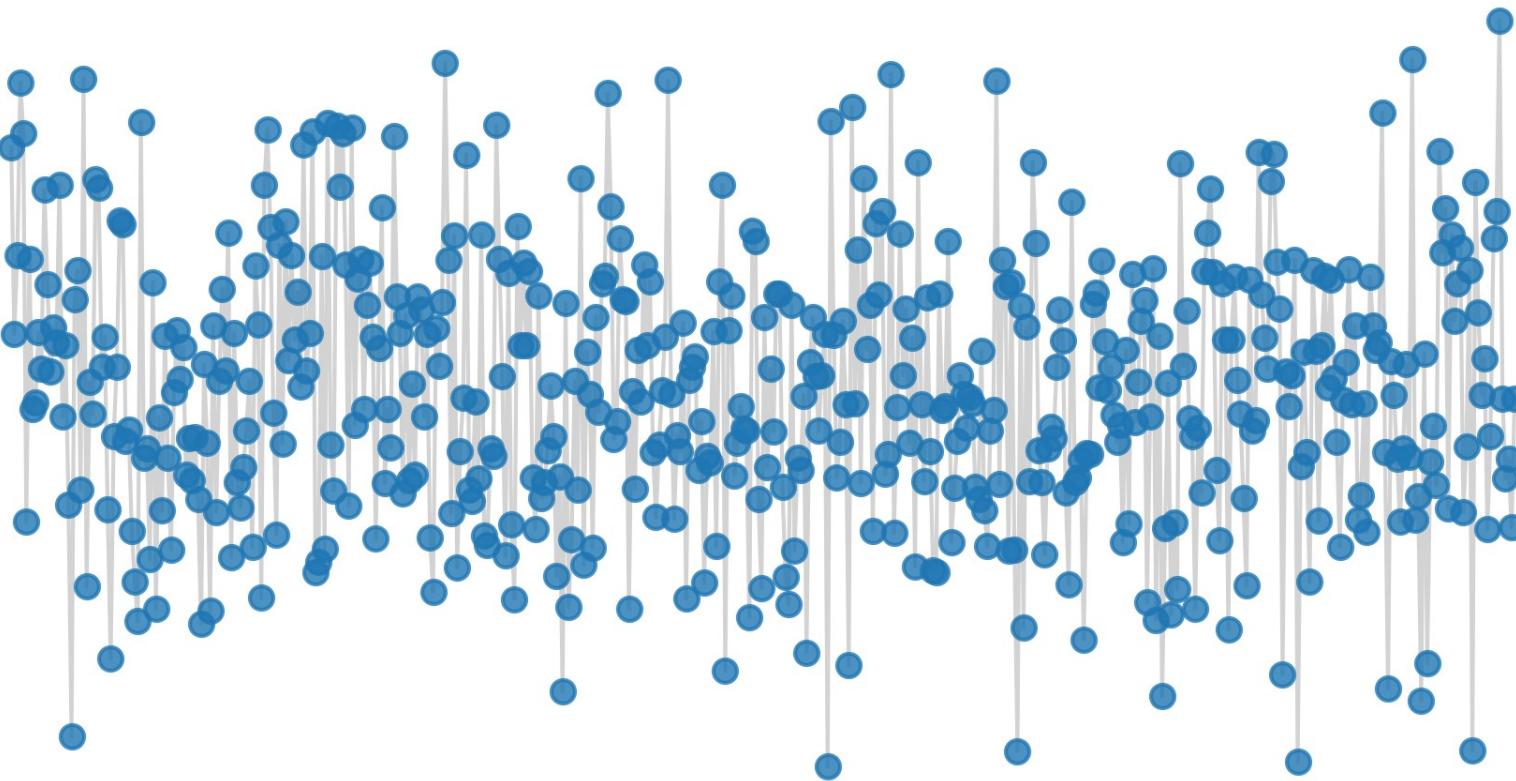
Numbers

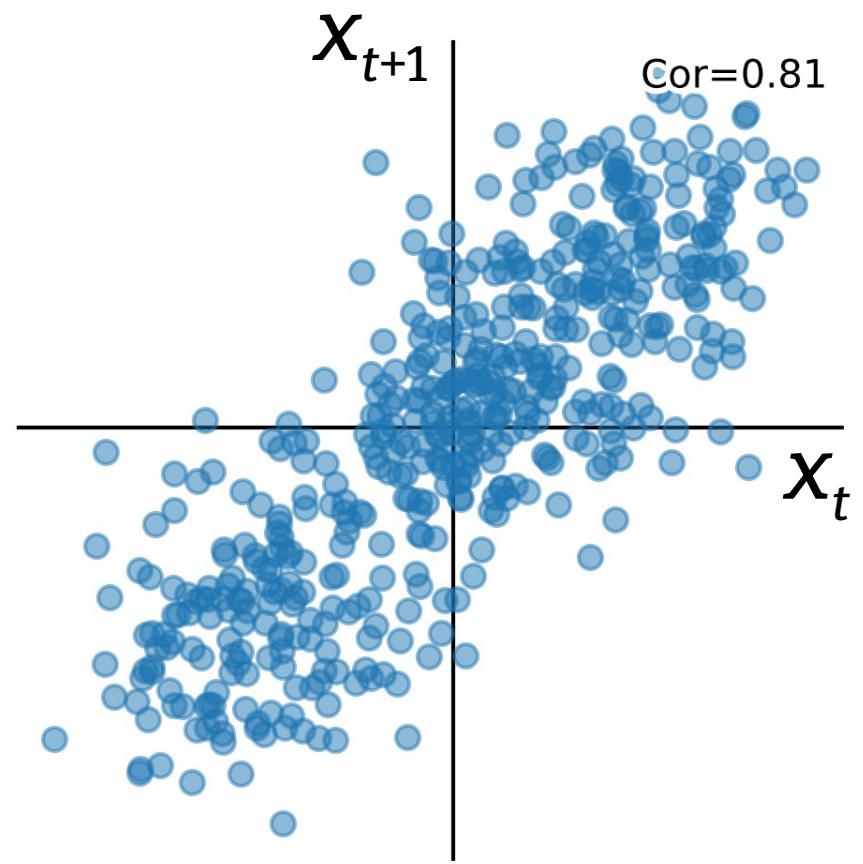
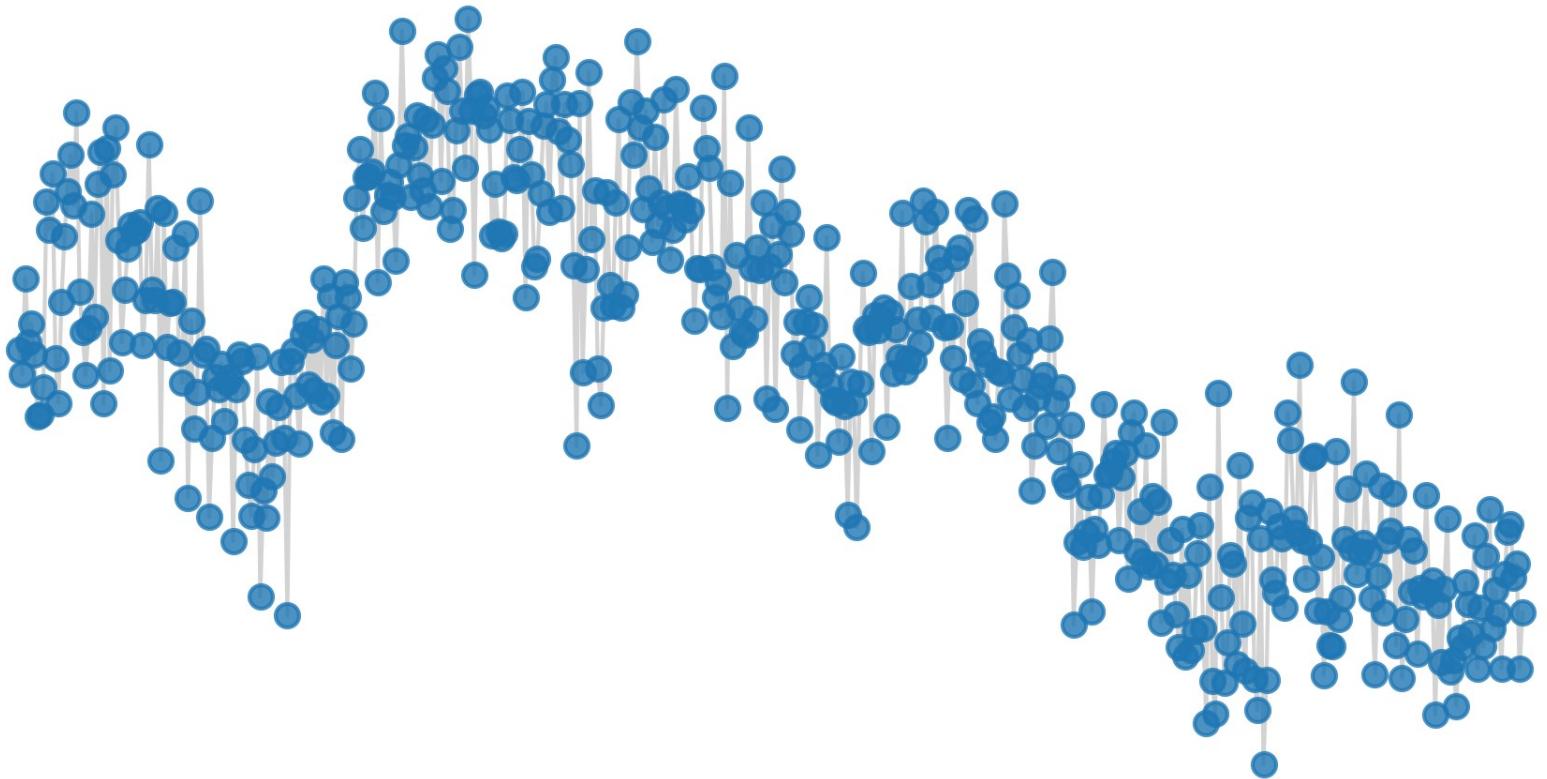
- Same as for univariate data

Numbers

- Same as for univariate data
- Auto-correlation: $\text{Cor}(x_t, x_{t+k})$

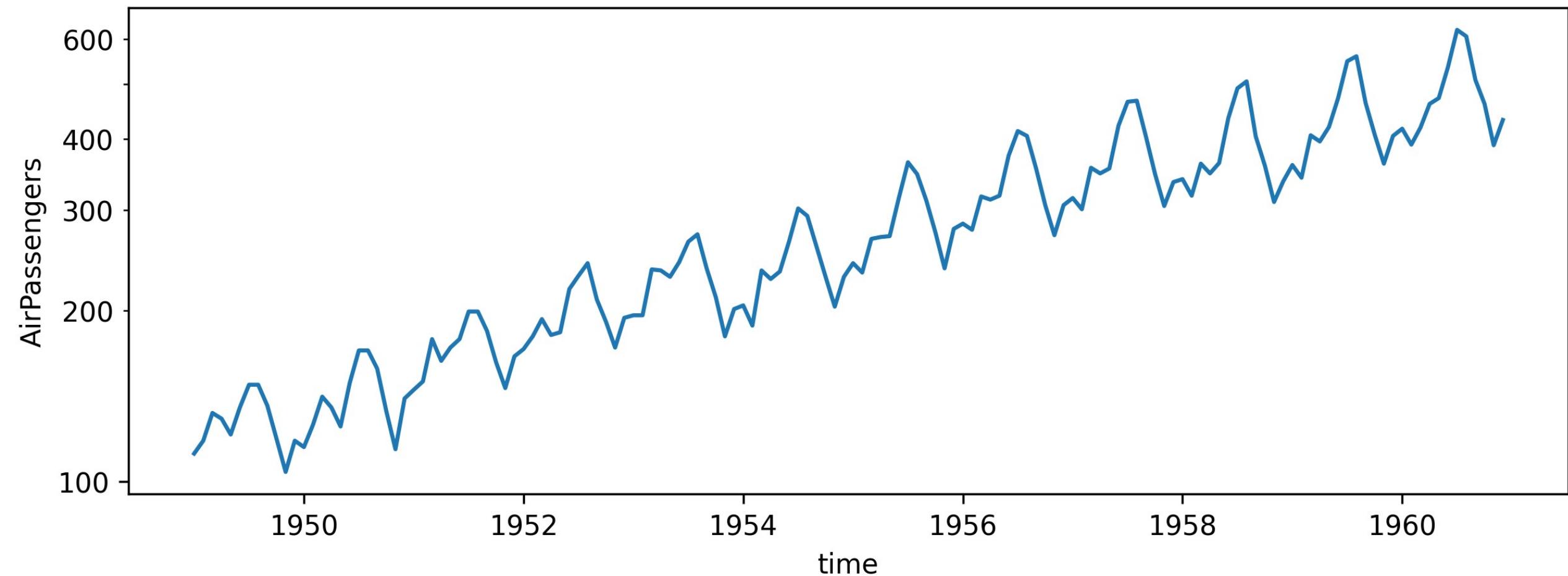
	date	previous	current
	1949-01-31	NaN	112.0
	1949-02-28	112.0	118.0
	1949-03-31	118.0	132.0
	1949-04-30	132.0	129.0
	1949-05-31	129.0	121.0
	1949-06-30	121.0	135.0
	1949-07-31	135.0	148.0
	1949-08-31	148.0	148.0
	1949-09-30	148.0	136.0
	1949-10-31	136.0	119.0
	1949-11-30	119.0	104.0
	1949-12-31	104.0	118.0

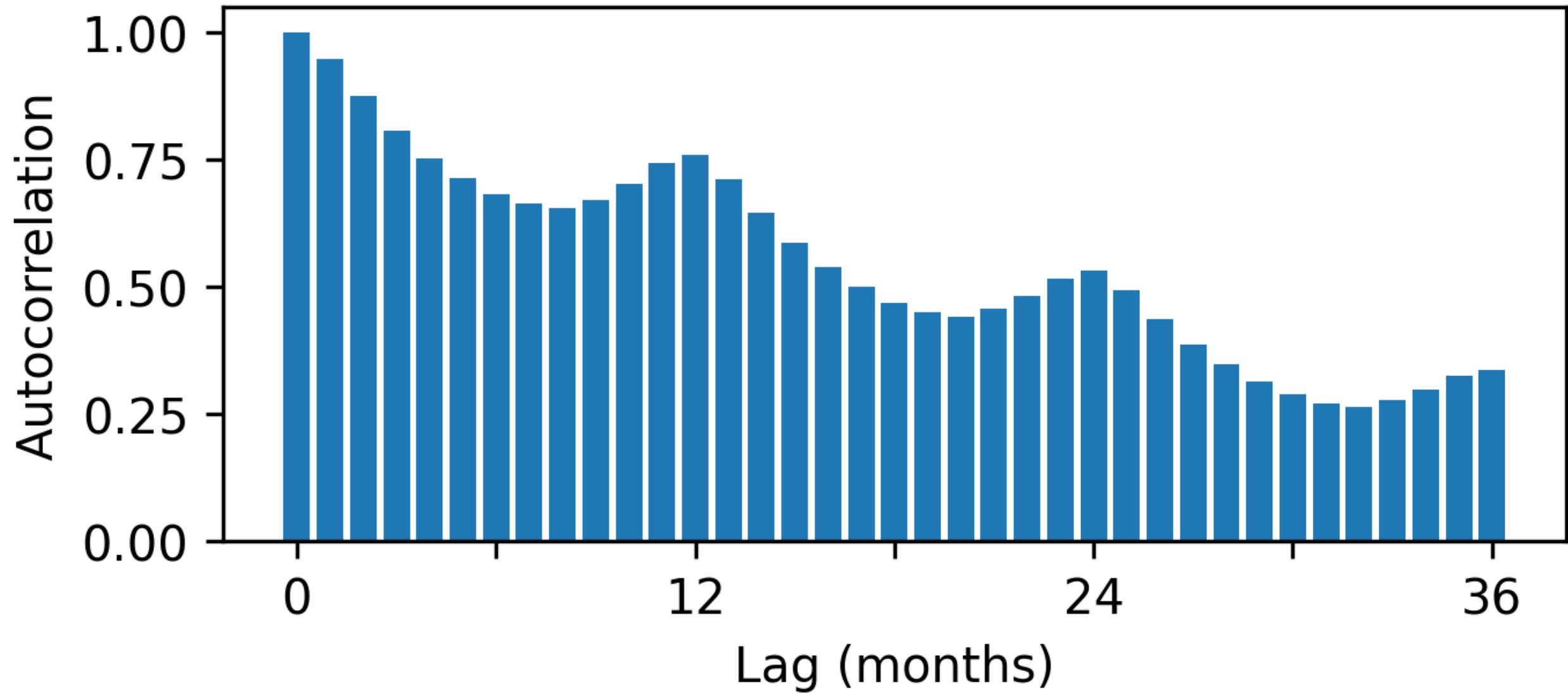


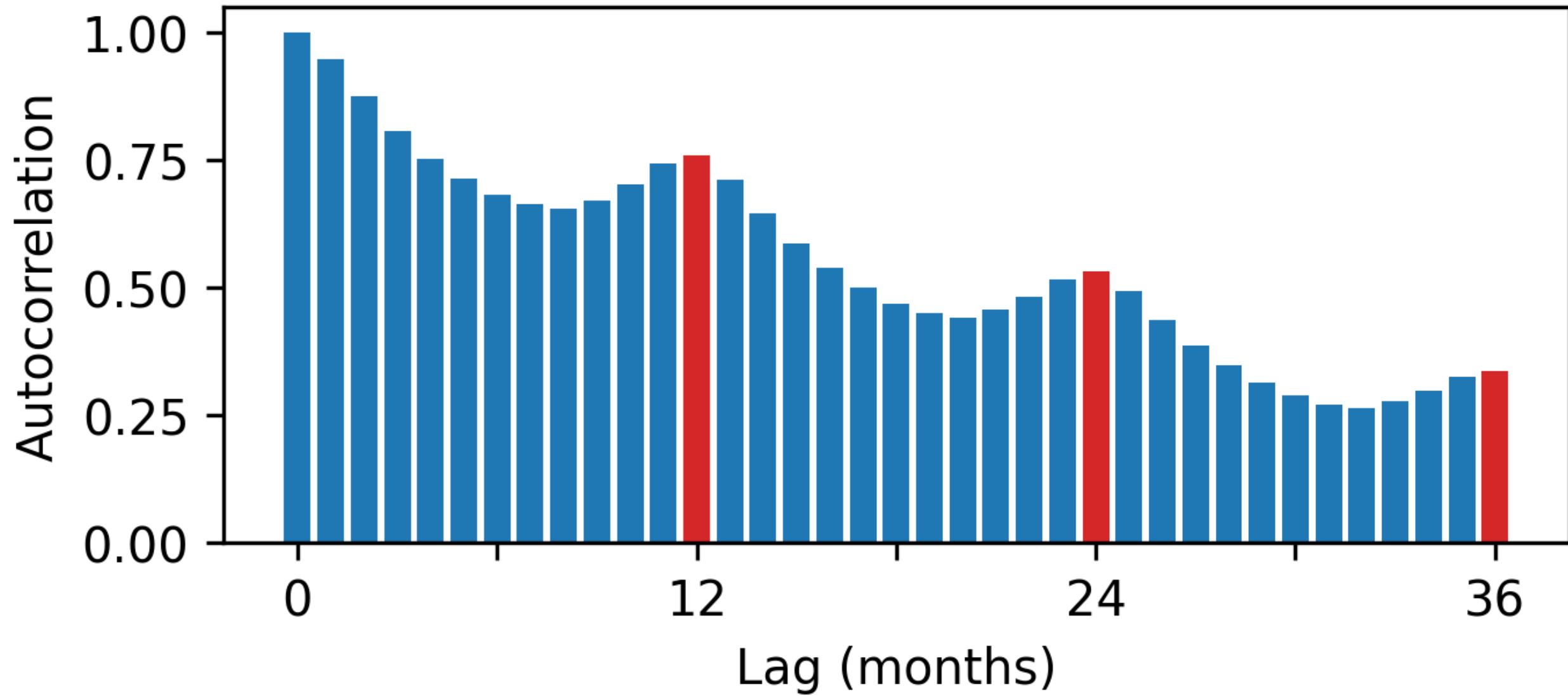


Plots

- Time series
- Auto-correlation



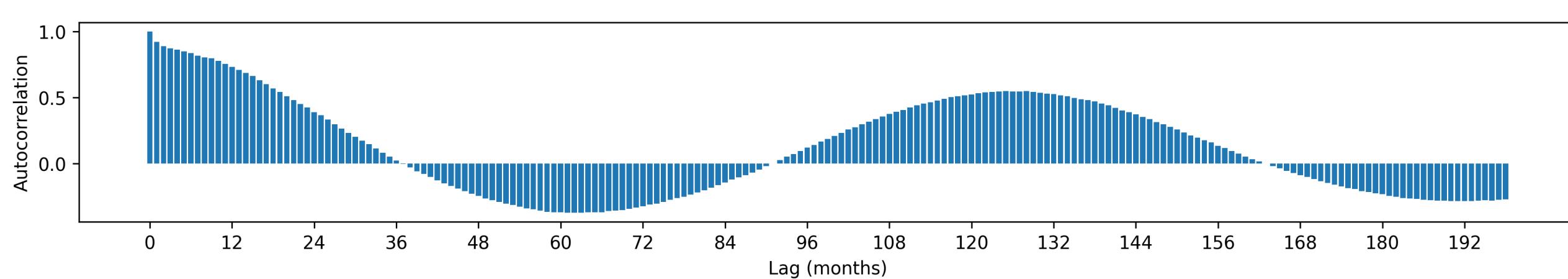
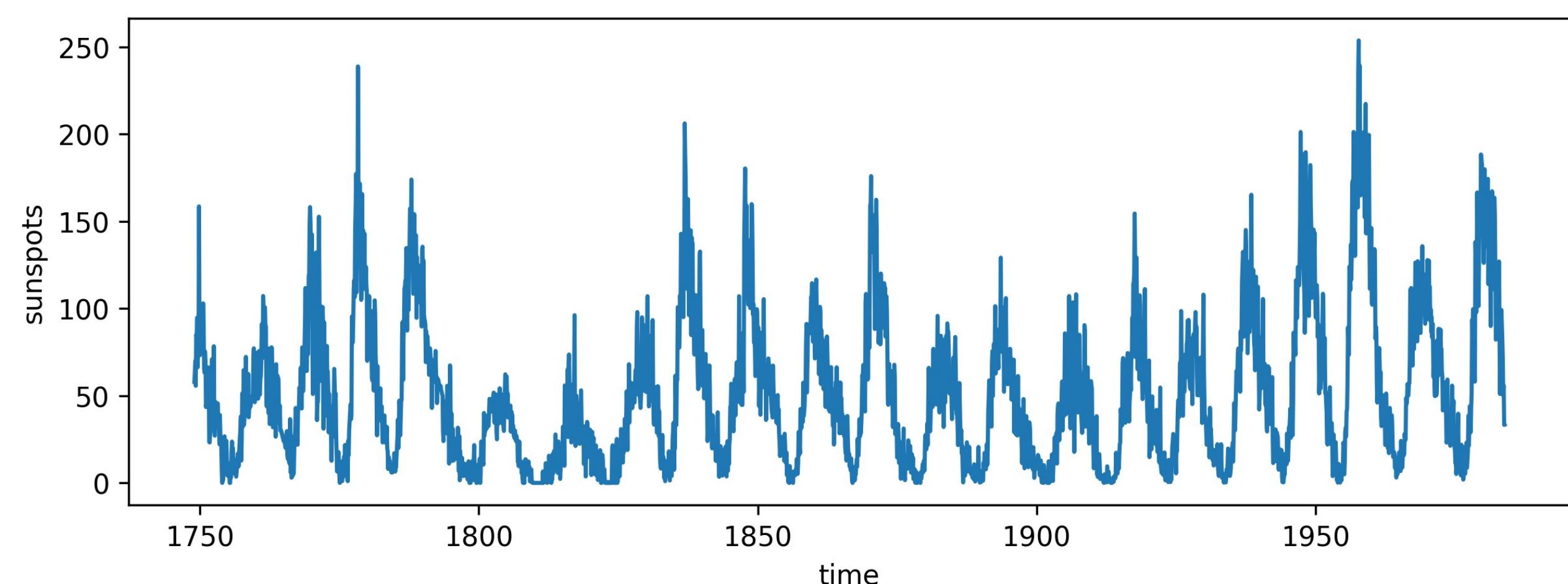




Exercise

Exercise

- Plot the following time series
- Are there data issues?
- Are there seasonal patterns?
With which period?



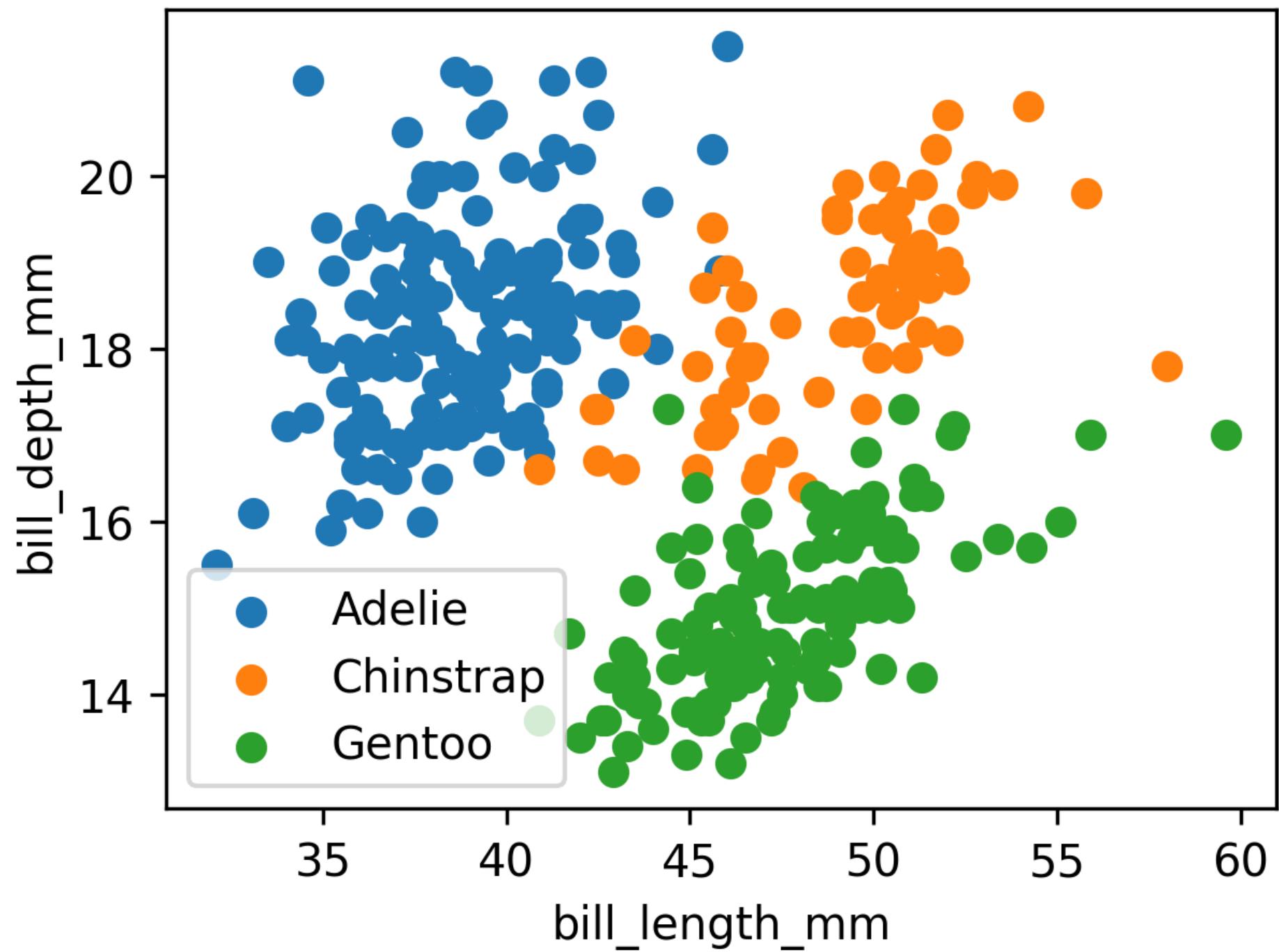
**Afternoon
Session**

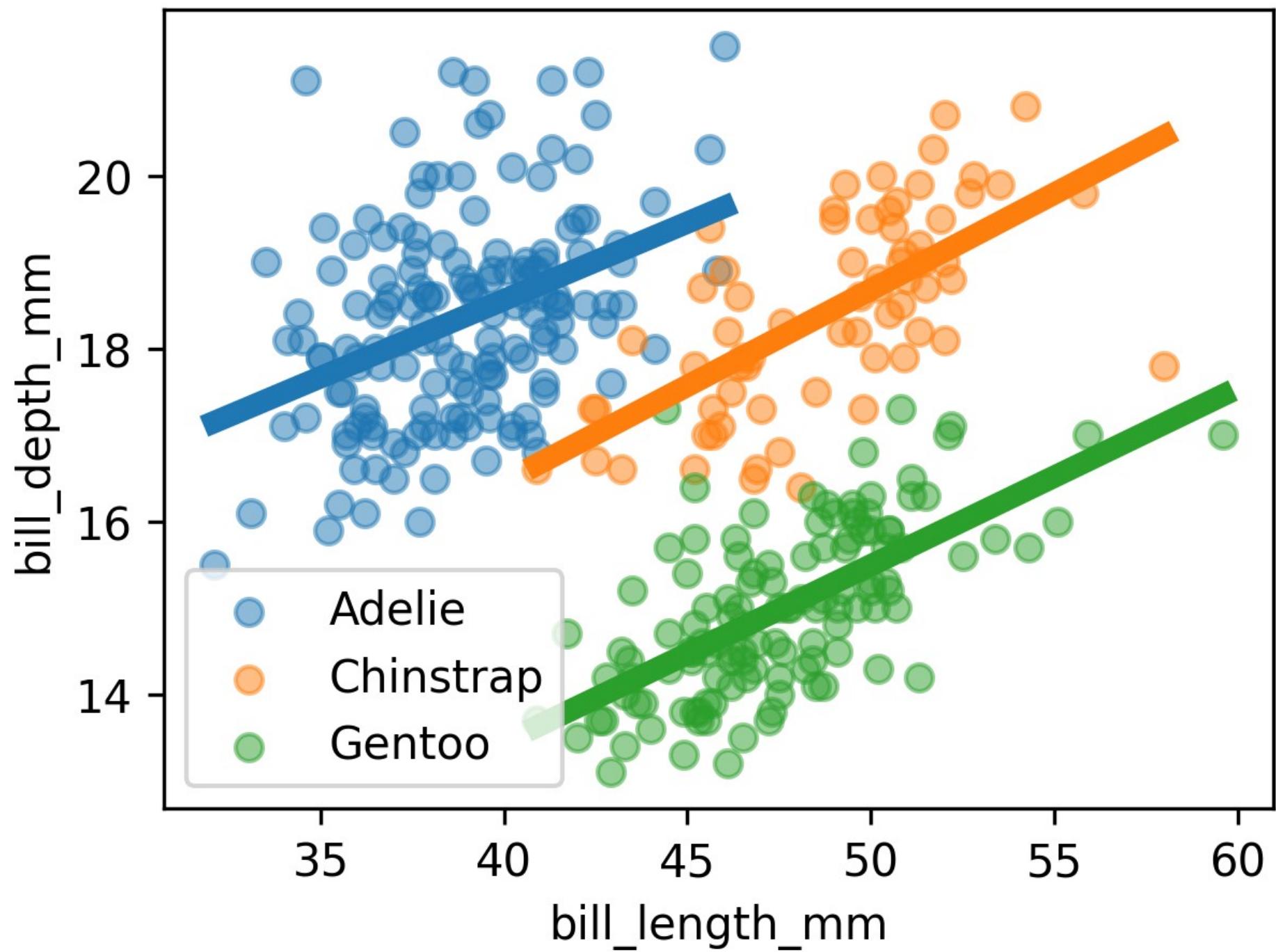
**Multivariate
with
qualitative data**

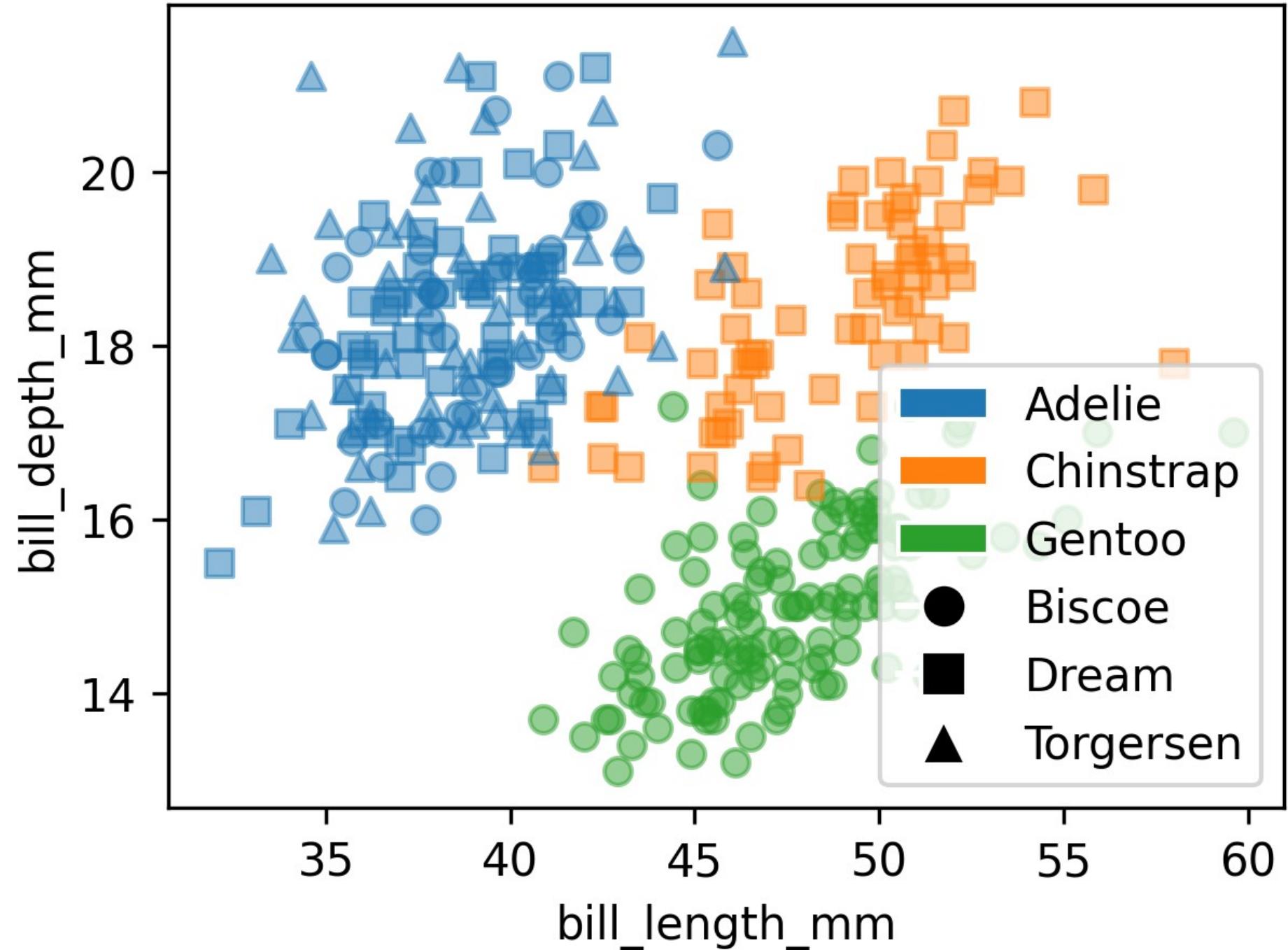
Exercise



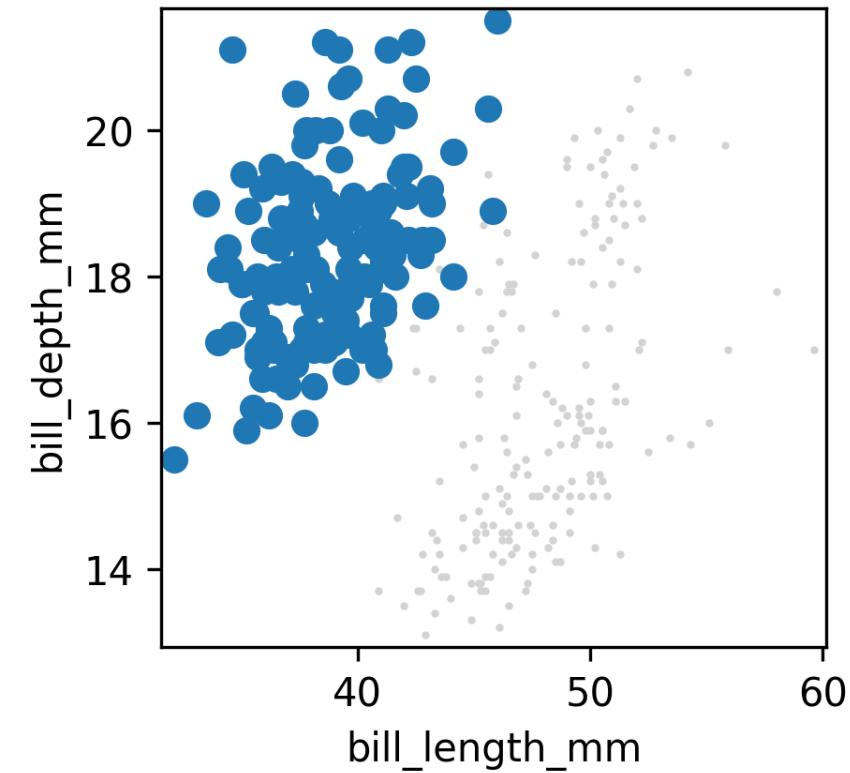
species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
Adelie	Torgersen	36.7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



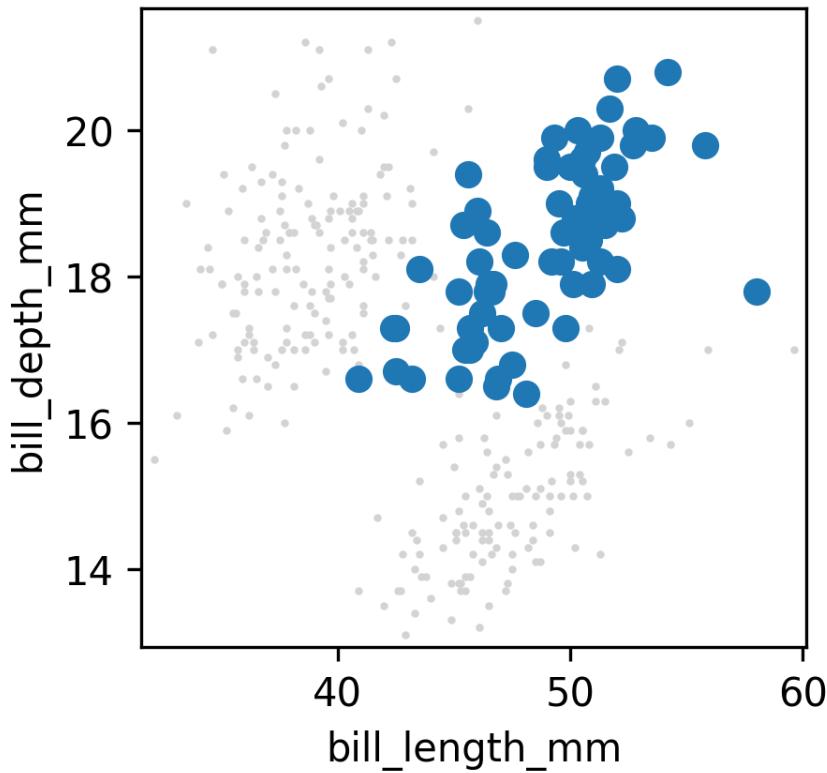




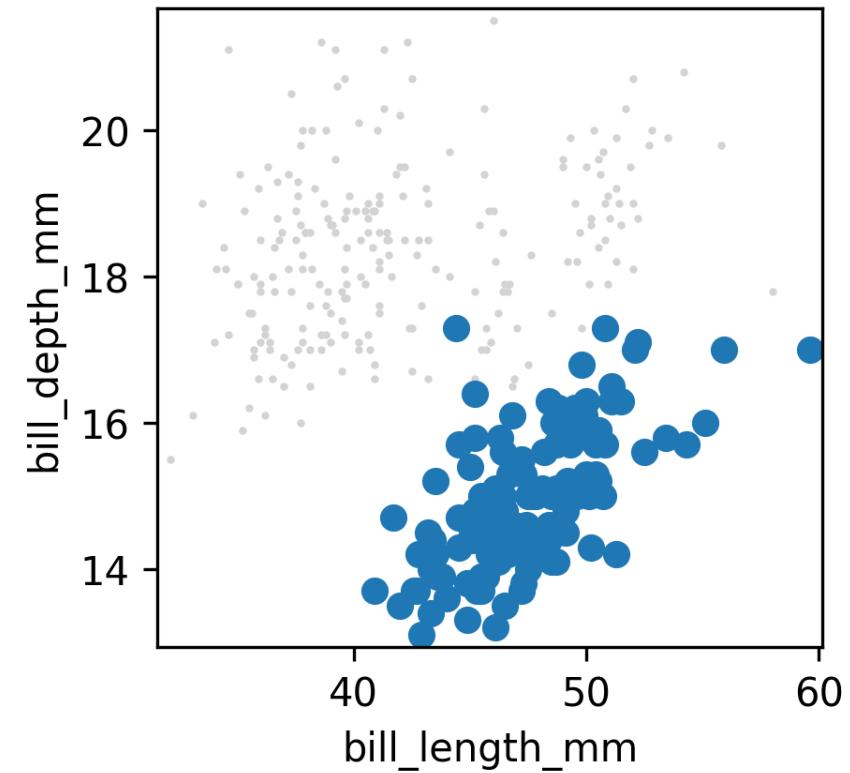
Adelie

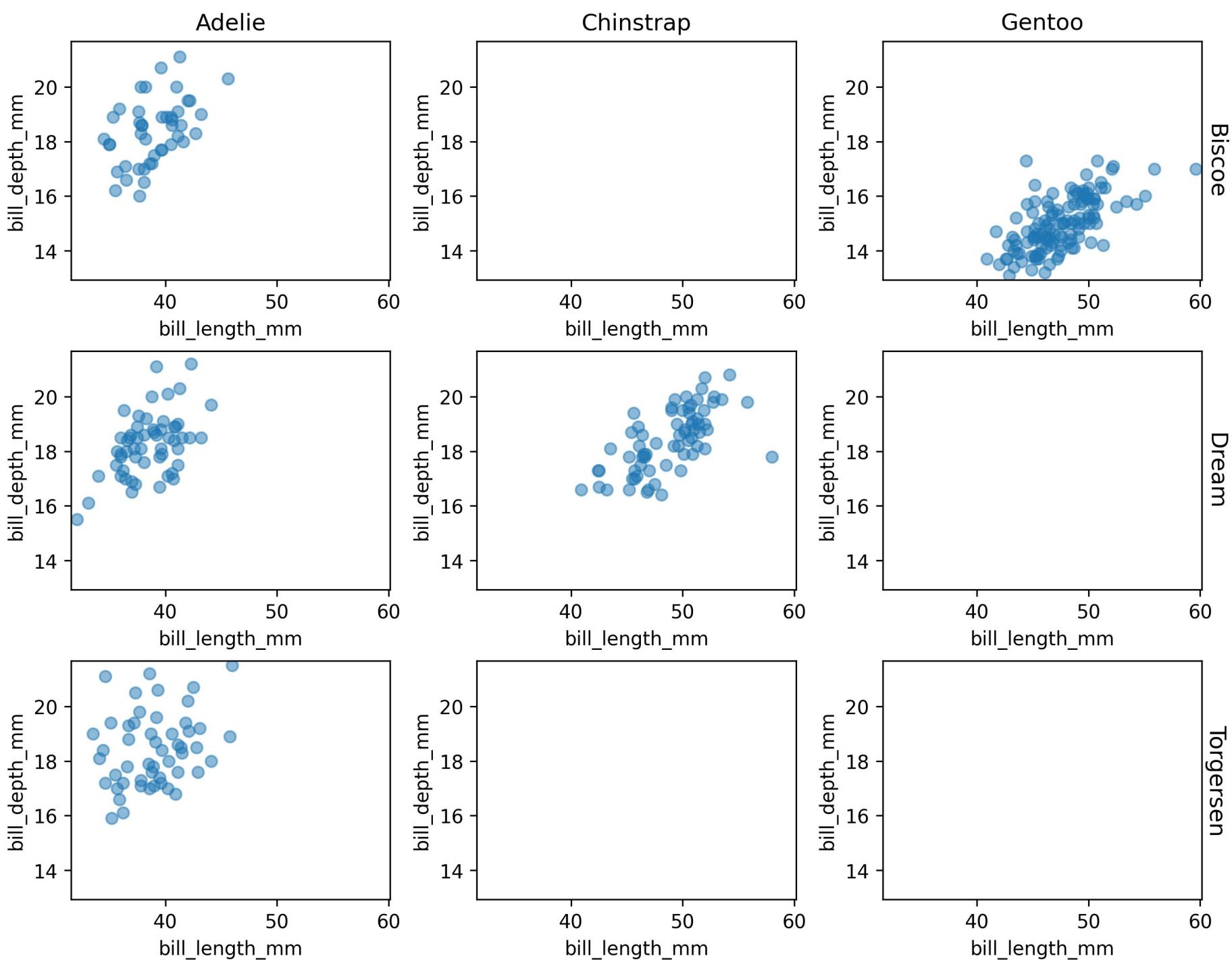


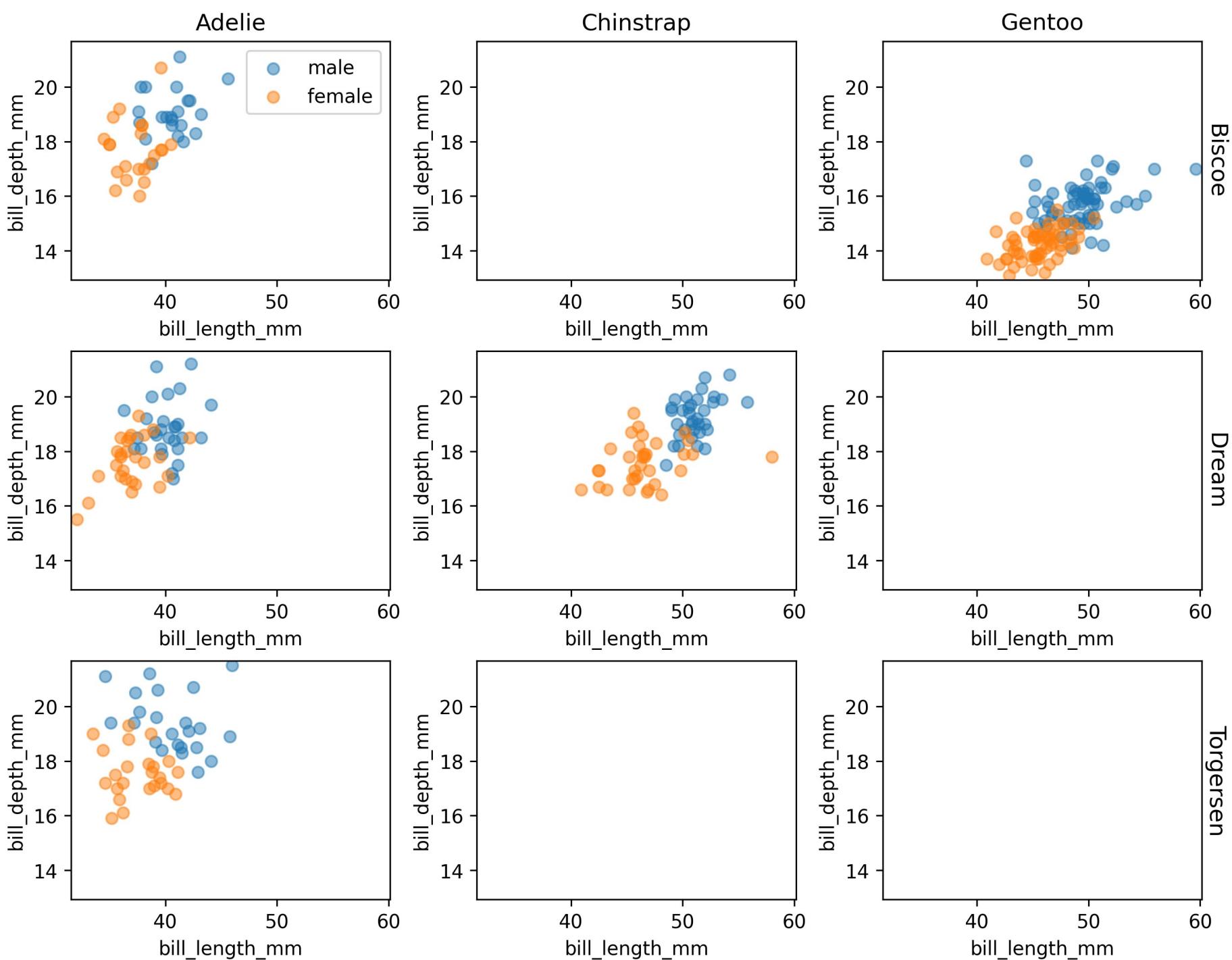
Chinstrap



Gentoo

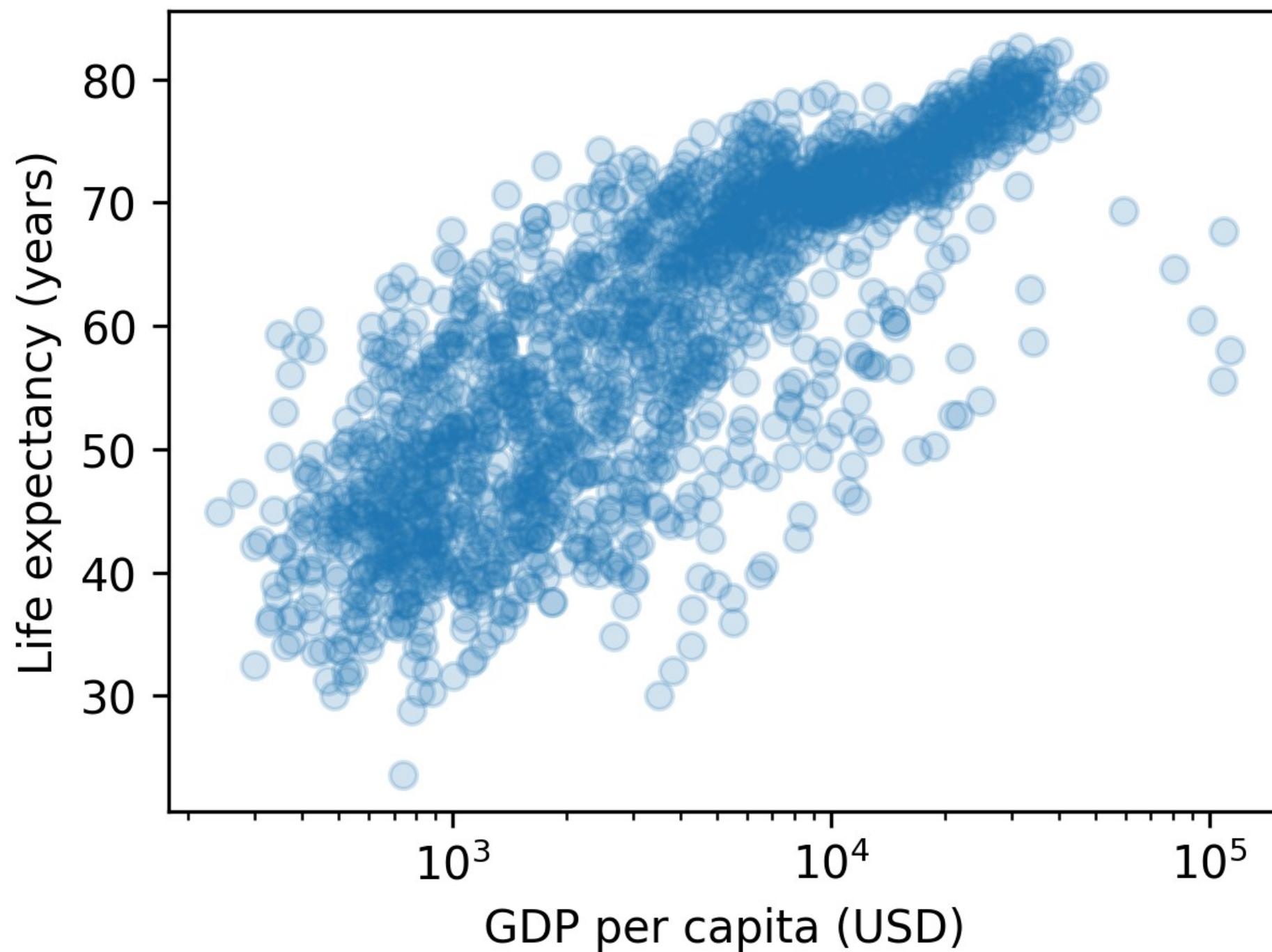


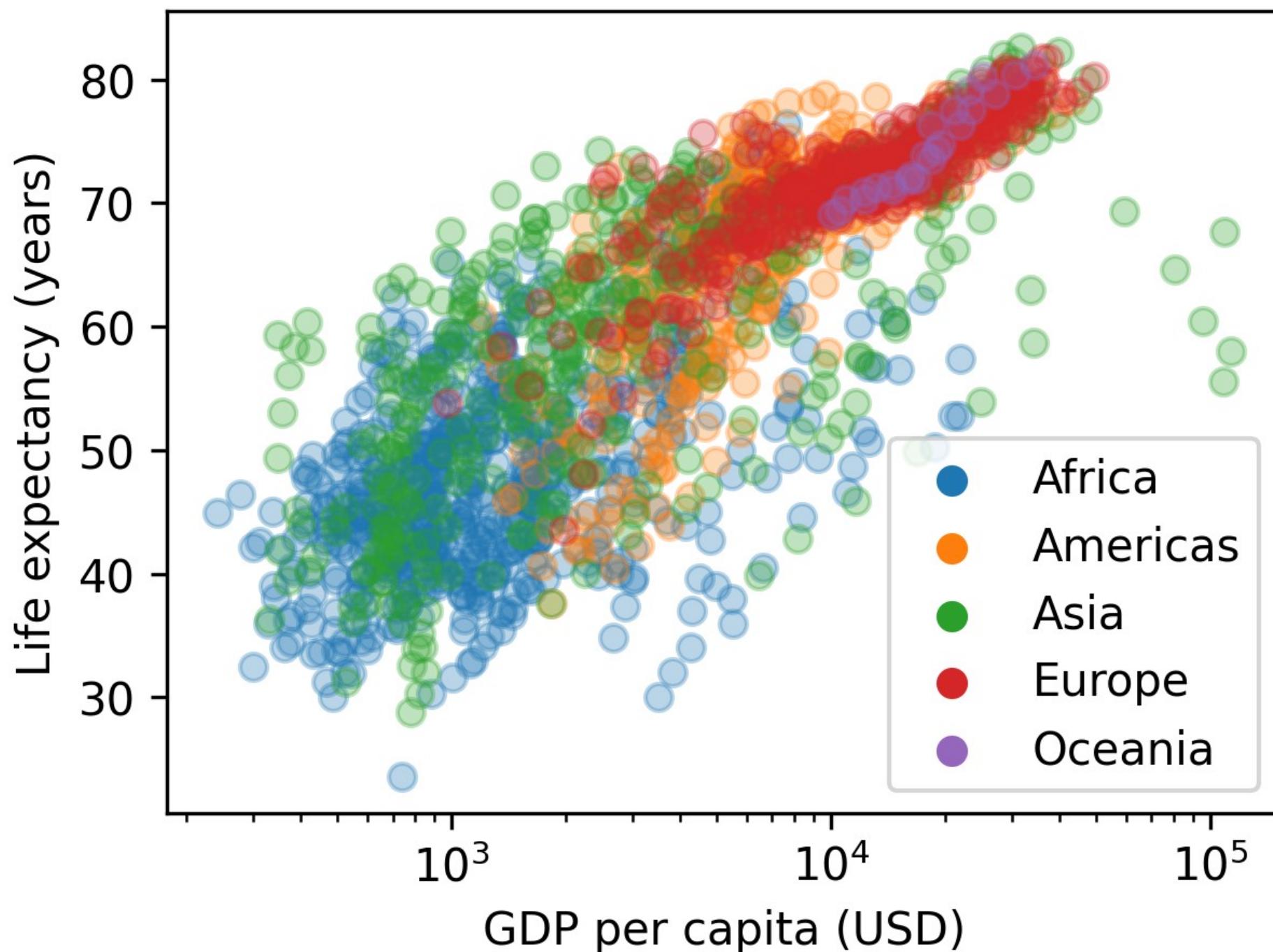


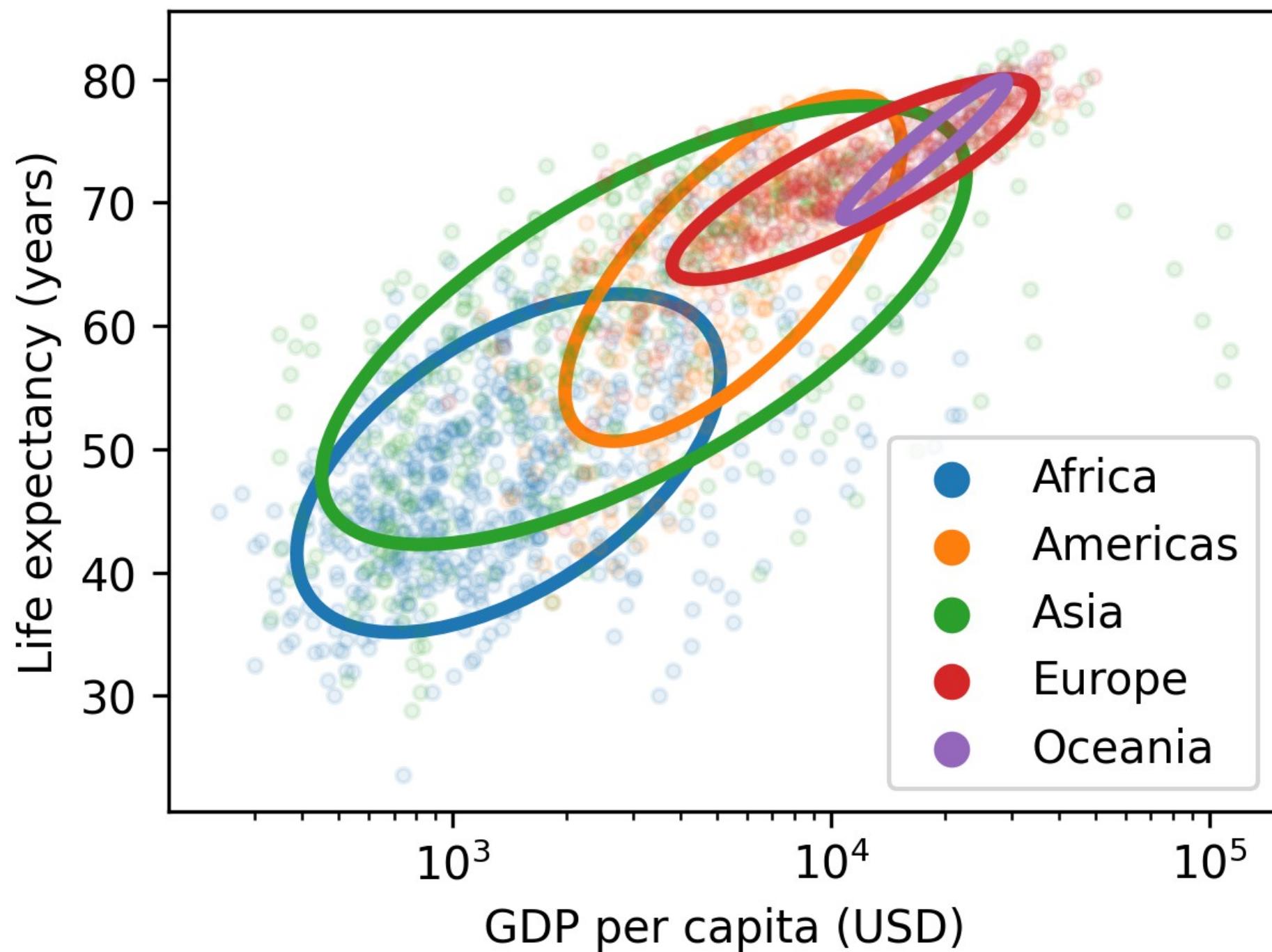


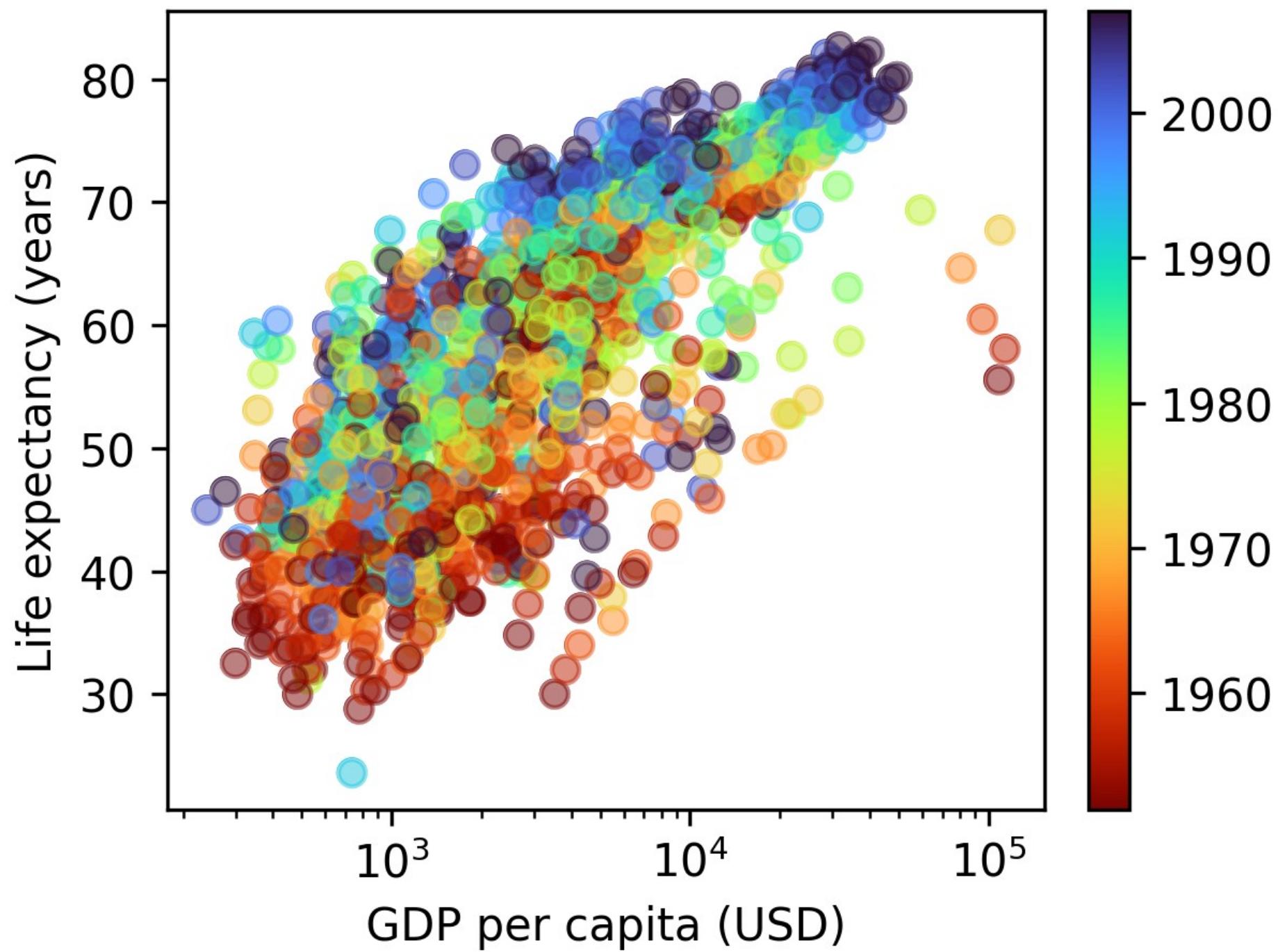
Exercise

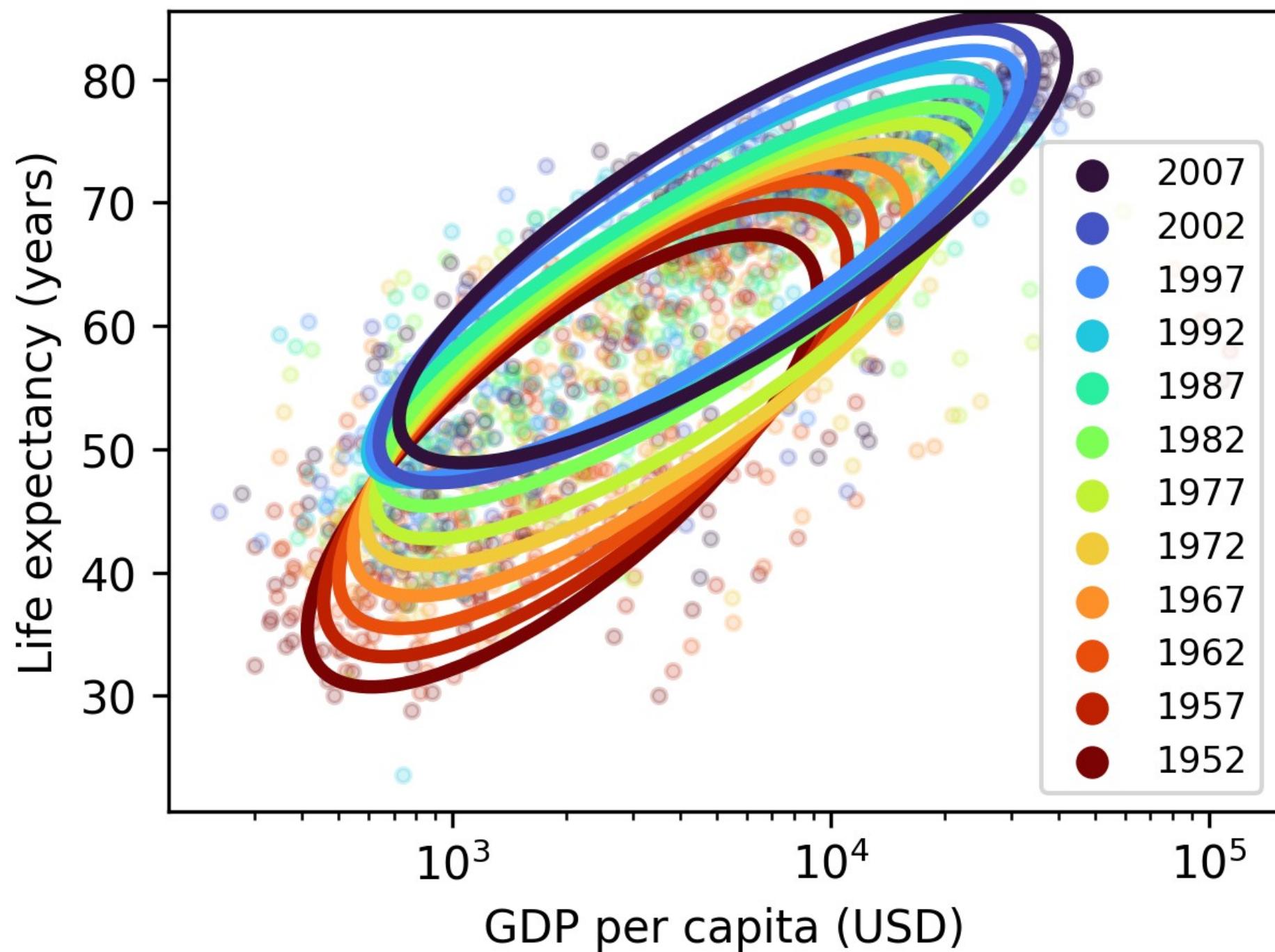
country	continent	year	lifeExp	pop	gdpPerCap
Afghanistan	Asia	1952	28.801	8425333	779.445314
Afghanistan	Asia	1957	30.332	9240934	820.853030
Afghanistan	Asia	1962	31.997	10267083	853.100710
Afghanistan	Asia	1967	34.020	11537966	836.197138
Afghanistan	Asia	1972	36.088	13079460	739.981106



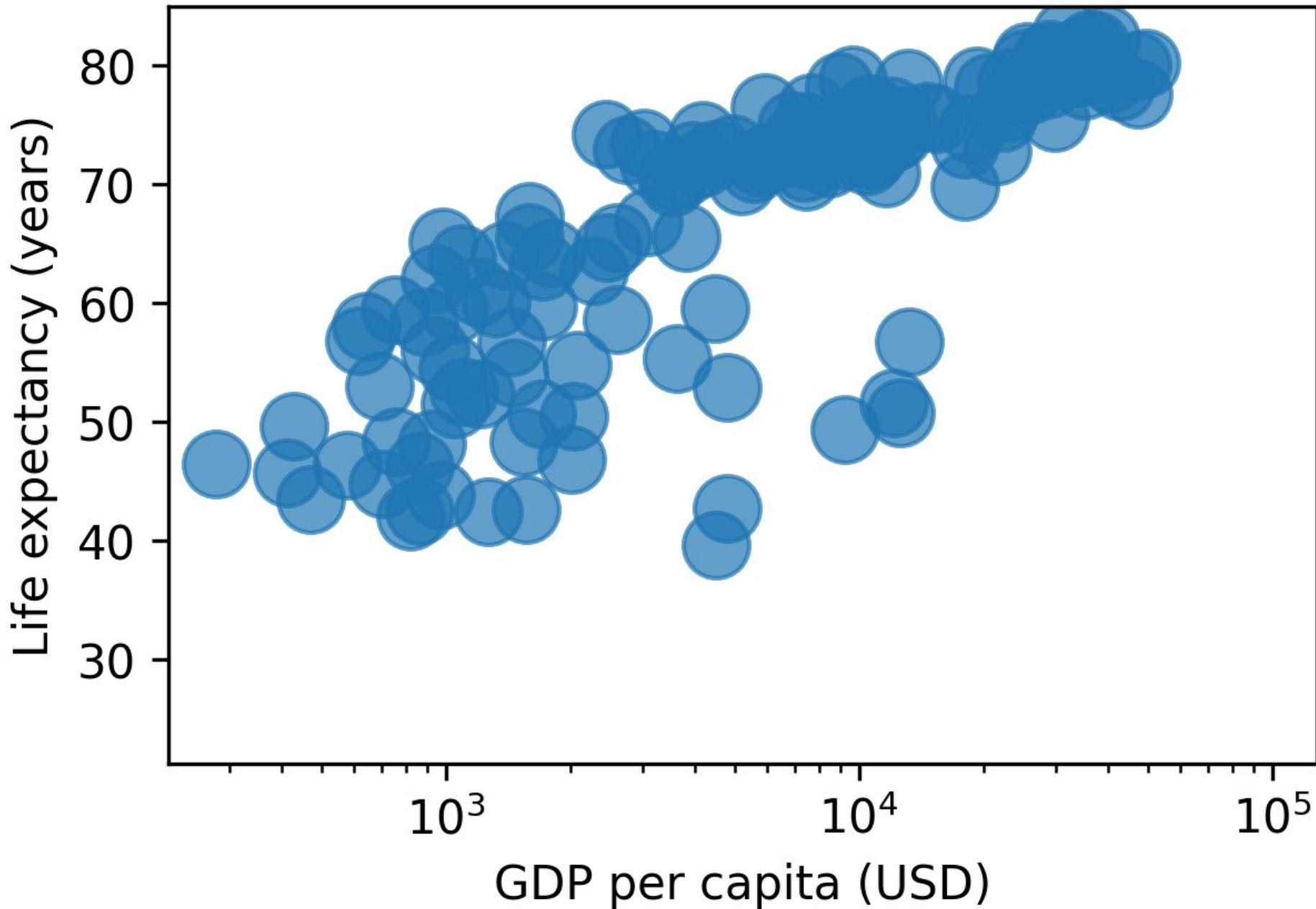




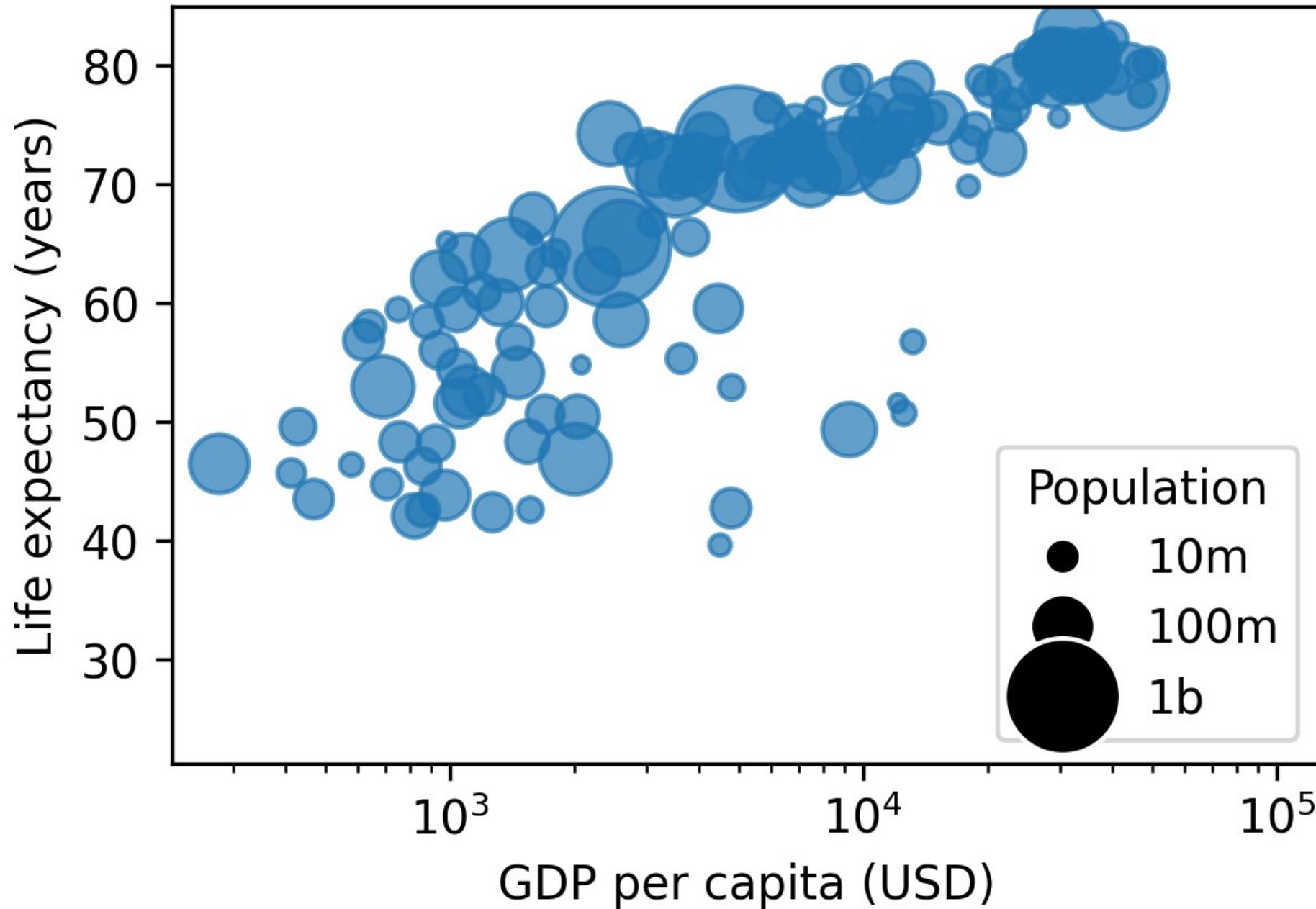




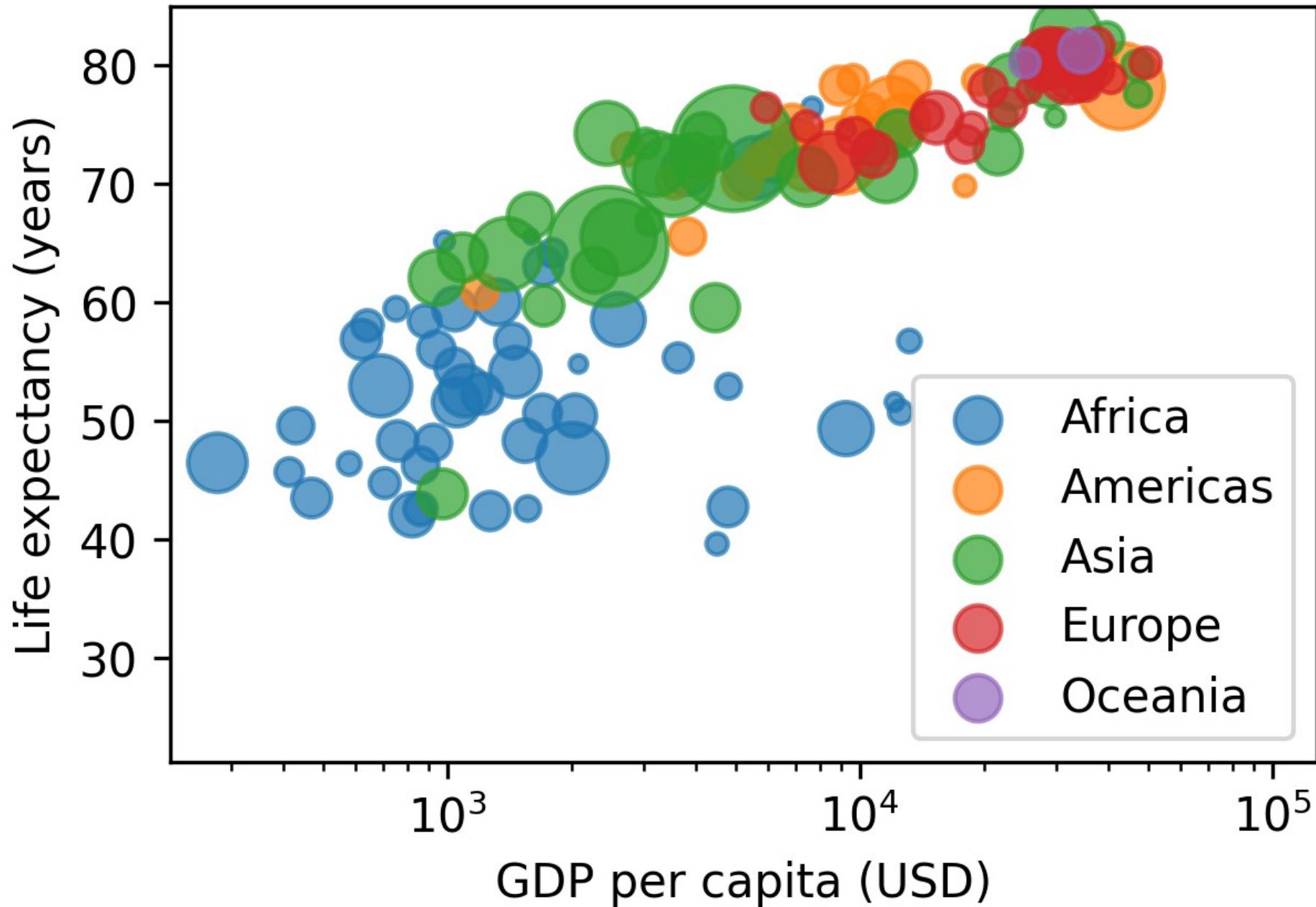
2007

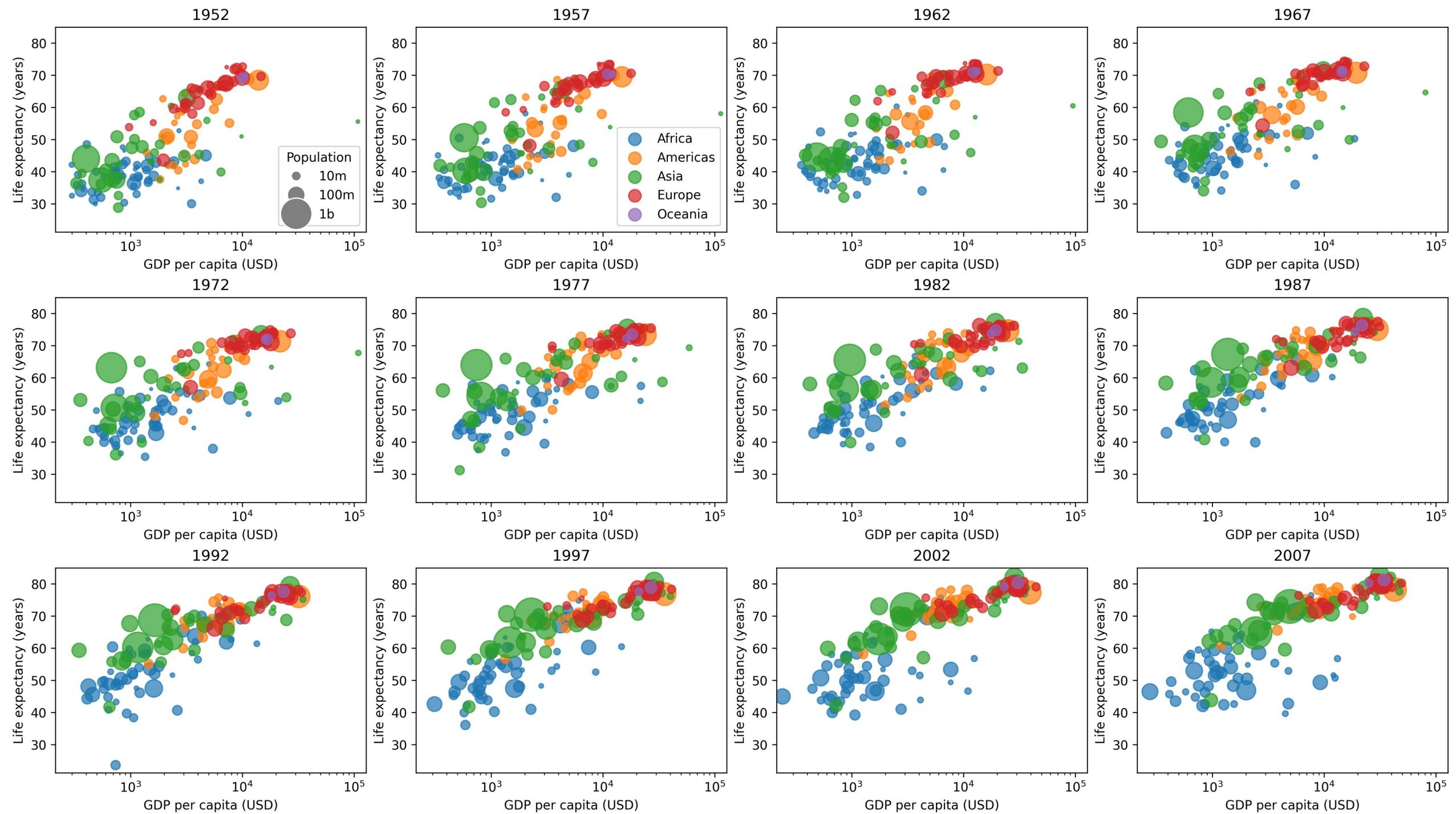


2007



2007





Automated EDA

Automated EDA

- There are libraries to do most of this automatically.

```
from ydata_profiling import ProfileReport  
ProfileReport(d)
```

Exercise

- Examine one of the datasets above with `ydata_profiling`.
- Is there anything we have overlooked?

Summary

Potential Problems

- Data types
- Missing values
- Magic numbers
- Zero inflation
- Outliers
- Fat tails
- Skewness
- Multimodality
- Bivariate outliers
- Non-linearity
- Heteroskedasticity
- Different units
- Long tail
- Non-stationarity

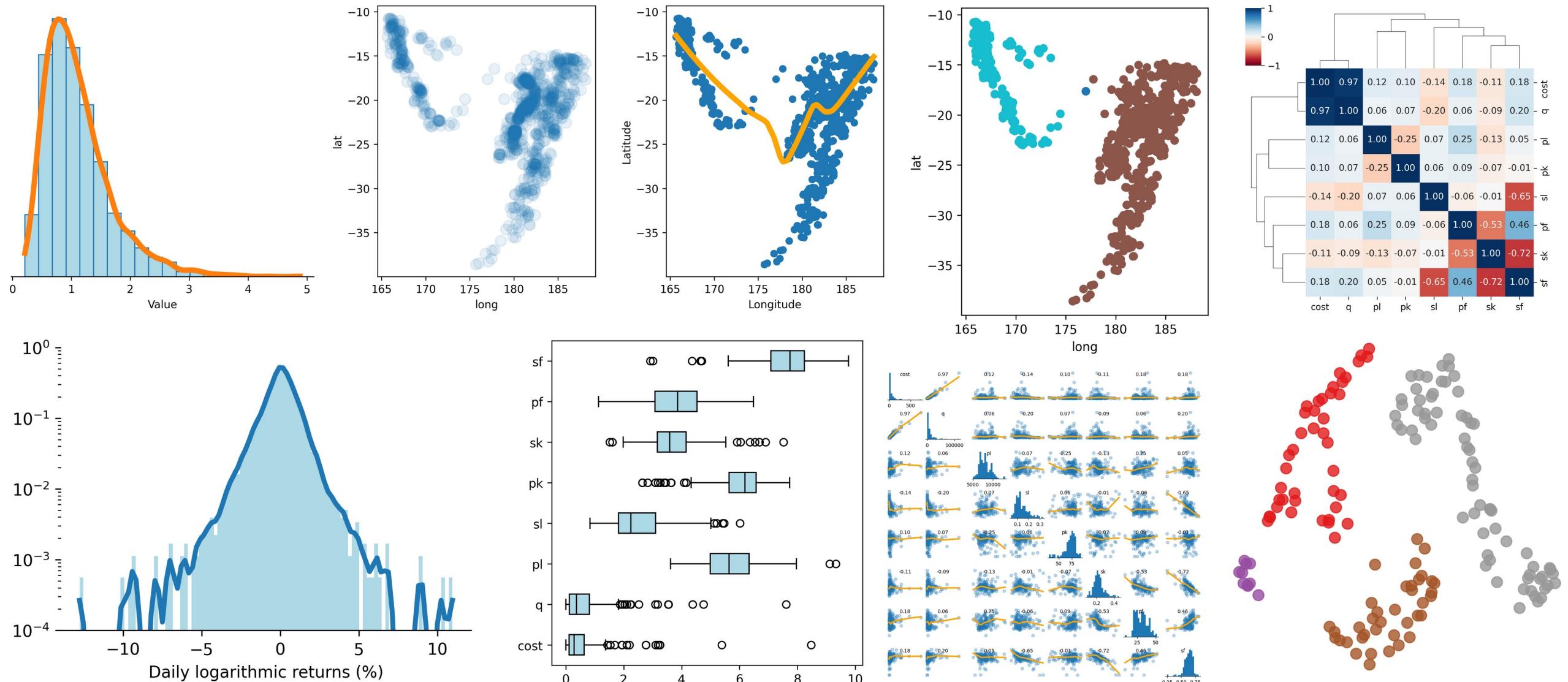
Numbers

- Mean, median, mode
- Standard deviation
- Quantiles, IQR
- Skewness
- Kurtosis
- Missing values
- Correlation

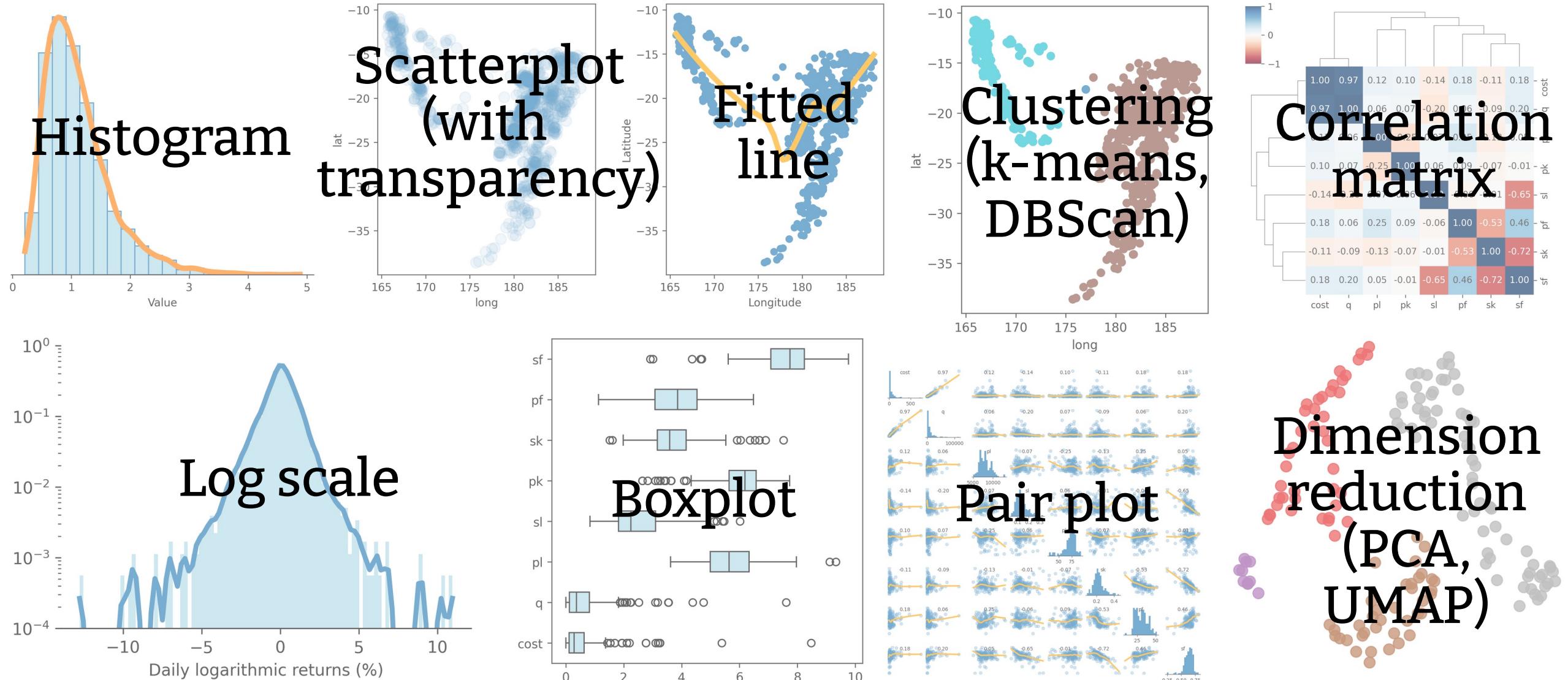
Plots

- Histogram, density (+log scale)
- Scatterplot (+curve fitting, clustering)
- Clustering: k-means, DBScan
- Pairplot
- Boxplots
- Correlation matrix (+hierarchical clustering)
- Dimension reduction: PCA, UMAP (+clustering)

Plots



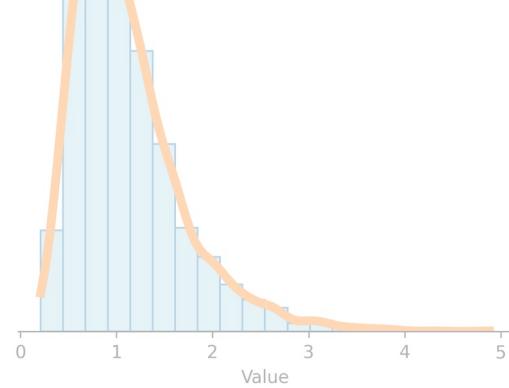
Plots



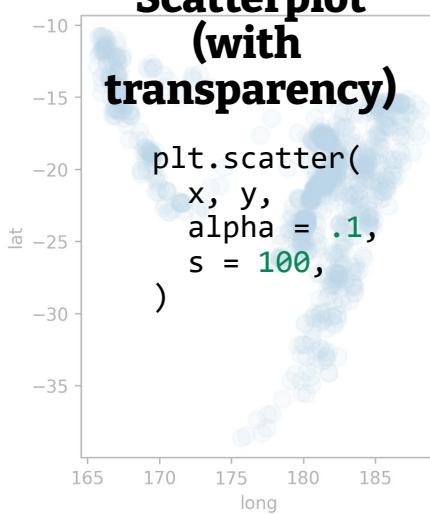
Plots

Histogram

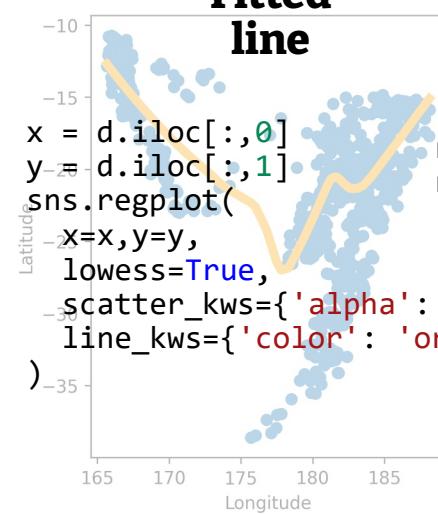
```
x.plot.hist(density=True)  
x.plot.density(linewidth=5)
```



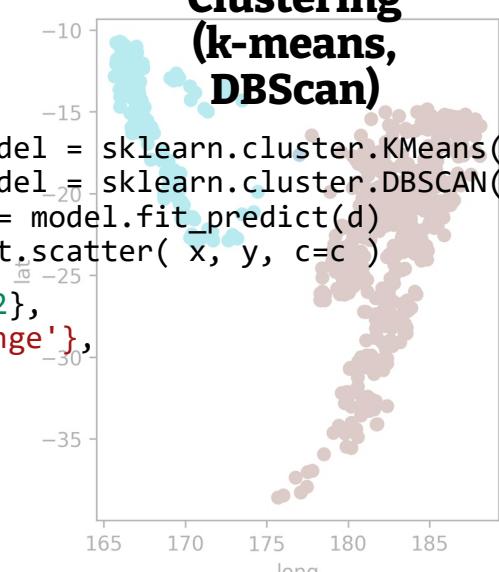
Scatterplot (with transparency)



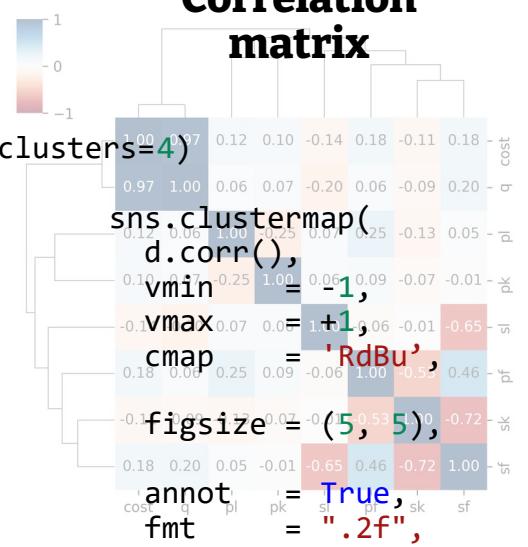
Fitted line



Clustering (k-means, DBScan)

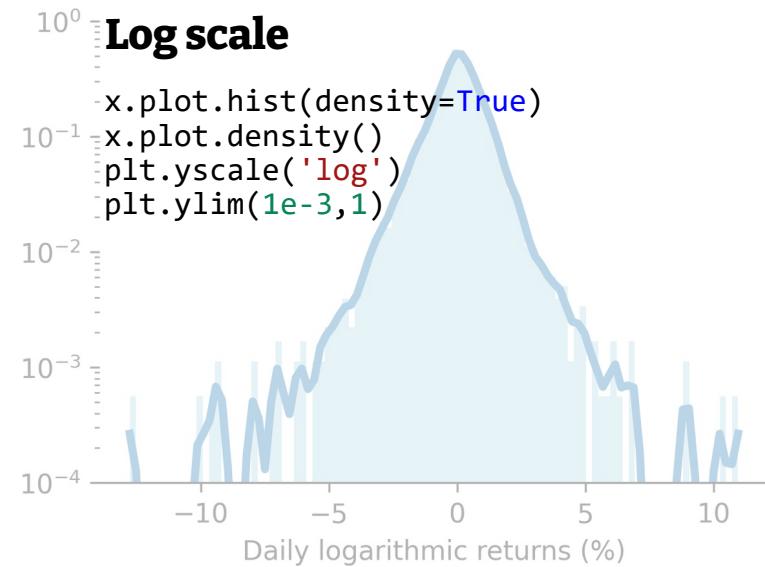


Correlation matrix

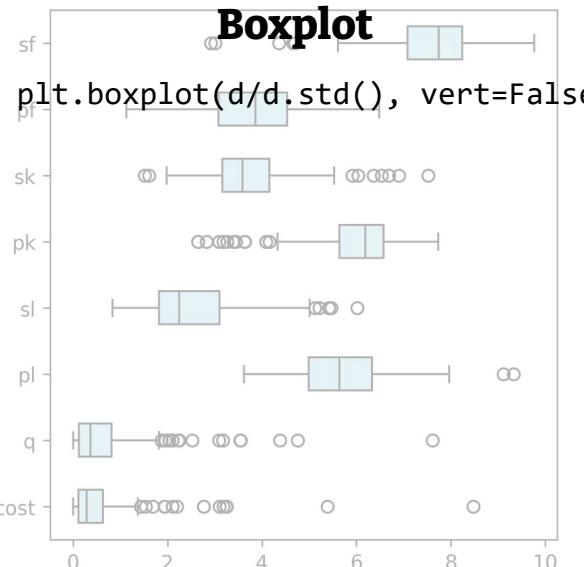


Log scale

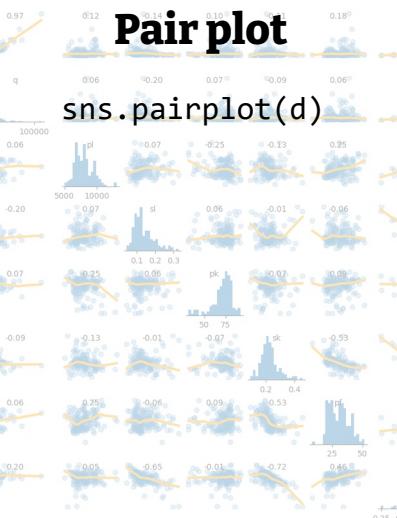
```
x.plot.hist(density=True)  
x.plot.density()  
plt.yscale('log')  
plt.ylim(1e-4,1)
```



Boxplot

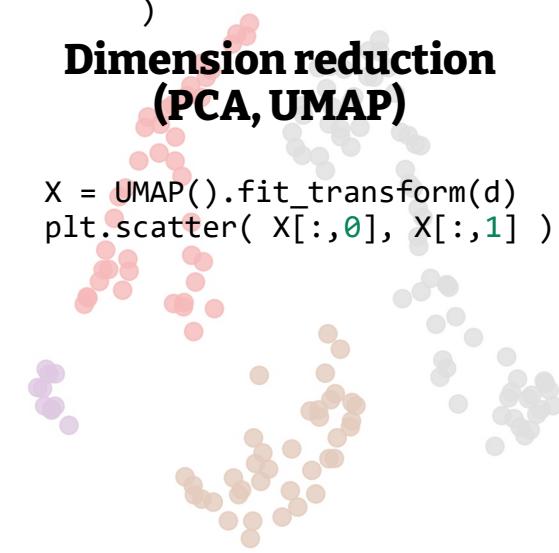


Pair plot



Dimension reduction (PCA, UMAP)

```
X = UMAP().fit_transform(d)  
plt.scatter( X[:,0], X[:,1] )
```



References

References

Learn X in Y minutes, where X=Python

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

Statistical plots

V. Zonnekynd (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/>

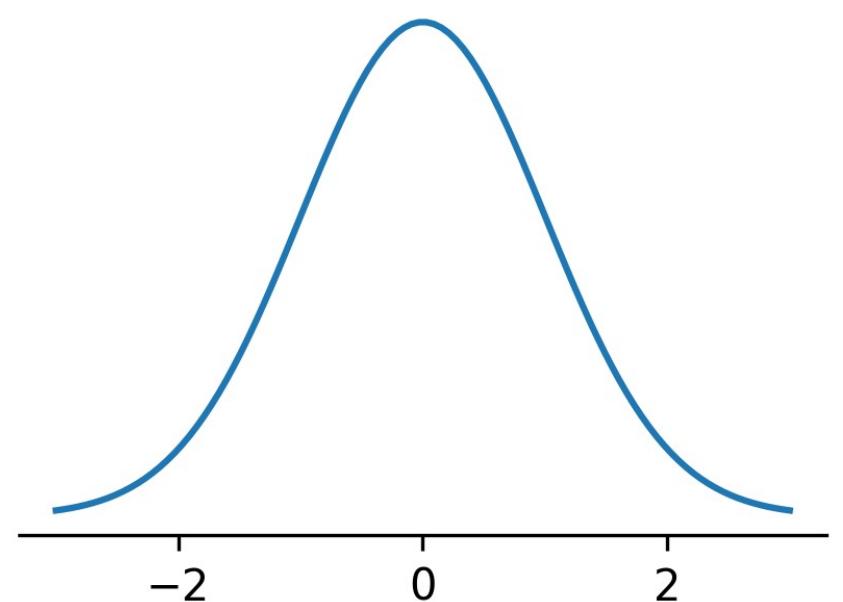
https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

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

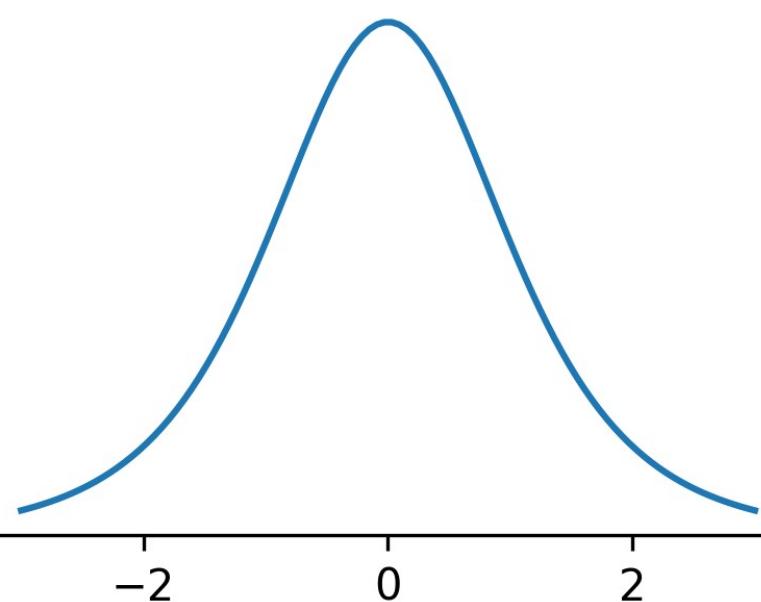
<https://www.datacamp.com/cheat-sheet/numpy-cheat-sheet-data-analysis-in-python>

Extra slides

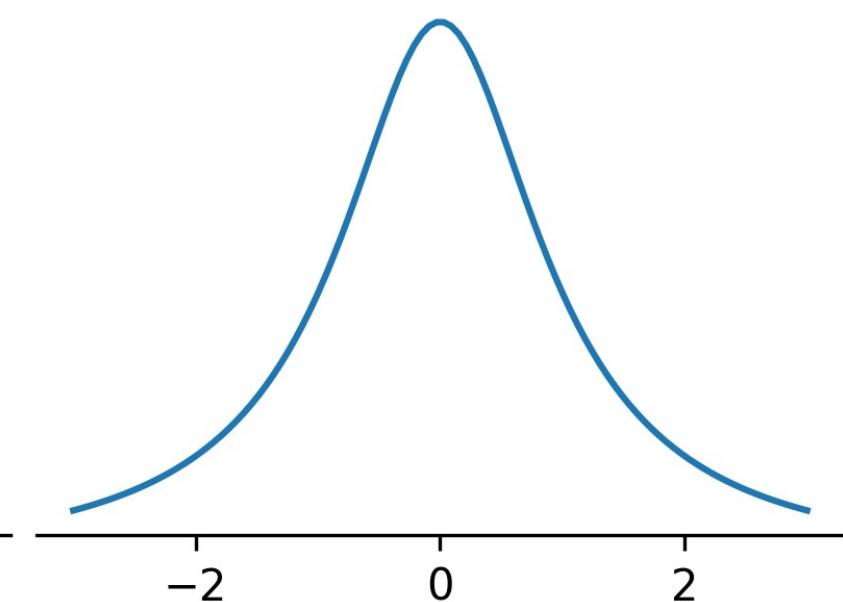
Gaussian



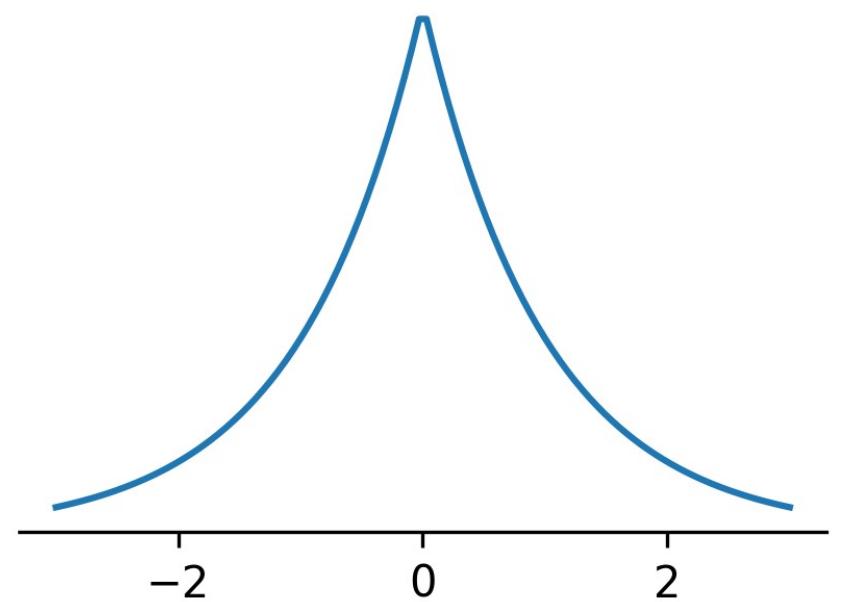
Student



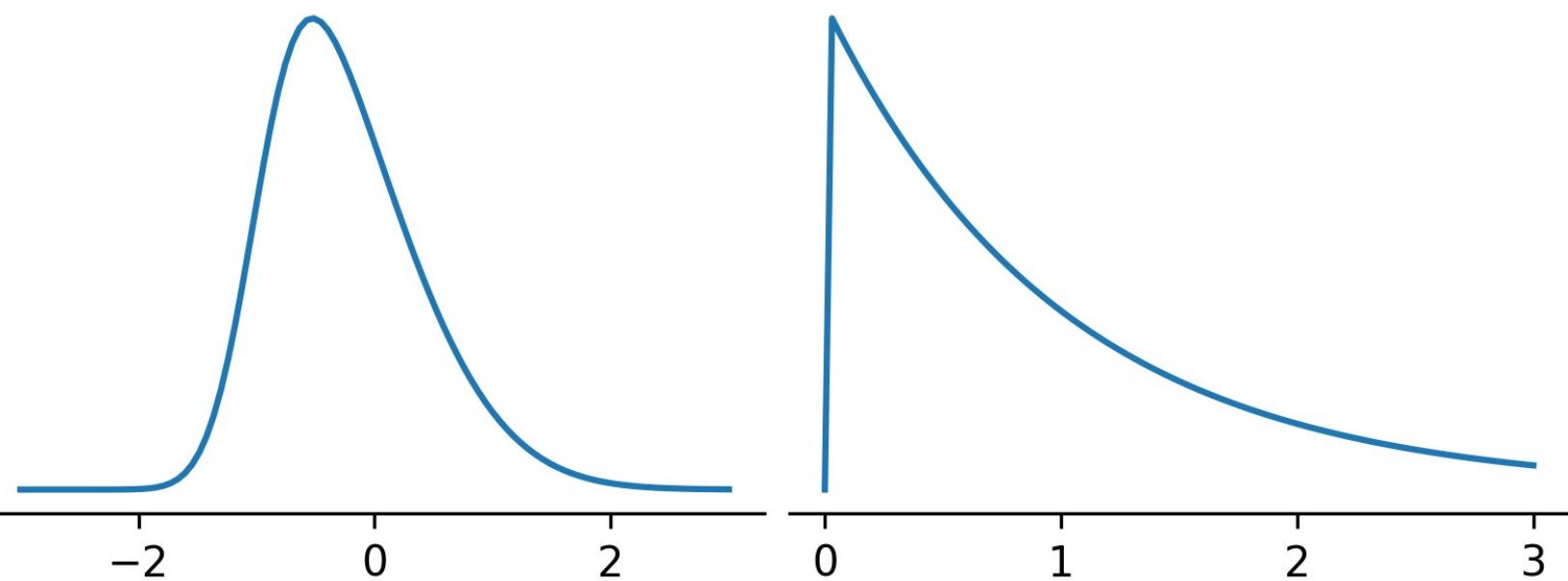
Cauchy



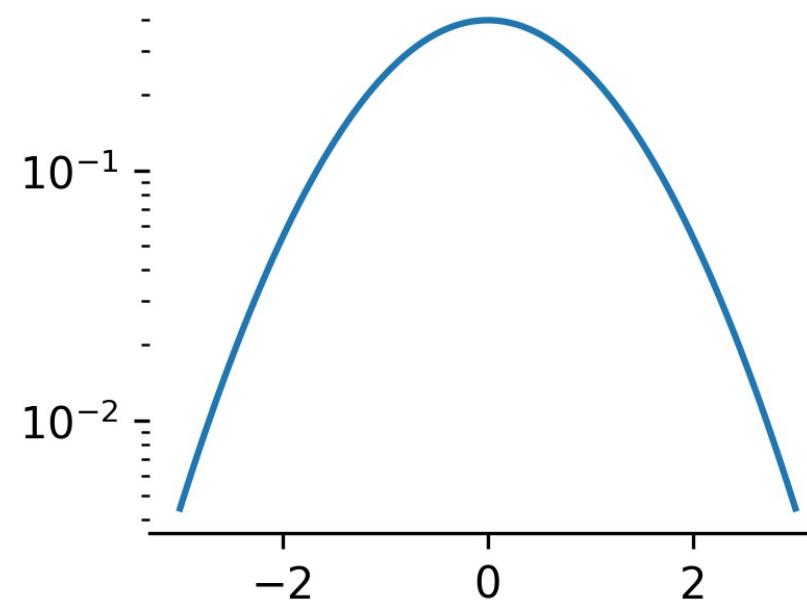
Laplace



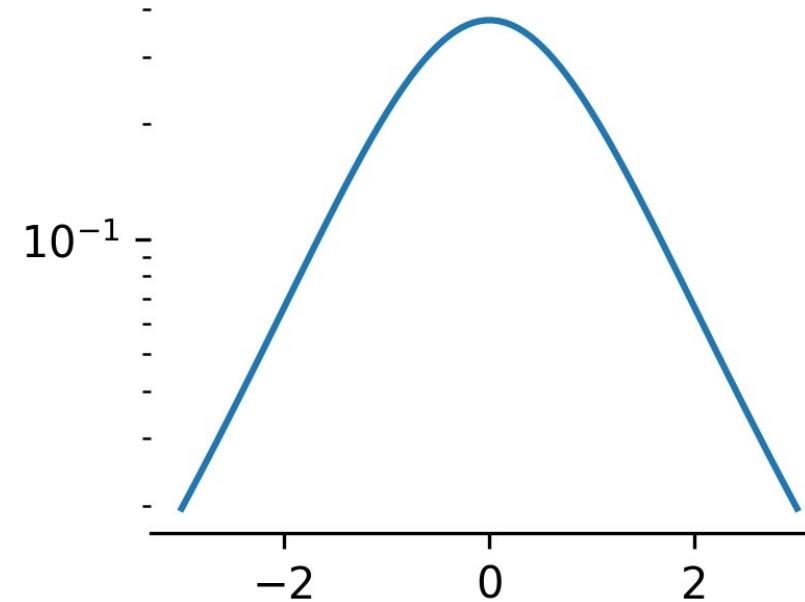
Exponential



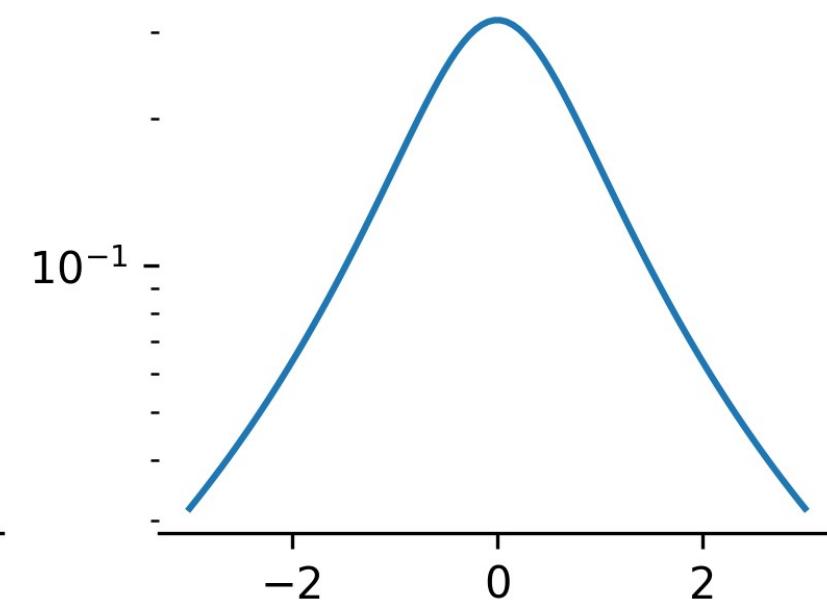
Gaussian



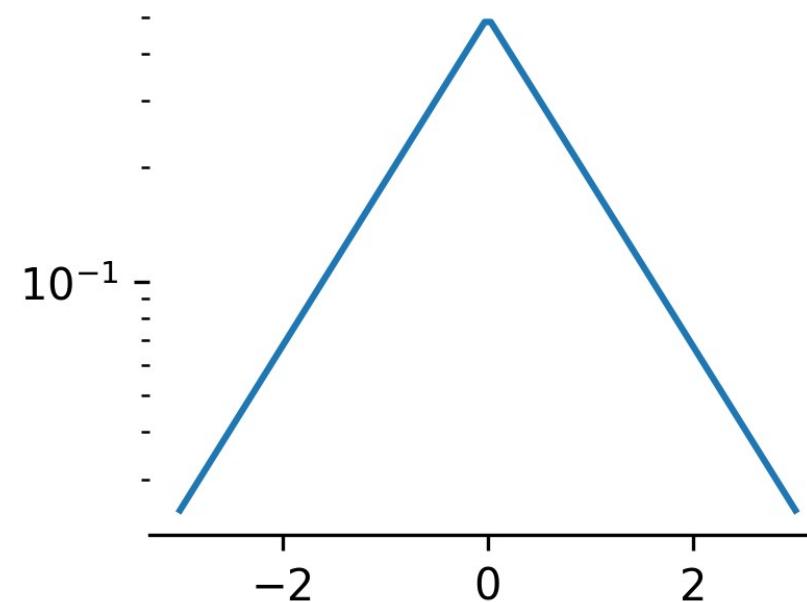
Student



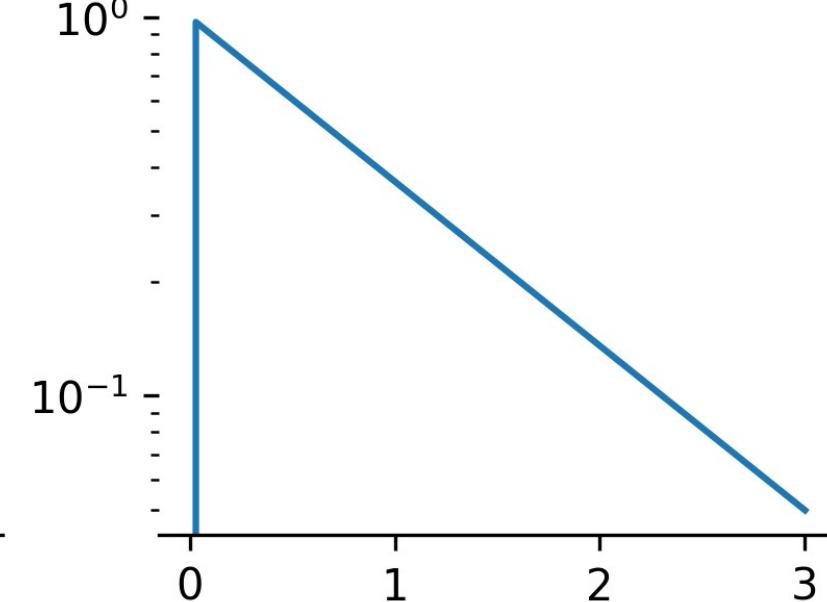
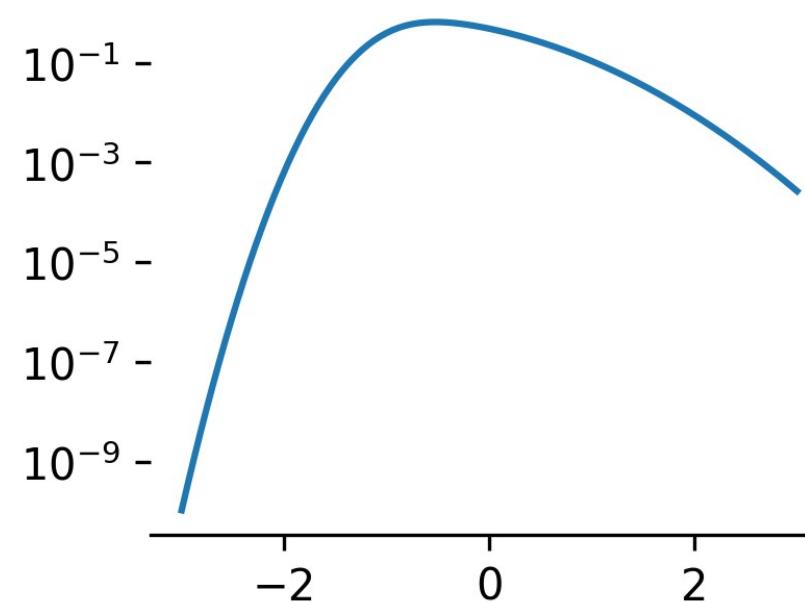
Cauchy



Laplace



Exponential



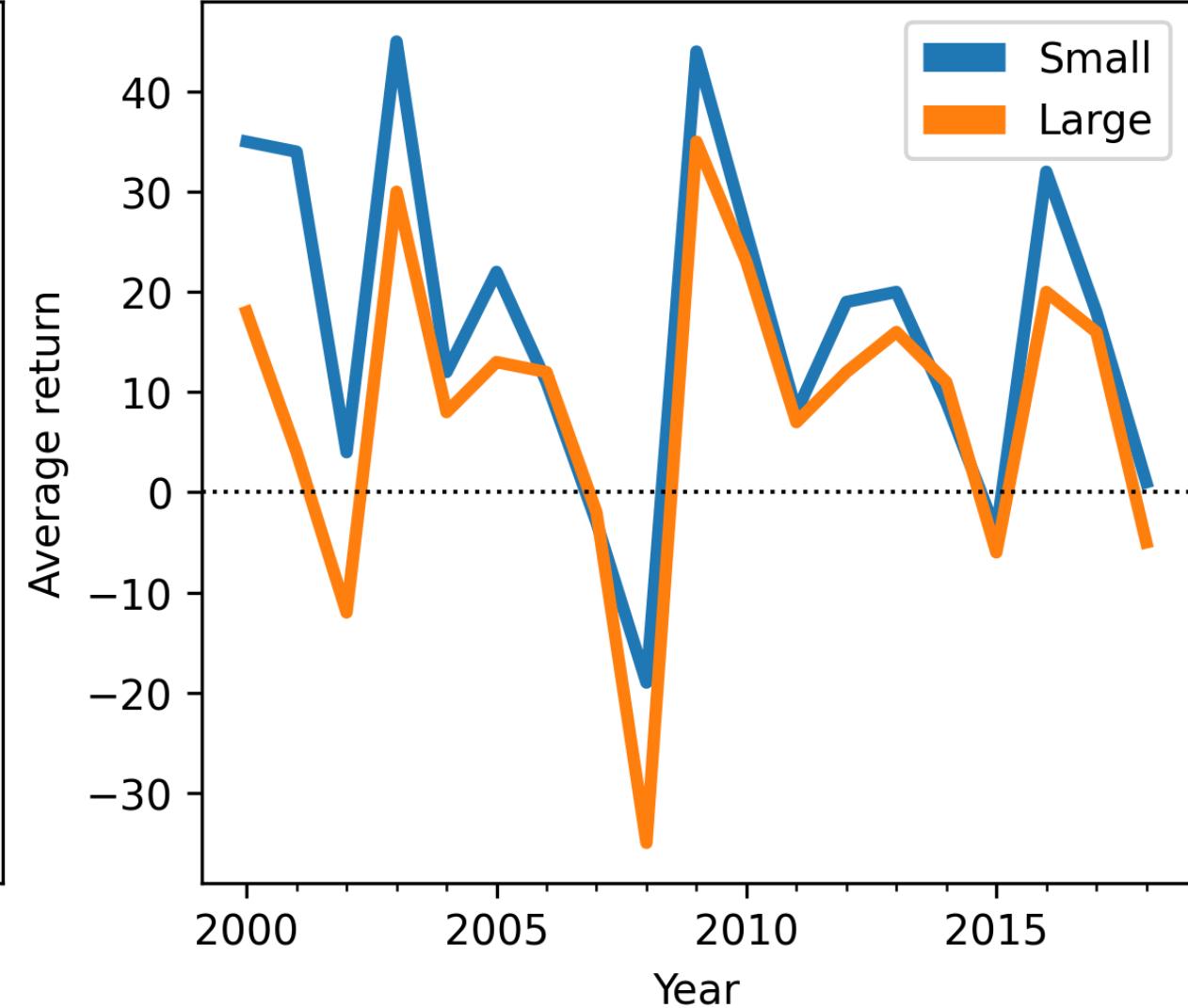
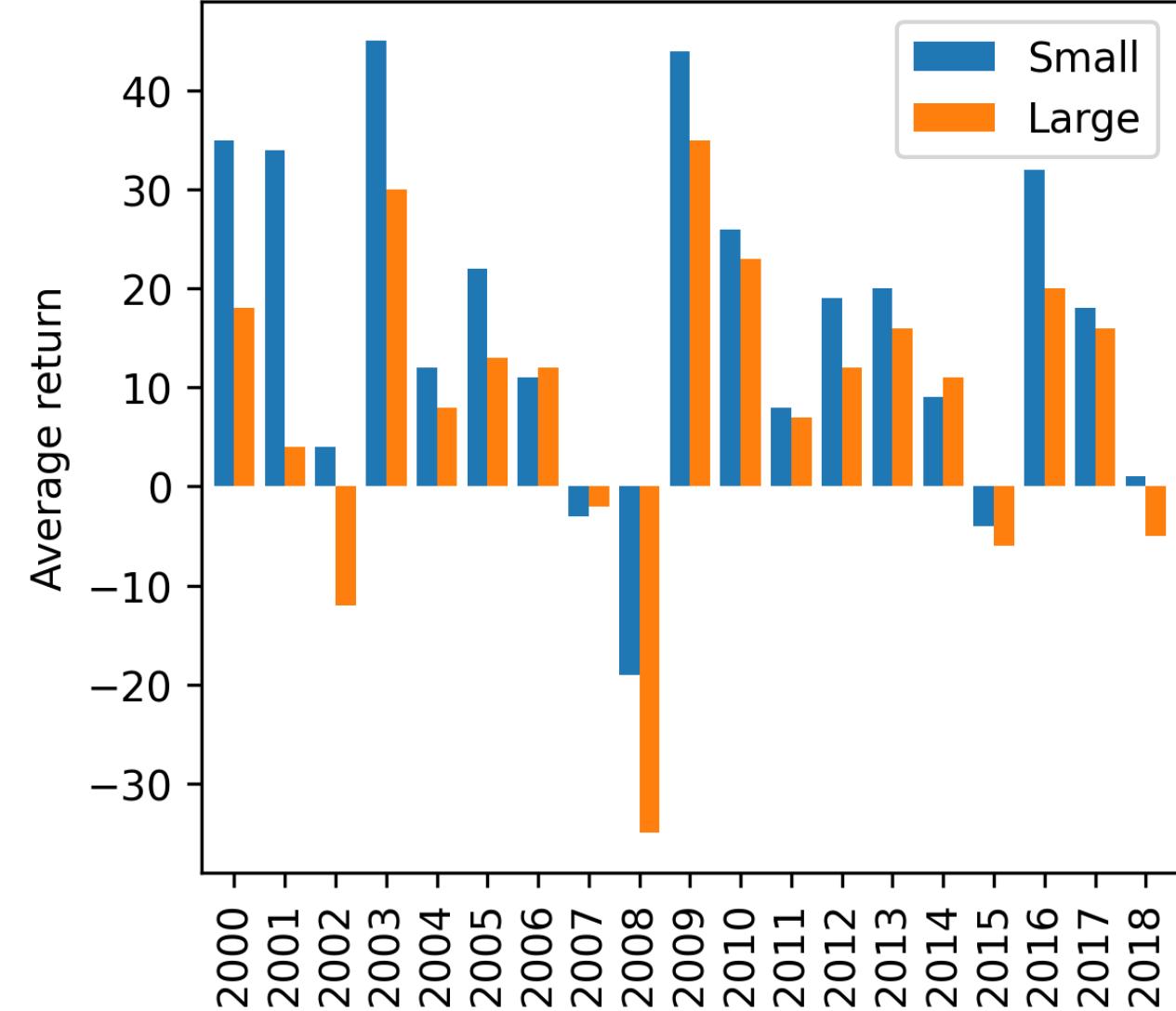
Plotting Advice

Plotting advice

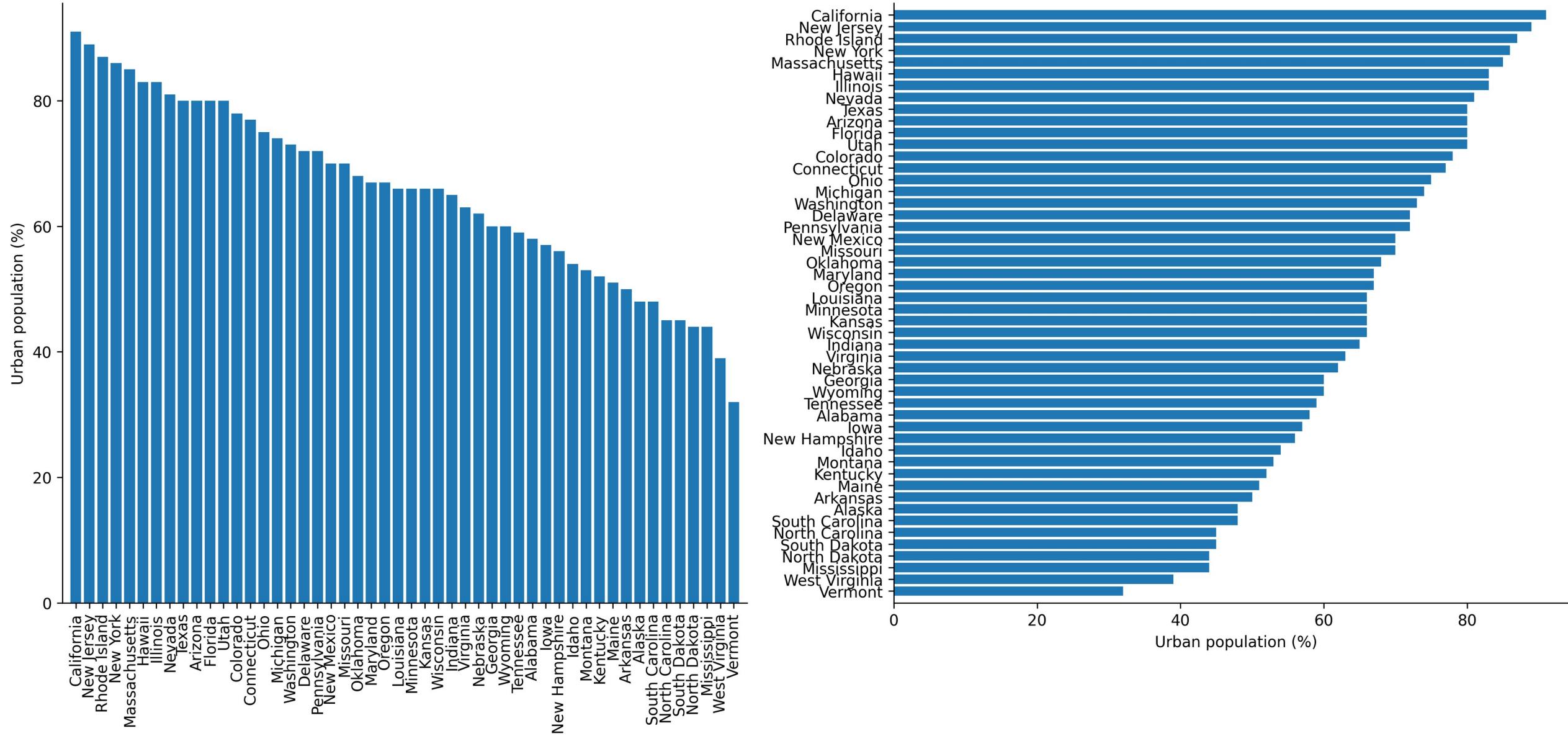
- Label the axes
- Line plots instead of Excel plots
- Horizontal labels
- Unambiguous date format
- Years are intervals, not points
- Discrete gradients
- Logarithmic scales
- Thick lines; even thicker lines in the legend
- Same order in the legend and in the plot
- Labels on the plot instead of in the legend
- Colour palettes
- Darker, more saturated colours for line plots

Label the axes

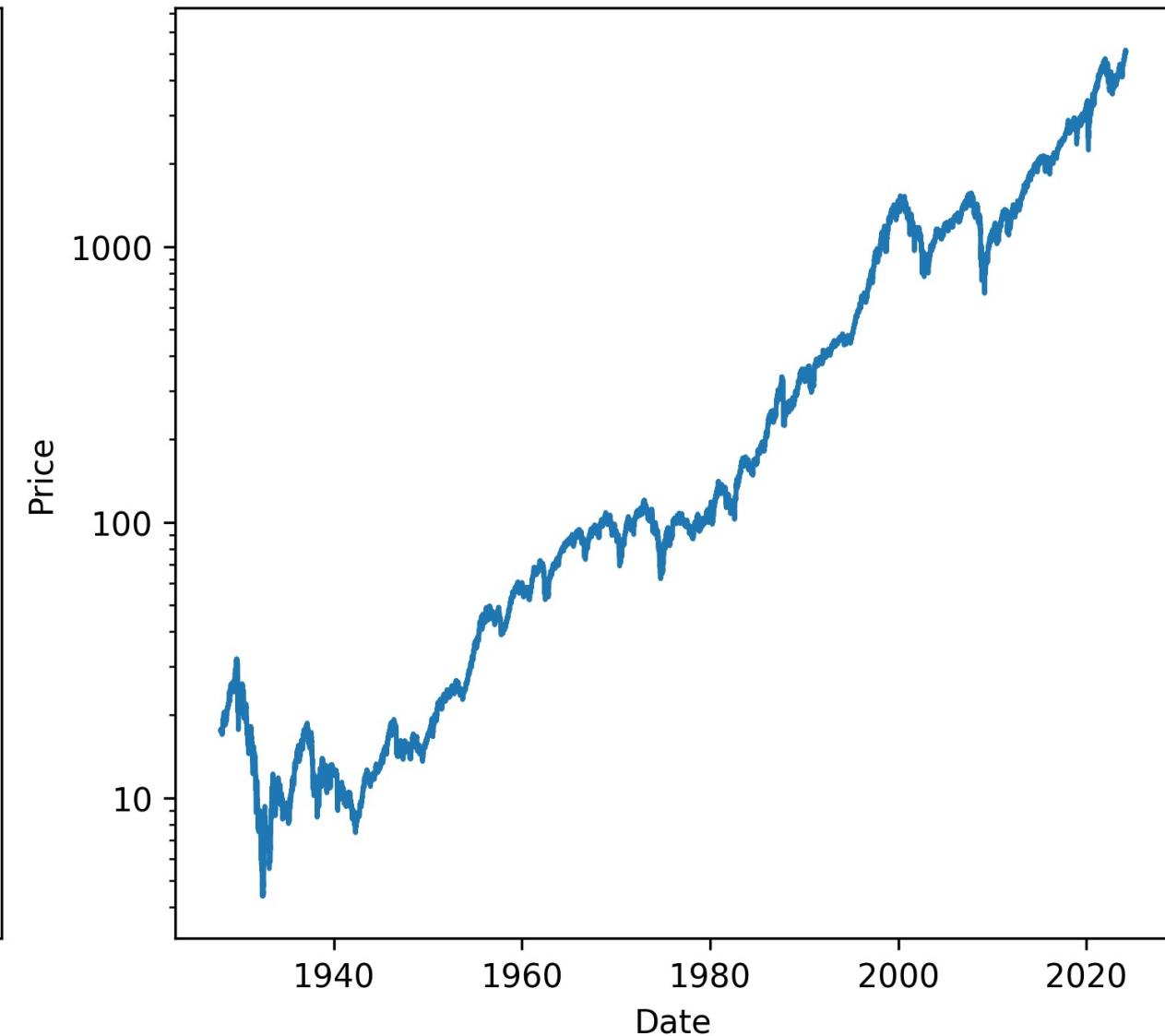
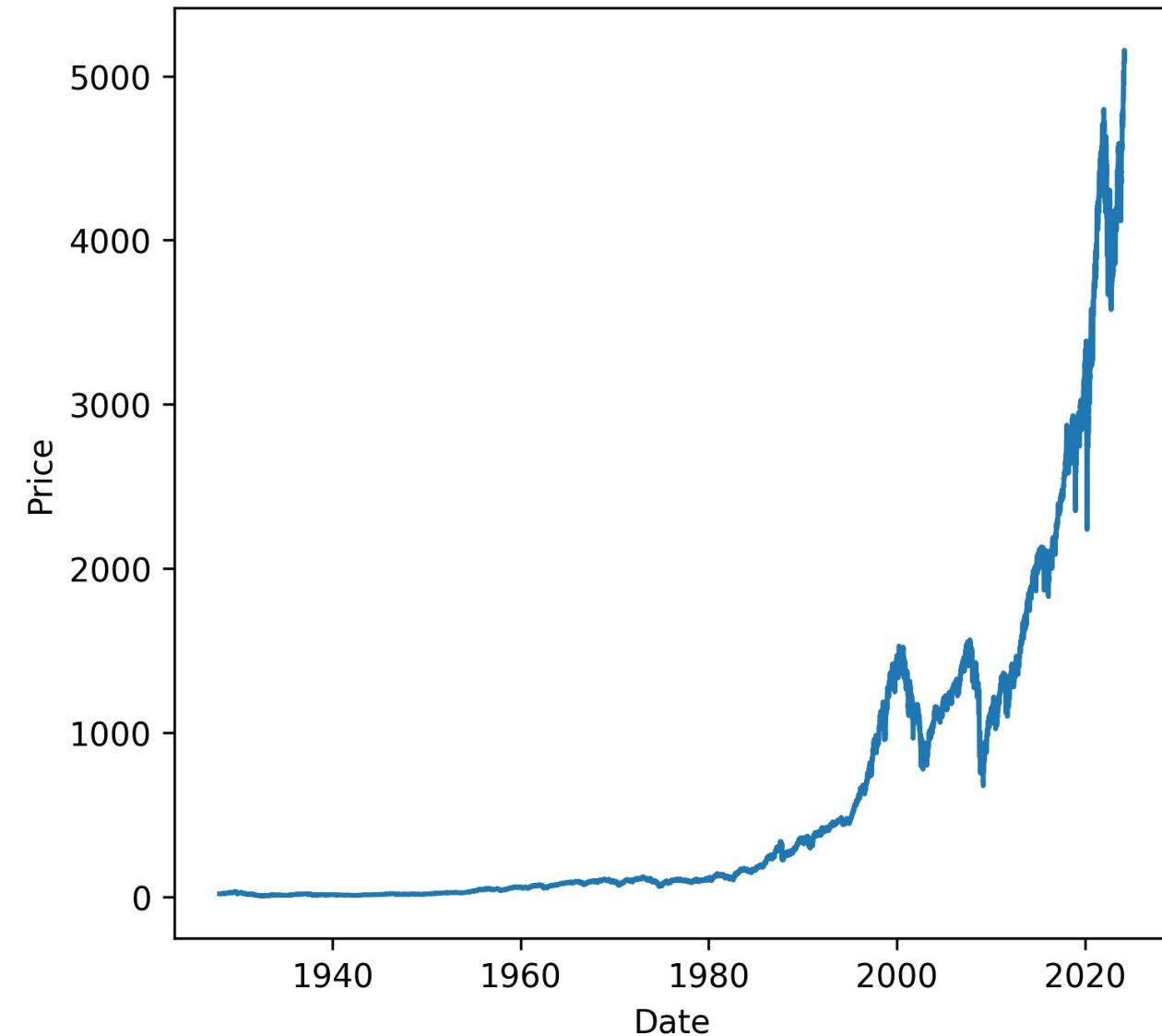
Line Plots



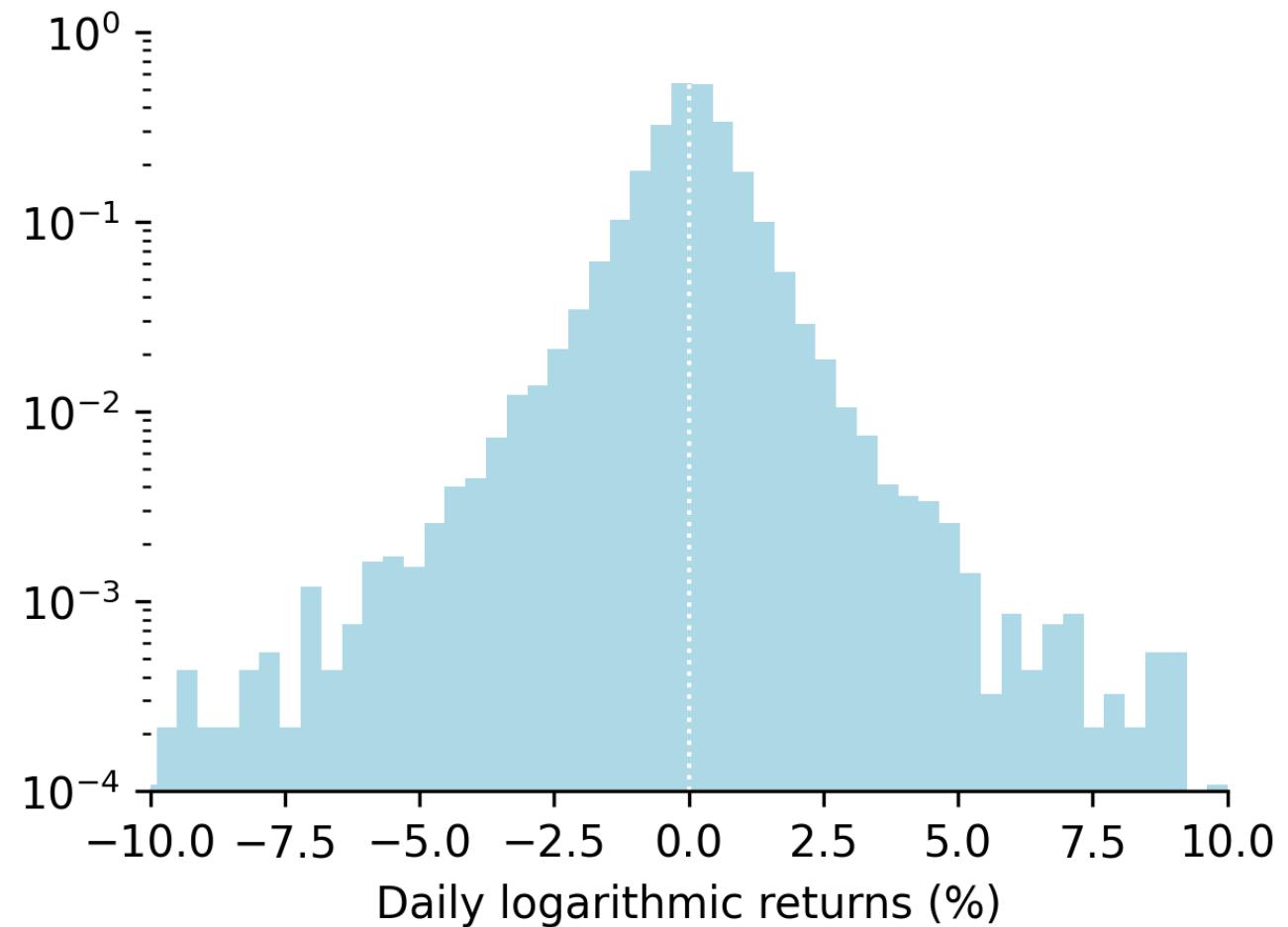
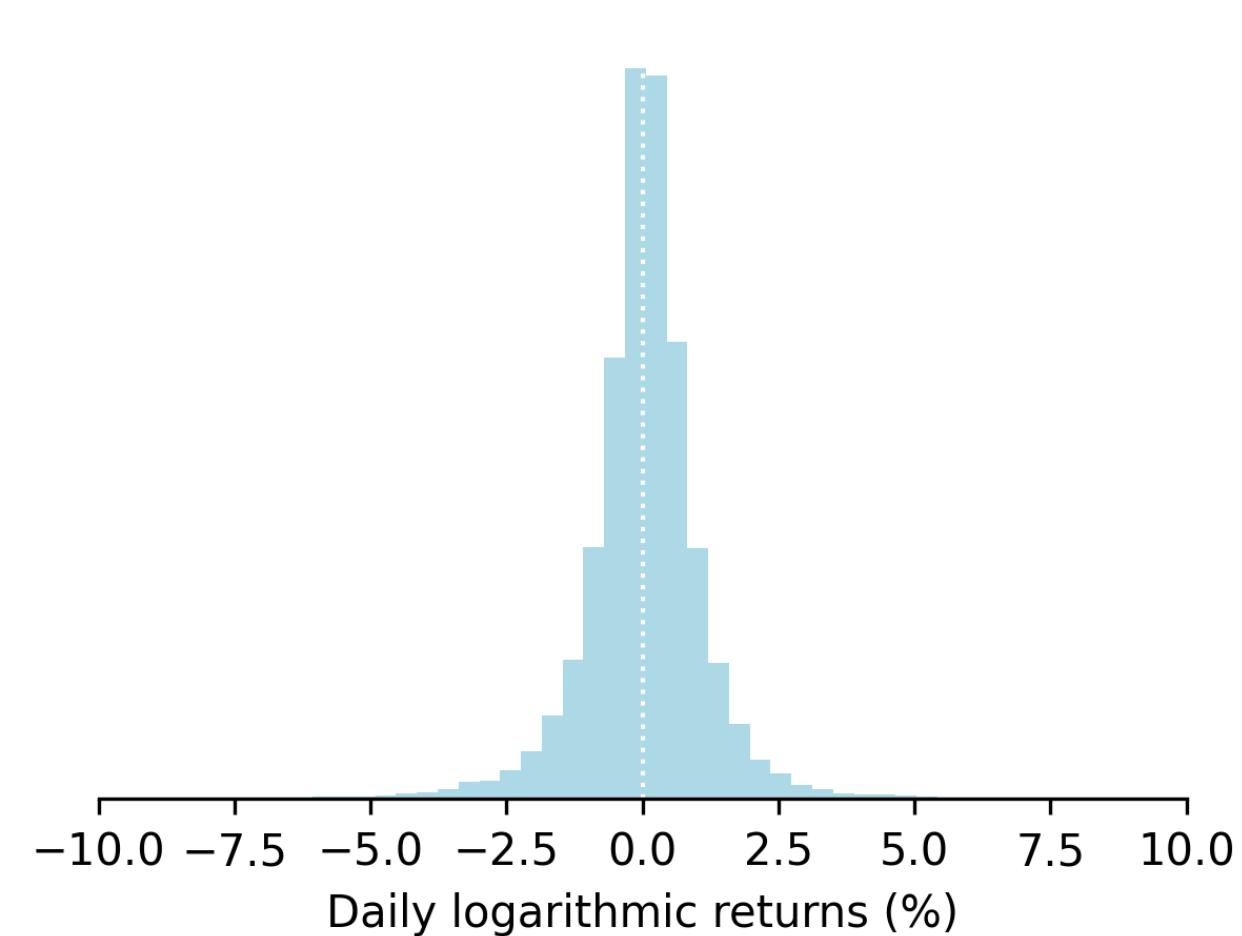
Horizontal Labels



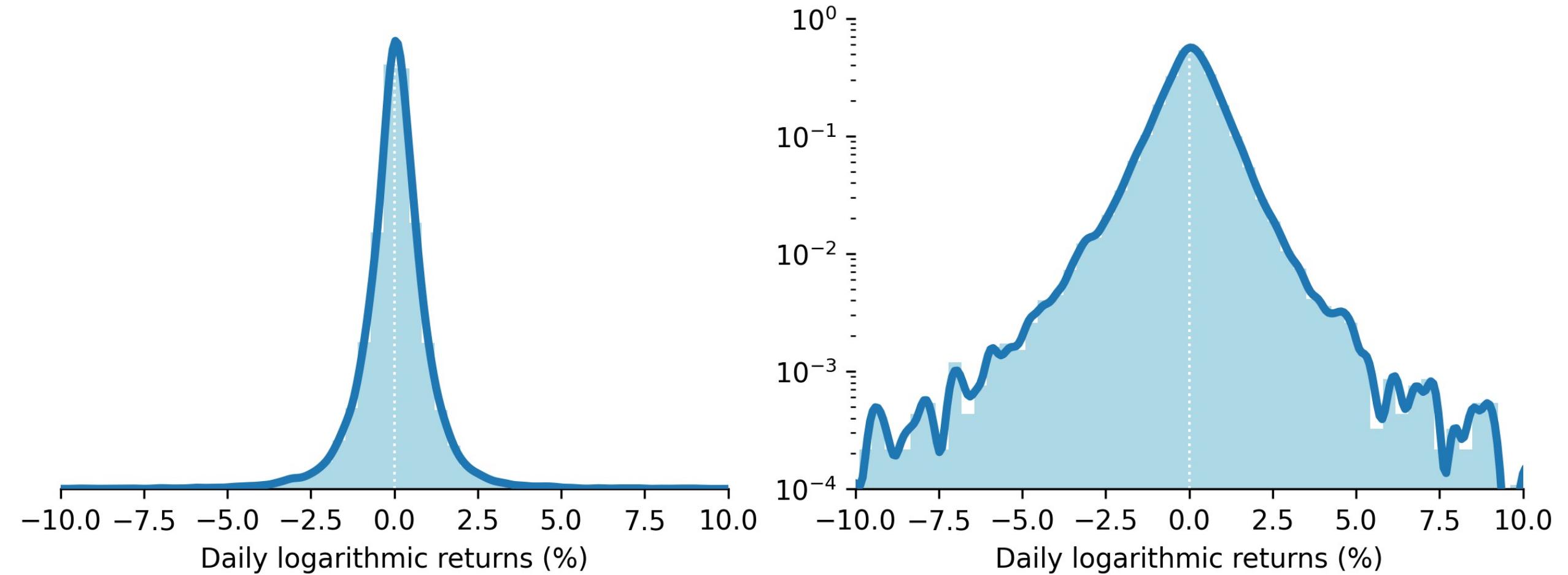
Logarithmic scales



Logarithmic scales

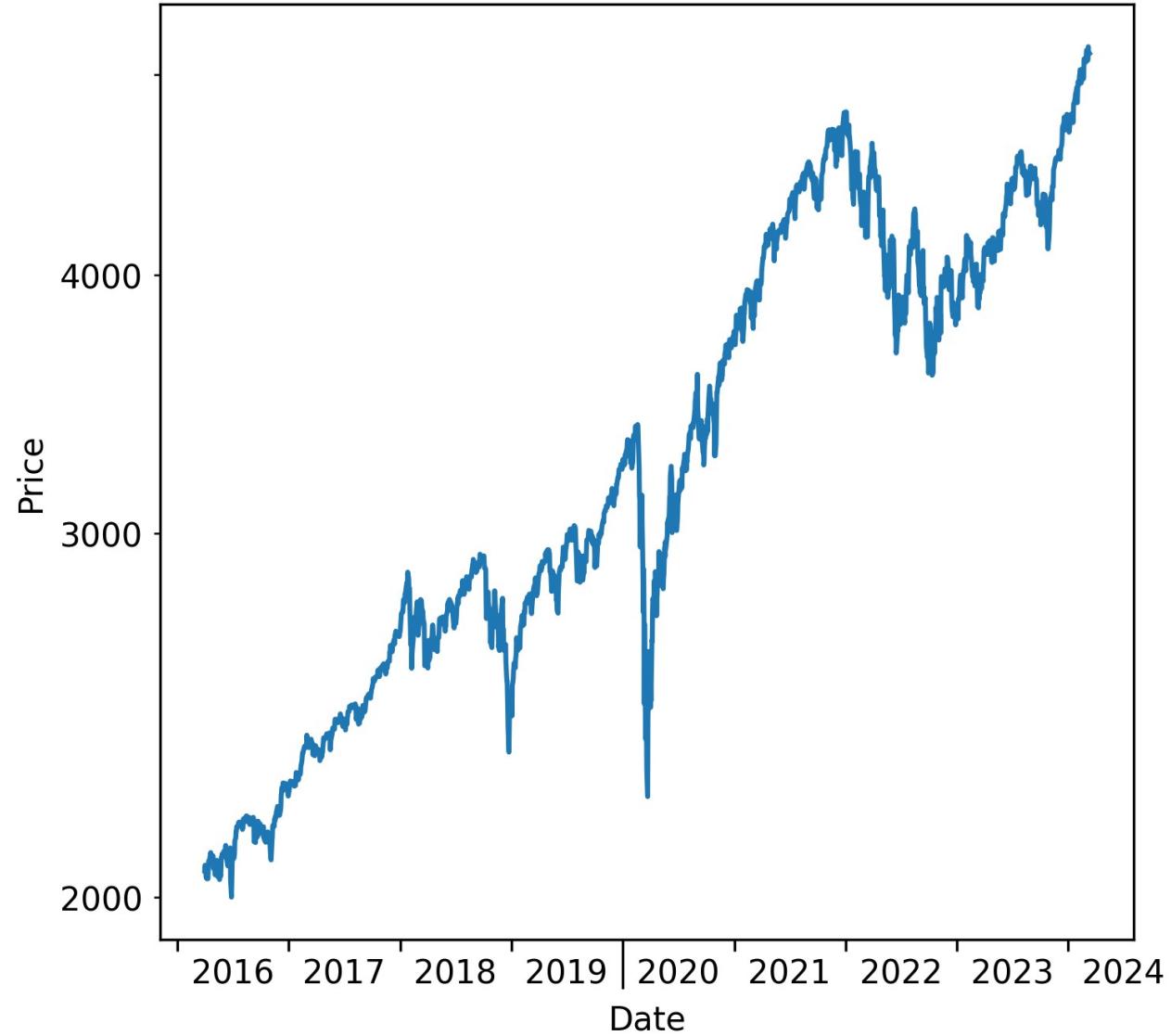
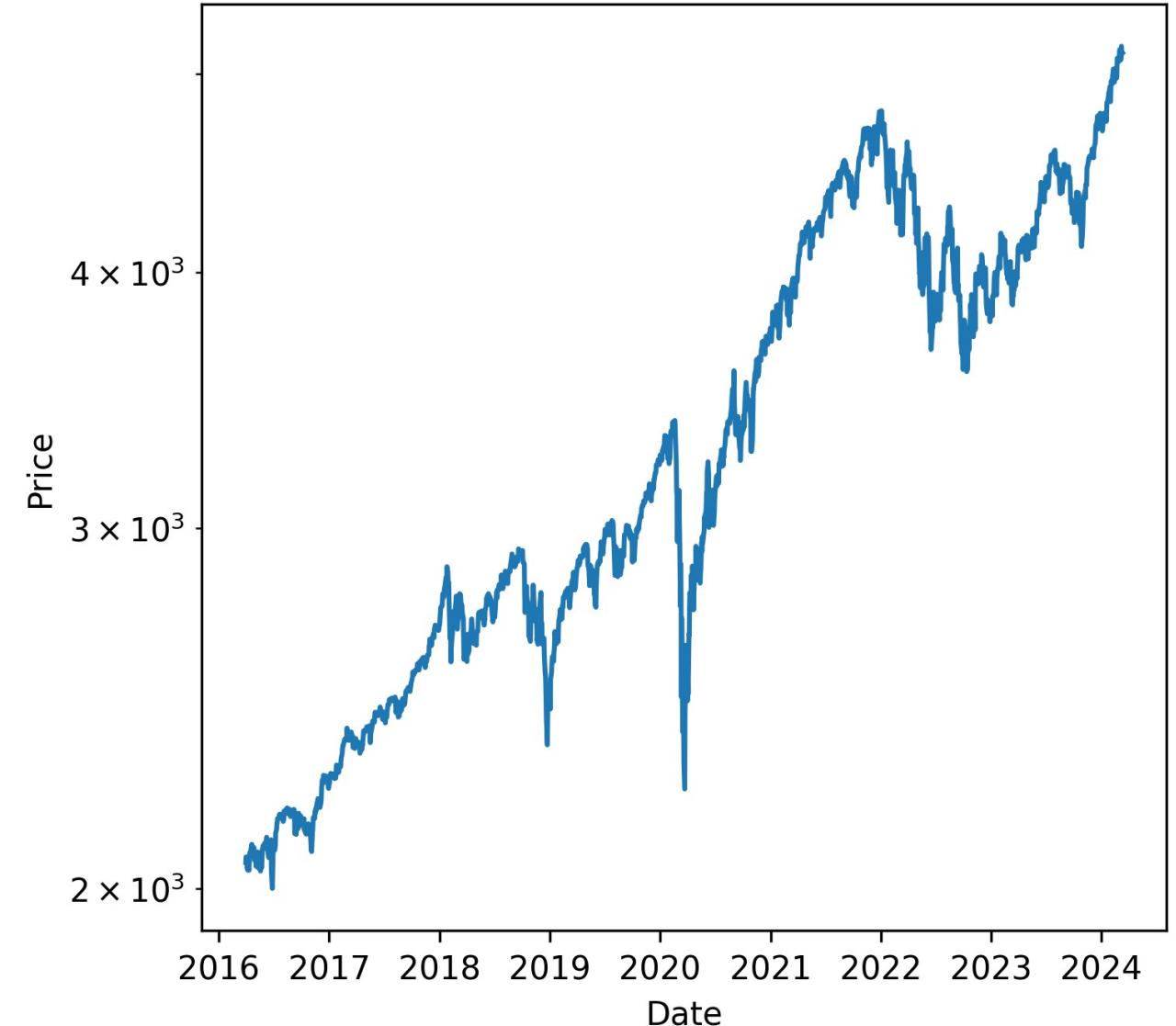


Logarithmic scales

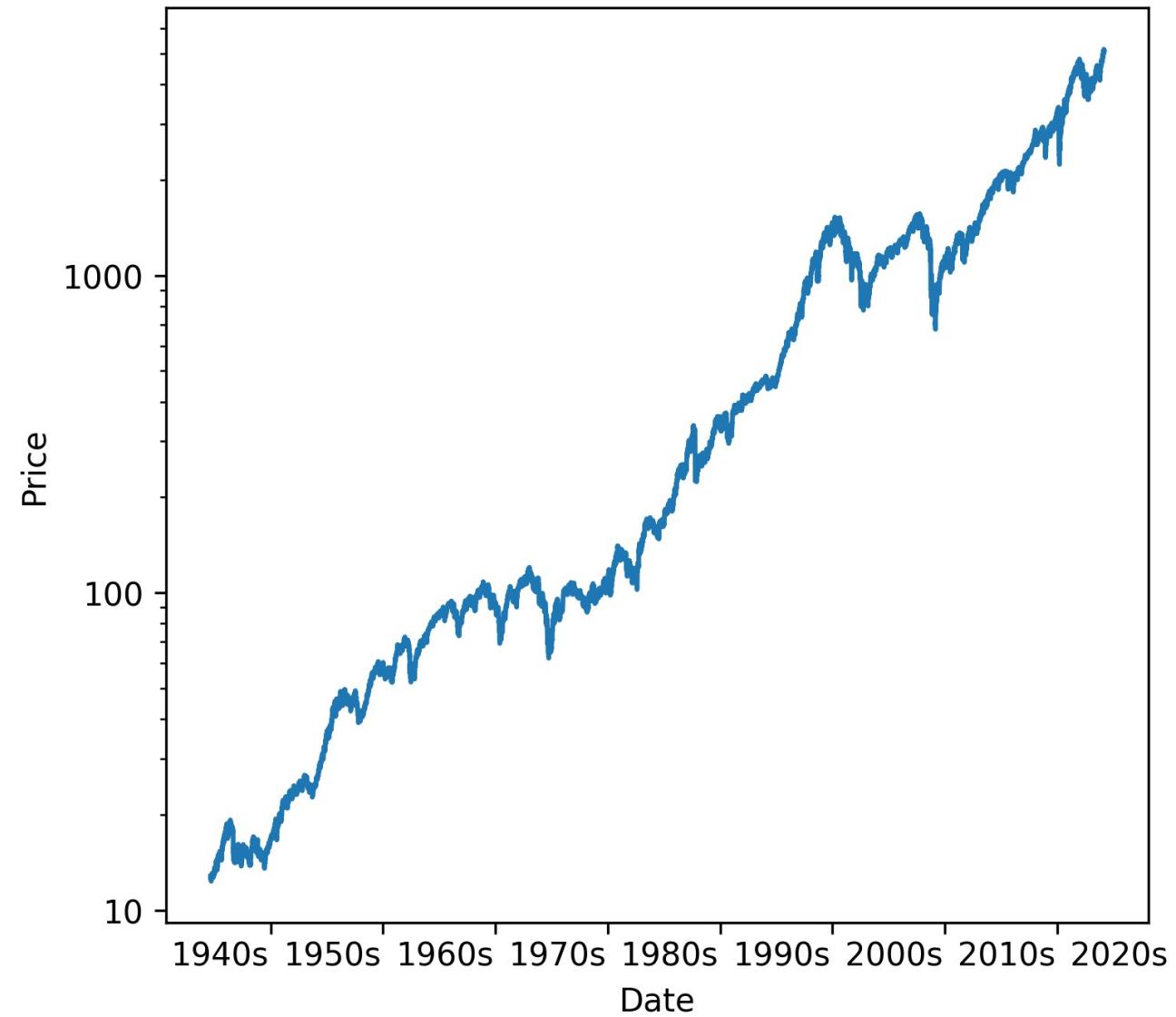
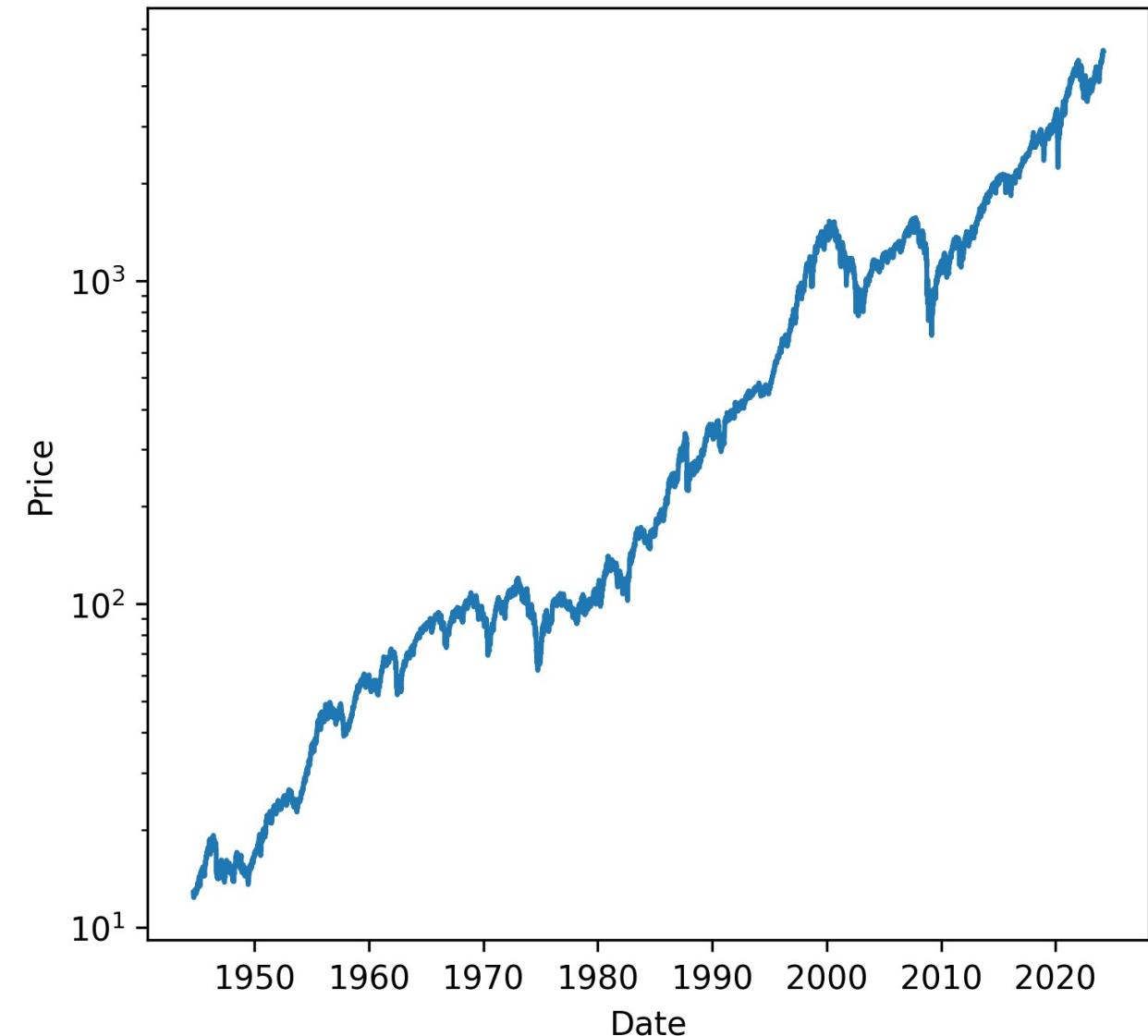


Unambiguous date format

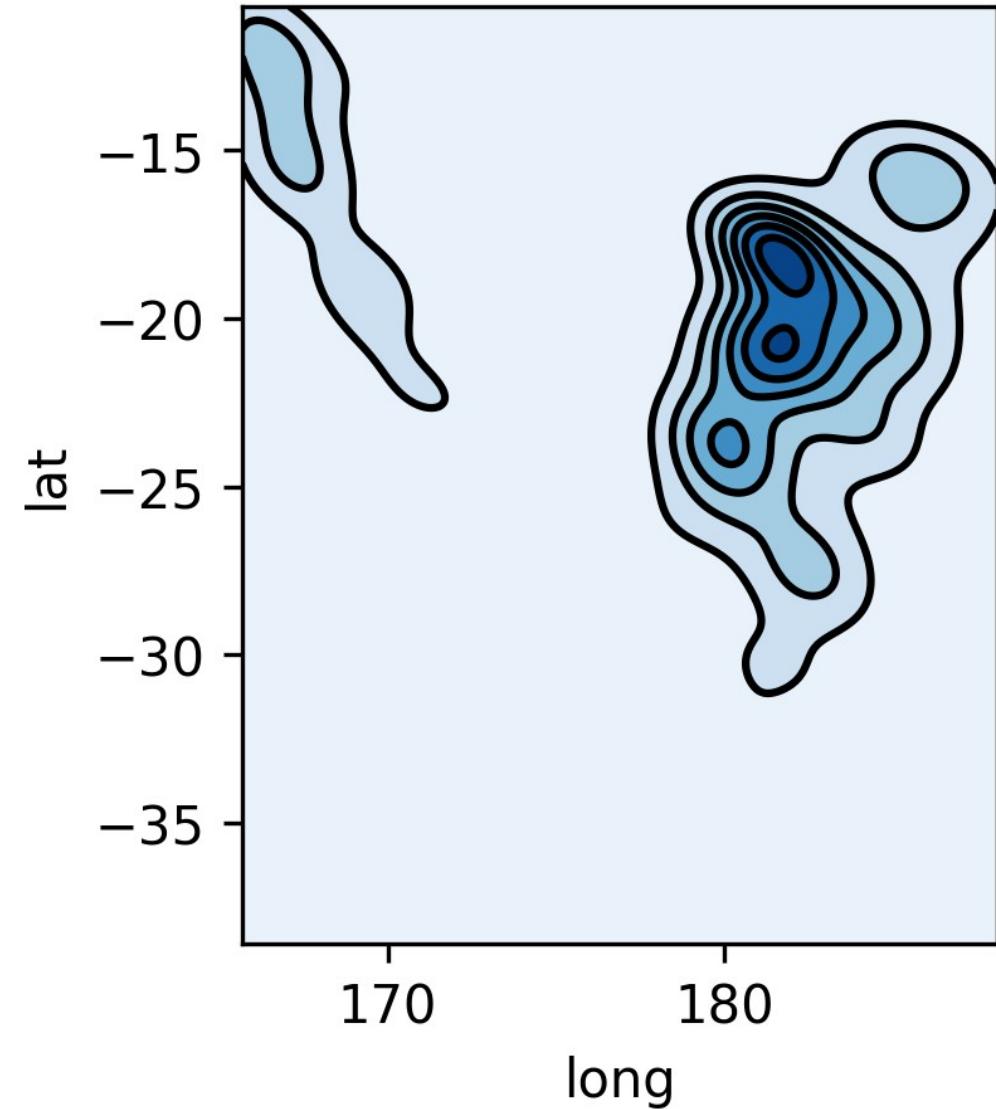
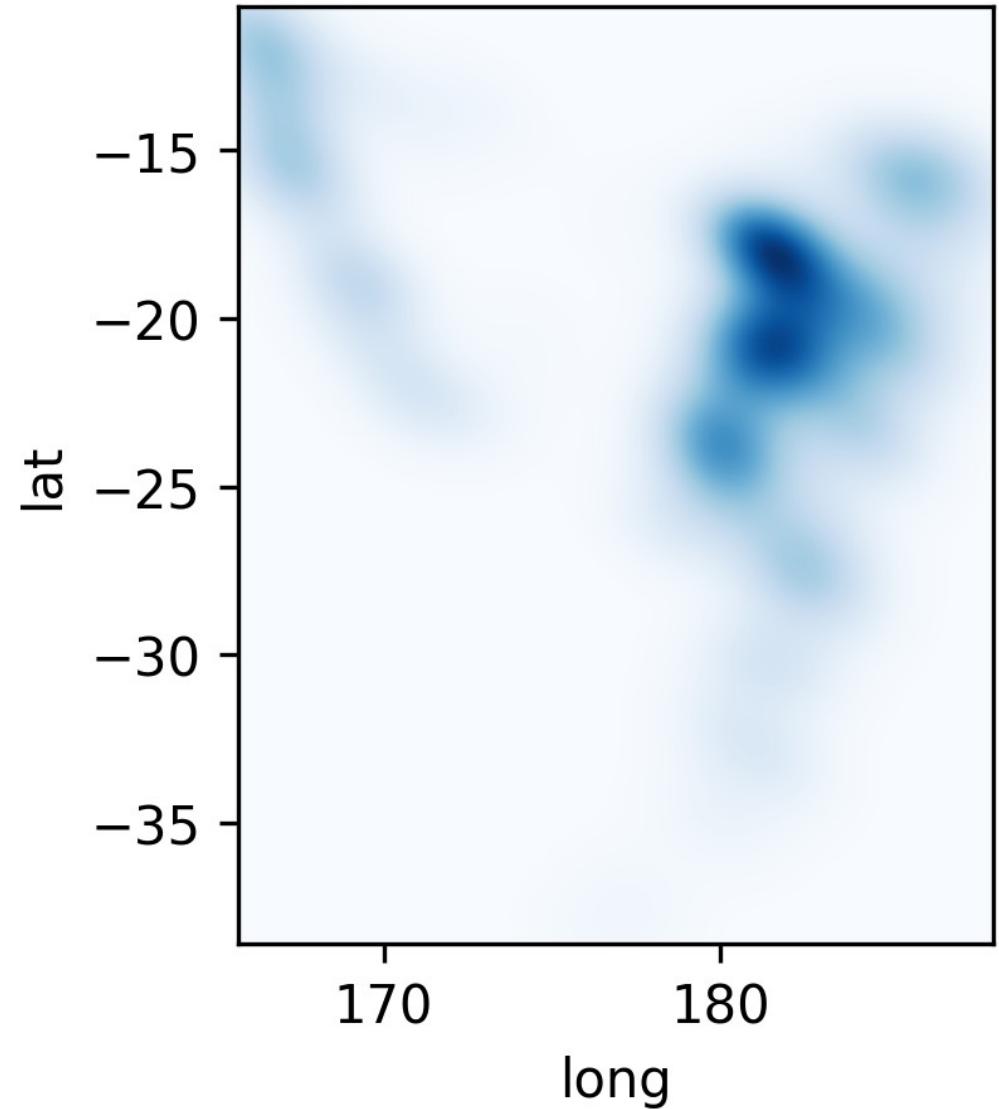
Years are intervals



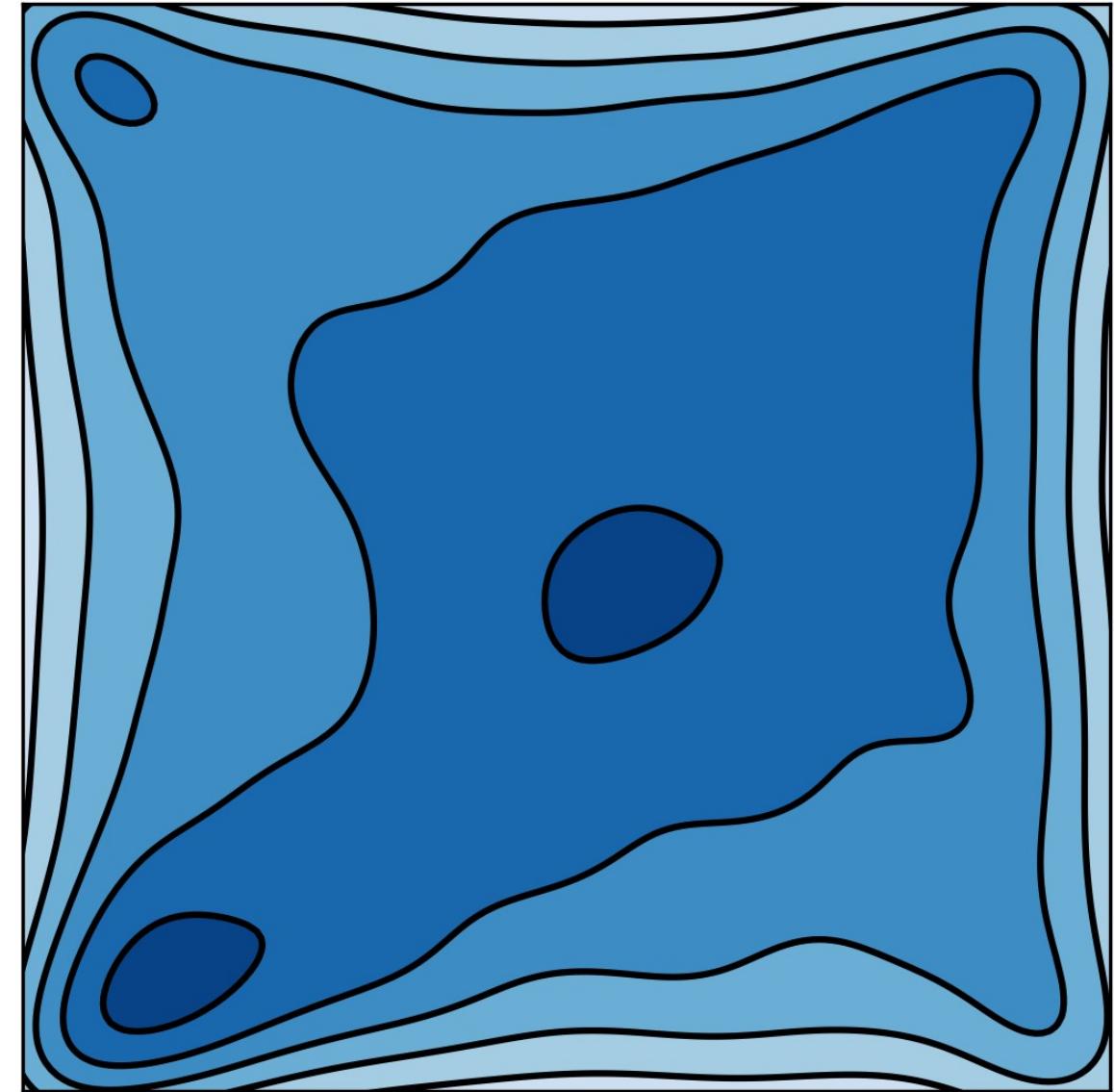
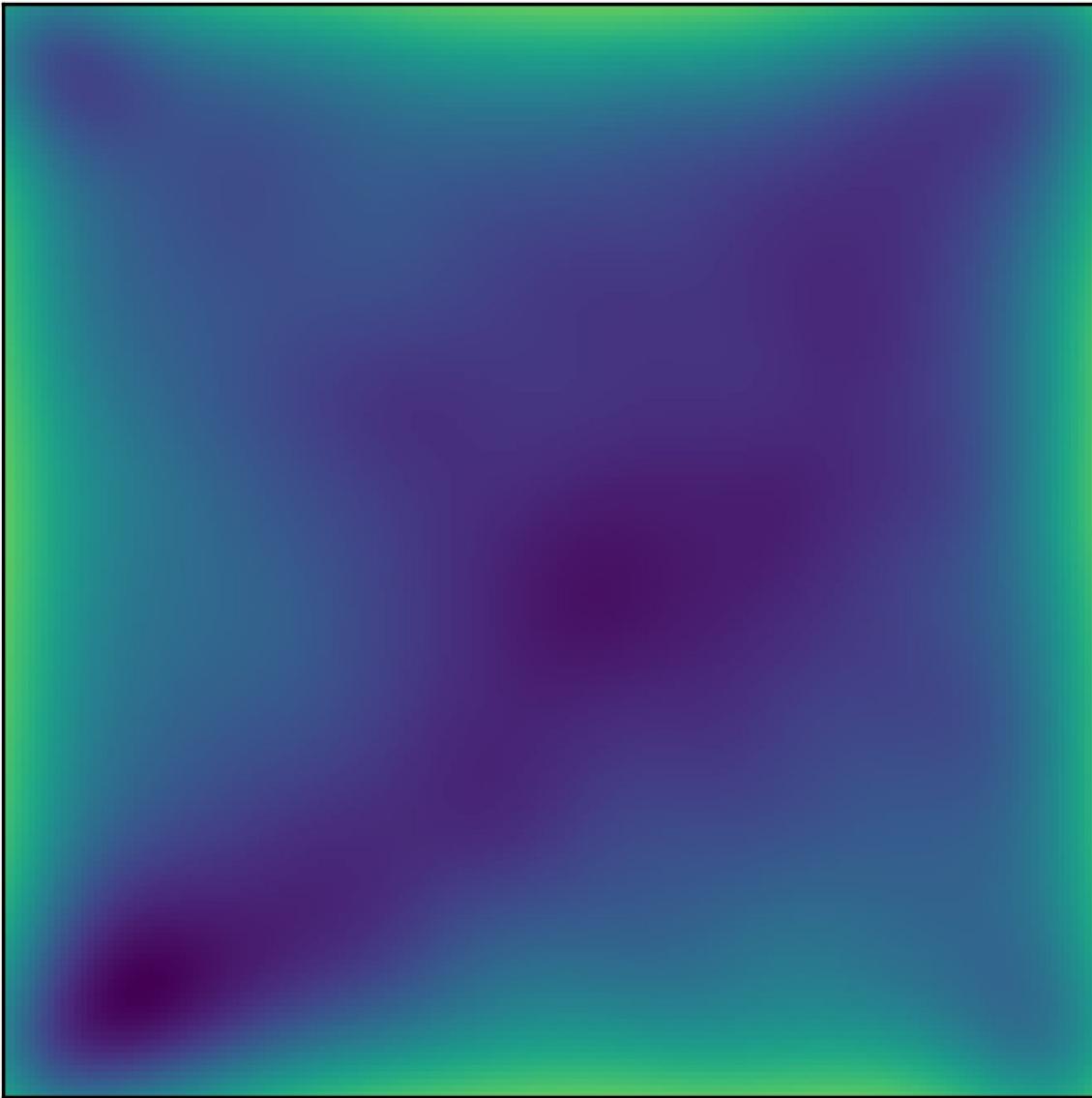
Years are intervals



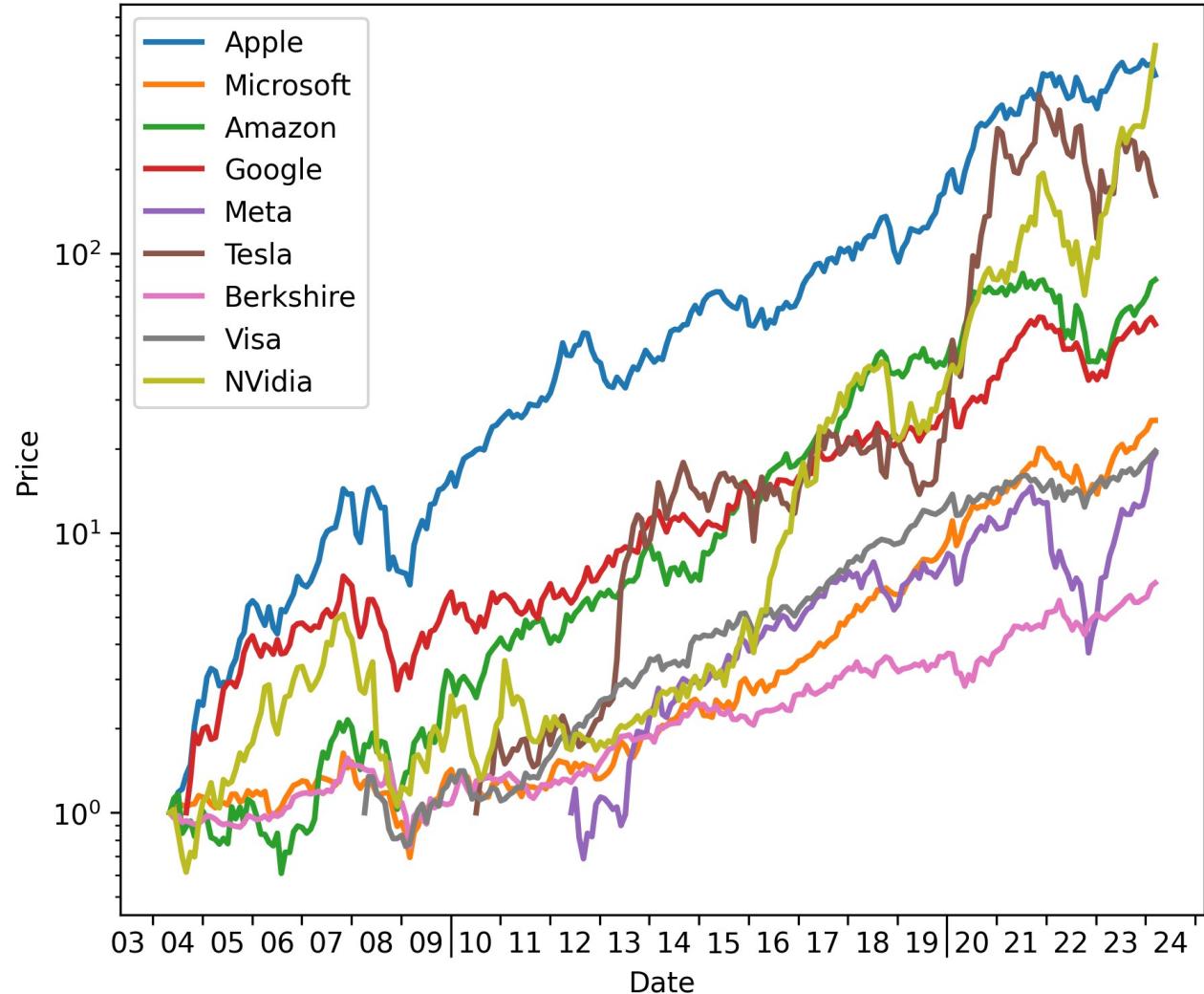
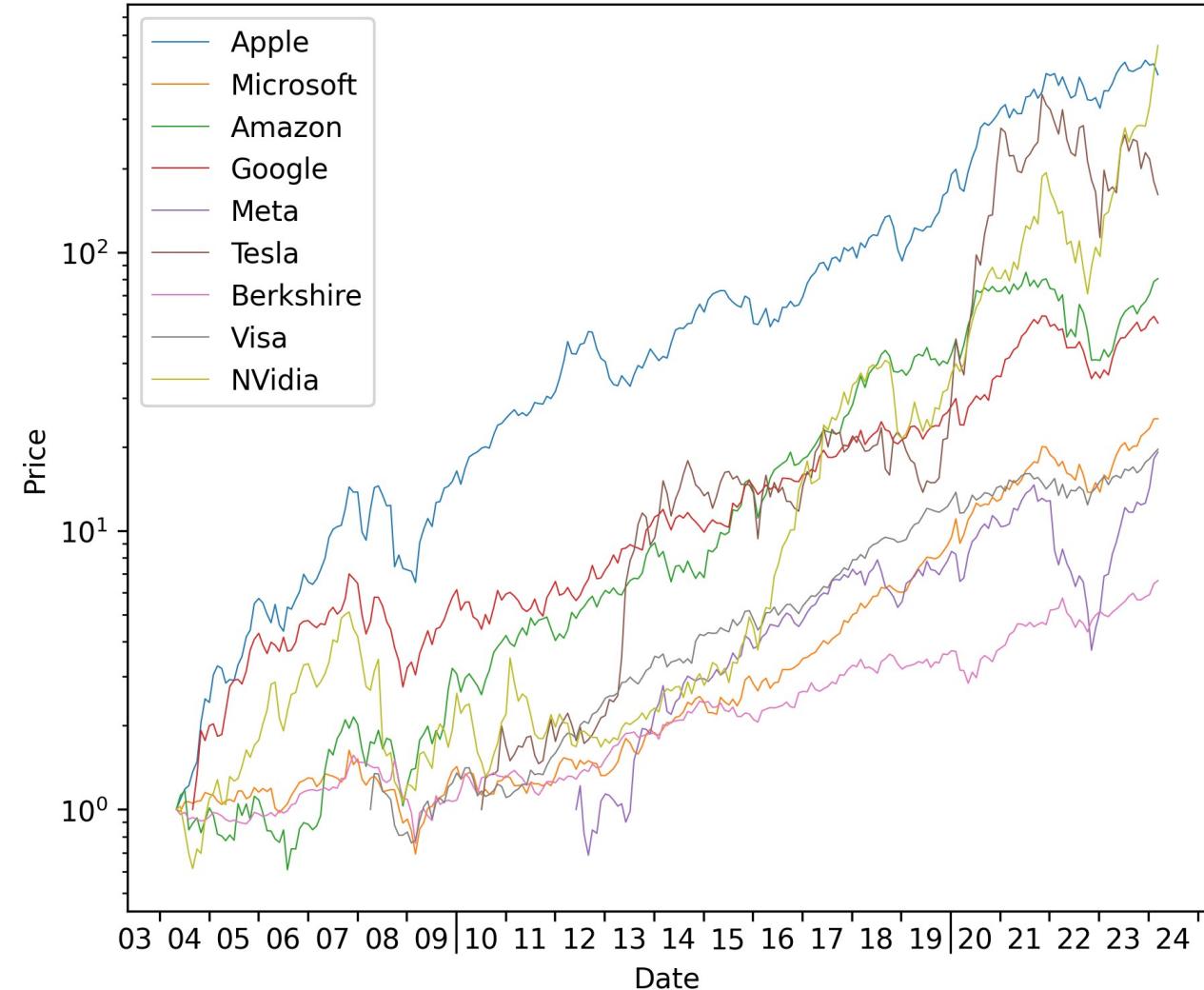
Discrete gradients



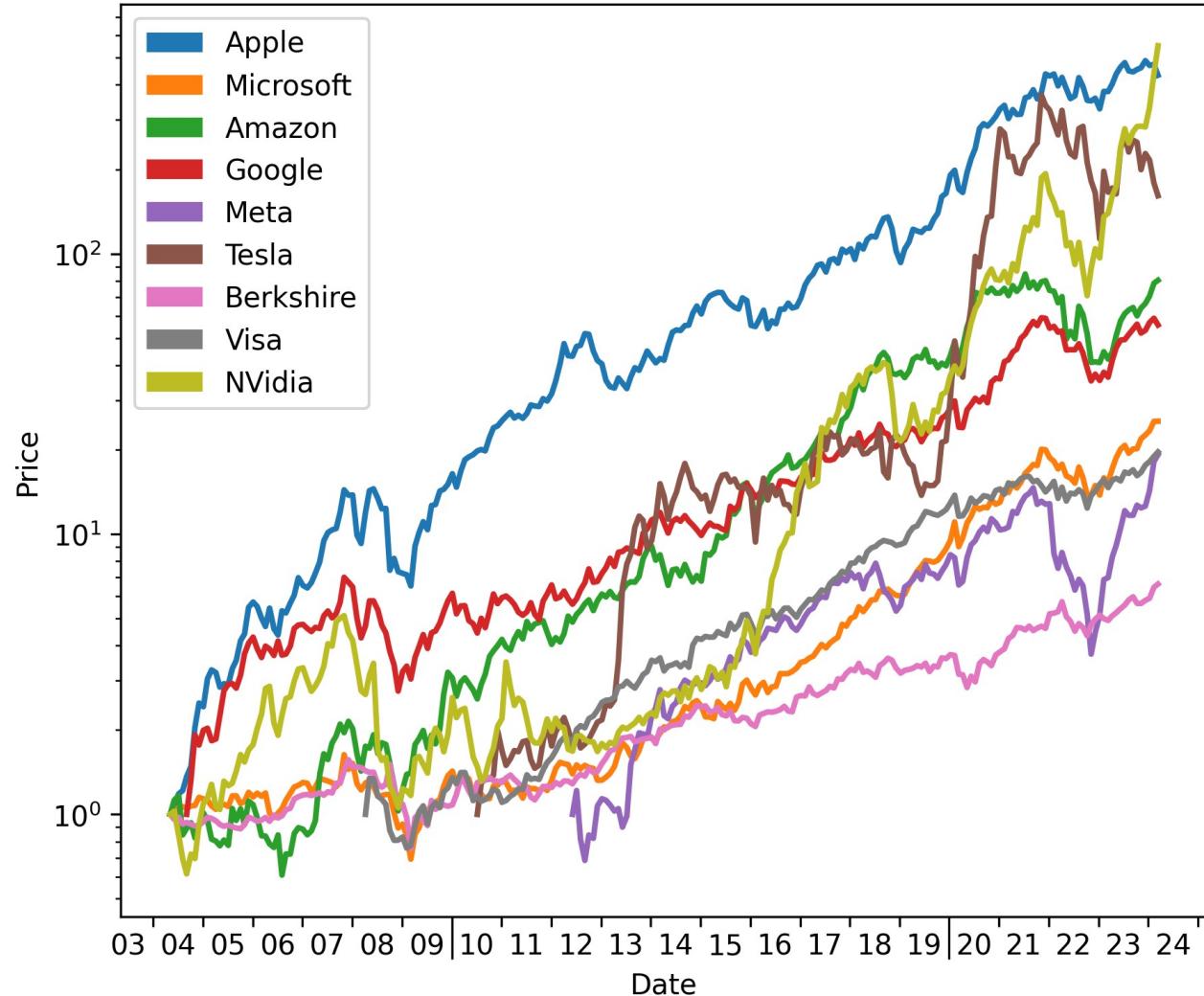
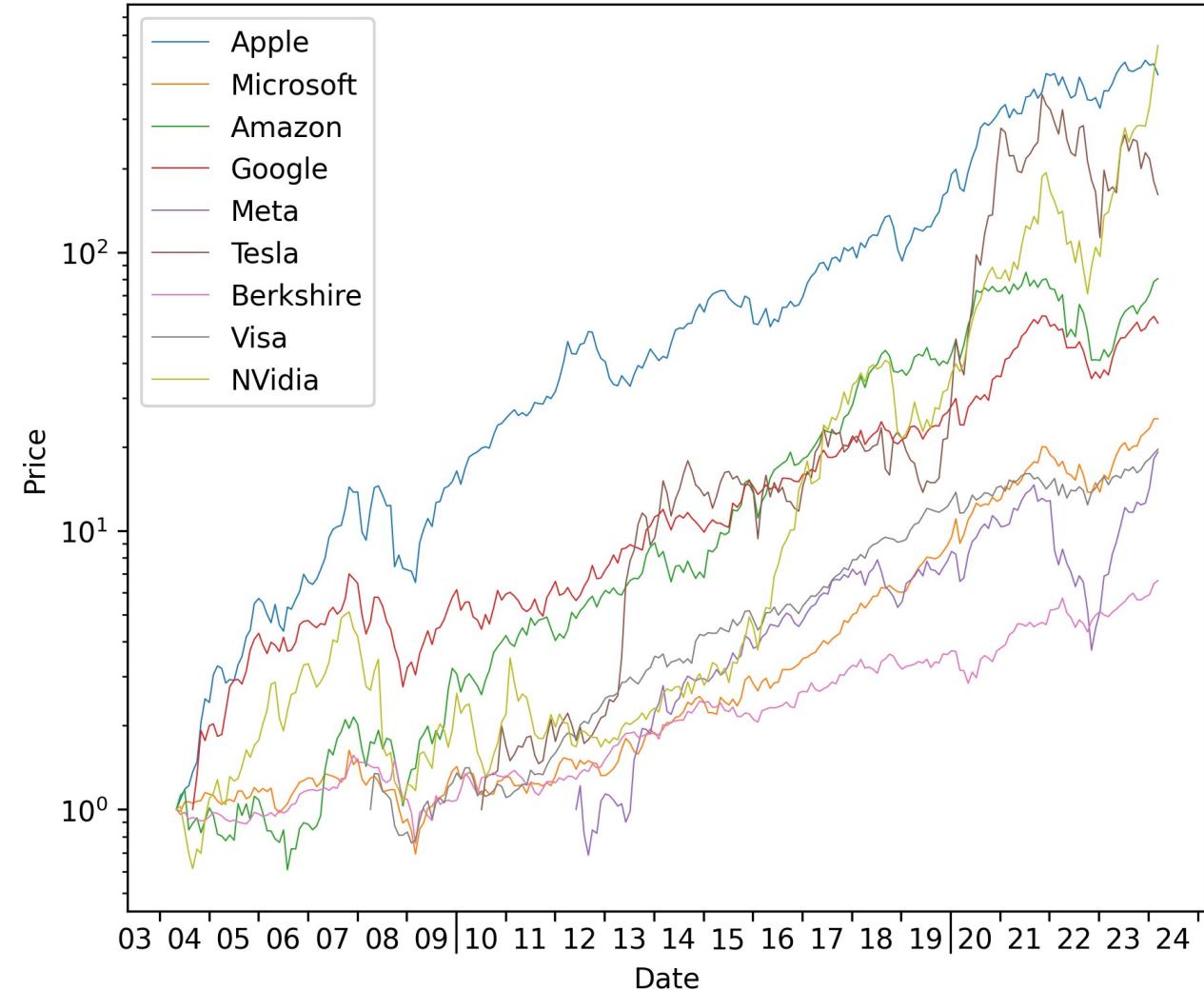
Discrete gradients



Thick lines

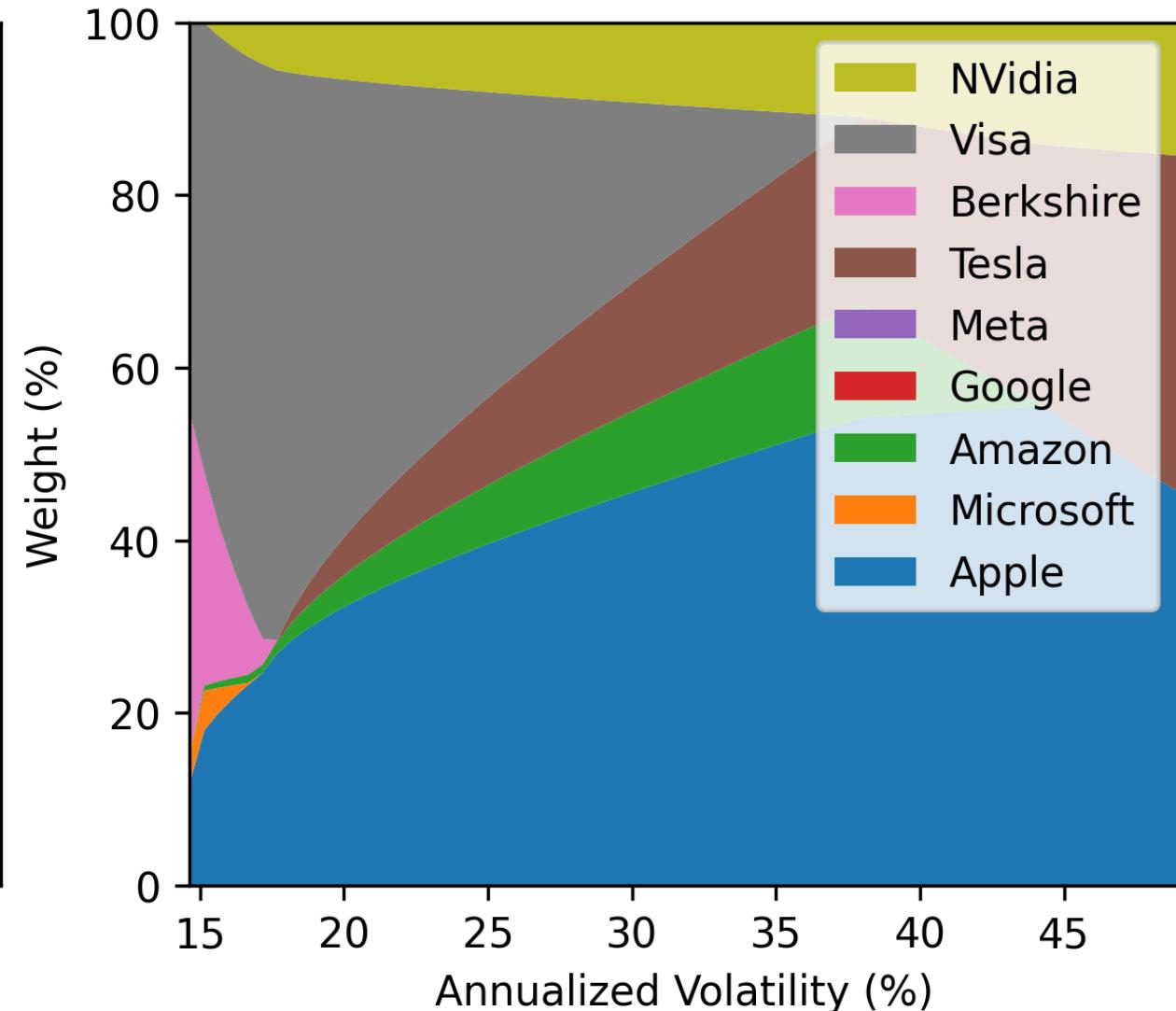
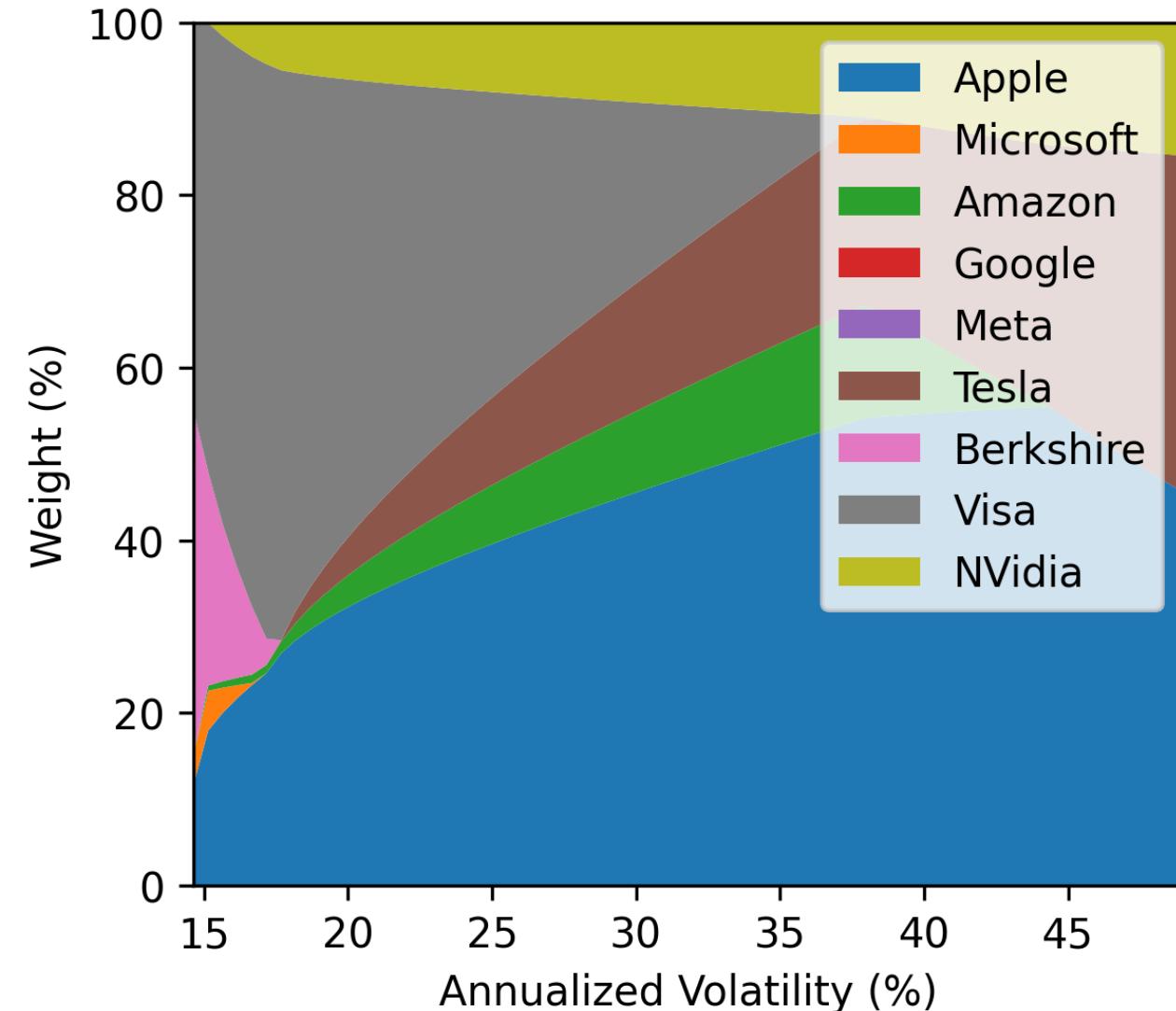


Thicker lines in the legend

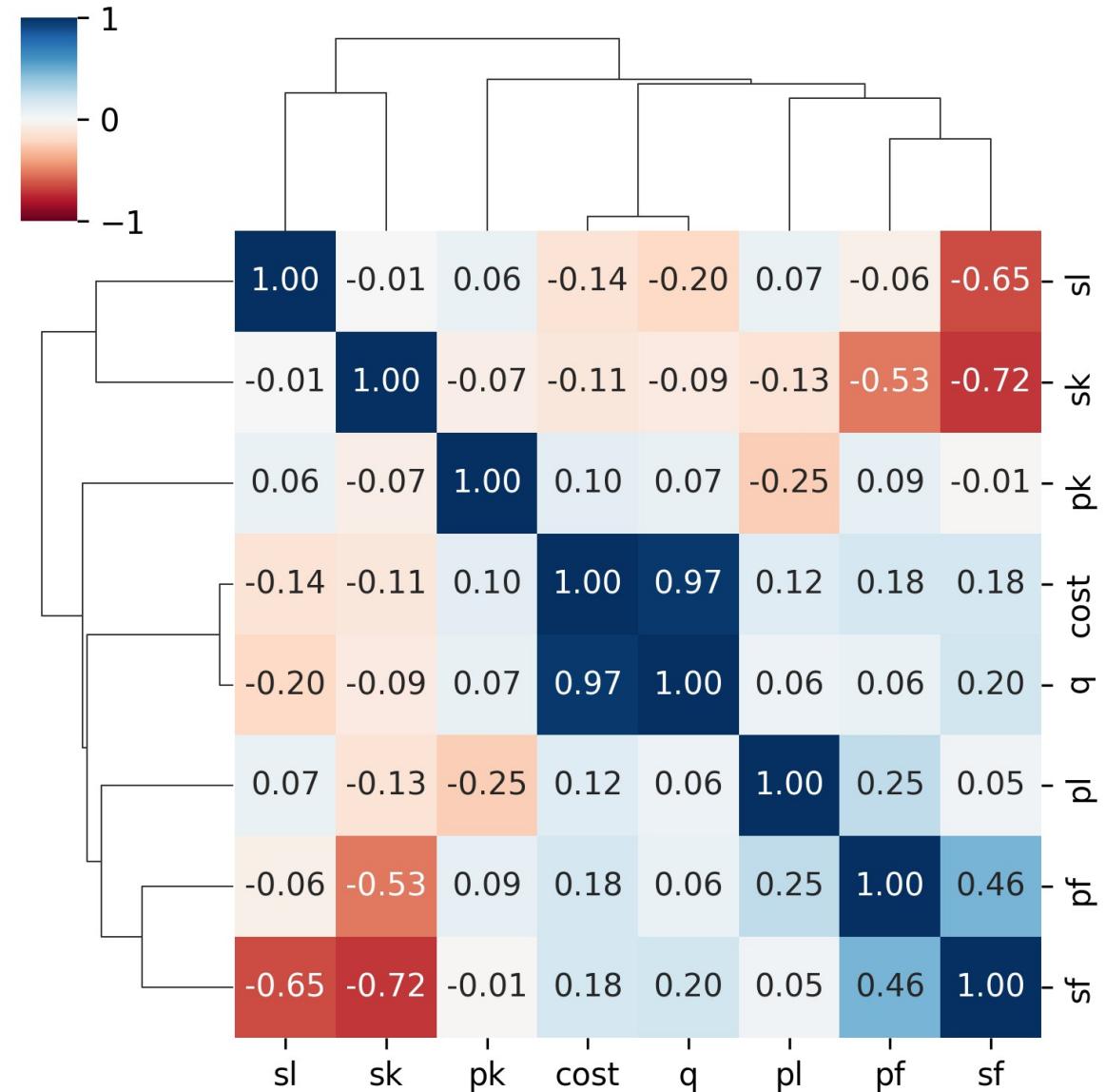
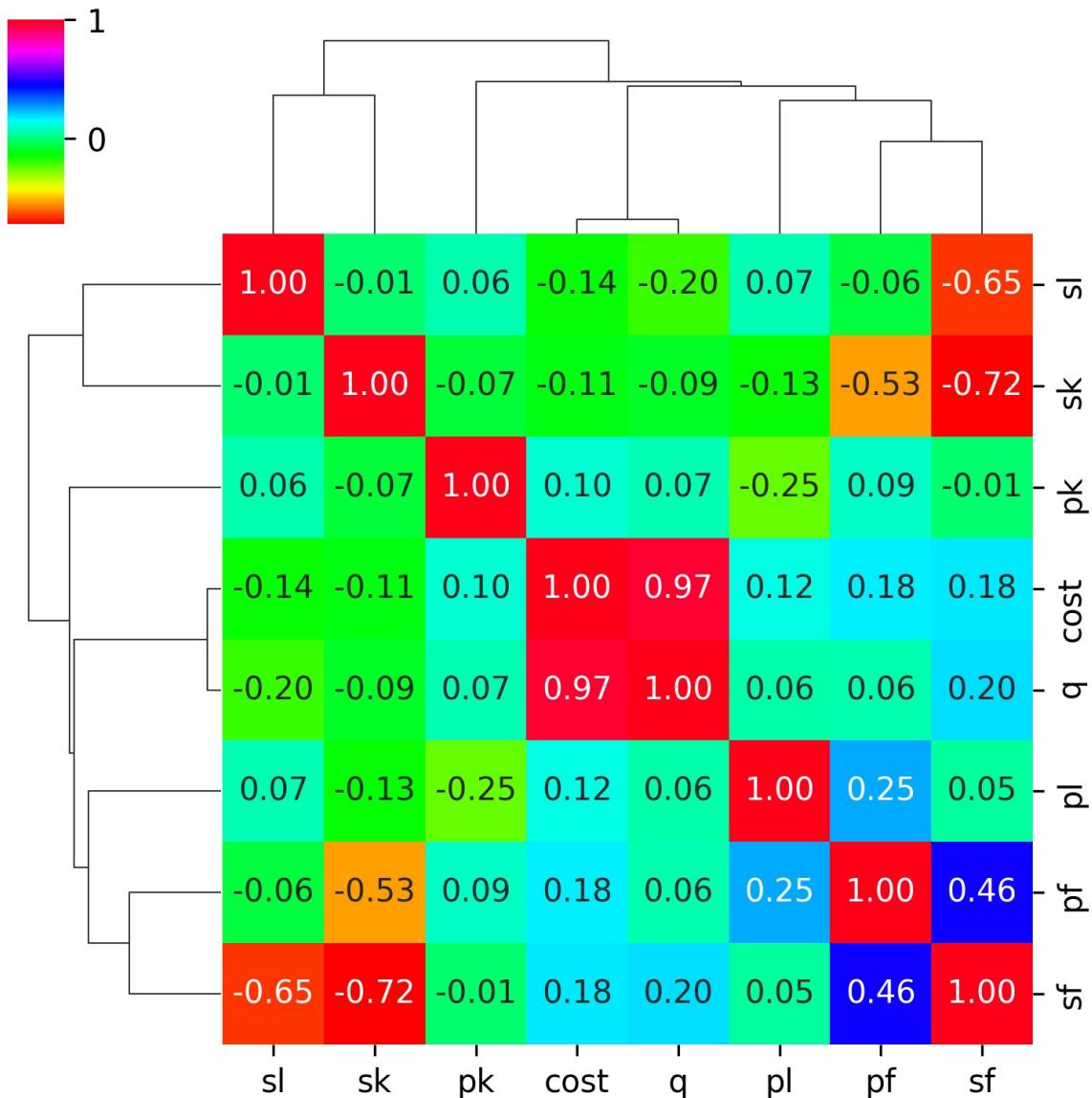


Labels on the plot

Legend order



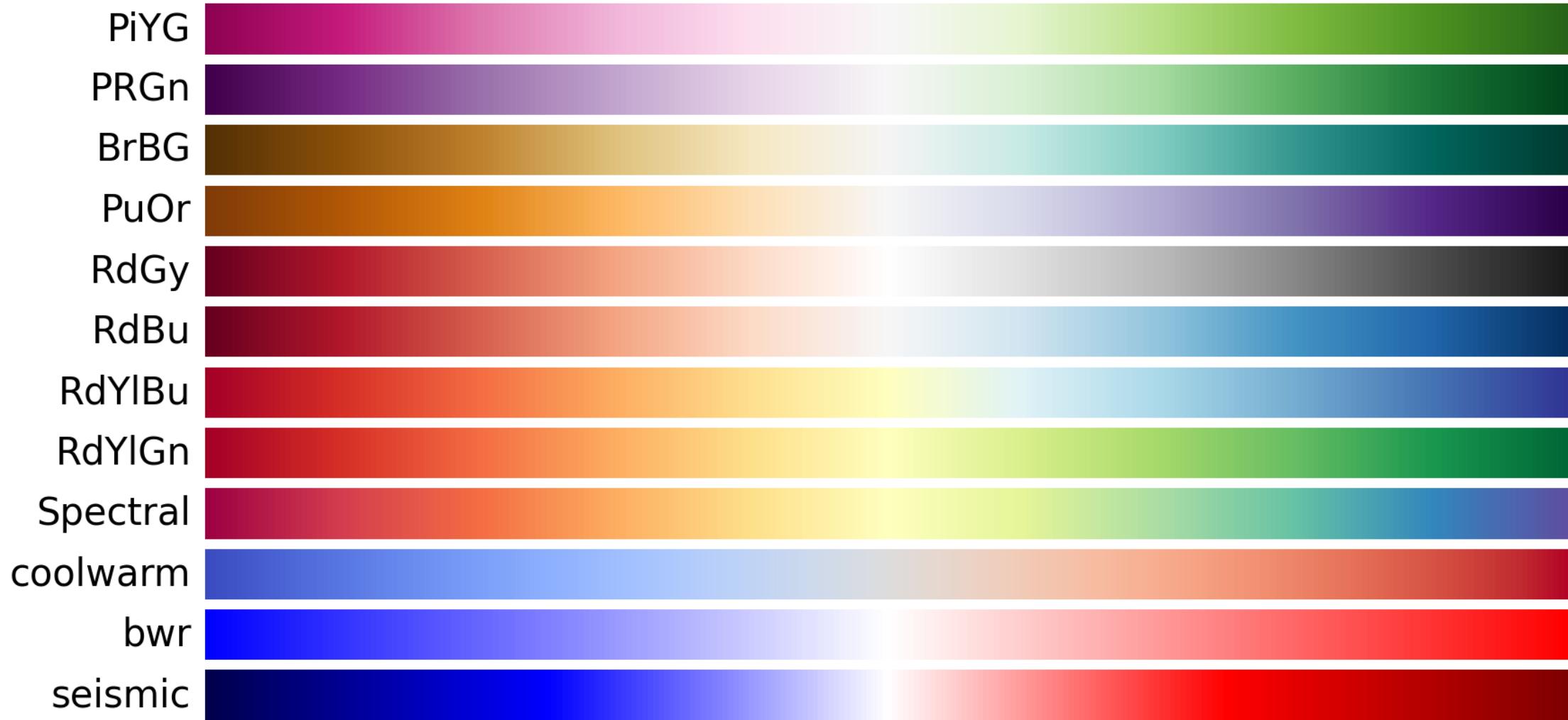
Colour palettes



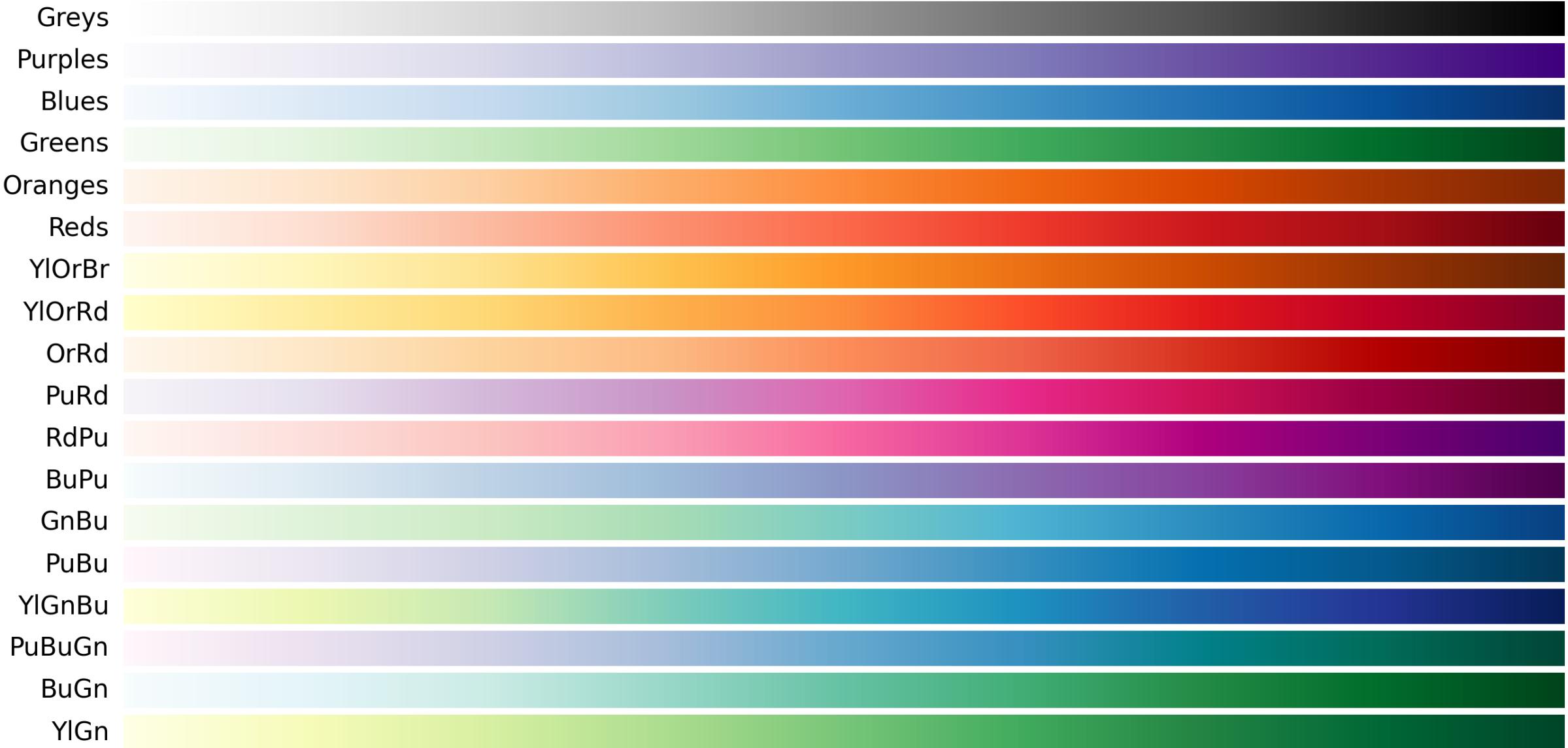
Colour palettes

- **Diverging palettes:** for positive and negative values; centered at 0 (white)
- **Sequential palettes:** smooth gradient from one colour (often white or black, corresponding to zero) to another
- **Qualitative palettes:** discrete colours (no gradients)

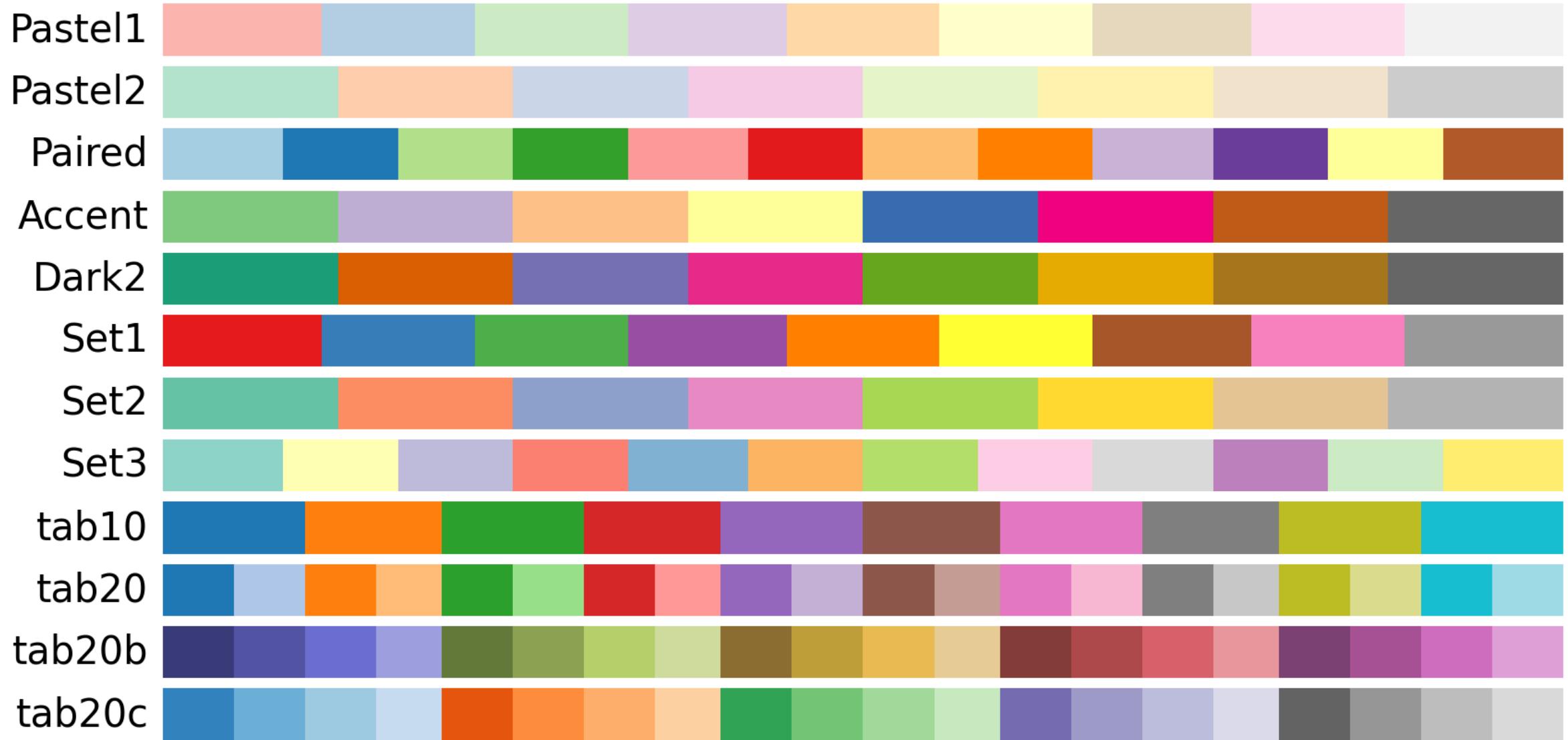
Colour palettes: diverging



Colour palettes: sequential



Colour palettes: qualitative



Darker, more saturated lines

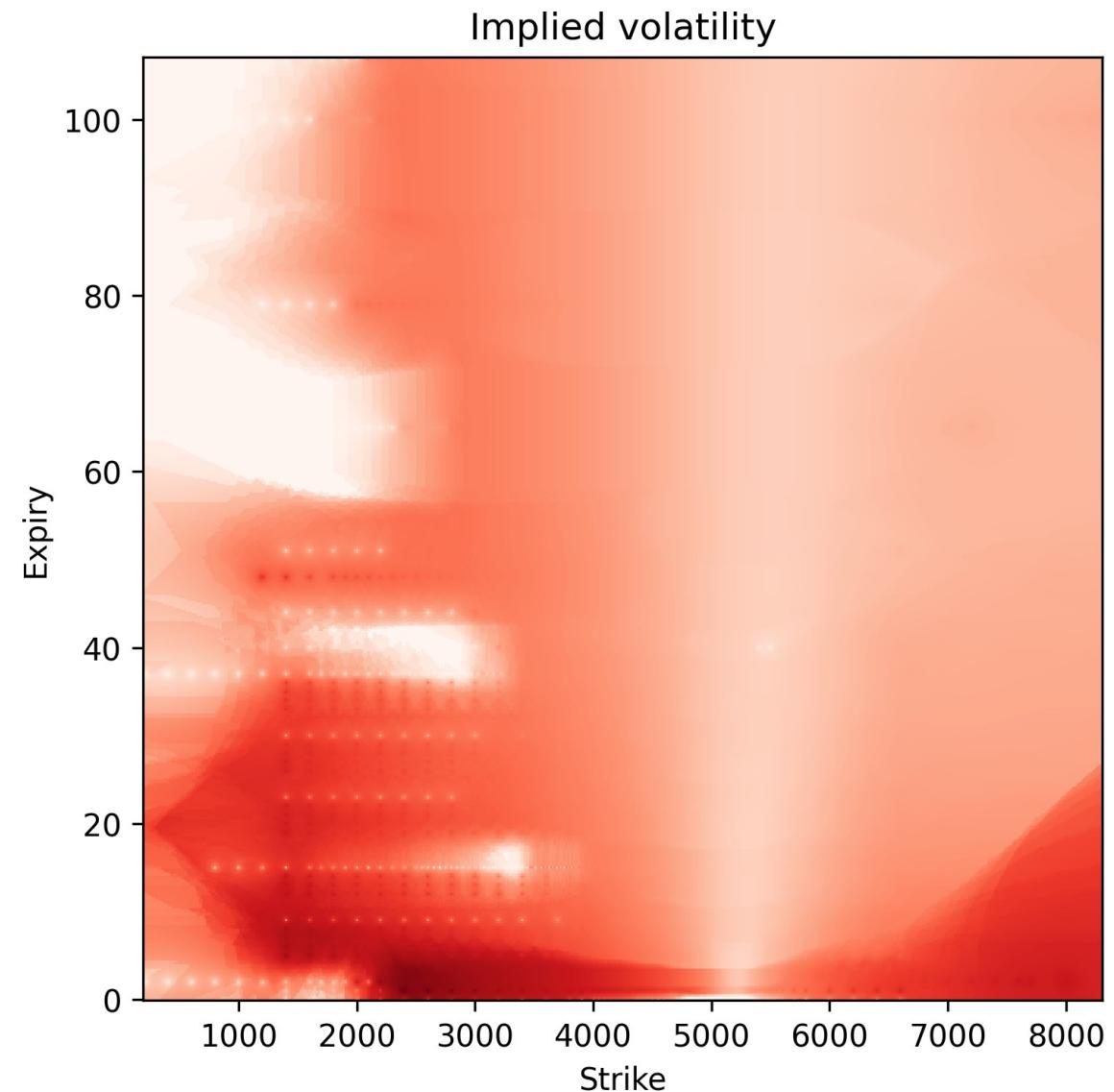
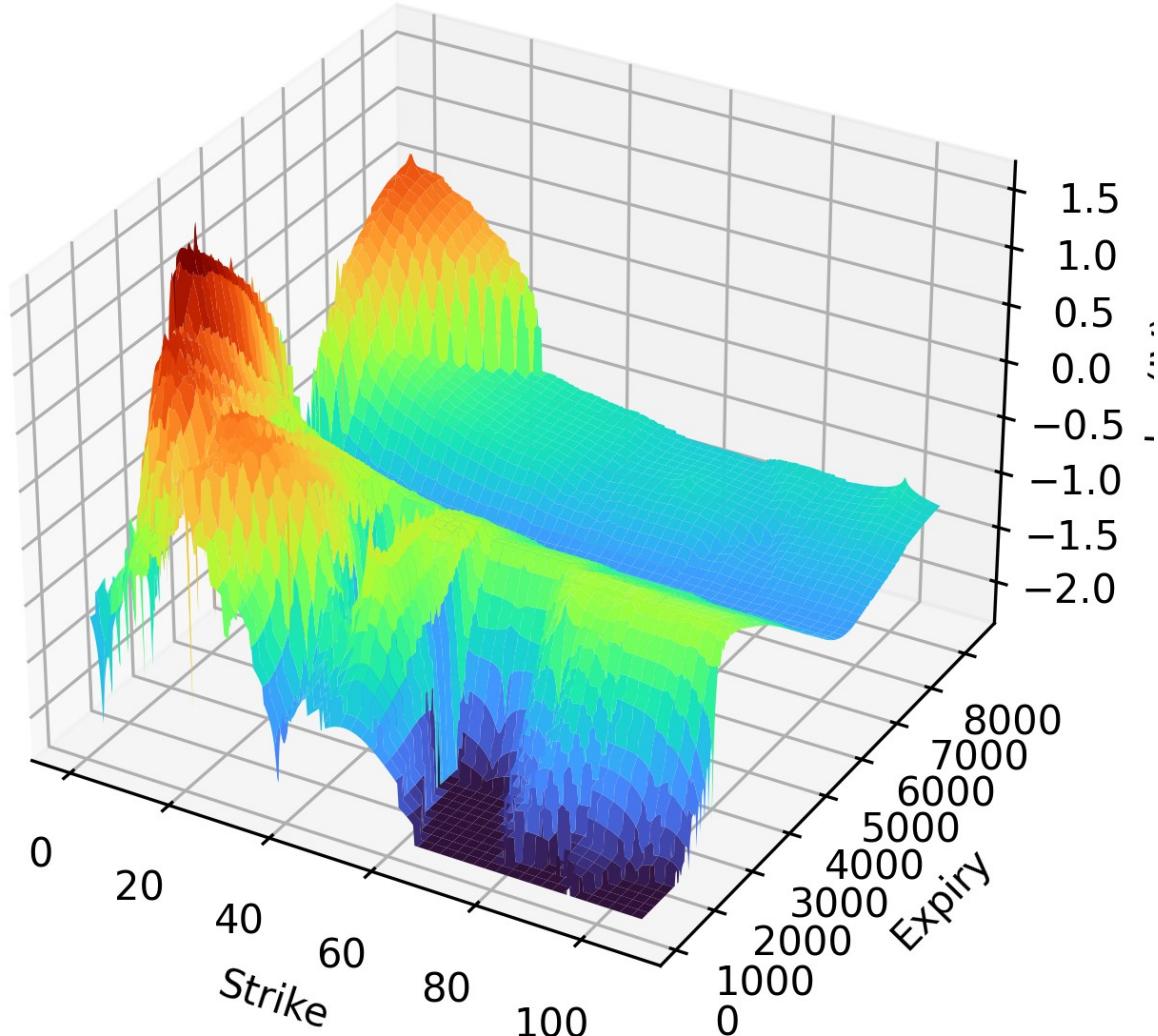
Numeric Precision

Plots to avoid

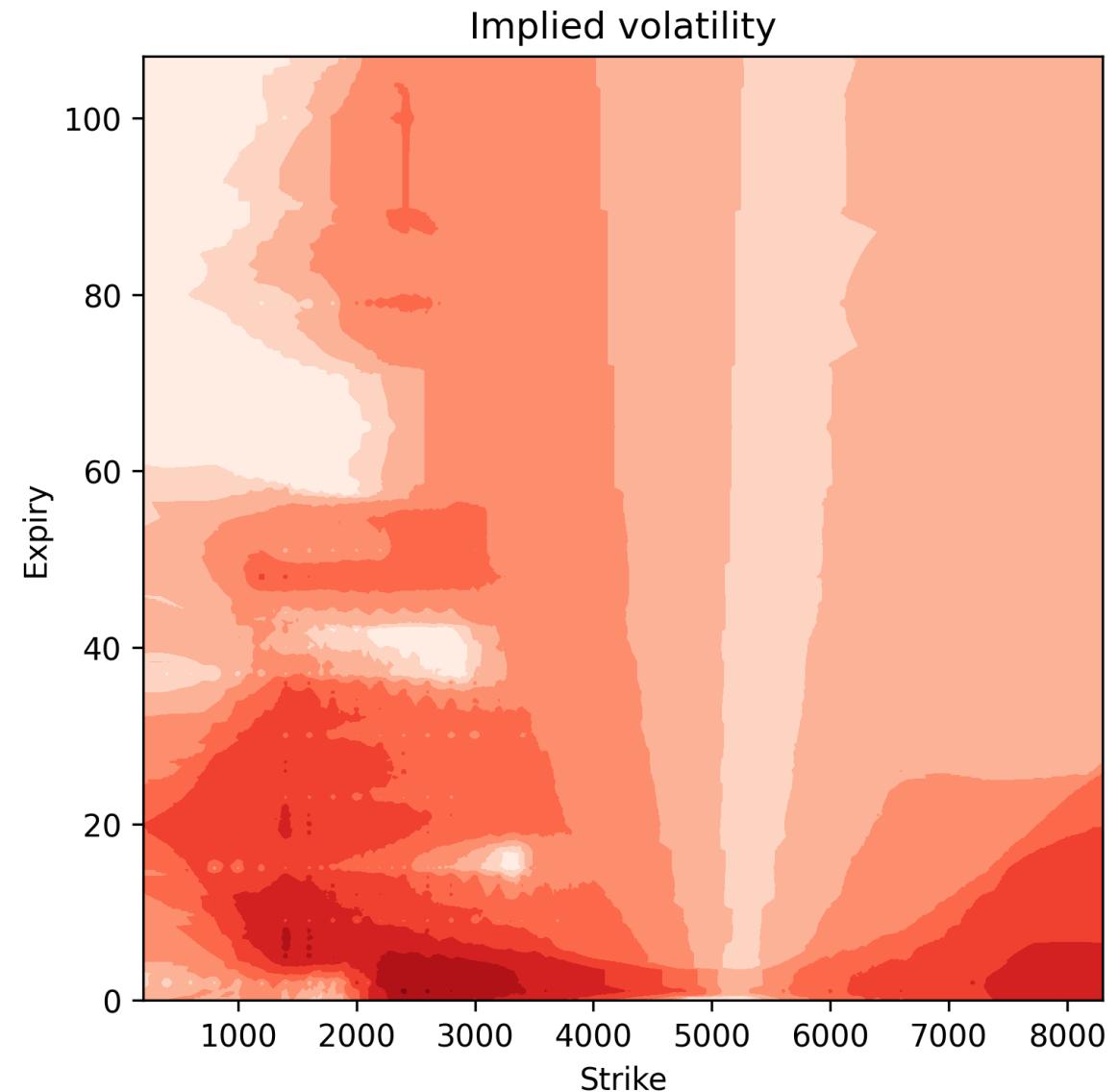
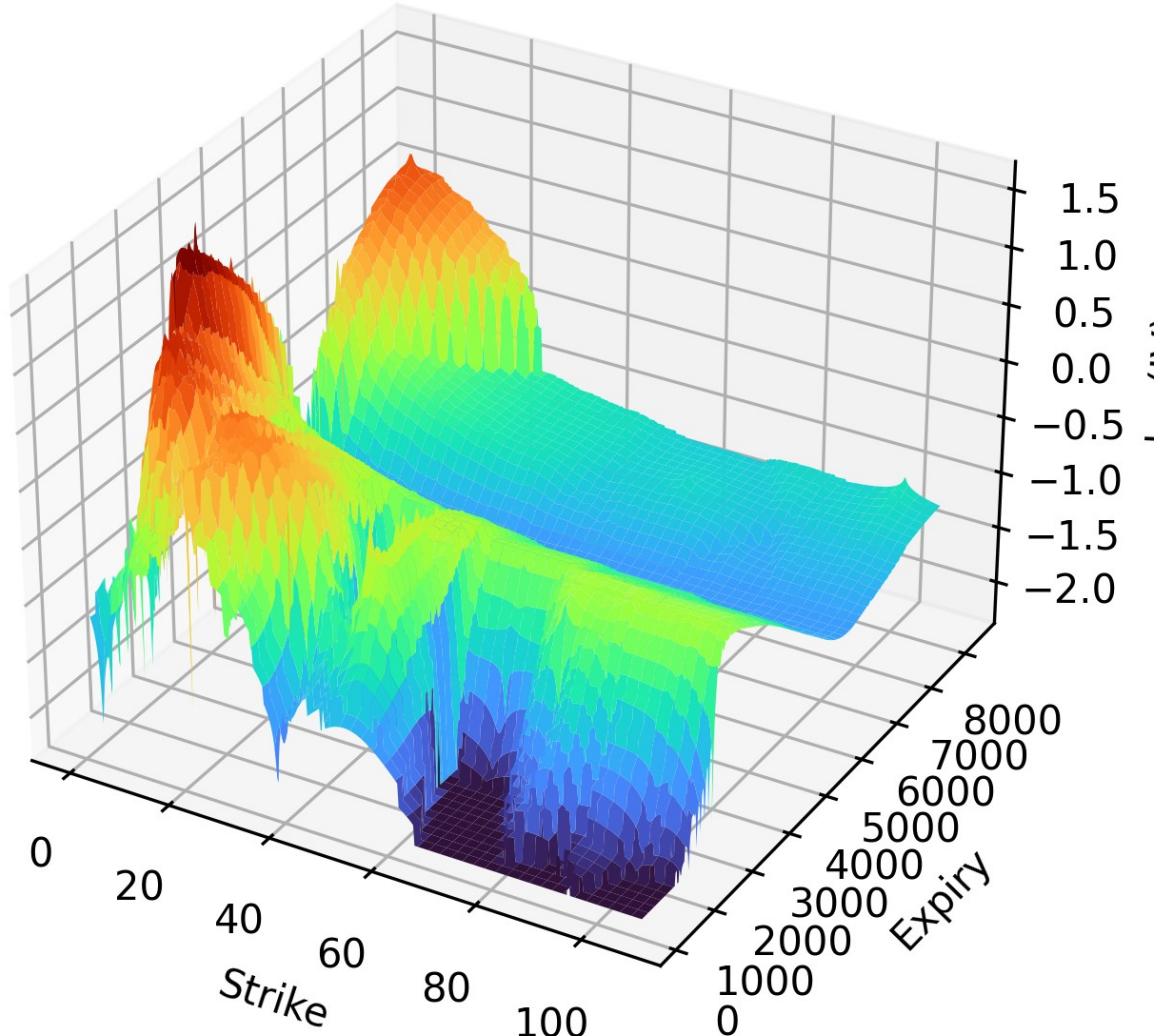
Plots to avoid

- 3D
- Piecharts
- Excel charts (when they can be replaced by a linechart)

Avoid 3D plots

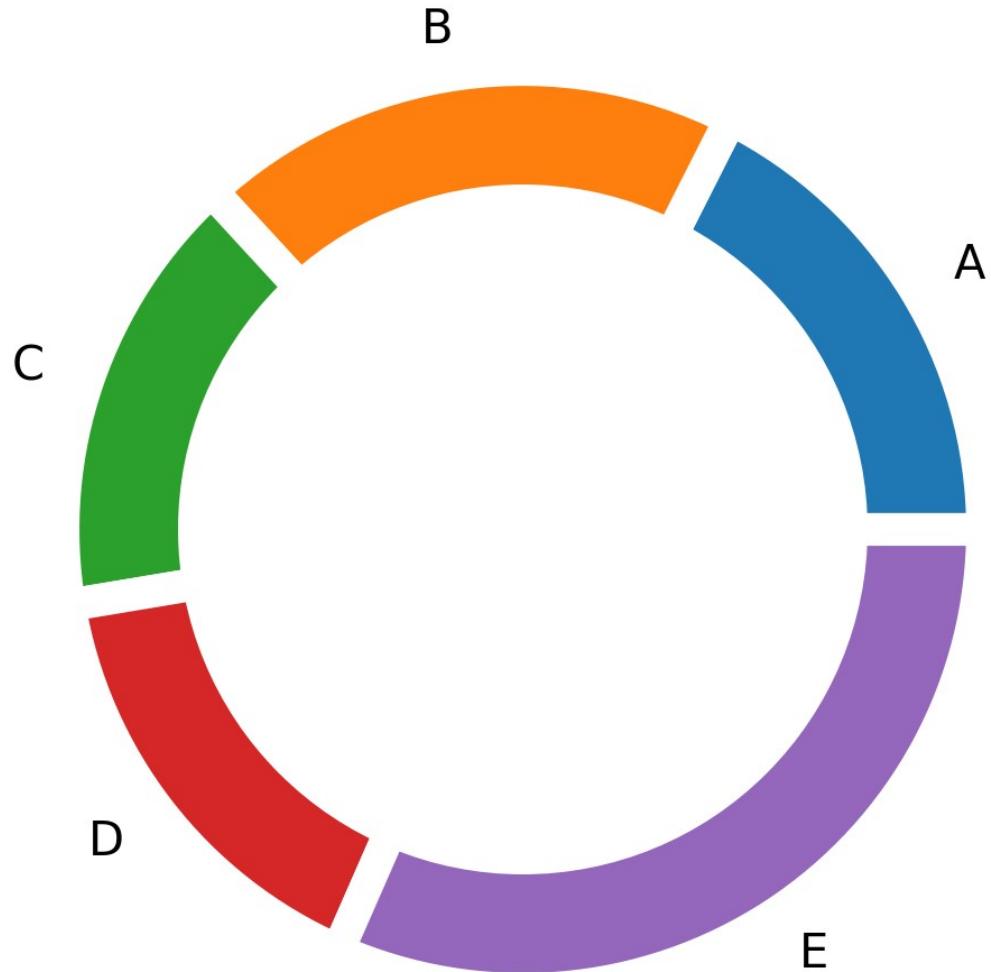
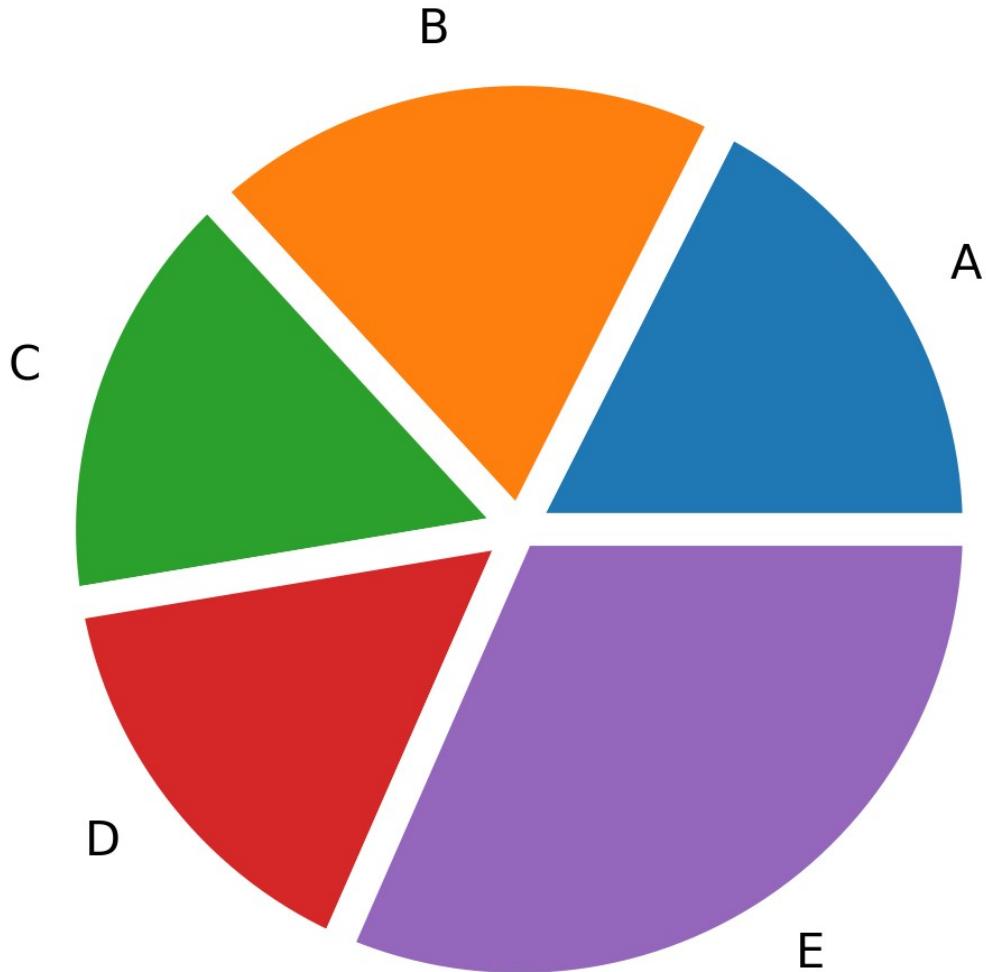


Avoid 3D plots

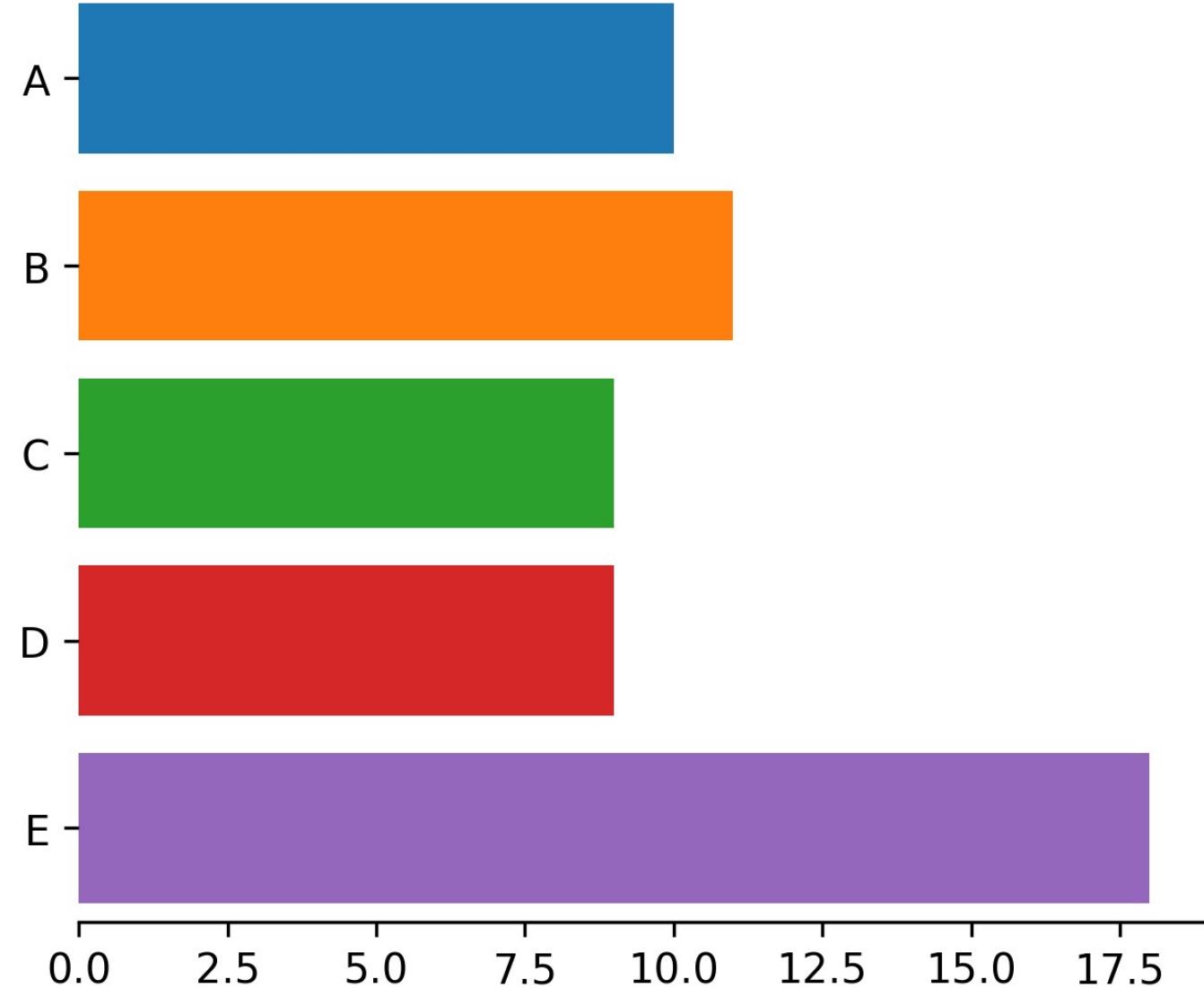
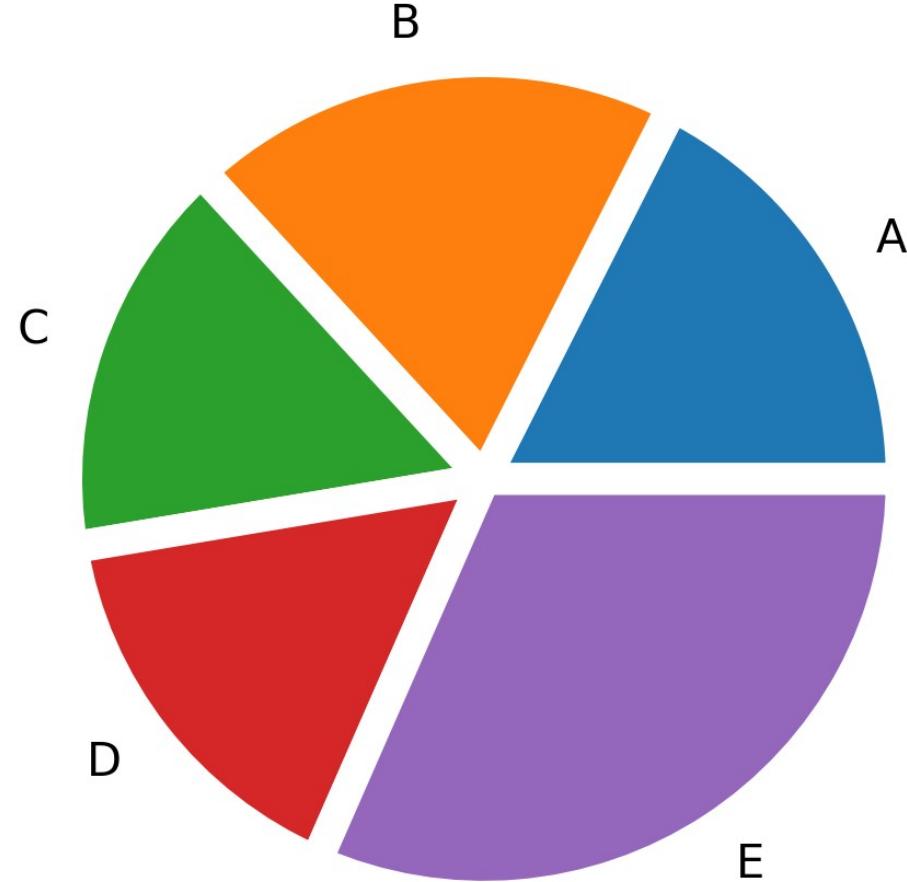


Avoid 3D plots

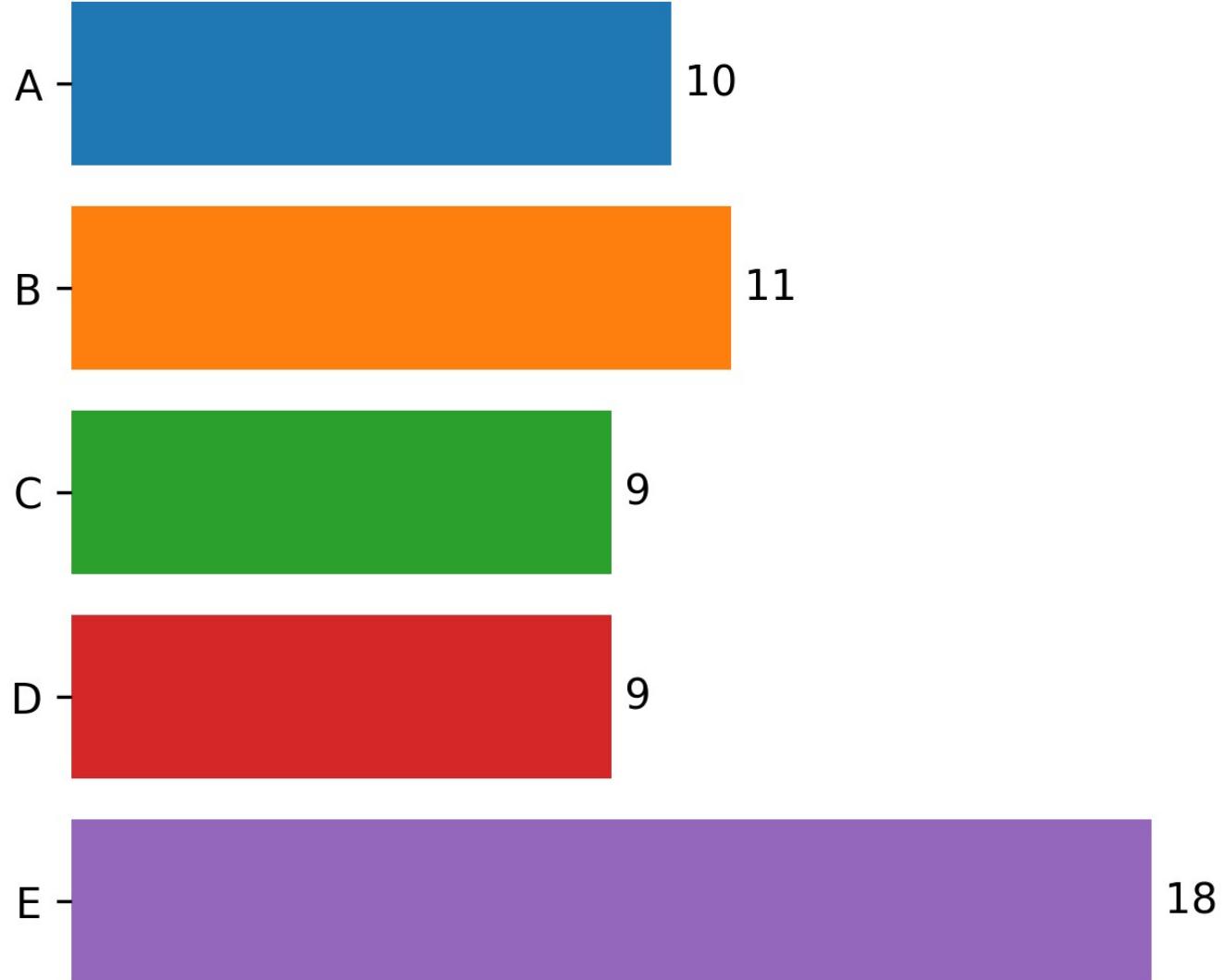
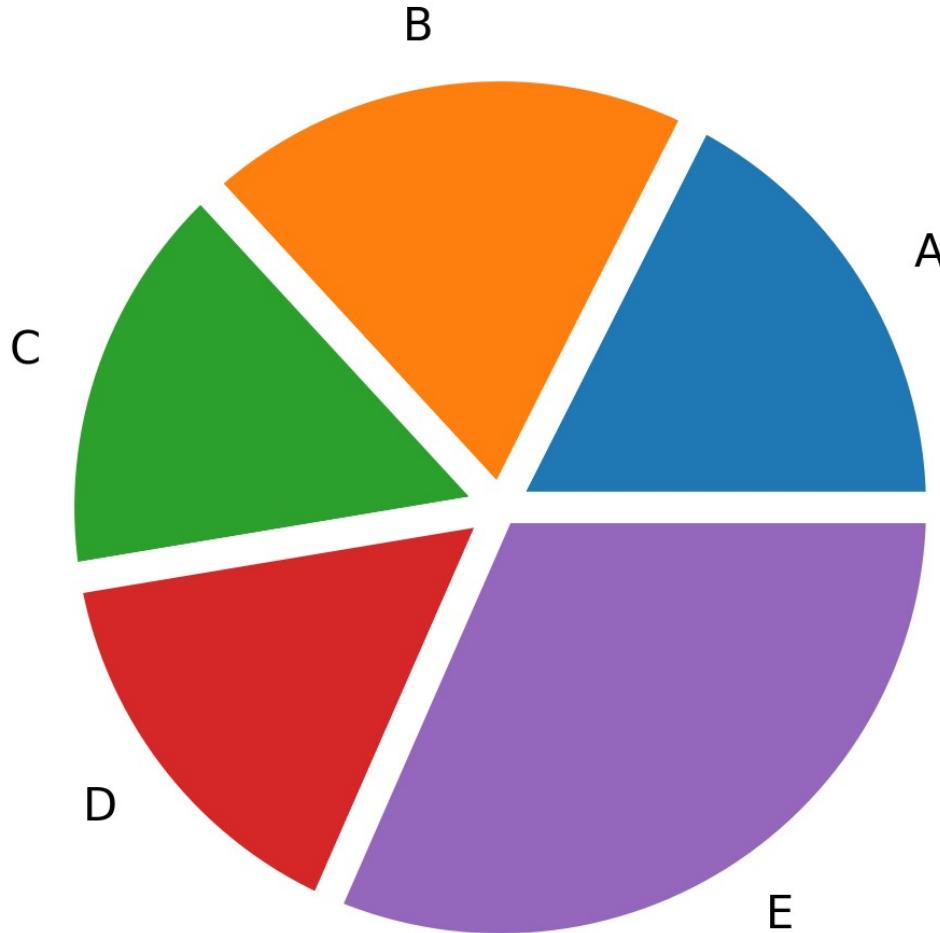
Avoid pie charts



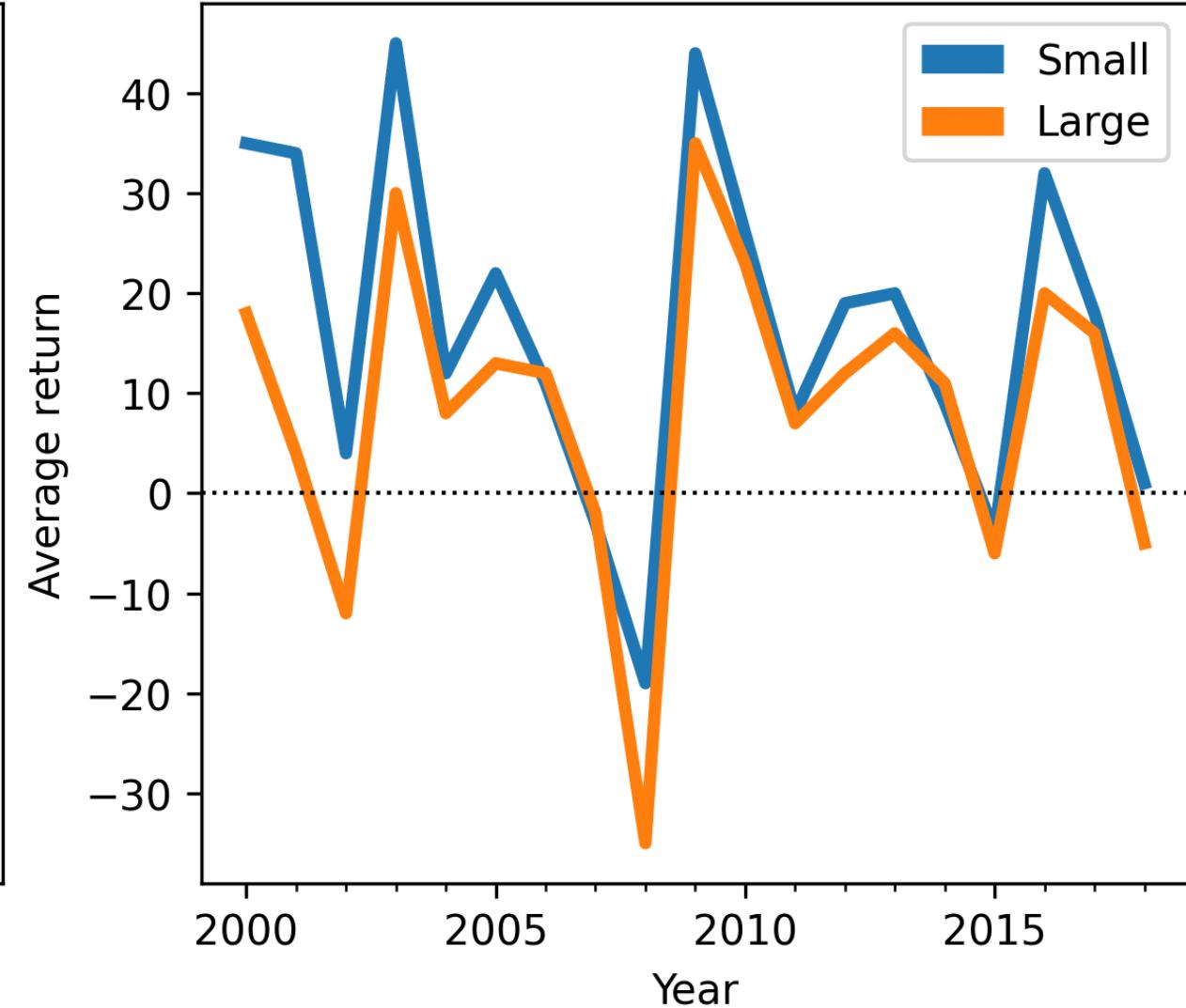
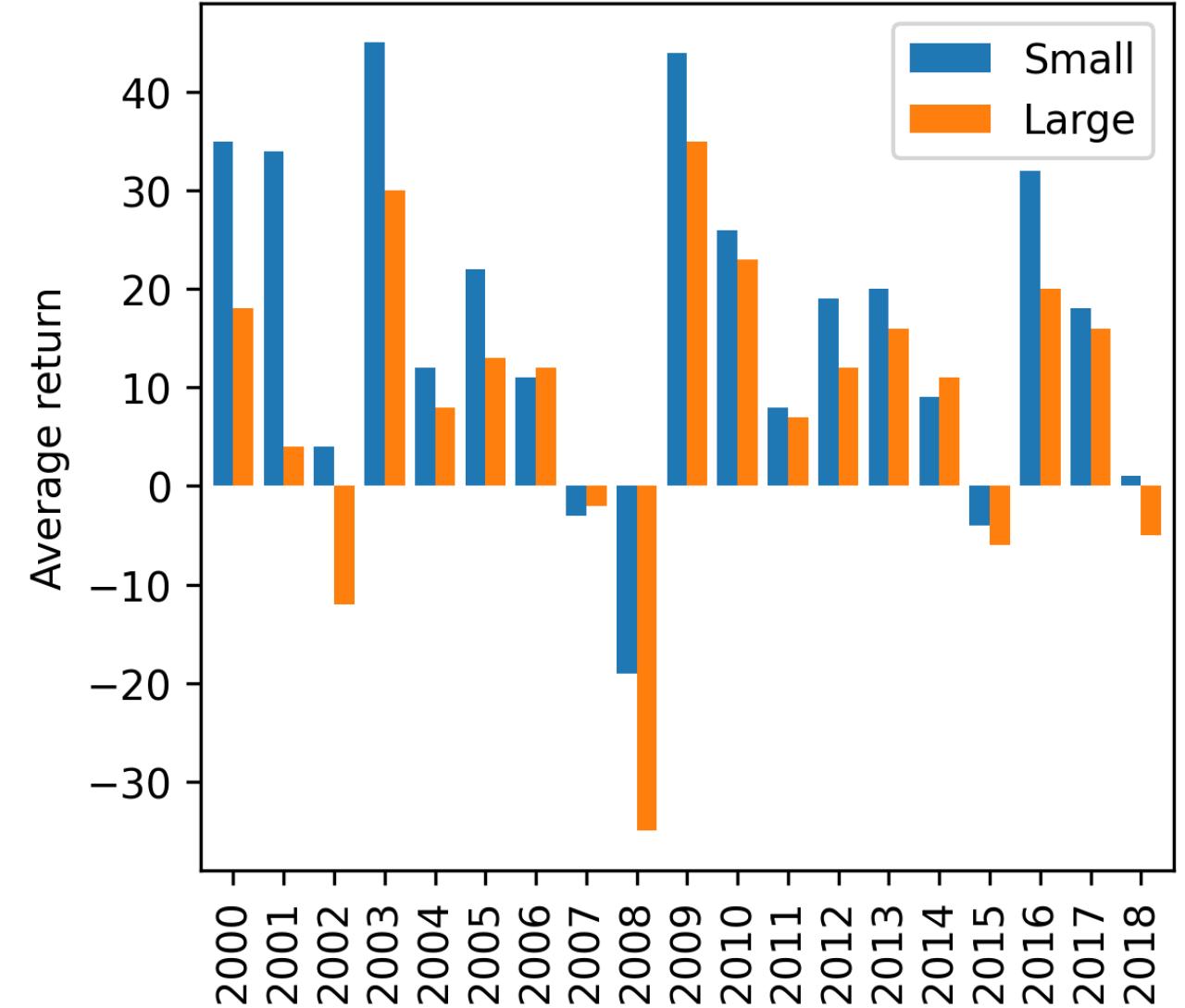
Avoid pie charts



Avoid pie charts



Avoid “Excel plots”

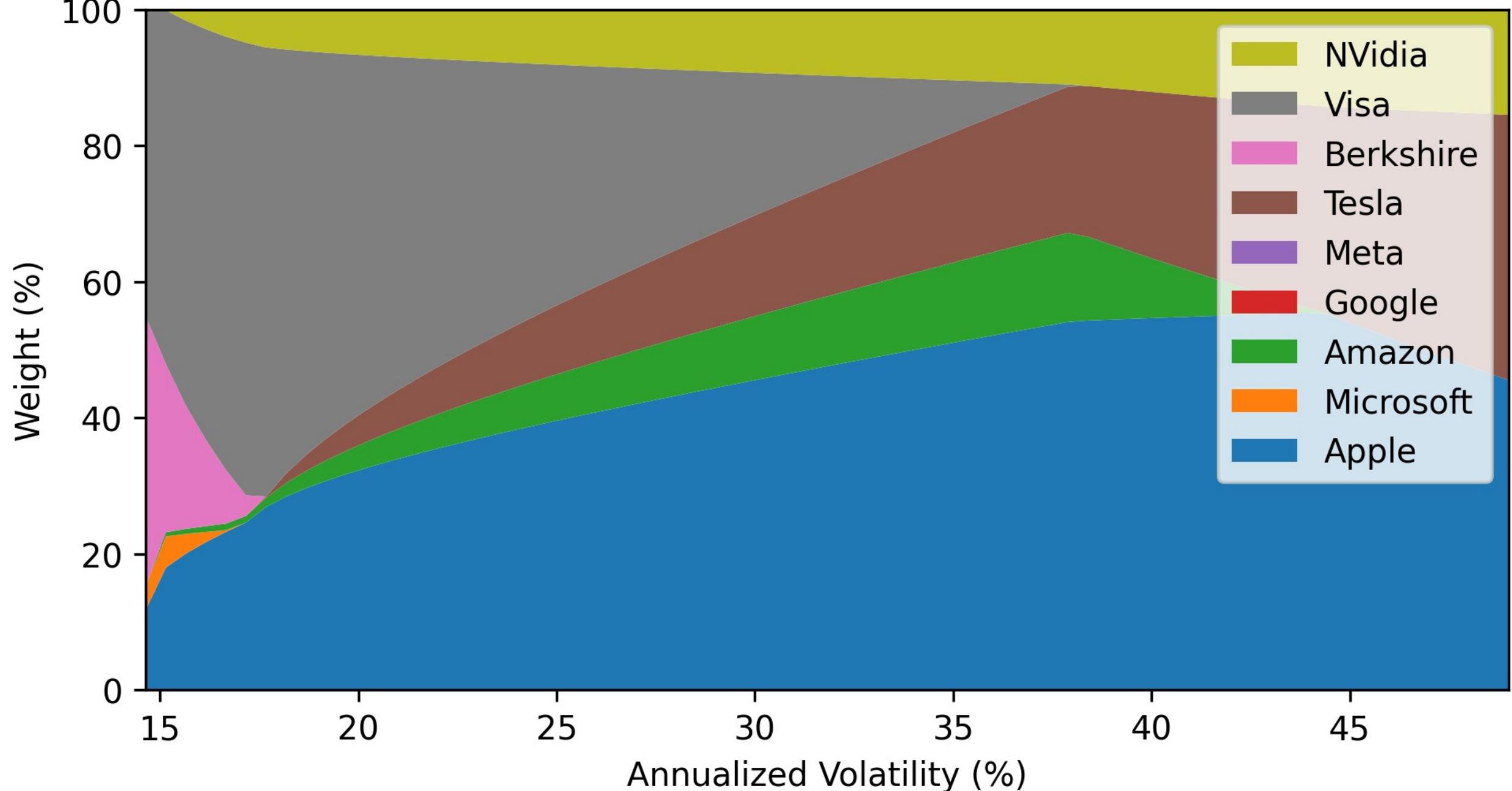


Colour theory

Value, Hue, Saturation

Out-of-scope

Optimal portfolios



Mixed Bivariate Data

Numbers

- Univariate numbers
- Univariate numbers within each group
- Polychoric correlation

Mixed bivariate data

- Boxplot, violinplot, (jittered) scatterplot (overlaid with a boxplot)
- Overlapping densities, ridgelines
- Cleveland plot
- Groupby (univariate metrics)

Bivariate Qualitative Data

clarity color

SI2 E

SI1 E

VS1 E

VS2 I

SI2 J

VVS2 J

VVS1 I

SI1 H

VS2 E

VS1 H

SI1 J

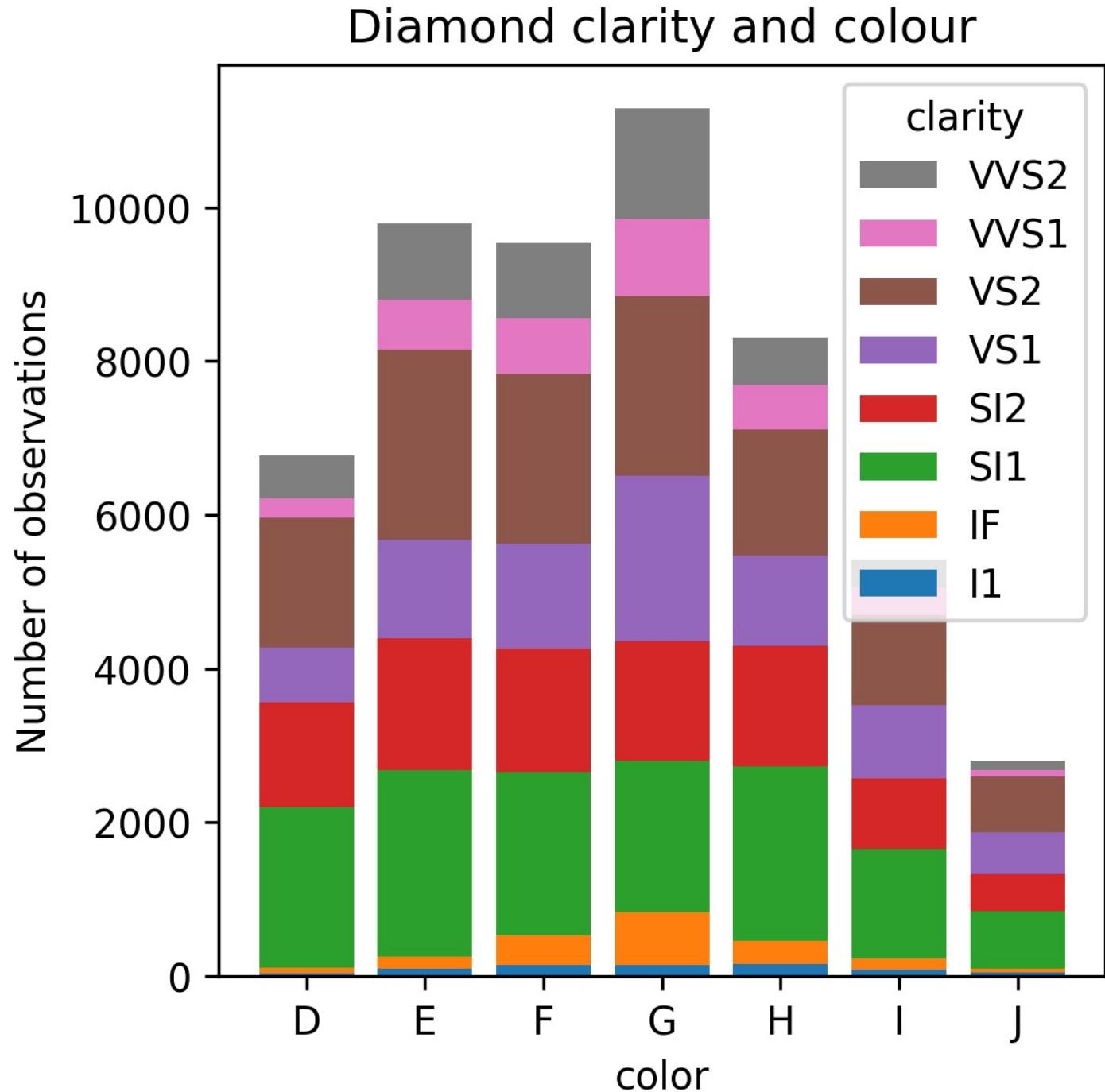
VS1 J

SI1 F

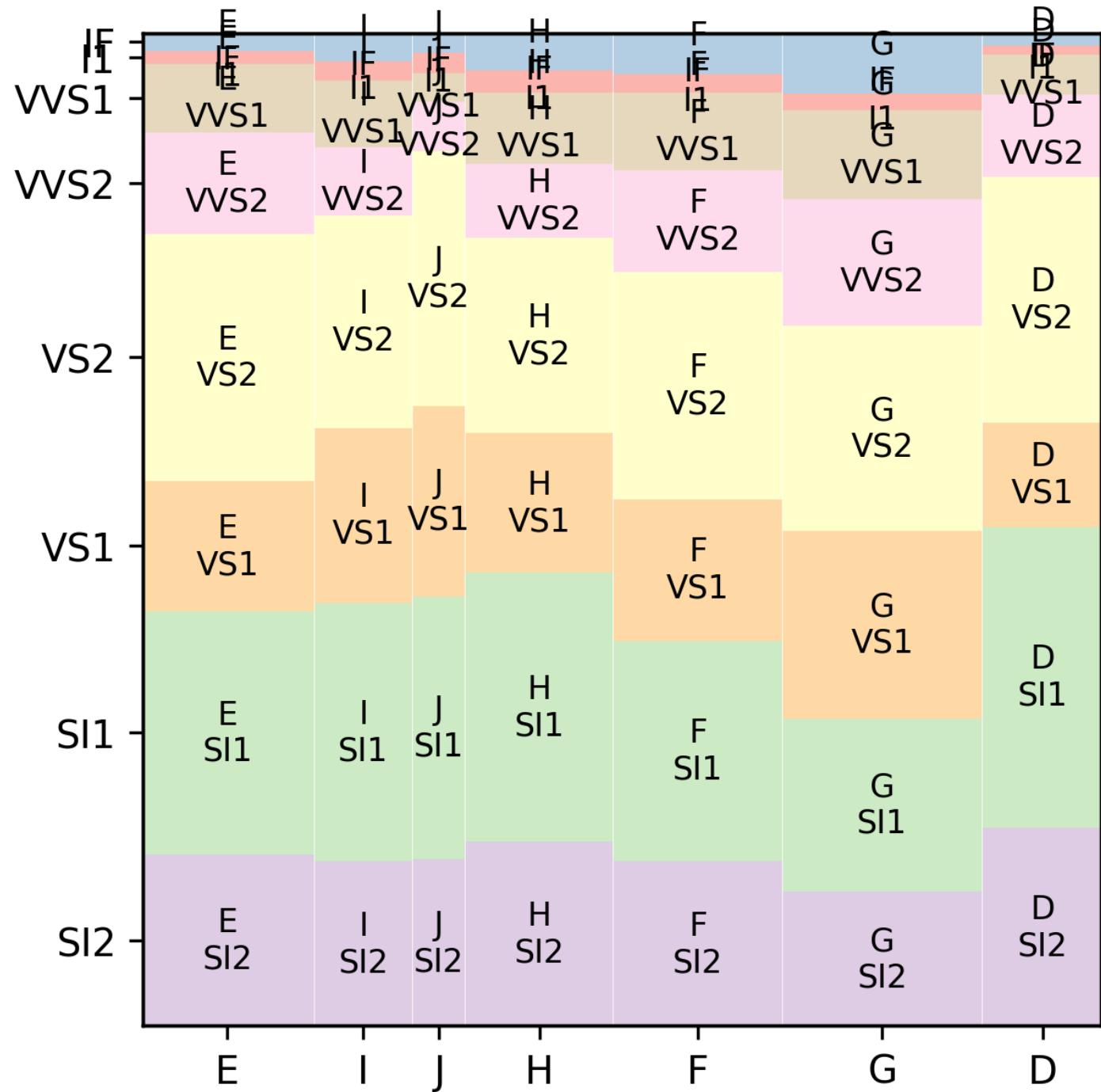

```

c = data("diamonds")[['color','clarity']].copy()
c['count'] = 1
c = (
    c.groupby(['color','clarity']).sum().reset_index()
    .pivot(
        index = 'color',
        columns = 'clarity',
        values = 'count',
    )
)
fig, ax = plt.subplots( layout = 'constrained' )
s = 0 * c.iloc[:,0]
S = c.T.sum()
for u in c.columns:
    ax.bar( c.index, c[u]/S, bottom = s, label = u )
    s += c[u] / S
ax.legend( reverse = True, title = c.columns.name )
ax.set_xlabel( c.index.name )
ax.set_ylabel( "Number of observations" )
plt.show()

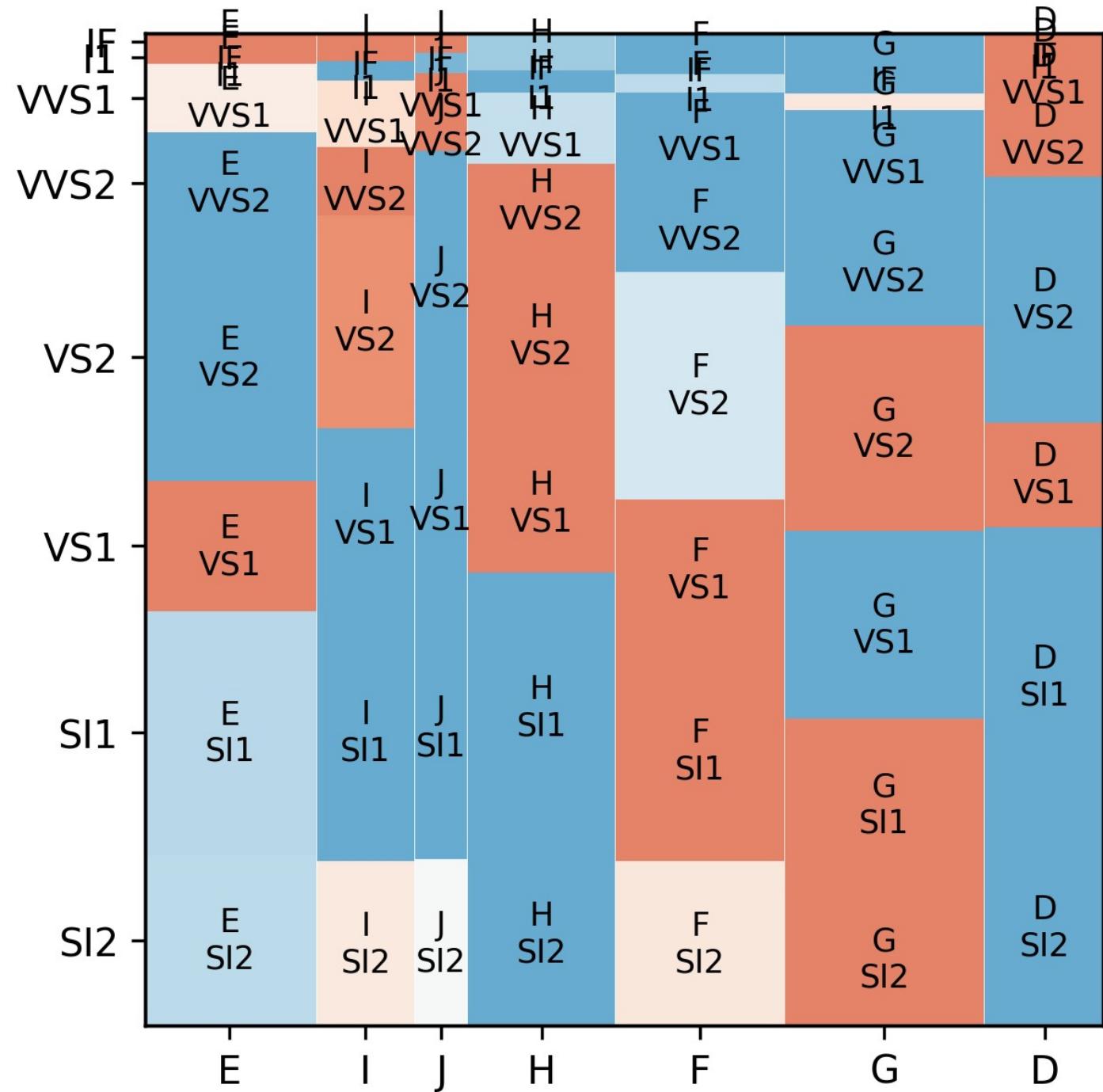
```



```
statsmodels.graphics.mosaicplot.mosaic(  
    d,  
    list( d.columns ),  
)
```



```
statsmodels.graphics.mosaicplot.mosaic(  
    d,  
    list( d.columns ),  
    statistic = True,  
)
```



Panel Data

Panel data

- 3 variables: time, identifier (patient, stock, etc.), value
- Can also be represented as a 2-dimensional array of values, with one row per timestamp, one column per identifier (or the opposite)

Panel Data

	date	stock	price
	2022-08-31	Apple	155.89
	2022-09-30	Apple	137.03
	2022-10-31	Apple	152.04
	2022-08-31	Microsoft	257.97
	2022-09-30	Microsoft	229.78
	2022-10-31	Microsoft	229.02
	2022-08-31	Amazon	126.77
	2022-09-30	Amazon	113.00
	2022-10-31	Amazon	102.44
	2022-08-31	Google	109.15

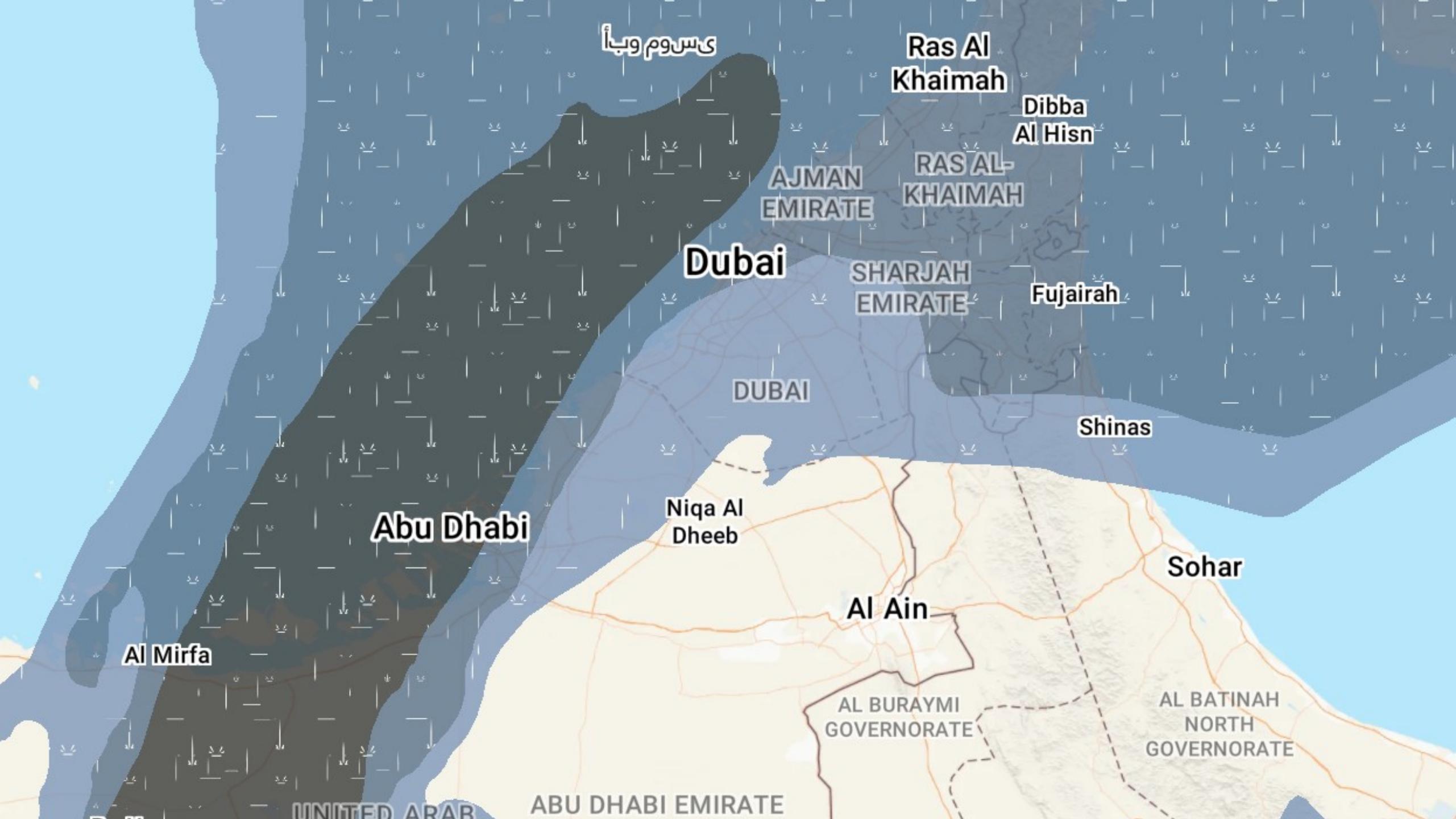
Panel Data

	Apple	Microsoft	Amazon	Google	Meta	Tesla	Berkshire	Visa	NVidia
2022-08-31	155.89	257.97	126.77	109.15	162.76	275.61	280.80	196.34	150.78
2022-09-30	137.03	229.78	113.00	96.15	135.54	265.25	267.02	175.53	121.30
2022-10-31	152.04	229.02	102.44	94.66	93.06	227.54	295.09	204.69	134.87
2022-11-30	147.02	252.43	96.54	101.45	117.97	194.70	318.60	214.89	169.15
2022-12-31	129.04	237.27	84.00	88.73	120.21	123.18	308.90	205.74	146.07
2023-01-31	143.31	245.18	103.13	99.87	148.81	173.22	311.52	227.97	195.27
2023-02-28	146.63	247.39	94.23	90.30	174.75	205.71	305.18	218.23	232.05

Panel Data

- This is outside the scope of this course

Geospatial Data



Geospatial Data

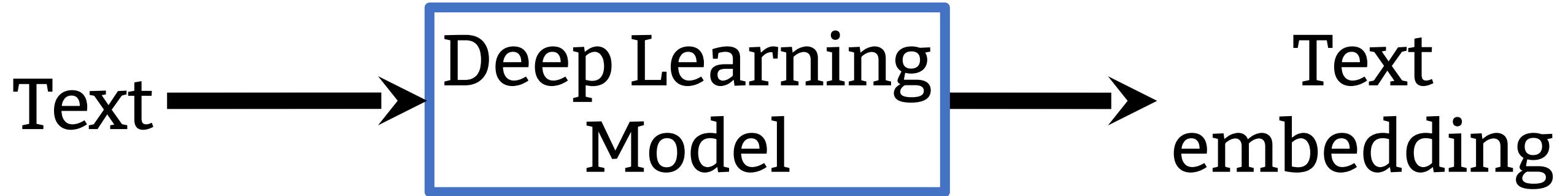
- This is outside the scope of this course

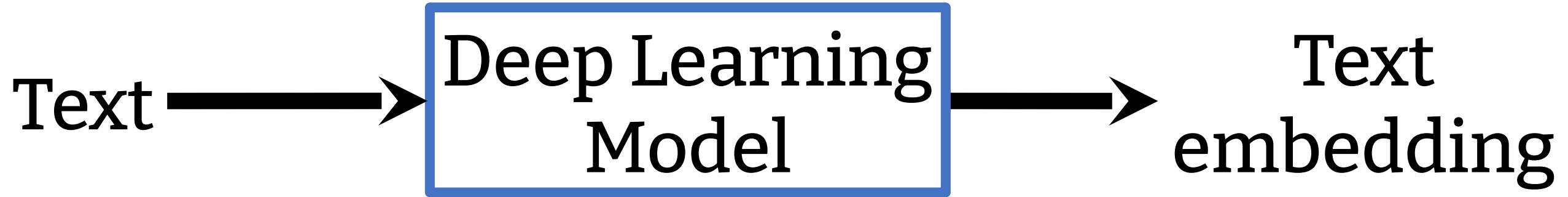
Unstructured Data

Unstructured Data

- Text
- Images
- Audio
- Video
- Graphs
- JSON, XML, etc.
- Etc.

text





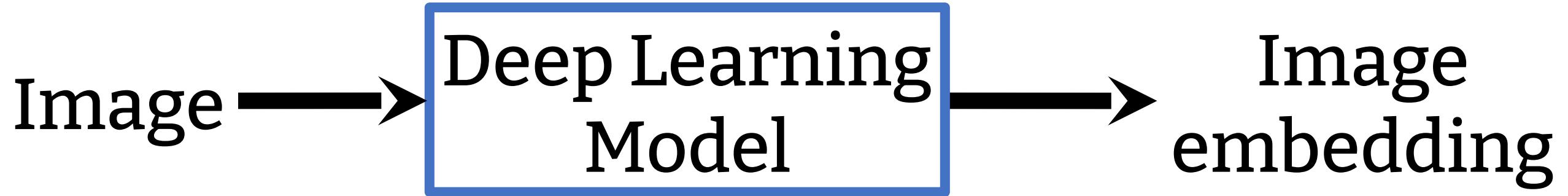
Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversations?"

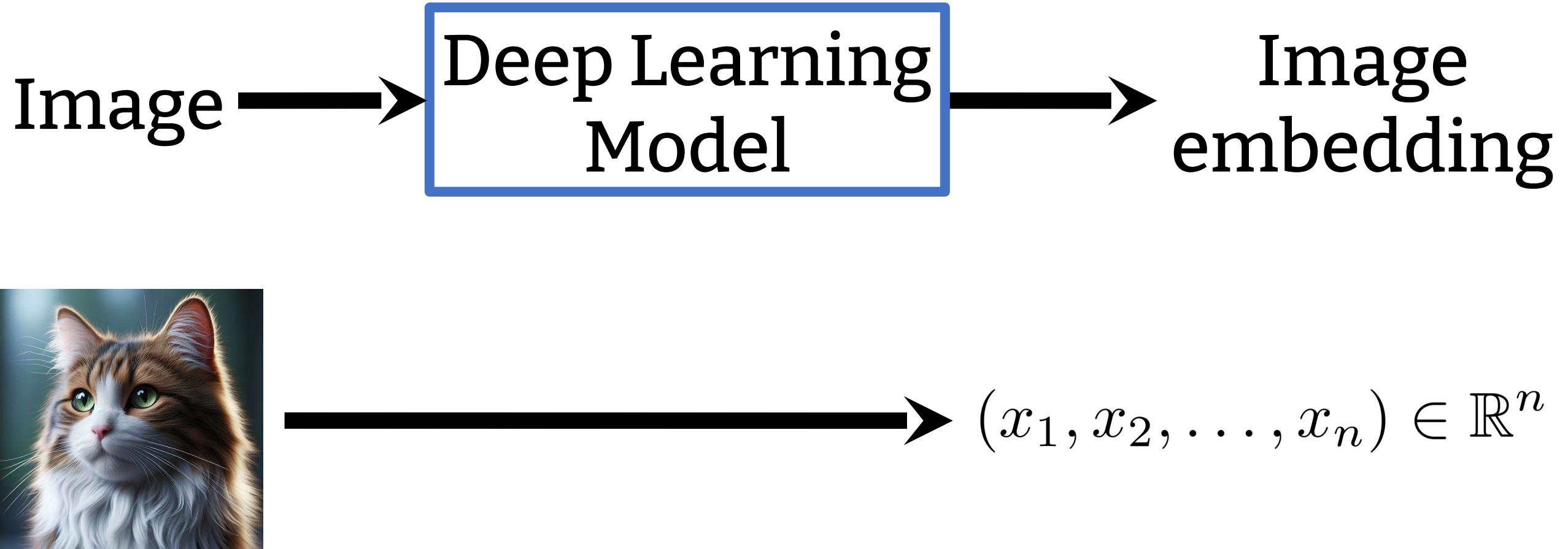
$$(x_1, x_2, \dots, x_n) \in \mathbb{R}^n$$

Text

- This is outside the scope of this course (but we may see a few examples on Friday)

Images

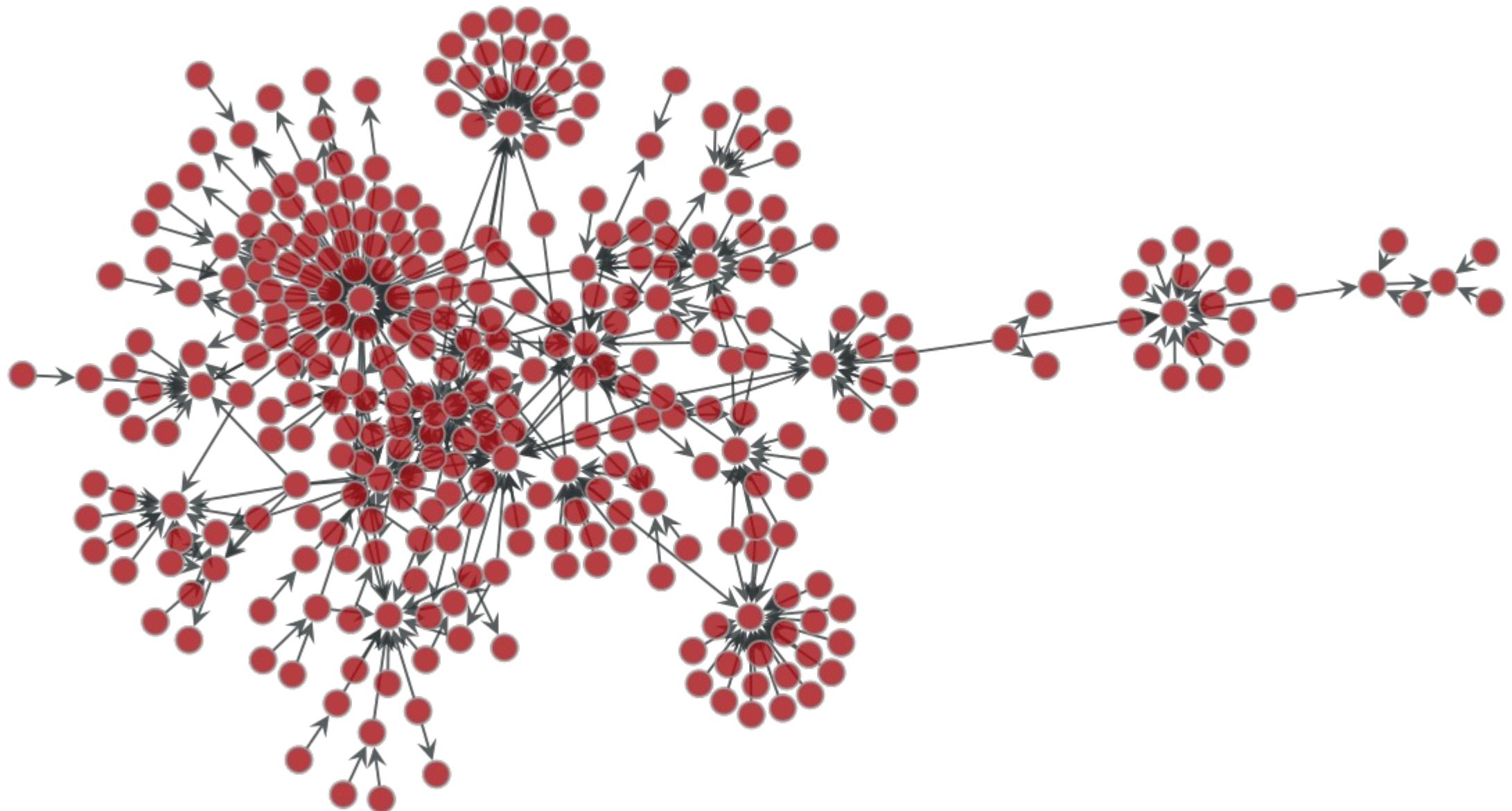


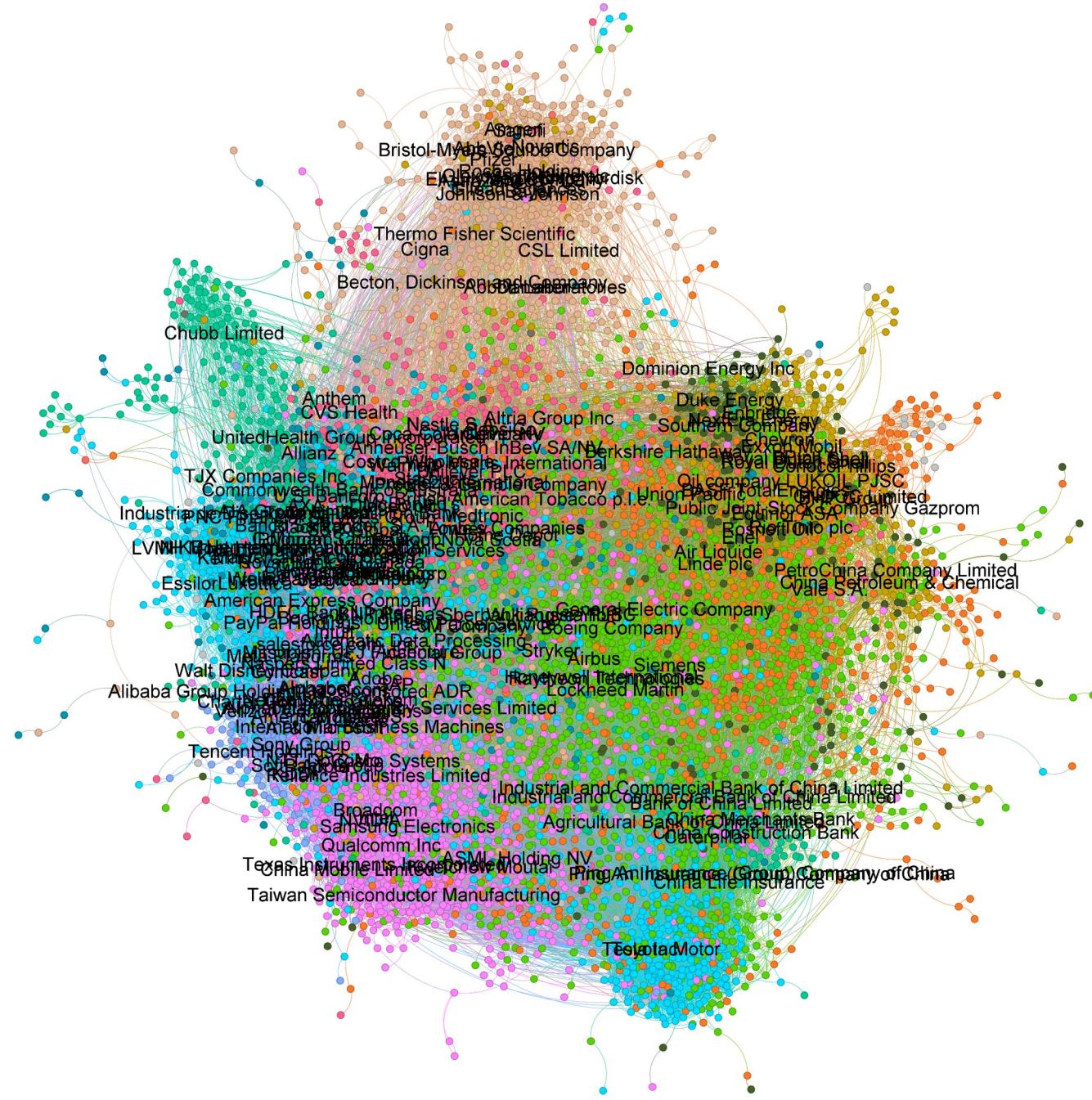


Images

- This is outside the scope of this course (but we may see a few examples on Friday)

Graphs





Graphs

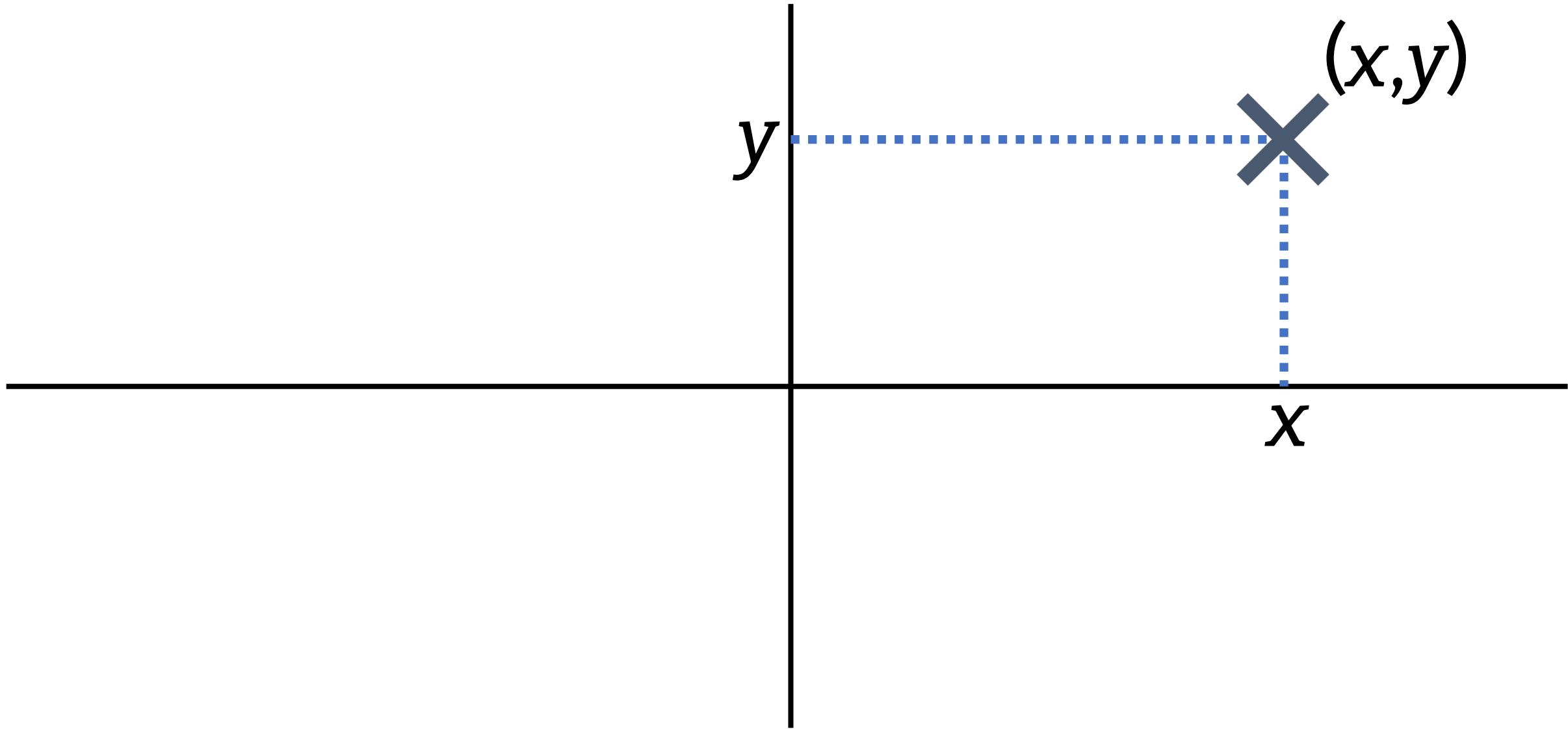
- This is outside of the scope of this course
- Check:
 - Networkx: computations
 - Gephi: plots (for large graphs)
 - <https://networks.skewed.de/>: graph data

Prerequisites

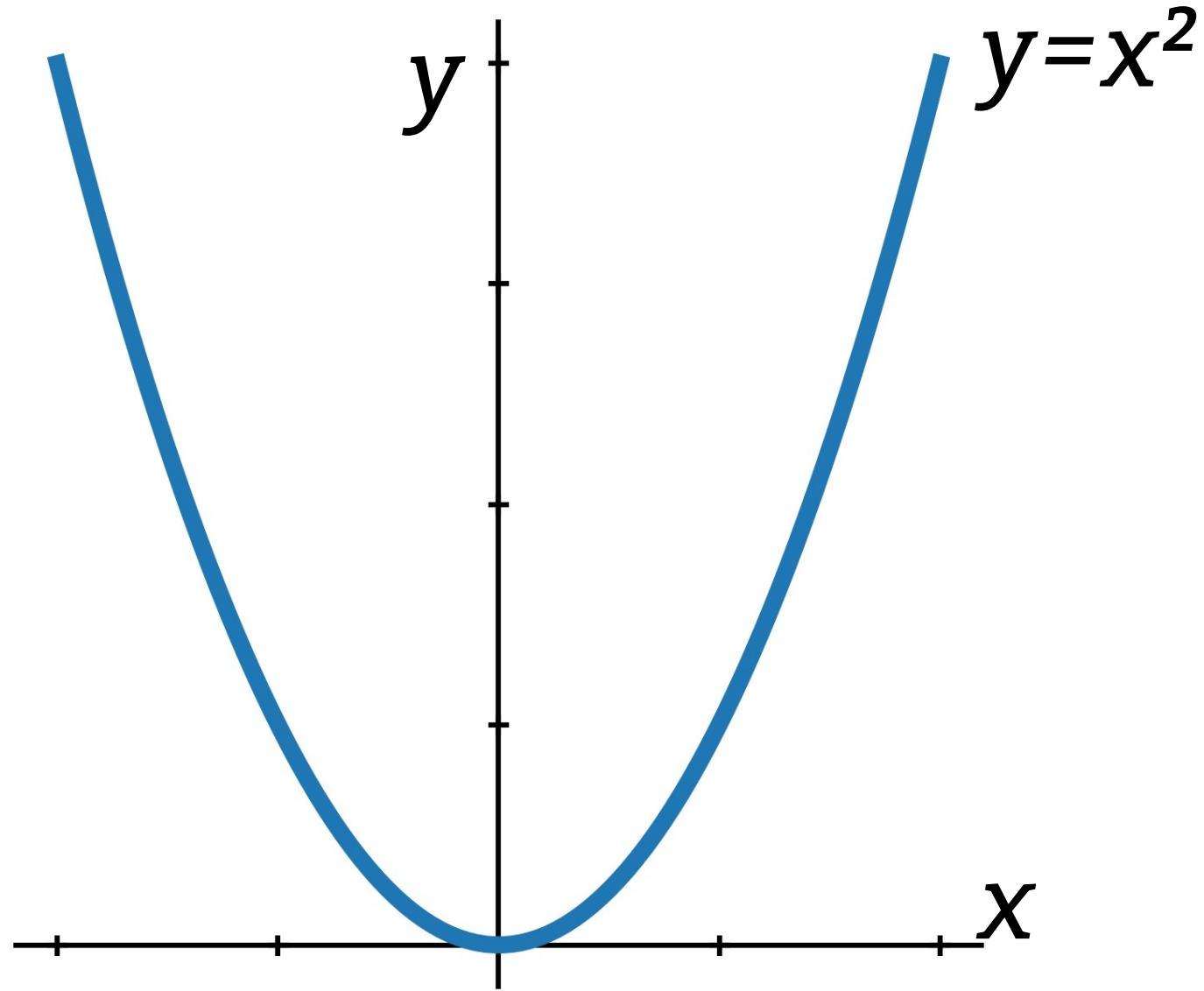
Function

$$f : \left\{ \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & x^2 \end{array} \right.$$

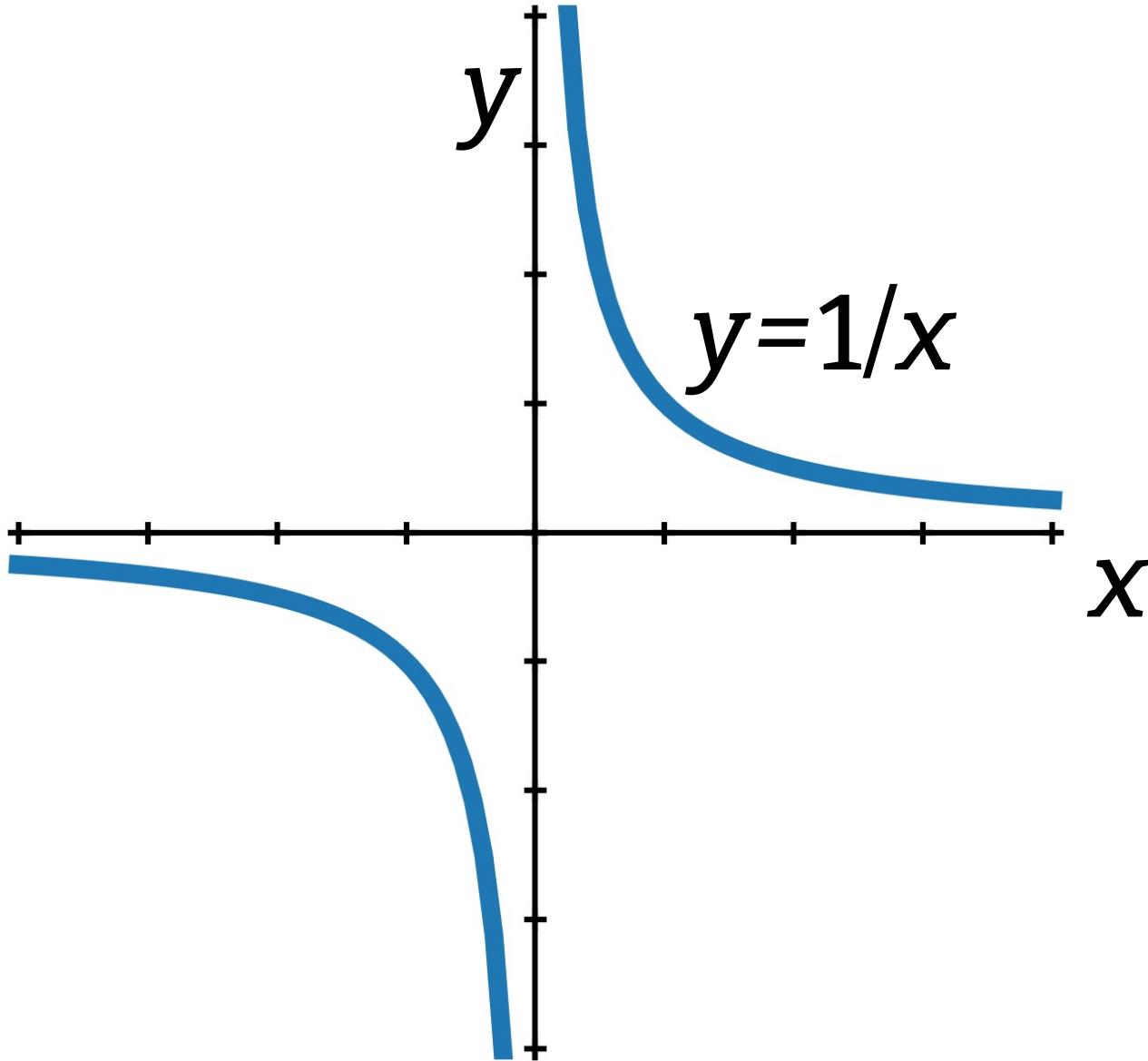
Coordinates



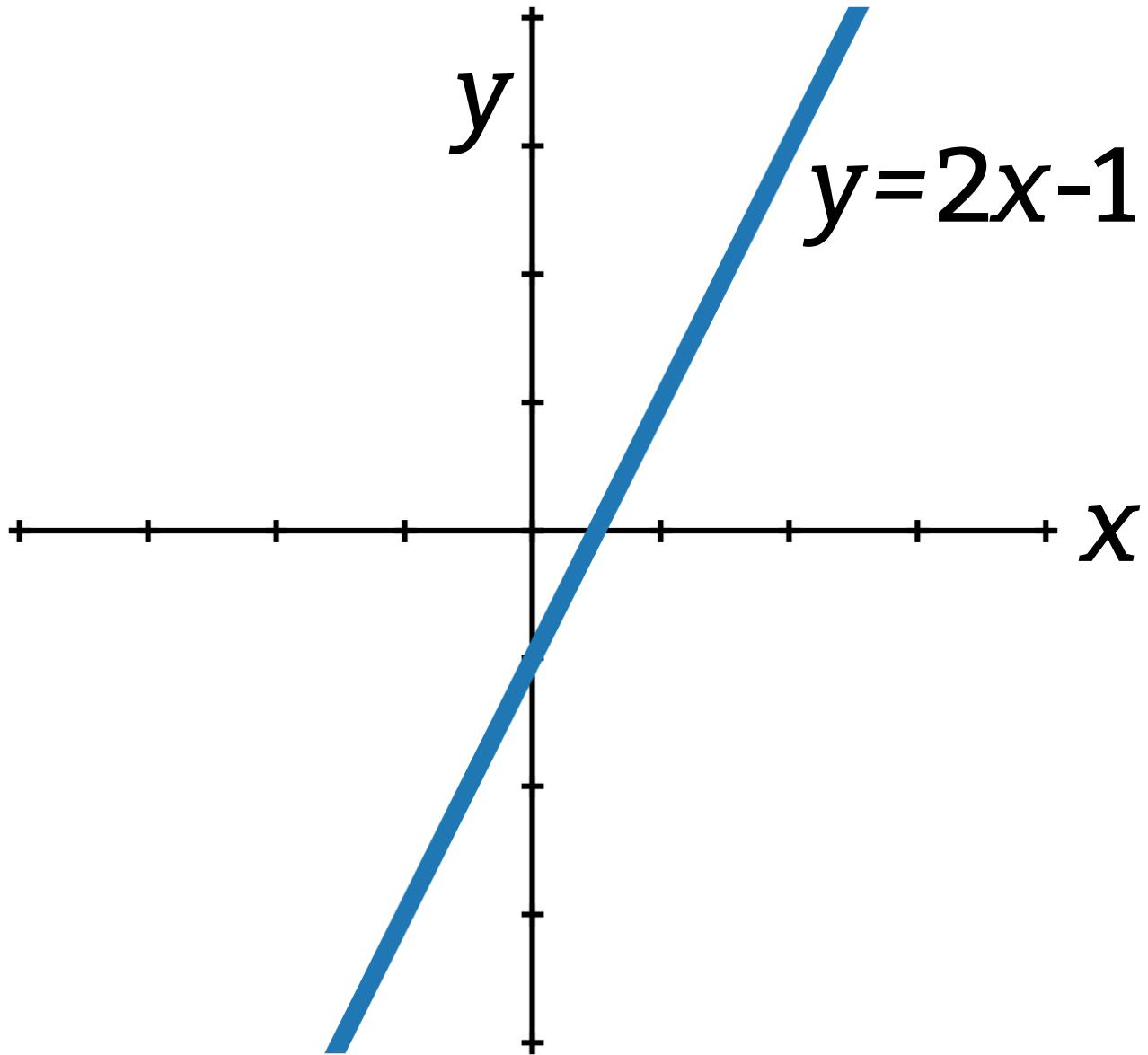
Graph of a function



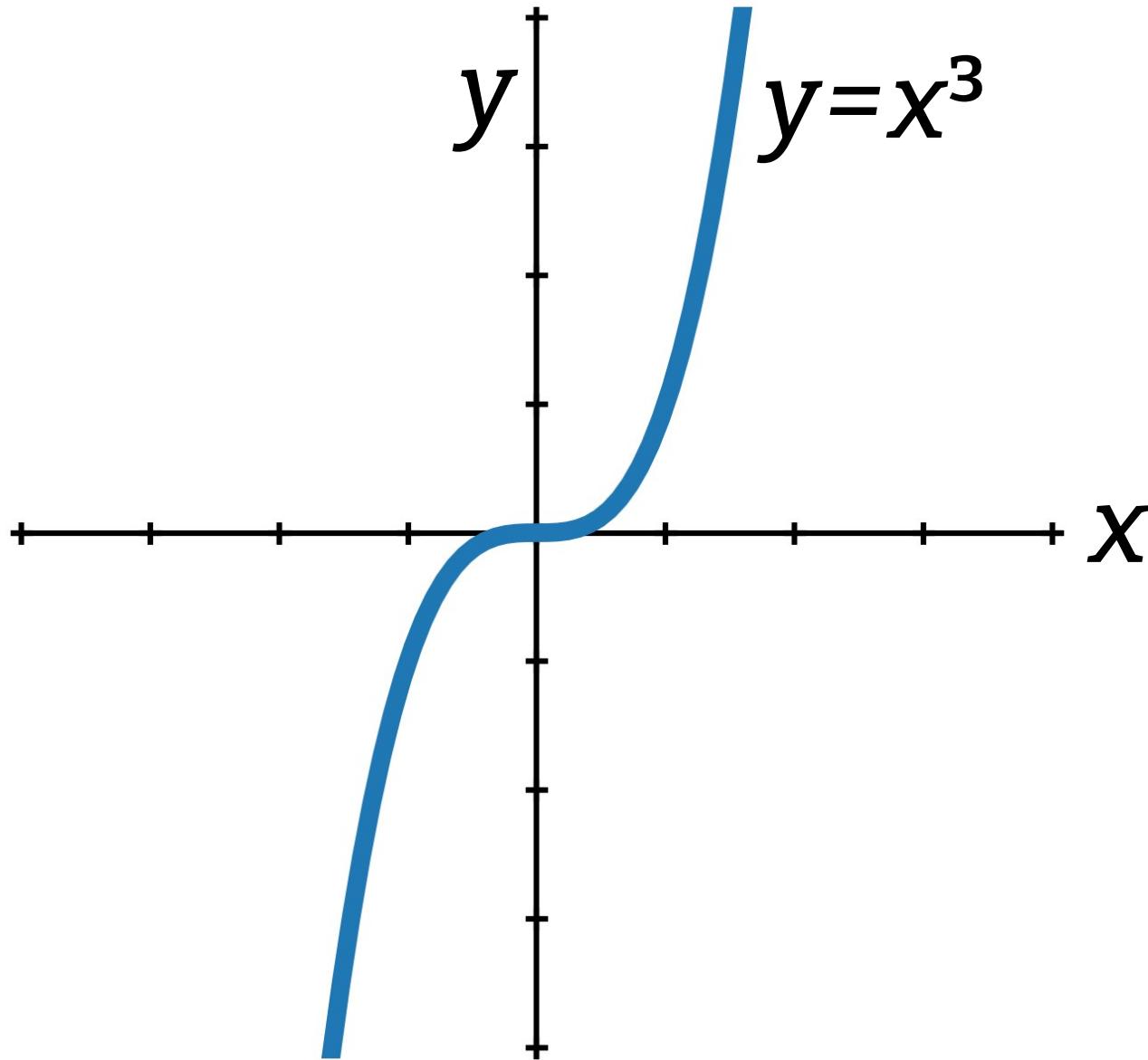
Graph of a function



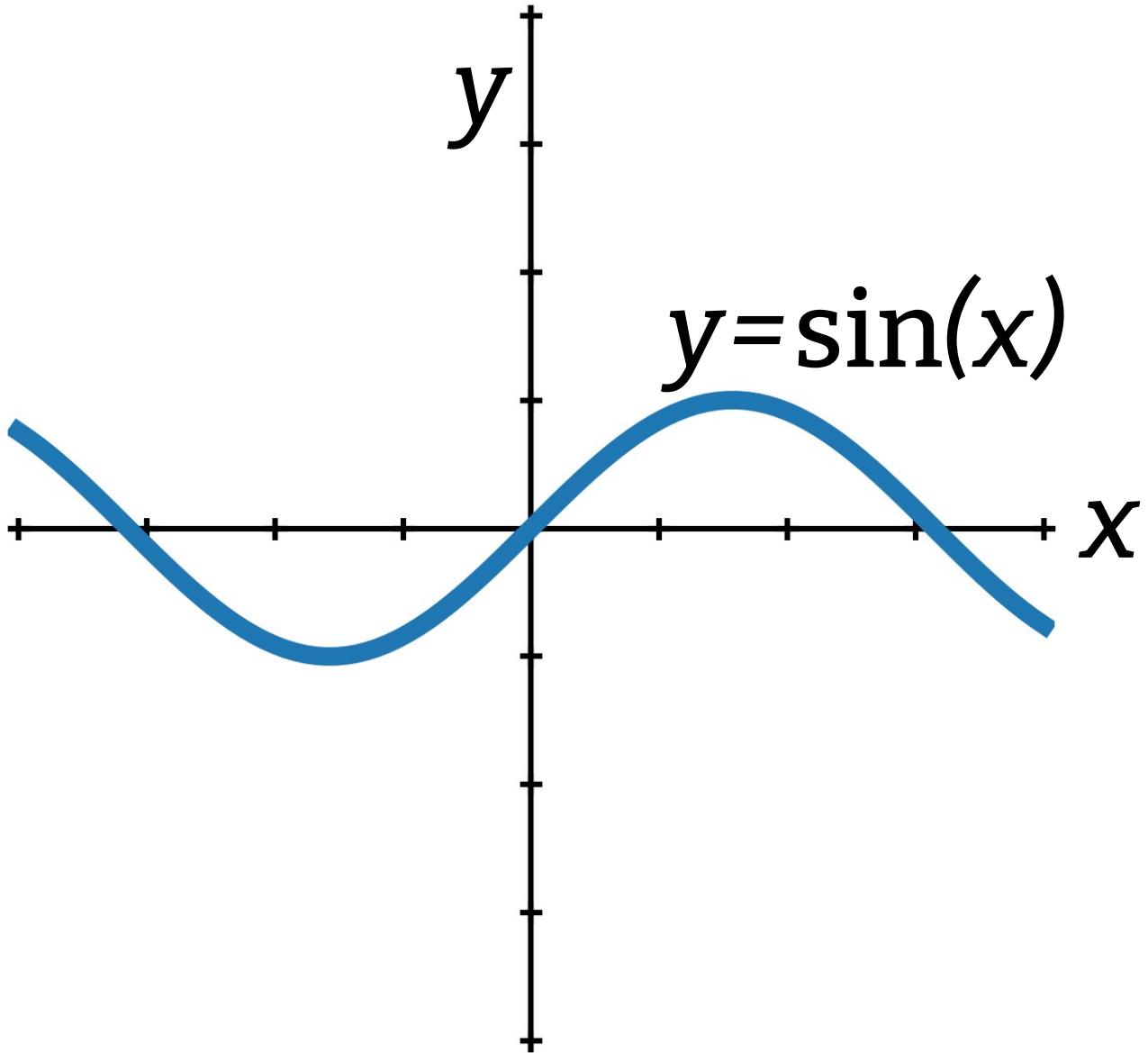
Graph of a function



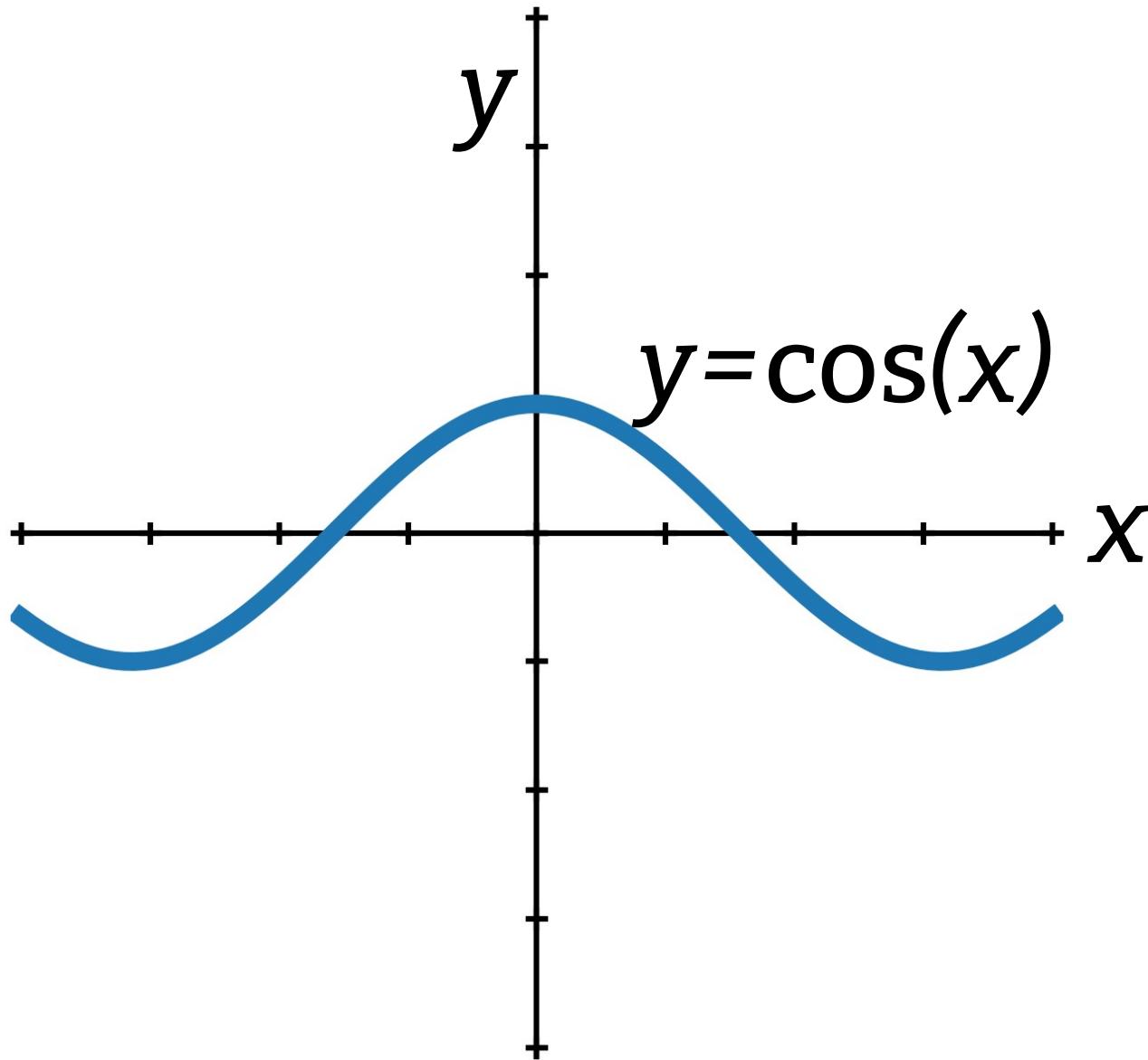
Graph of a function



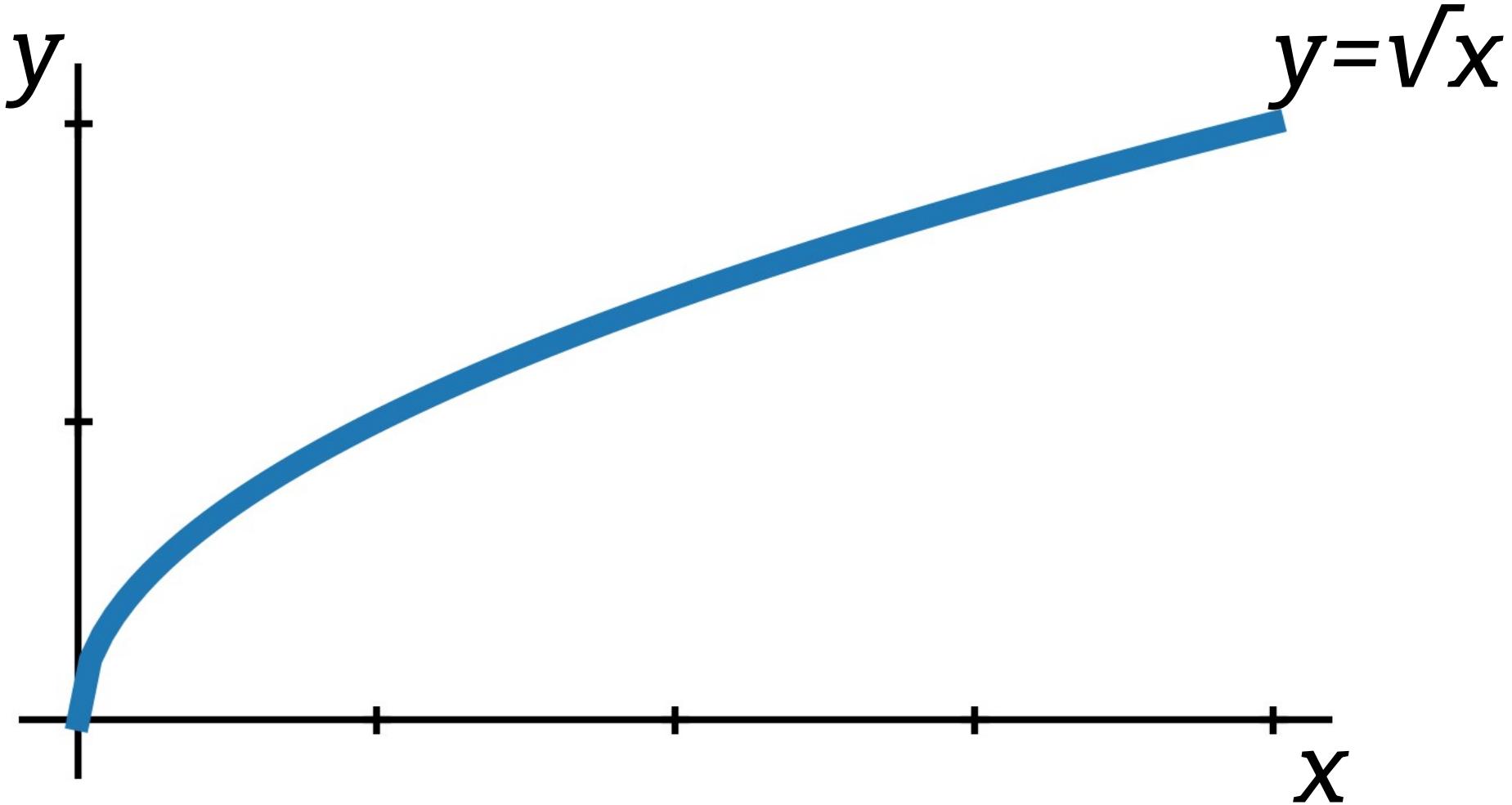
Graph of a function



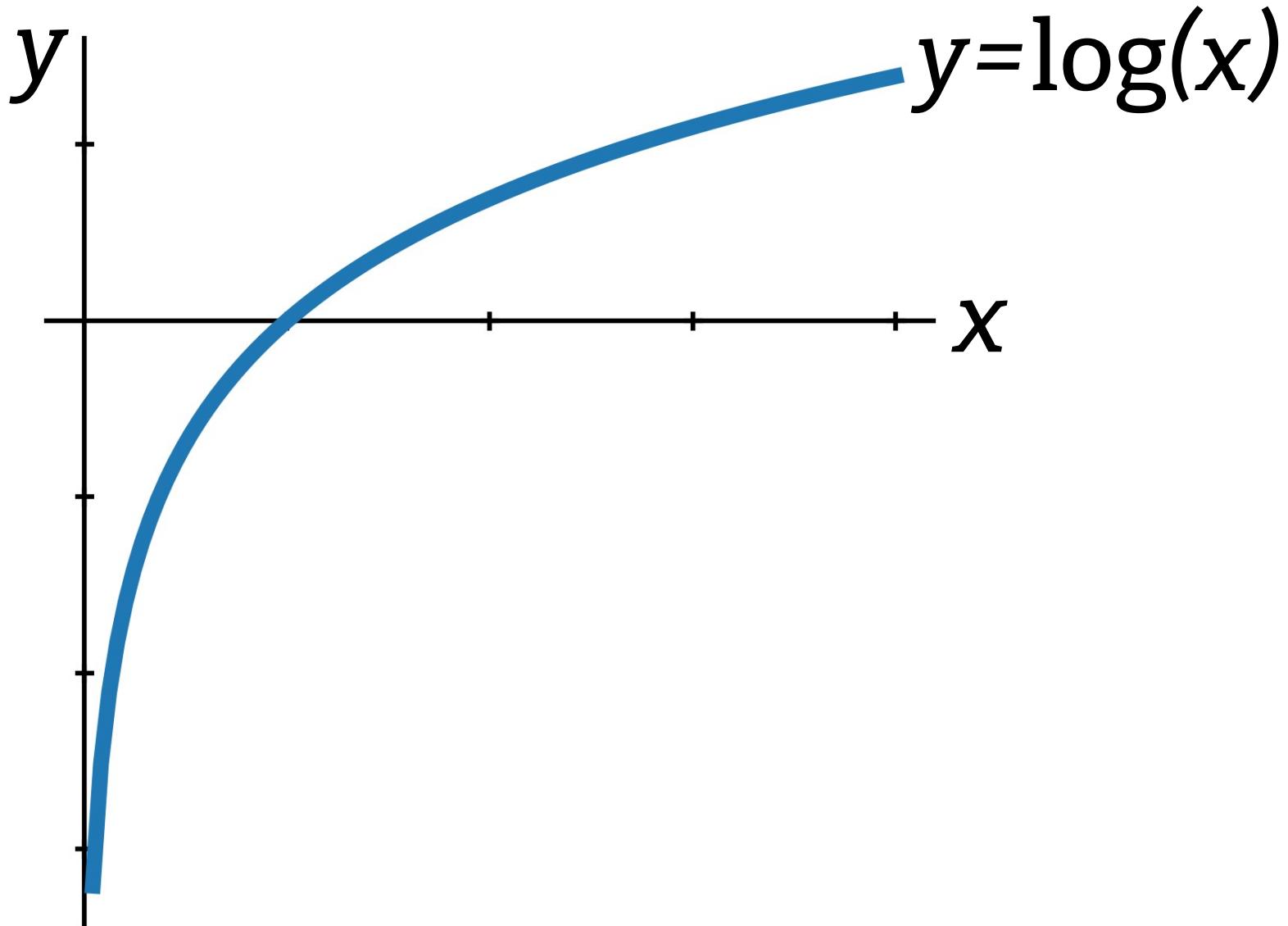
Graph of a function



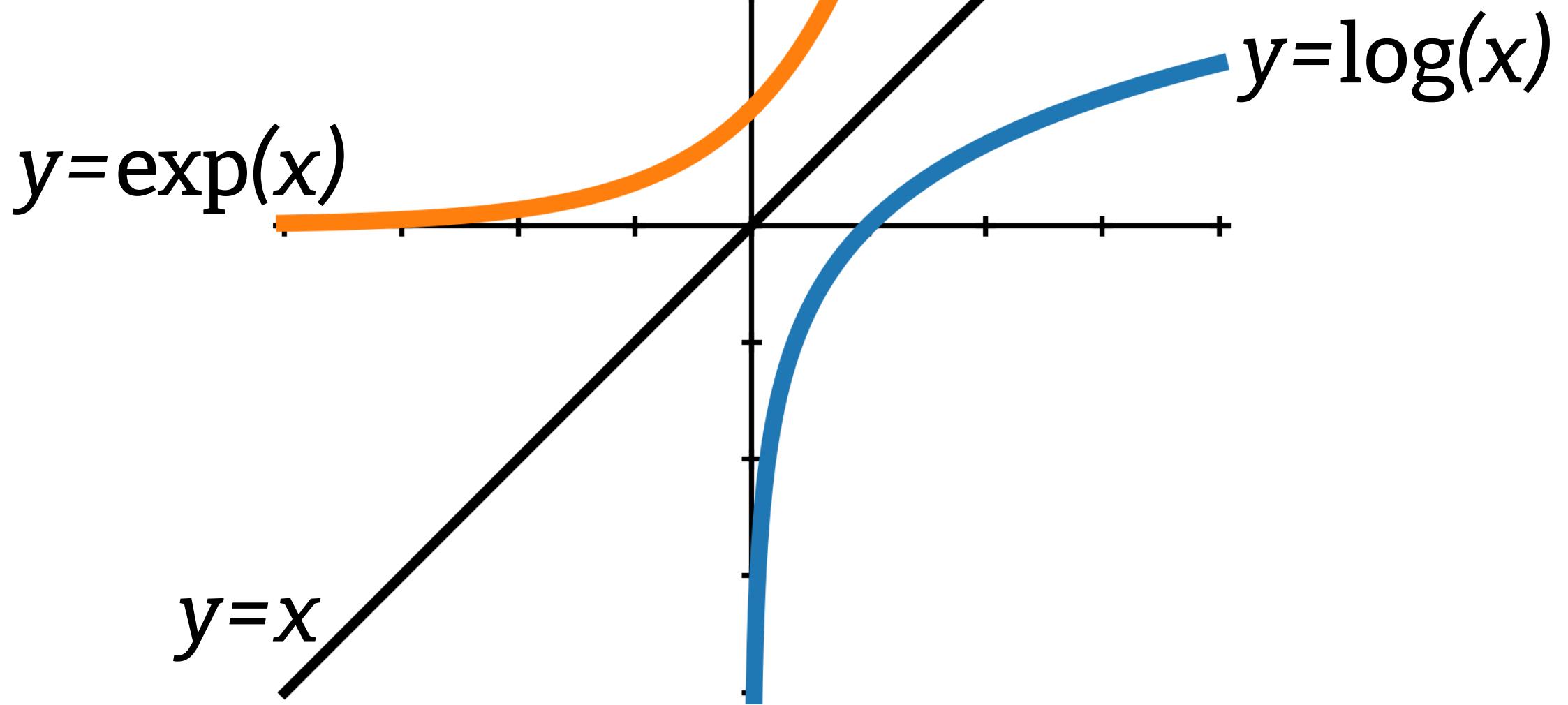
Graph of a function



Logarithm



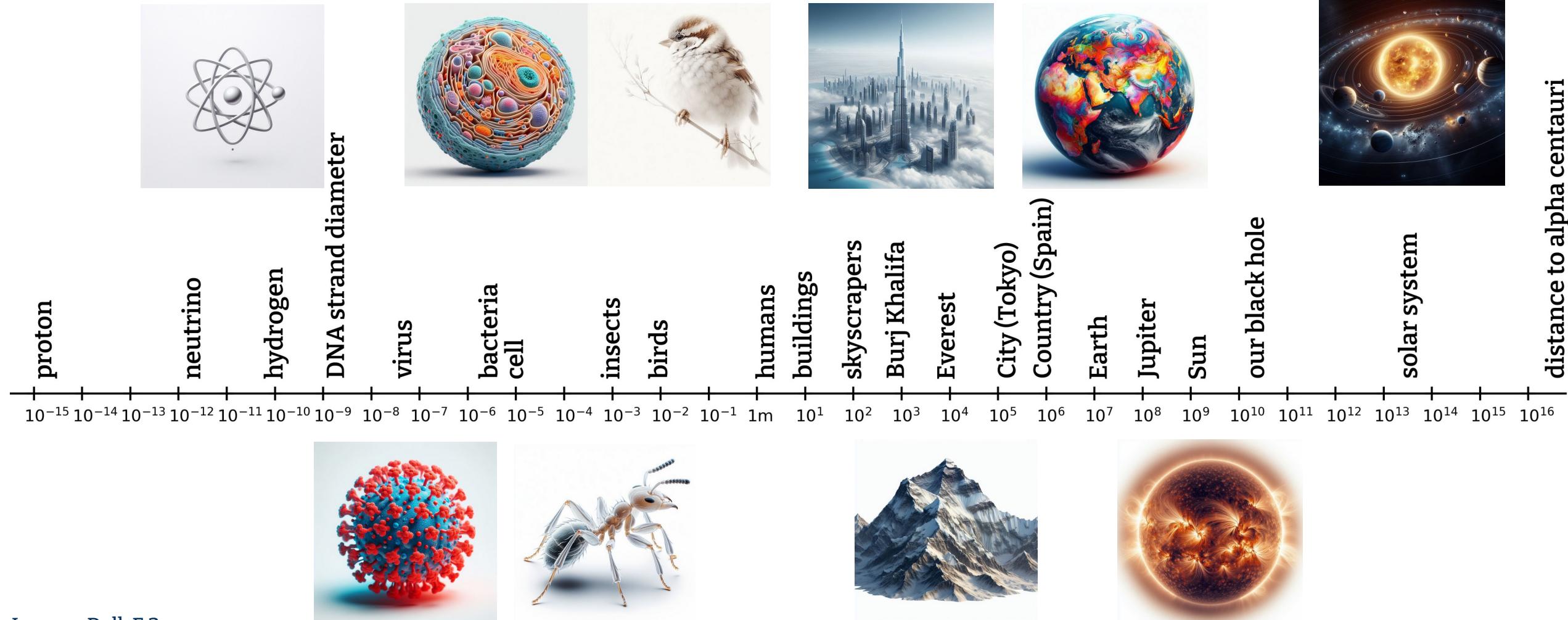
Logarithm



Logarithmic scales

- Exponential growth
- Data spanning several orders of magnitude:
 - Distances in the universe, from atoms to galaxies
 - Wavelengths, from radio waves to visible light, to X rays

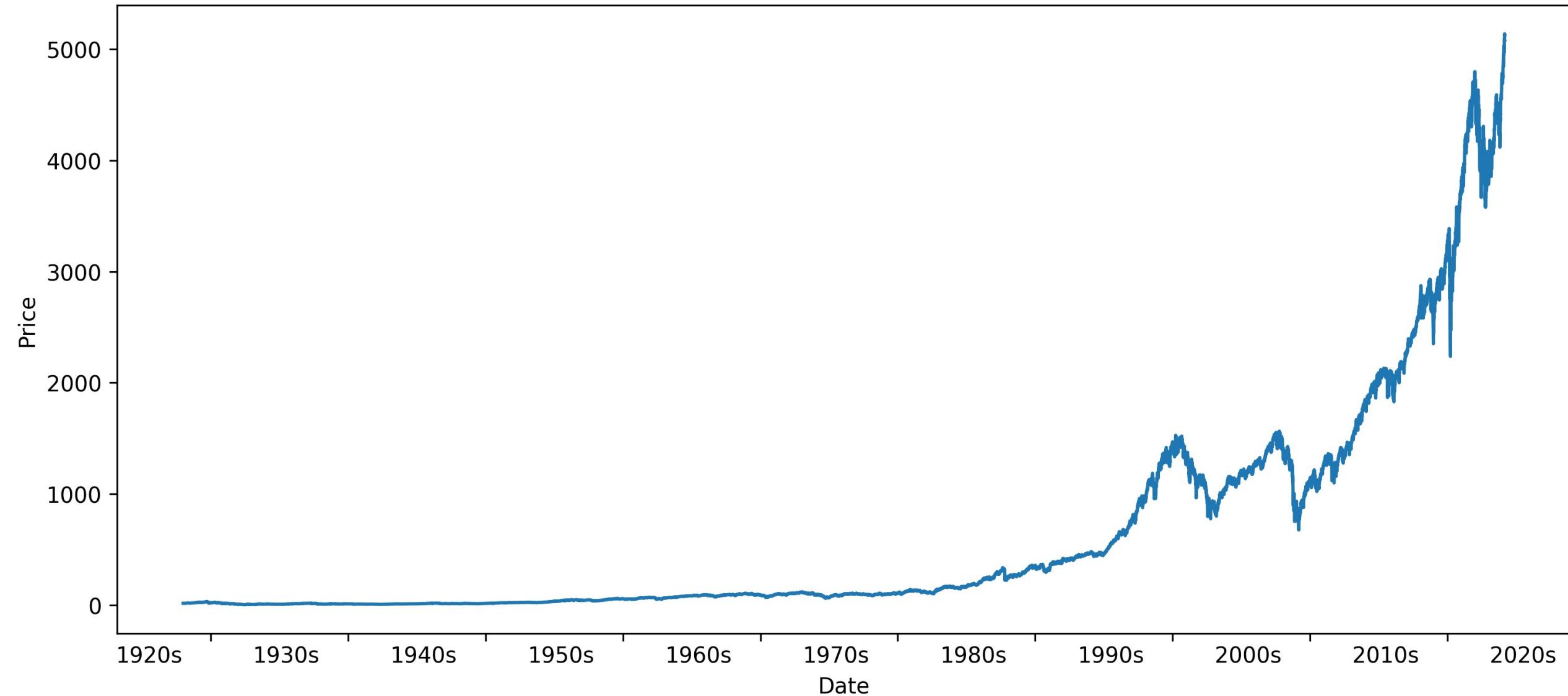
Logarithmic scale



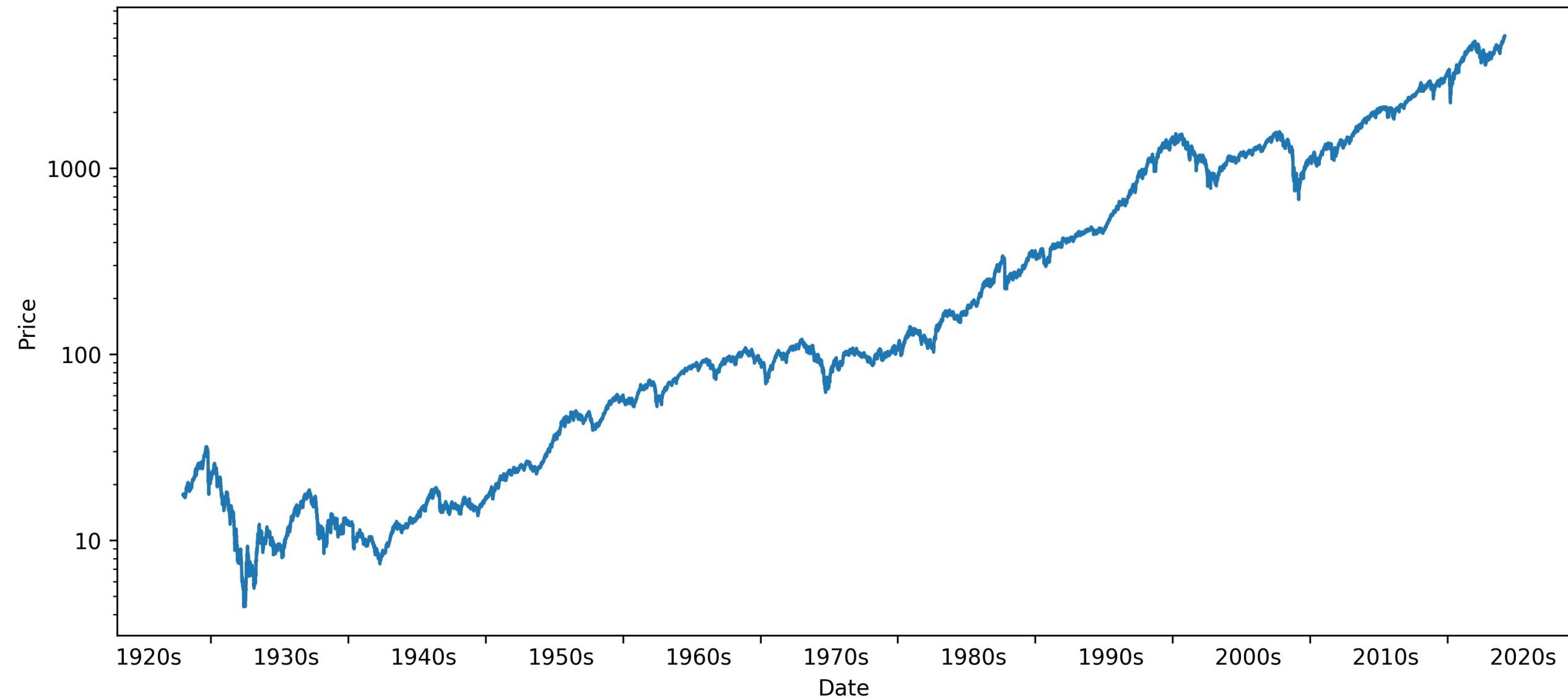
Images: Dall-E 3

Sizes: https://www.youtube.com/watch?v=Z_1Q0XB4X0Y

Linear scale



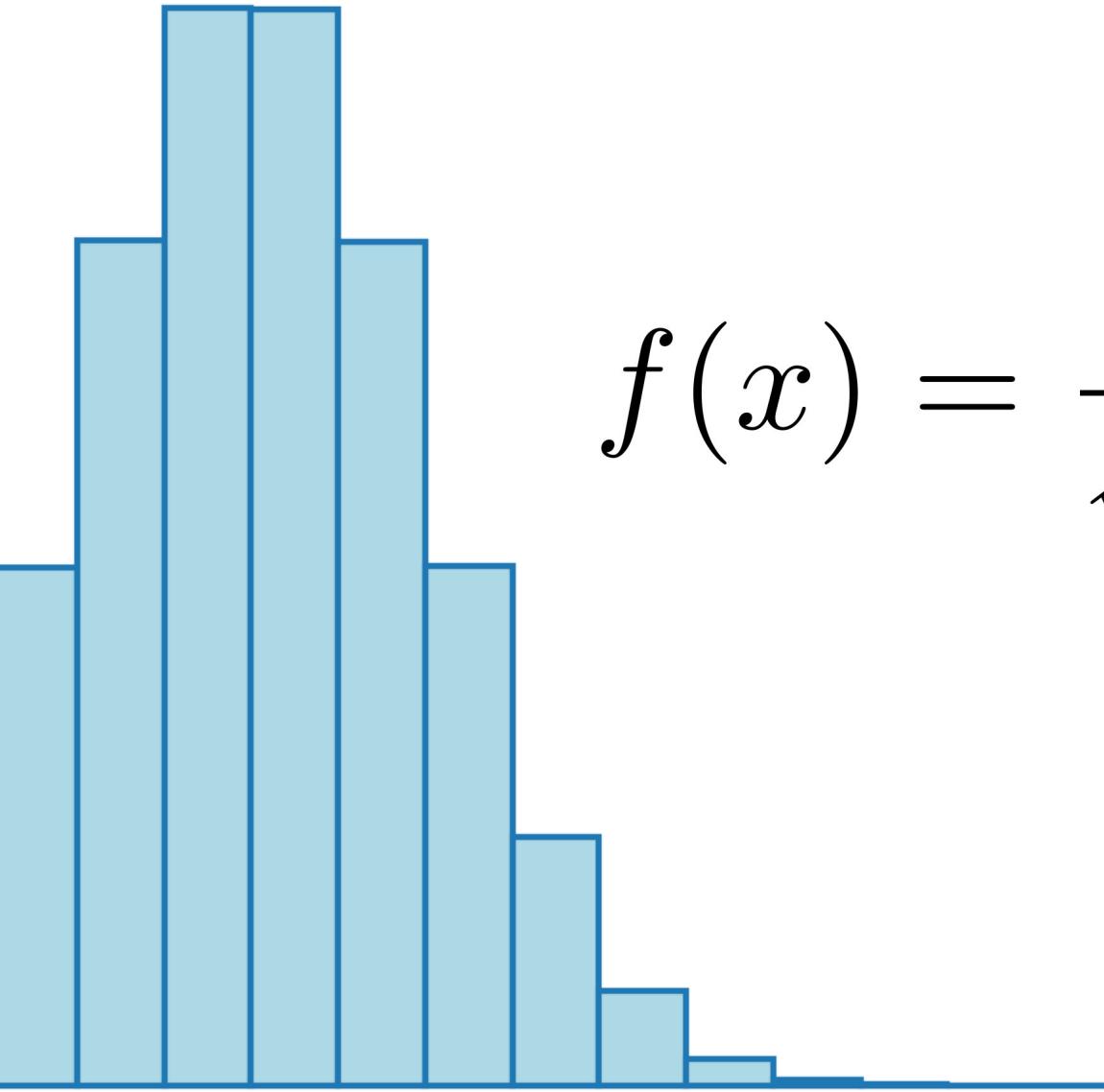
Logarithmic scale



Vectors, \mathbf{R}^n

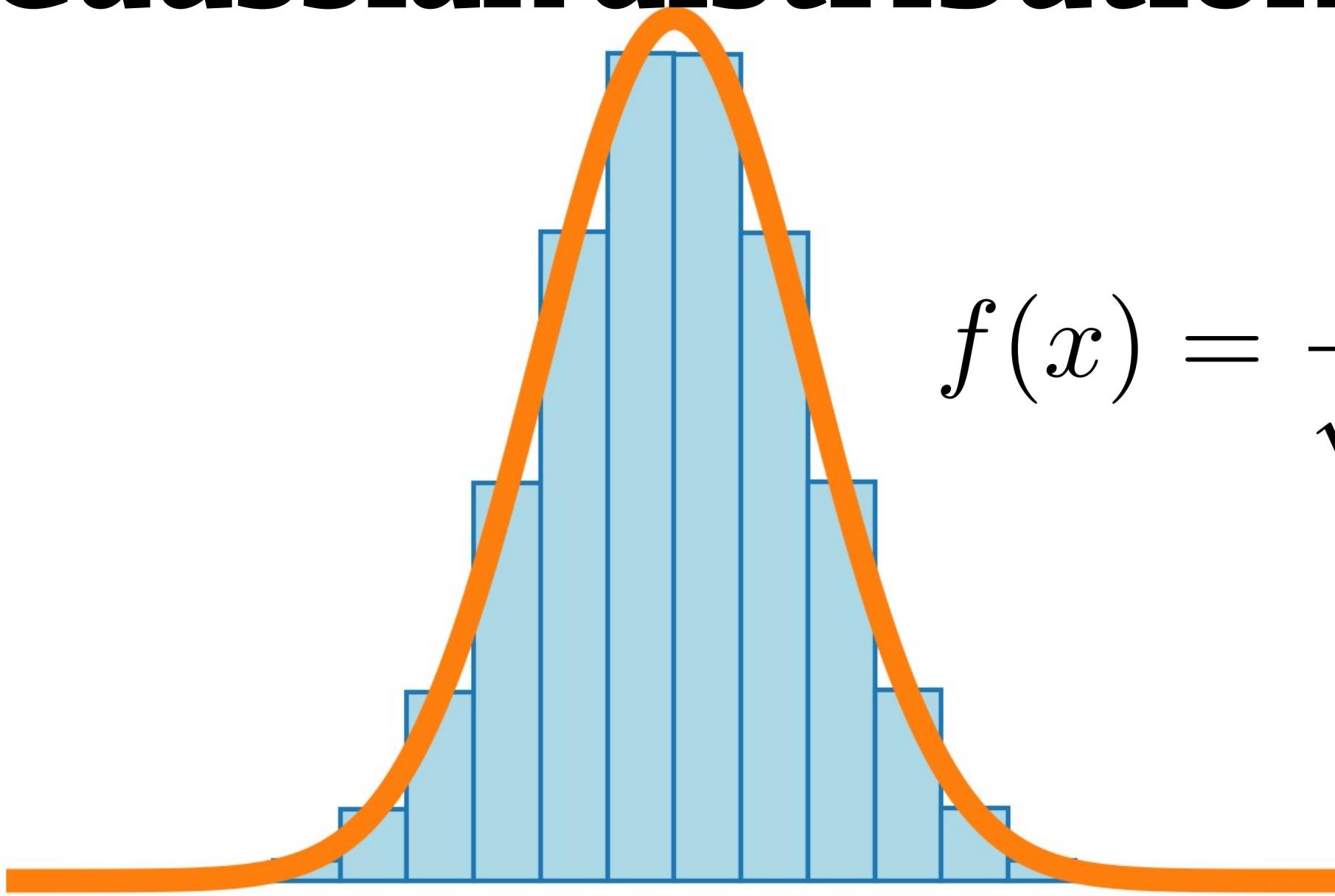
$$(x_1, x_2, \dots, x_n) \in \mathbb{R}^n$$

Gaussian distribution



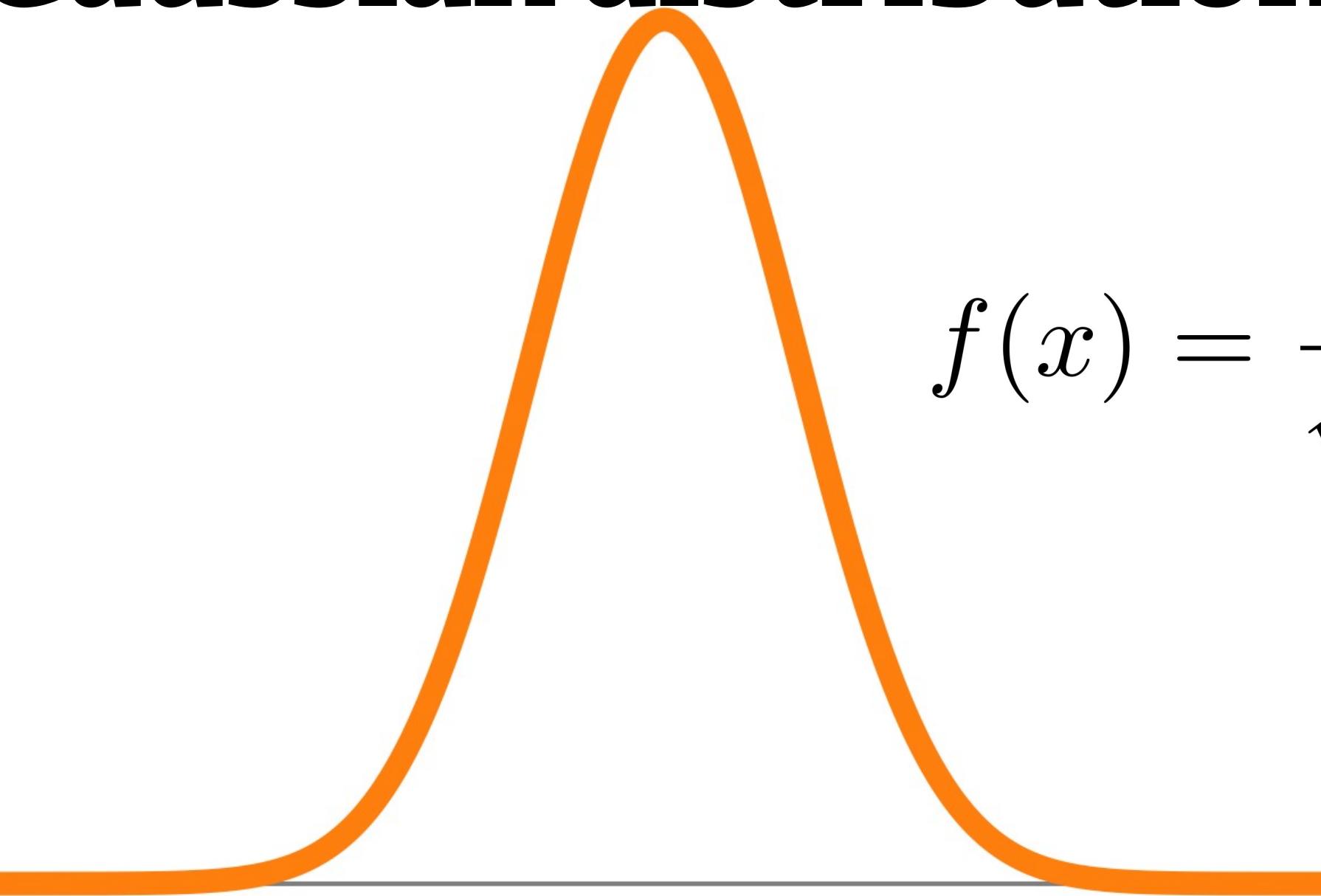
$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Gaussian distribution



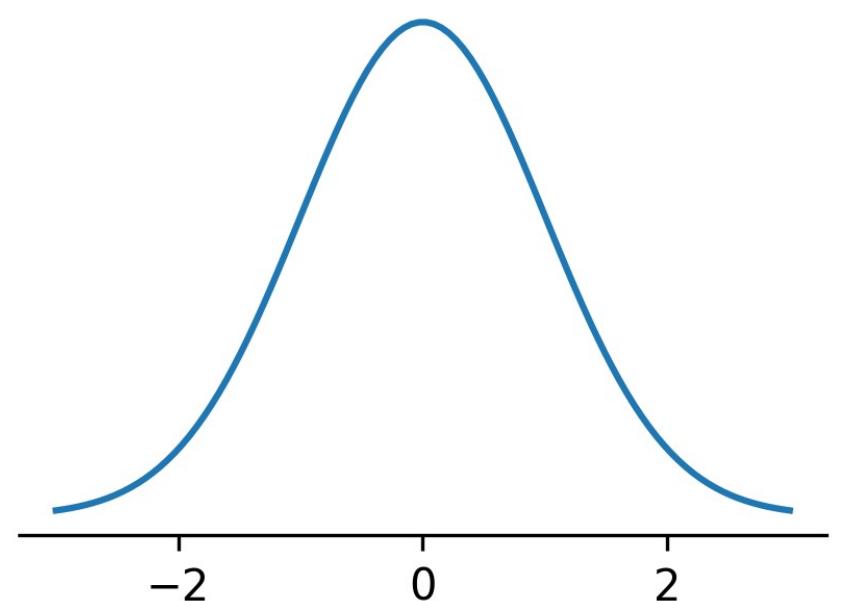
$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Gaussian distribution

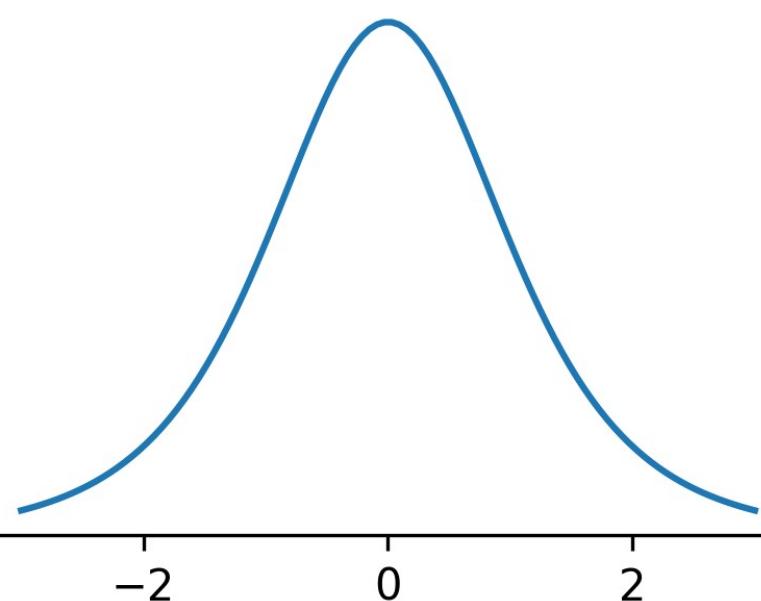


$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

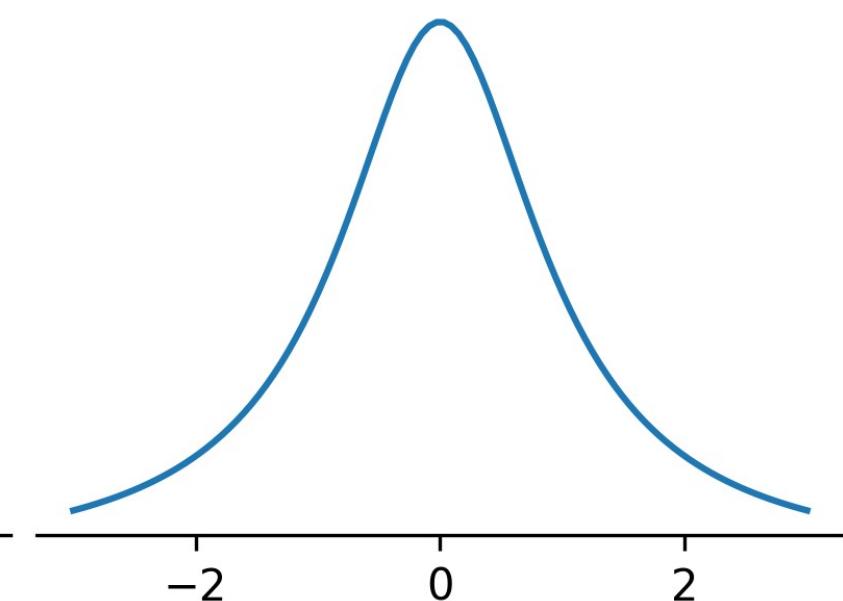
Gaussian



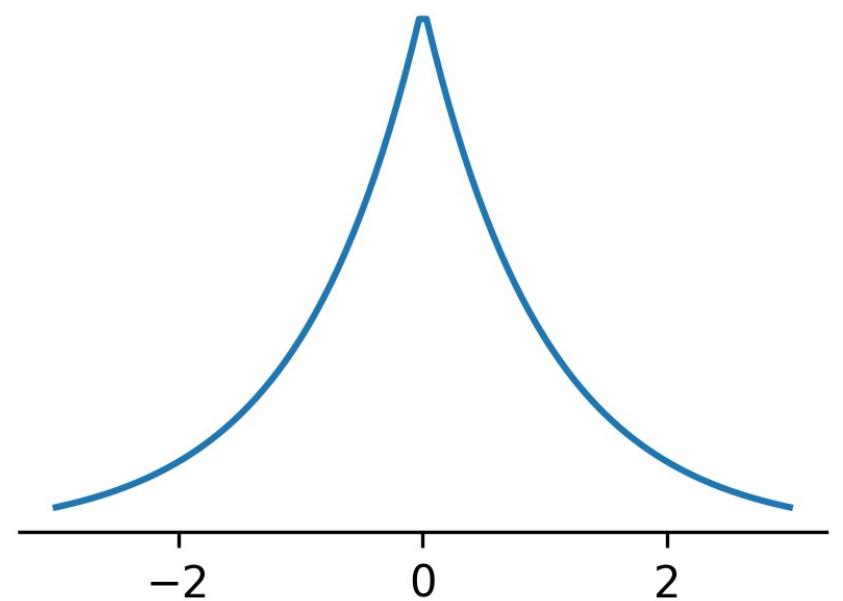
Student



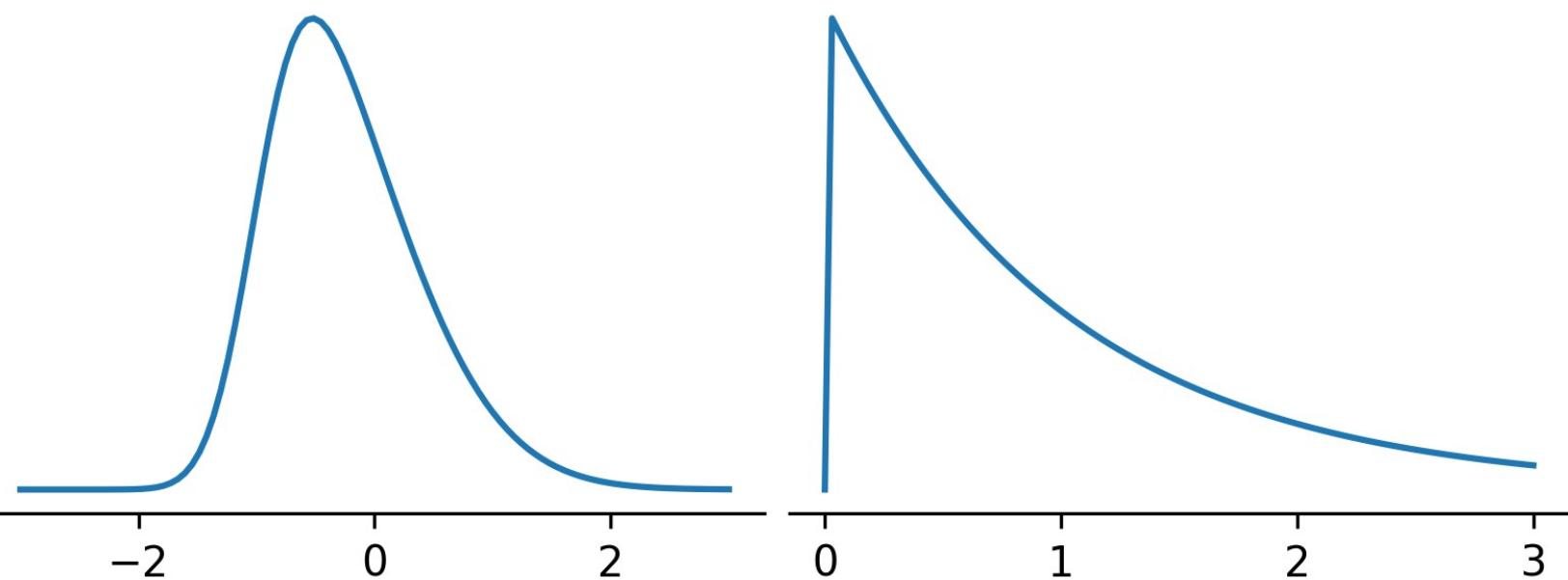
Cauchy



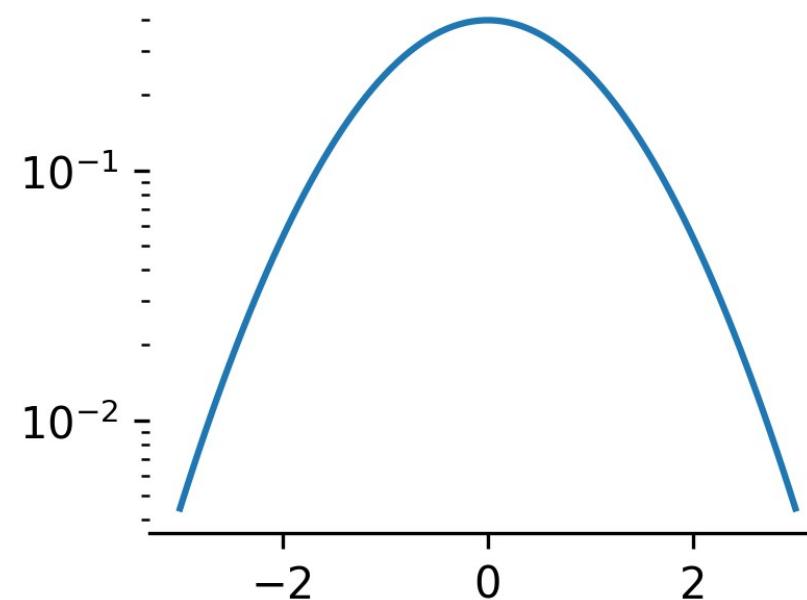
Laplace



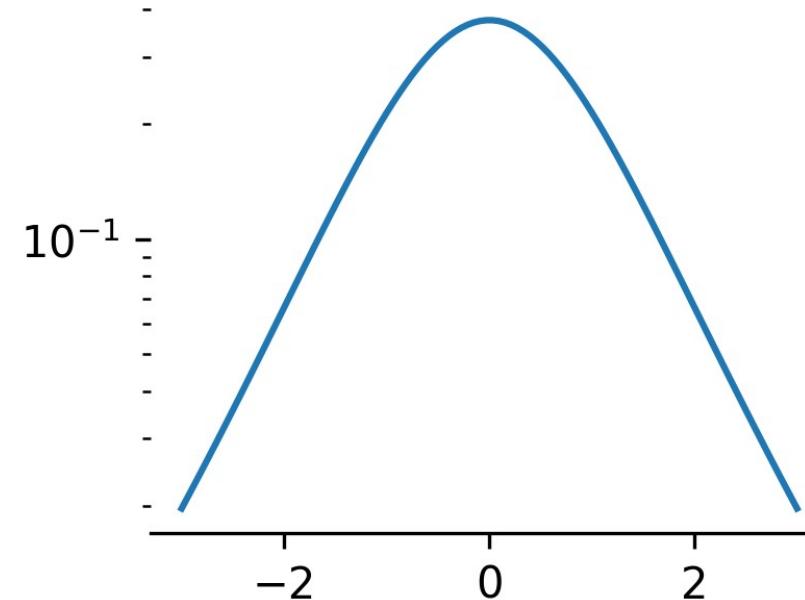
Exponential



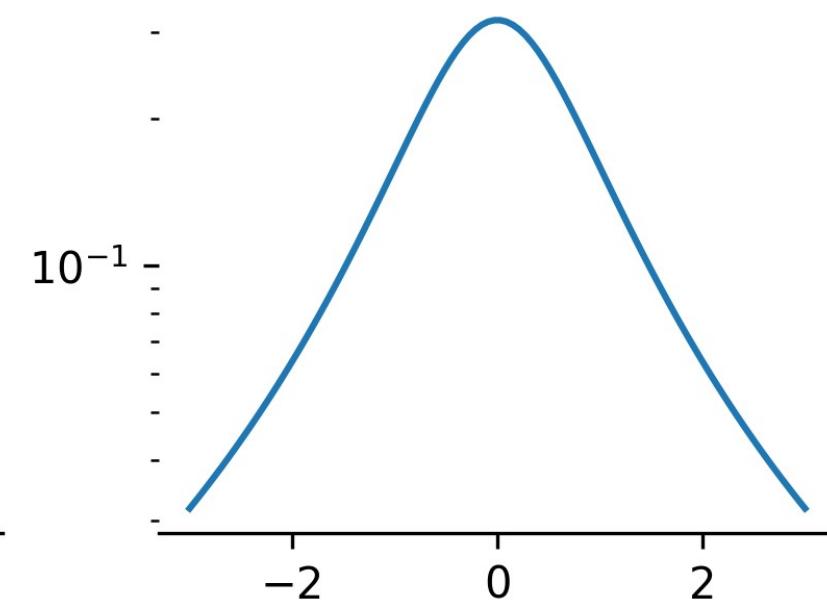
Gaussian



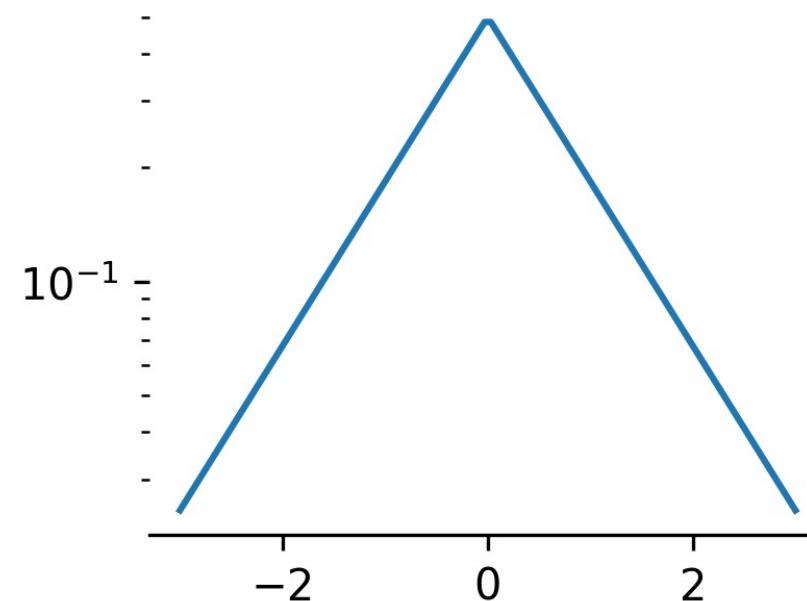
Student



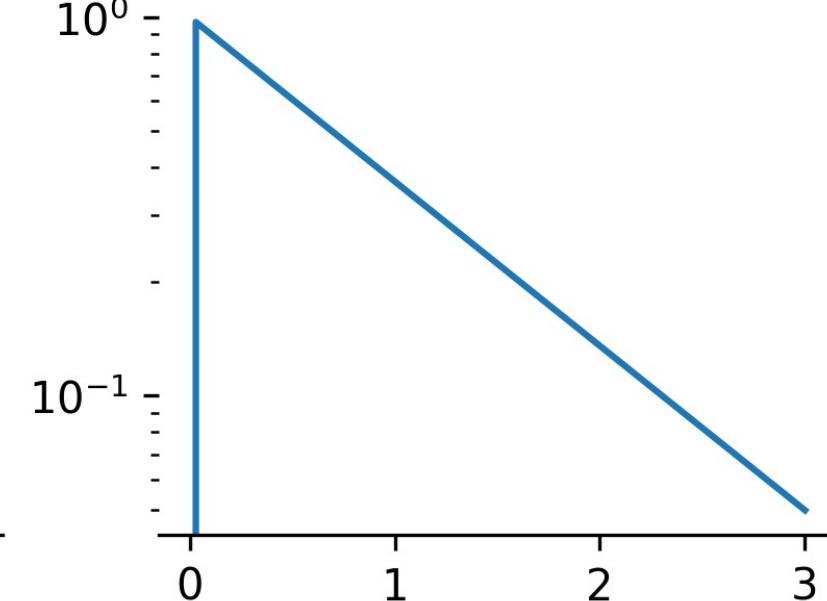
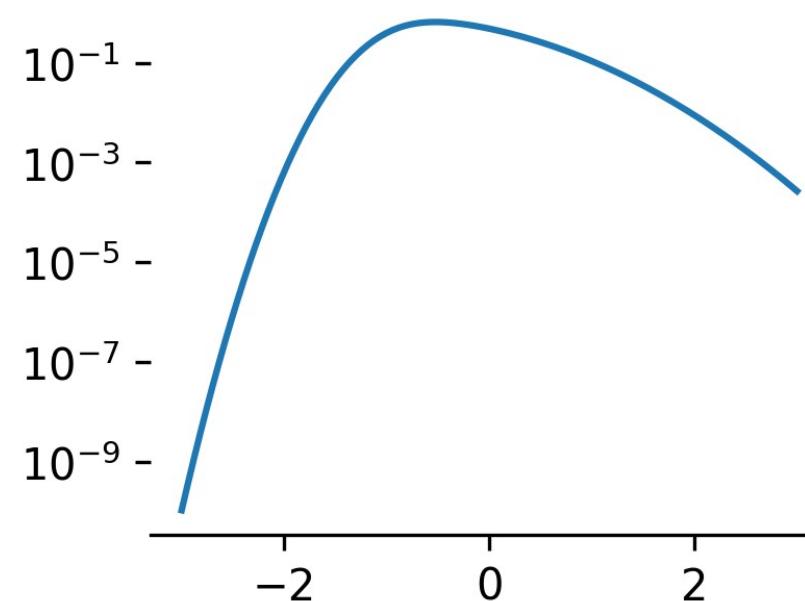
Cauchy



Laplace



Exponential



Python

```
def factorial(n):
    if n == 0:
        return 1
    x = 1
    for k in range(1,n+1):
        x *= k
    return x
```

Pandas

```
d = pd.read_csv("a.csv")
d.head()
d["x"]
d.loc[:, "x"]
d.iloc[:, 0]
d.iloc[0, :]
d.sum()
d.groupby("region").sum()
```

Numpy

```
d = pd.read_csv("a.csv")
x = d.values
x[:,0]
x[0,:]
np.sum(x, axis=0) # sum of columns
np.sum(x, axis=1) # sum of rows
```

Definitions

Definition. The **average** of a sample $x = (x_1, \dots, x_n) \in \mathbf{R}^n$ is ()

$$\bar{x} = \frac{1}{n} \sum_i x_i.$$

Definition. The **variance** of a sample $x = (x_1, \dots, x_n) \in \mathbf{R}^n$ is

$$\text{Var}[x] = \frac{1}{n} \sum_i (x_i - \bar{x})^2.$$

Definition. The **standard deviation** of a sample $x = (x_1, \dots, x_n) \in \mathbf{R}^n$ is the square root of the variance,

$$\sigma_x = \sqrt{\text{Var } x}.$$

Remark. The standard deviation is a measure of dispersion.

Definition. The **covariance** between two samples $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ is

$$\text{Cov}(x, y) = \frac{1}{n} \sum (x_i - \bar{x})(y_i - \bar{y}).$$

Remark. $\text{Var } x = \text{Cov}(x, x)$

Definition. The **correlation** between two samples $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ is

$$\text{Cor}(x, y) = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y}.$$

Remark. $\text{Cor}(x, y) \in [-1, 1]$

Definition. The **scalar product** of two vectors $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ is

$$\begin{aligned}\langle x, y \rangle &= x_1 y_1 + x_2 y_2 + \cdots + x_n y_n \\ &= \sum x_i y_i\end{aligned}$$

Definition. The **norm** of a vector $x = (x_1, \dots, x_n)$ is

$$\begin{aligned}\|x\| &= \sqrt{\sum x_i^2} \\ &= \sqrt{\langle x, c \rangle}.\end{aligned}$$

Remark. If the samples x and y are centered, i.e., $\bar{x} = \bar{y} = 0$, then

$$\text{Cor}(x, y) = \frac{\langle x, y \rangle}{\|x\| \cdot \|y\|} = \cos \theta$$

is the cosine of the angle between x and y .

Datasets

old faithful

waiting

79

54

74

62

85

55

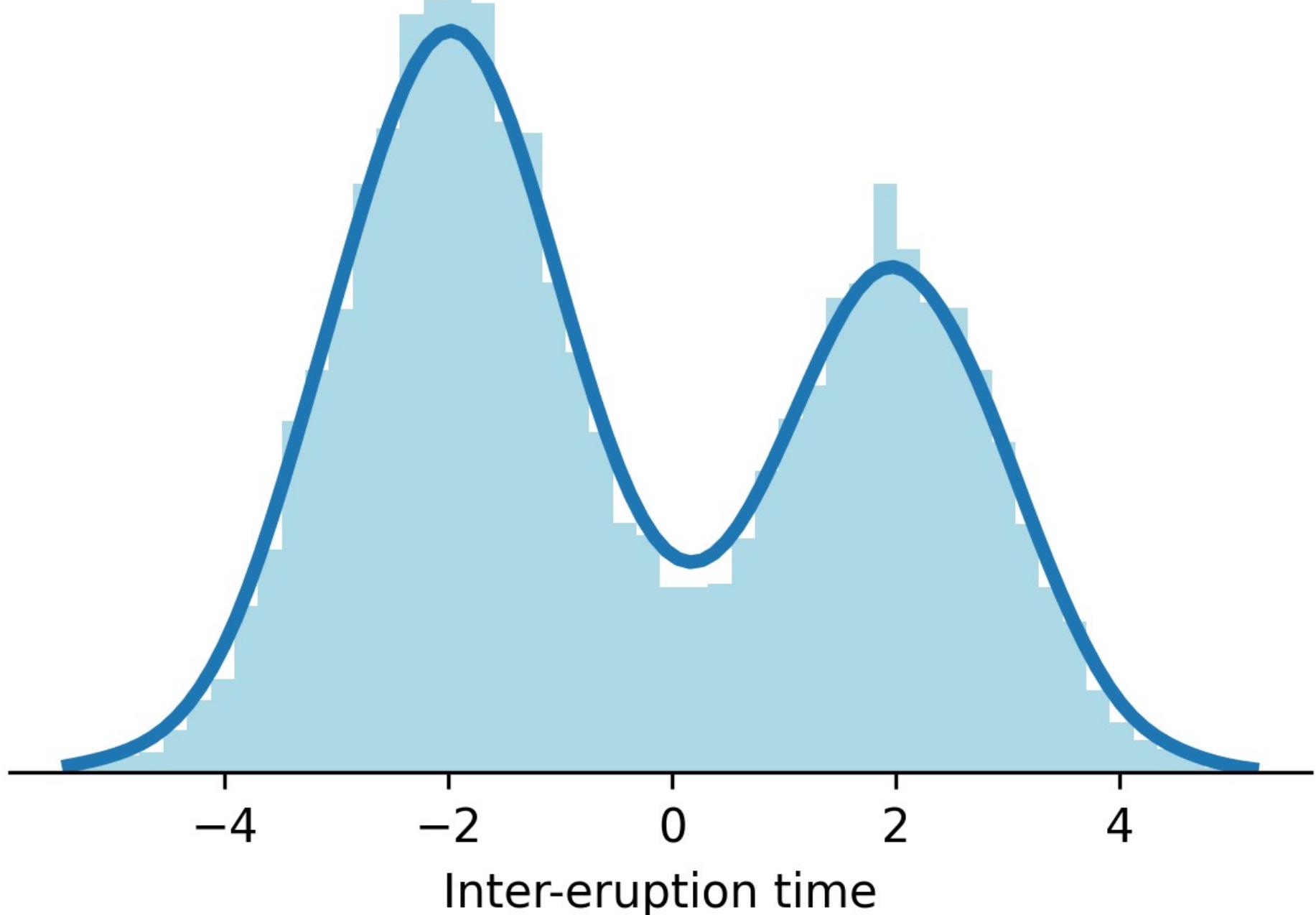
88

85

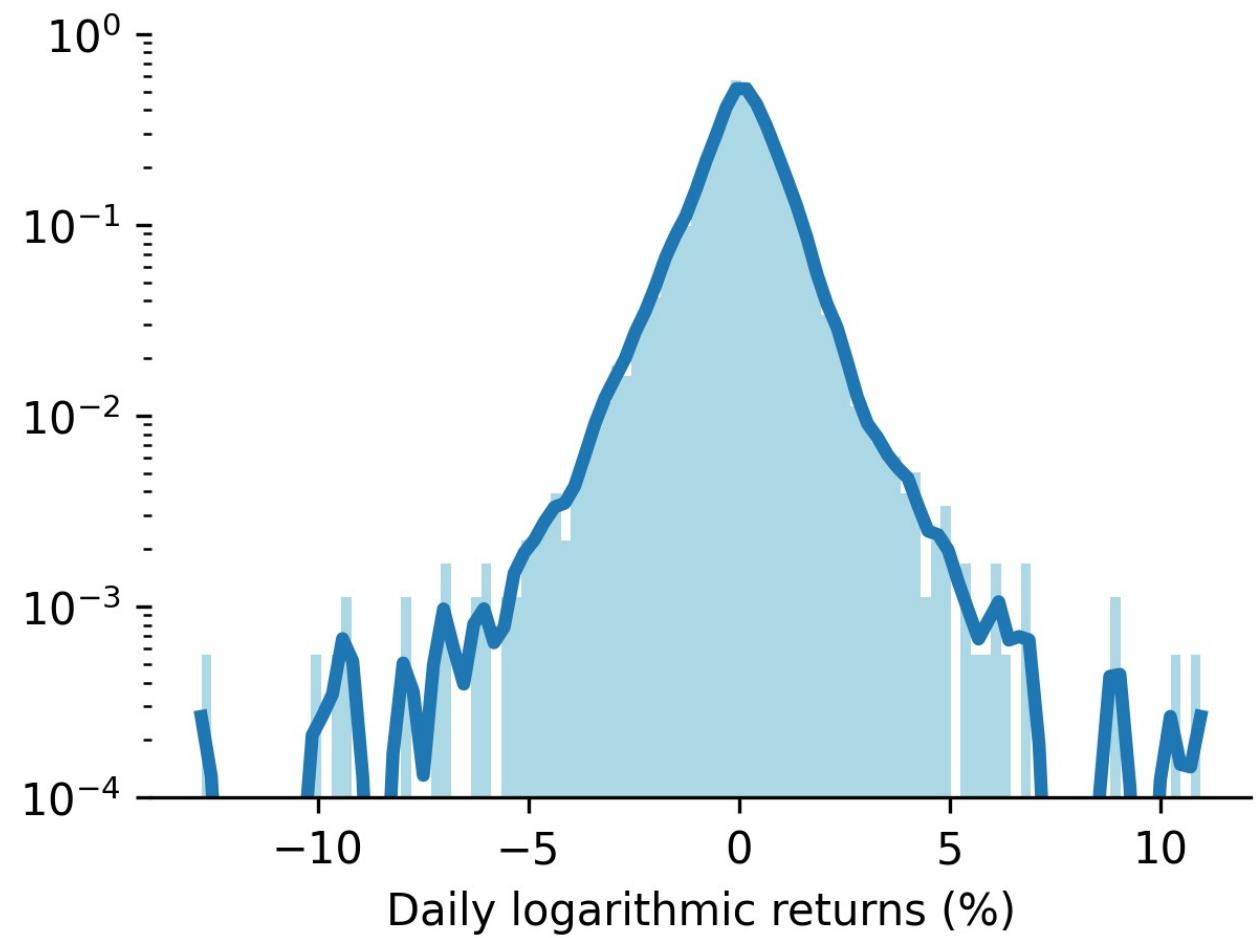
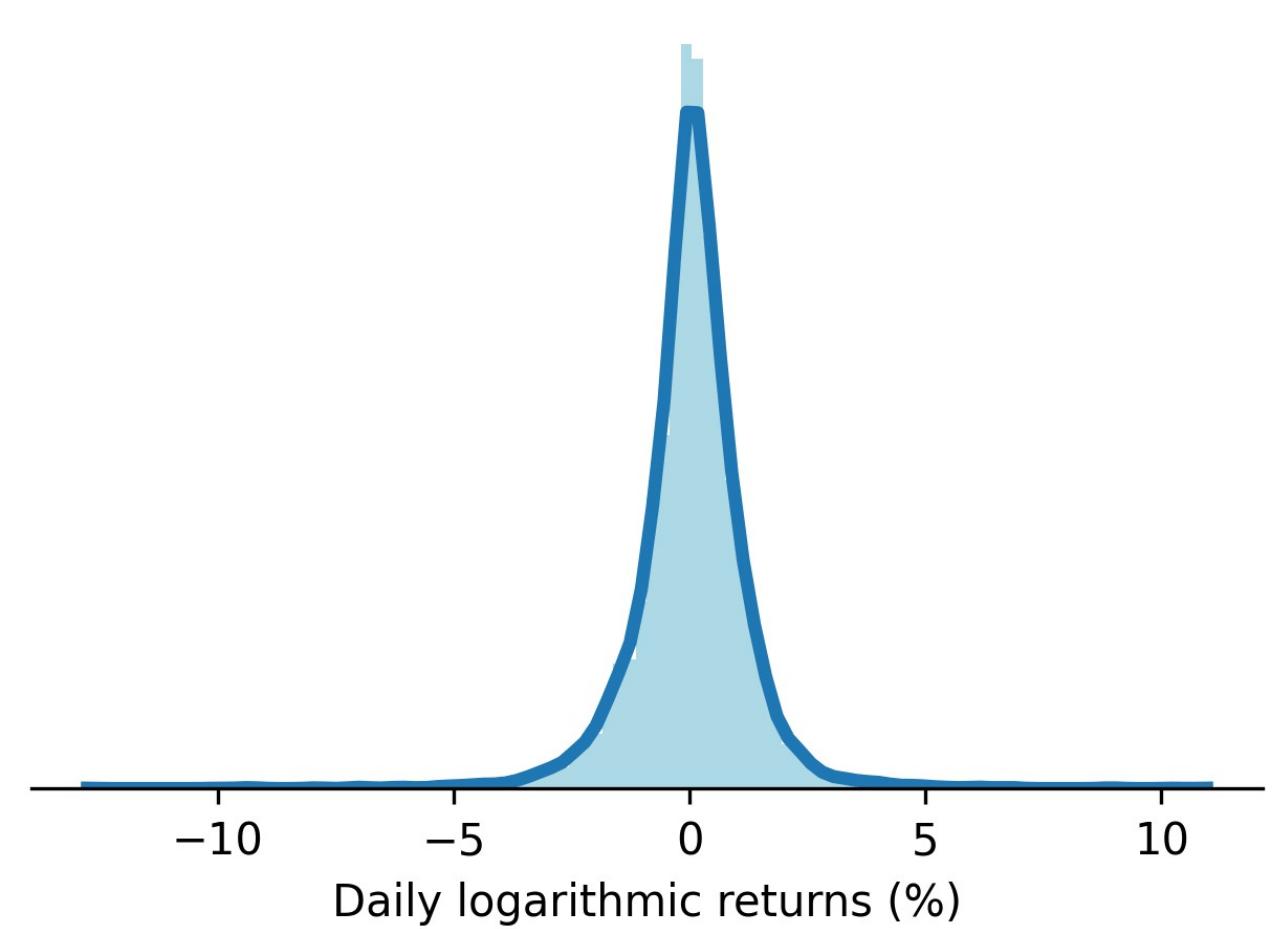
51

85

54

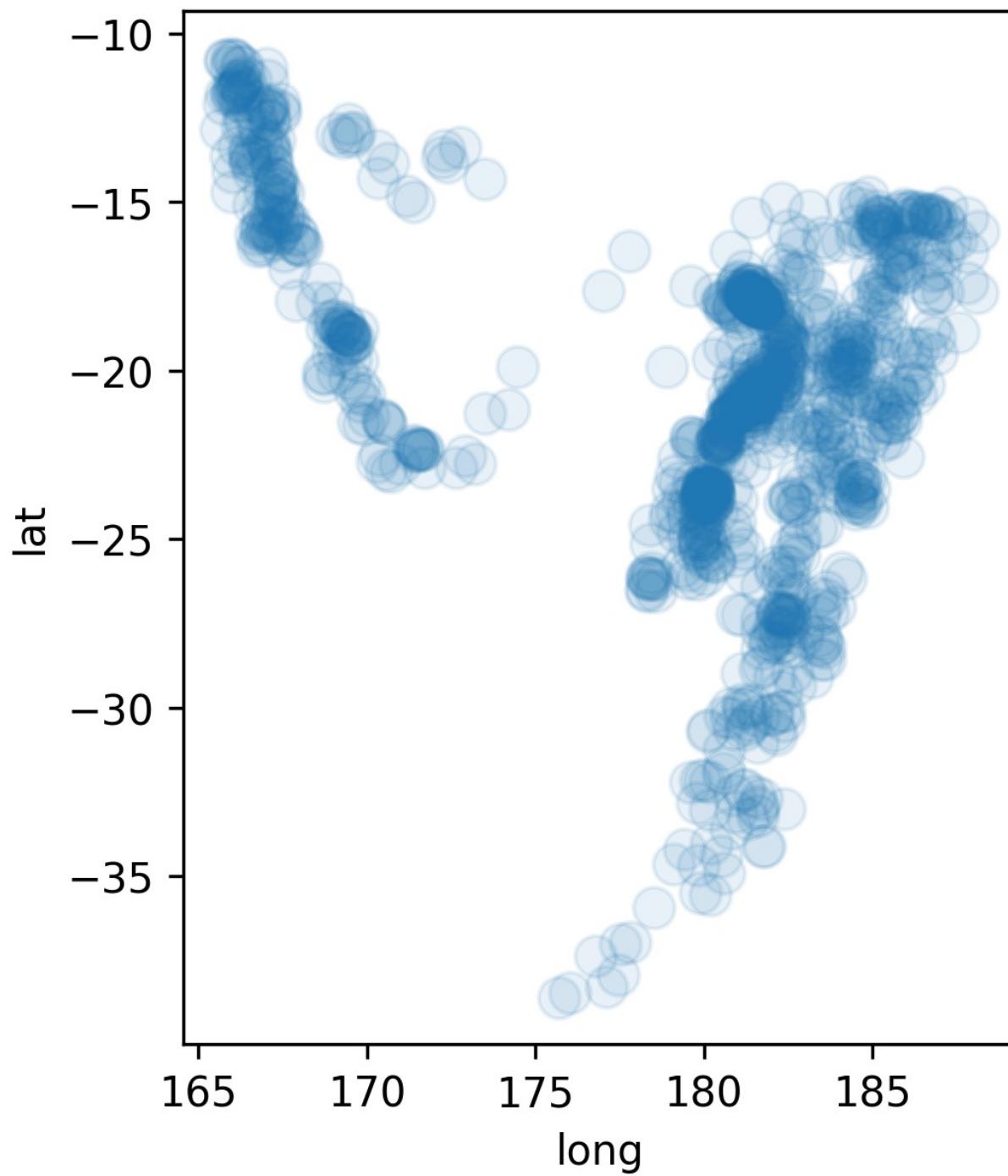


Stock Returns

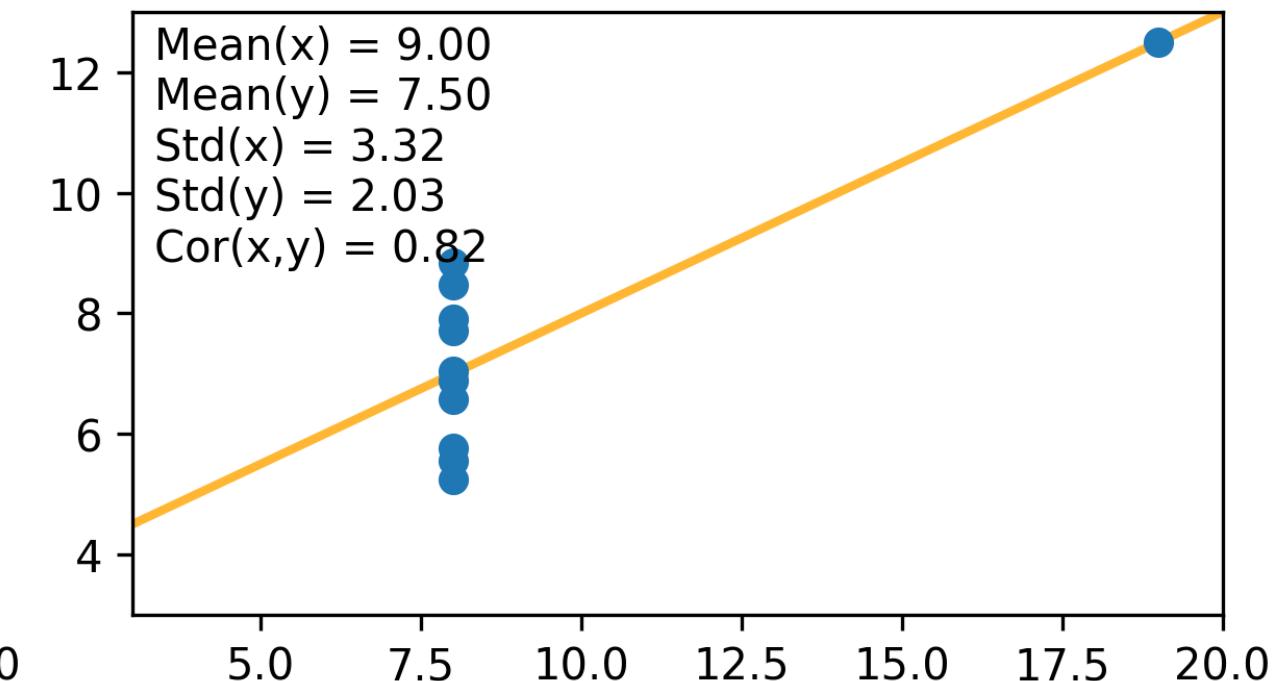
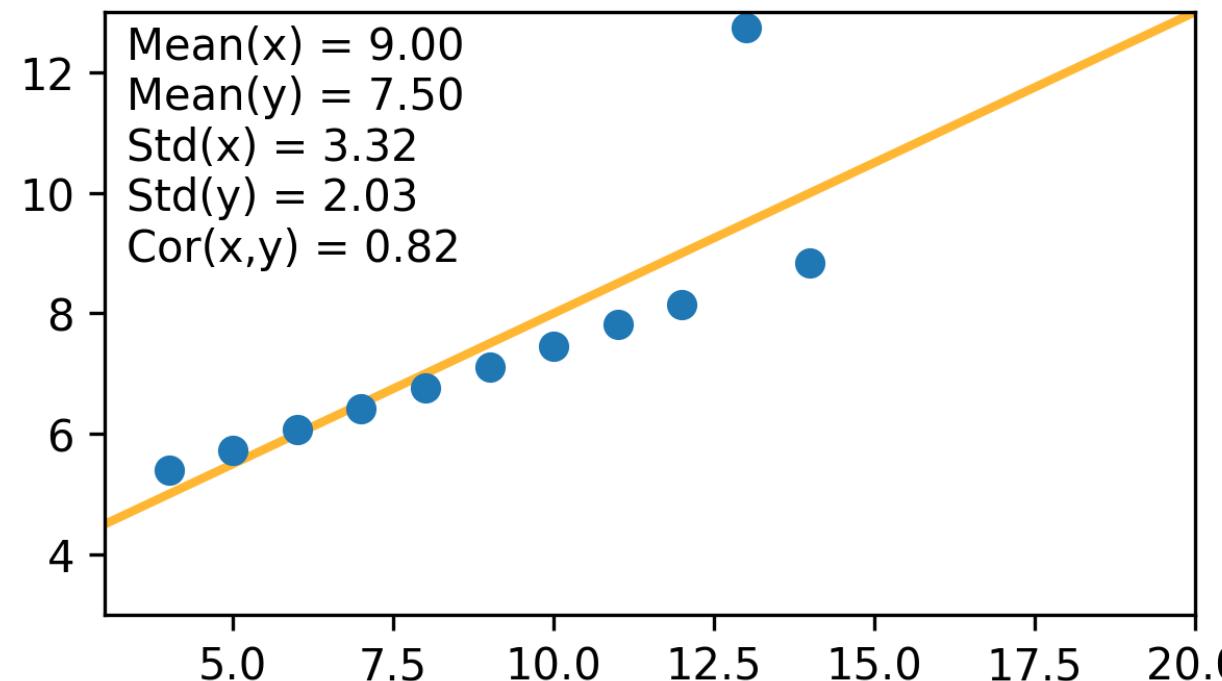
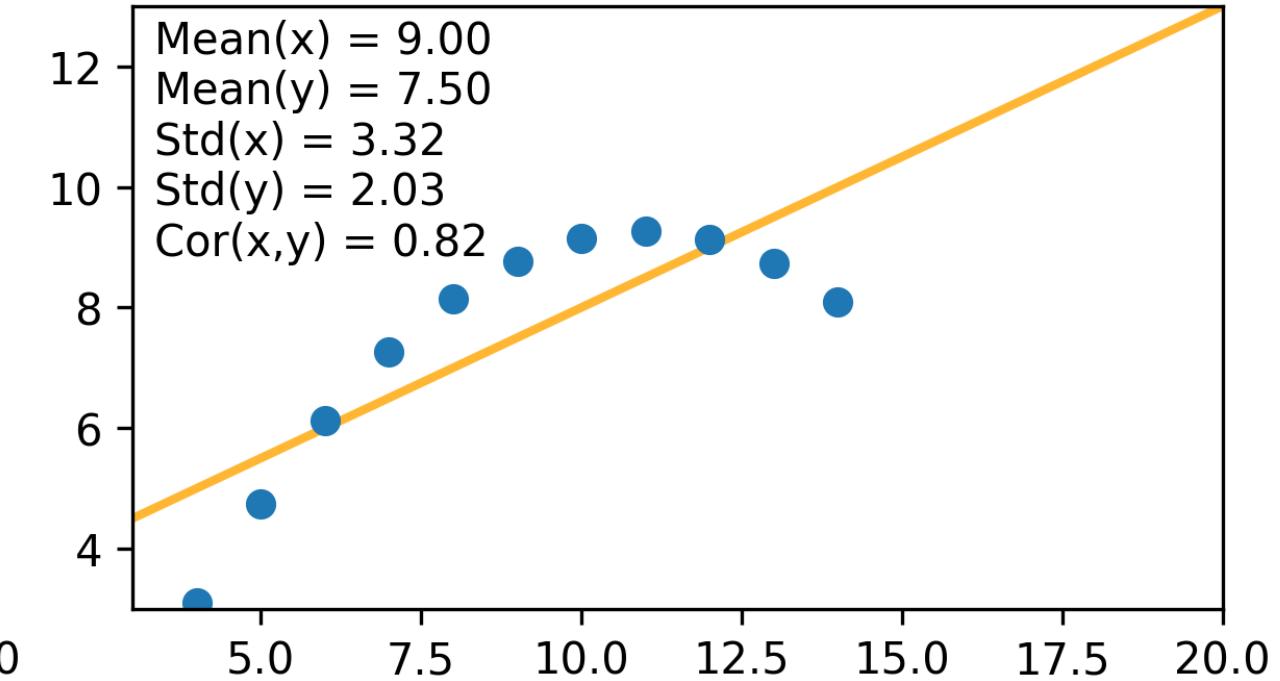
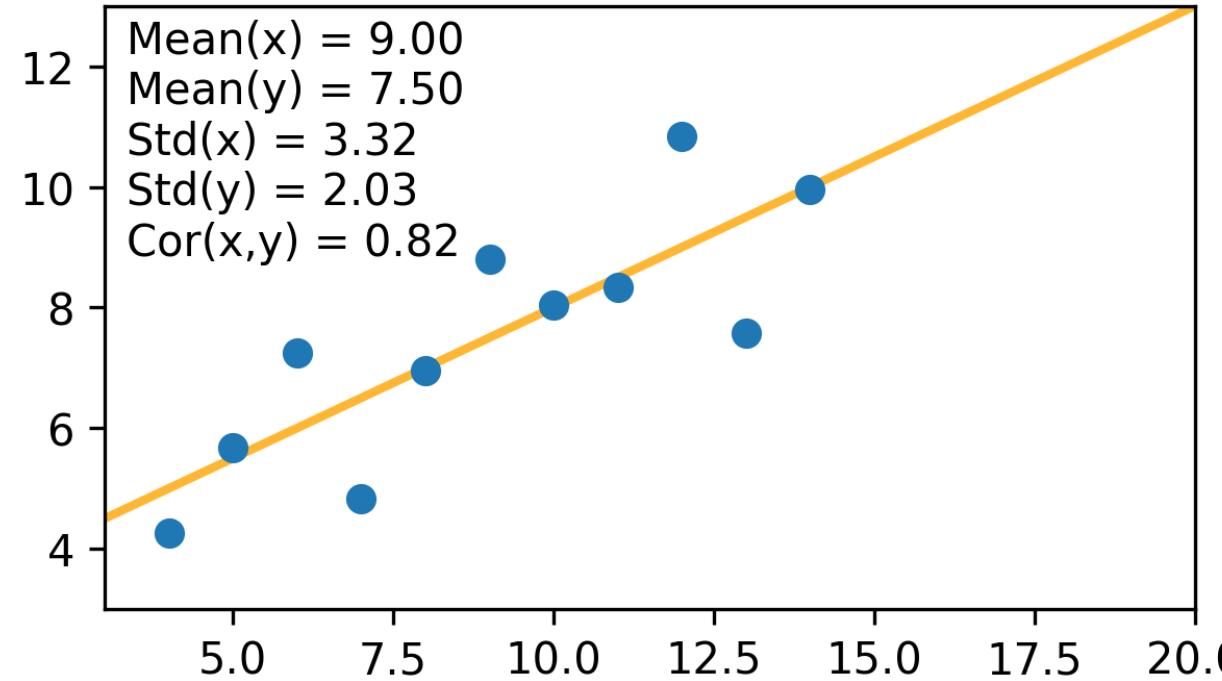


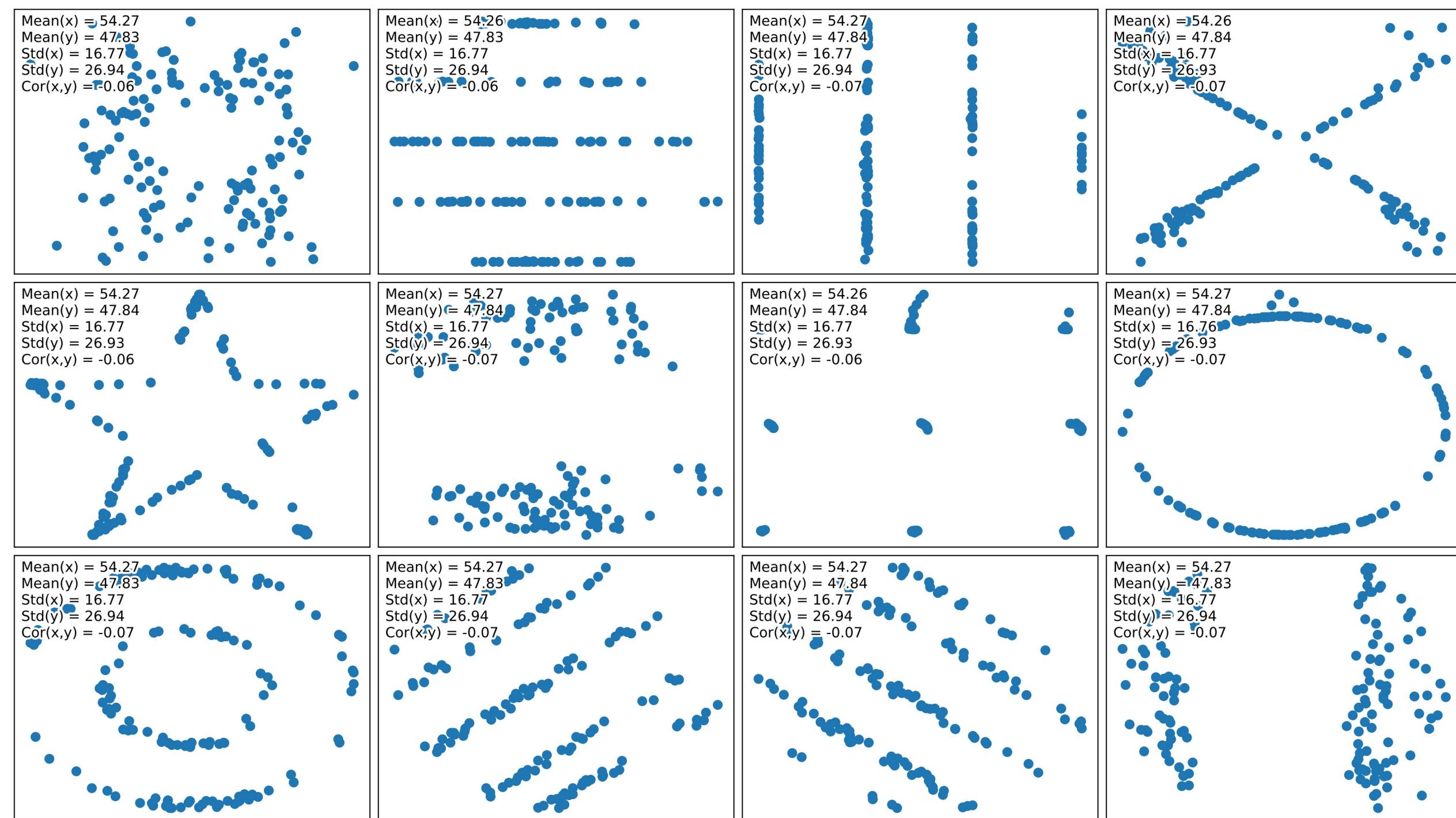
Quakes

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

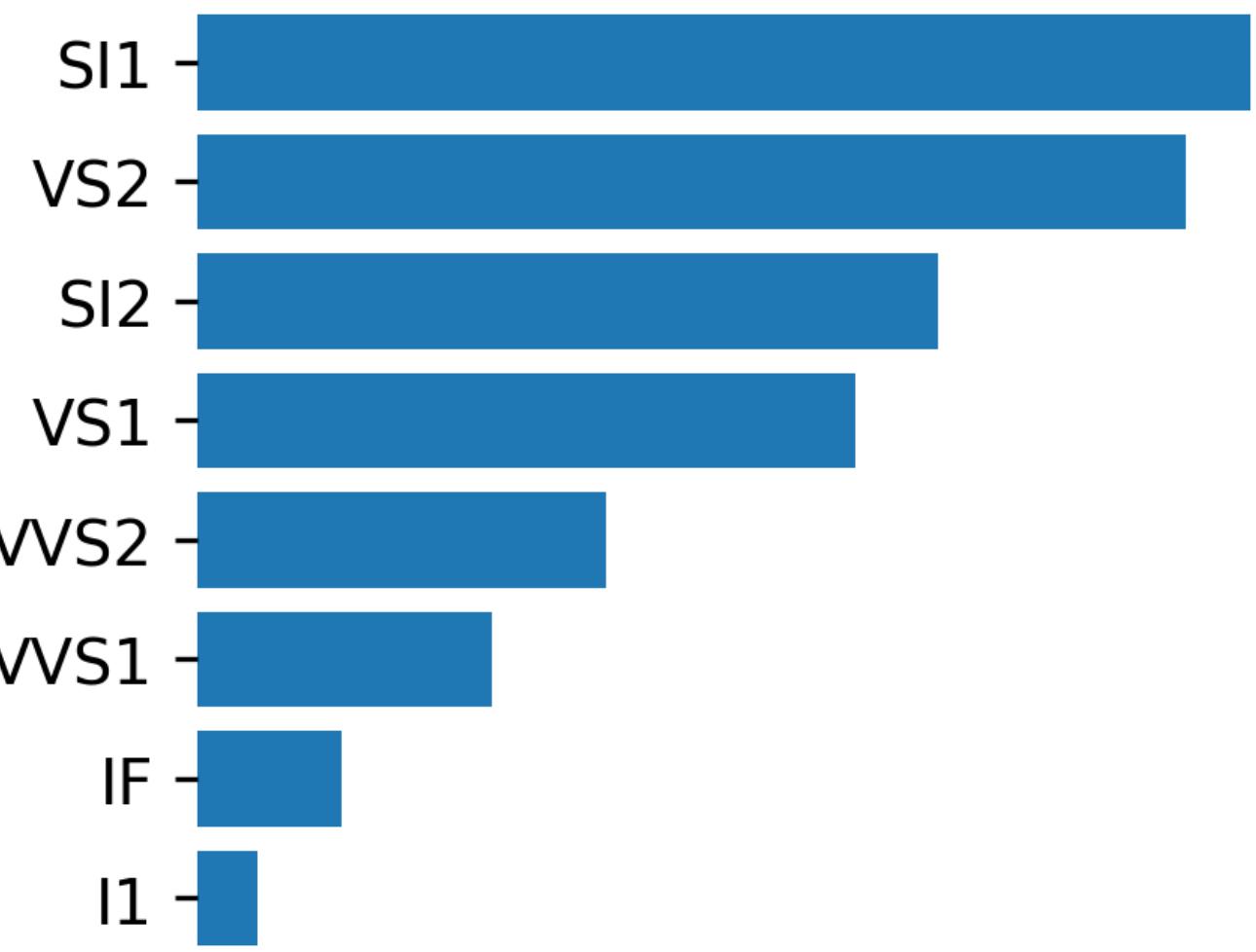
VS2

VS1

SI1

VS1

SI1

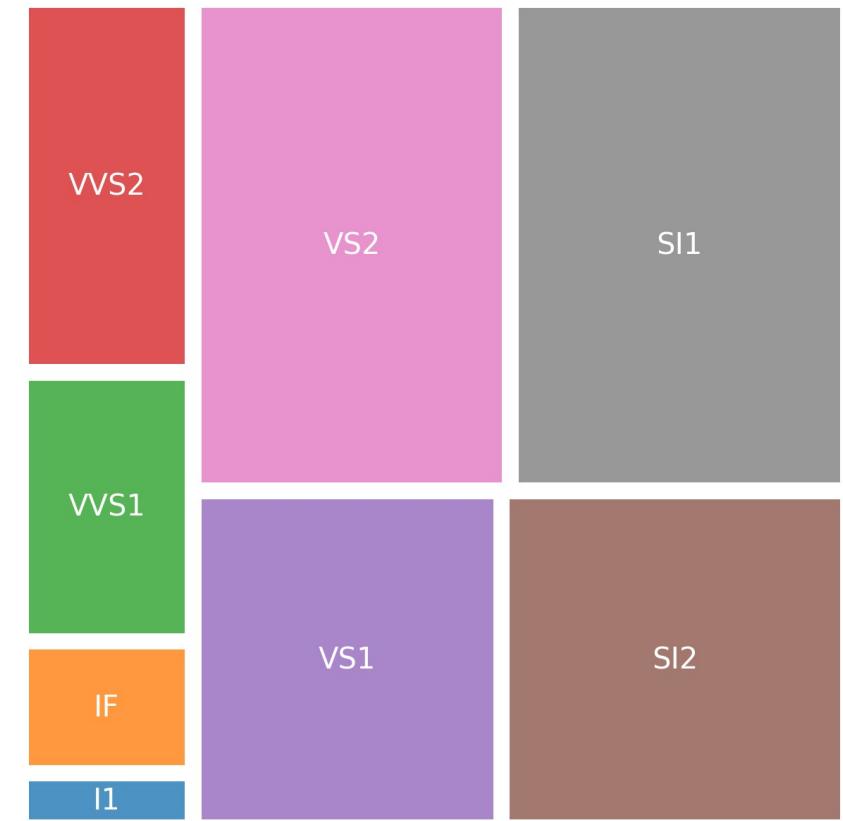
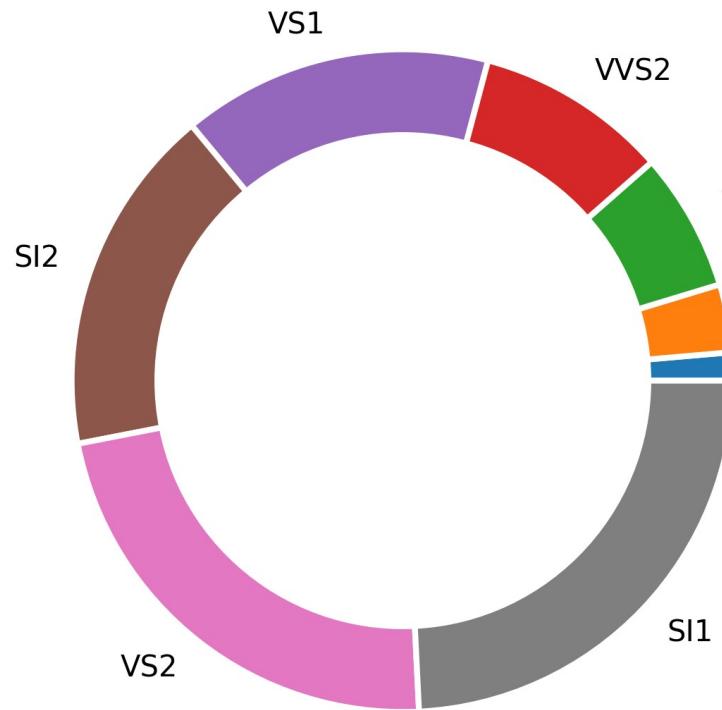
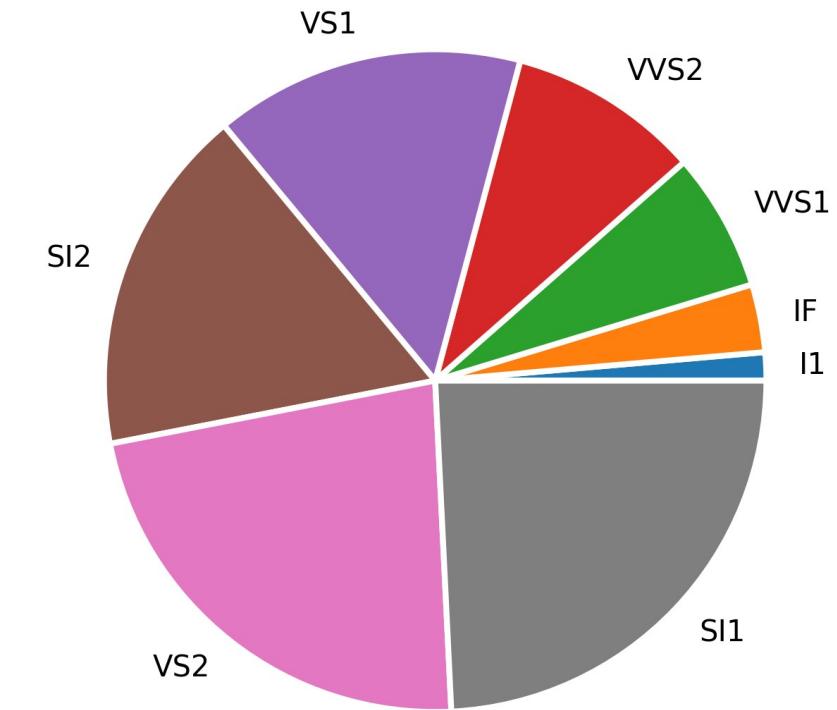


0

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

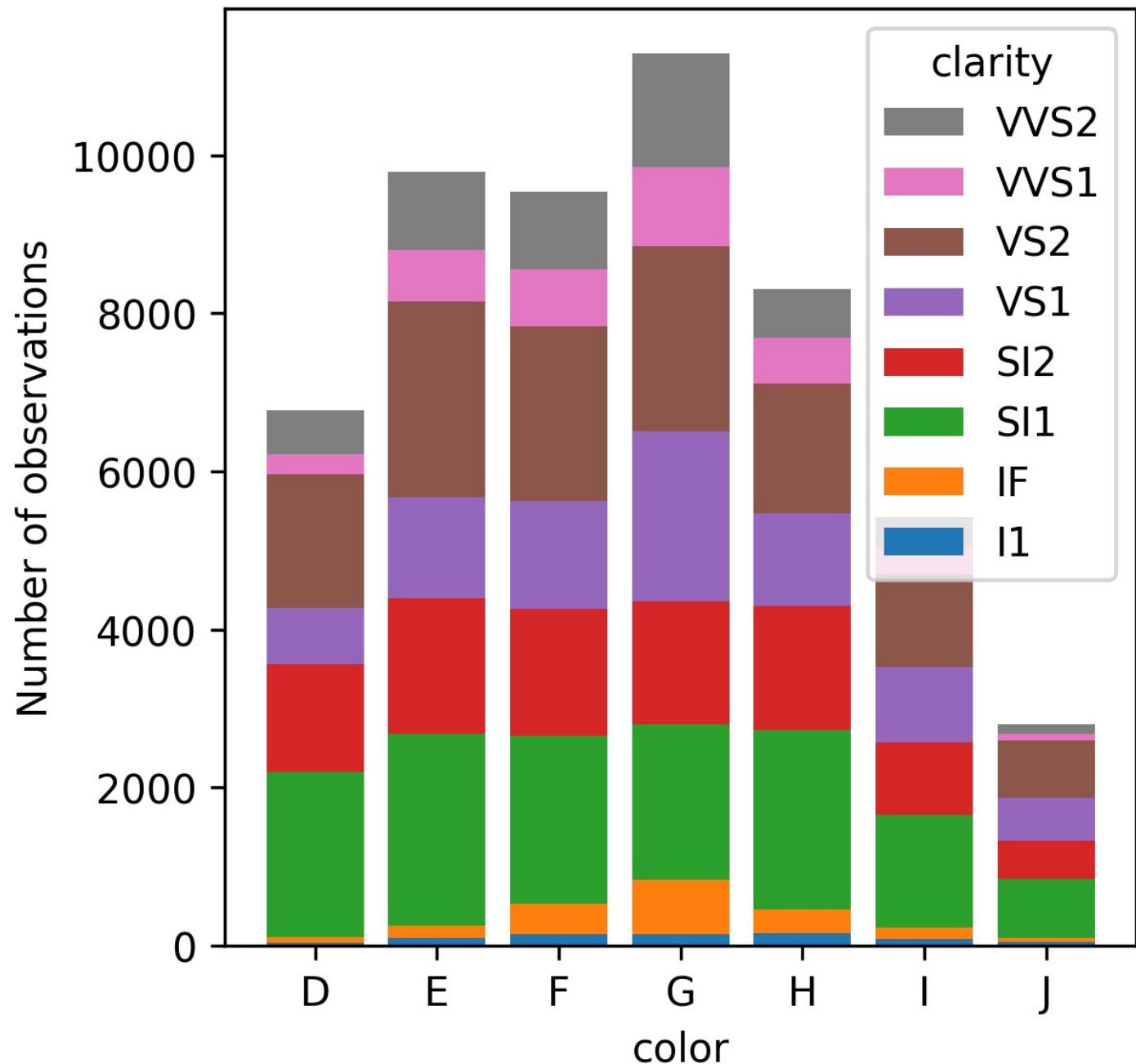
SI1 J

VS1 J

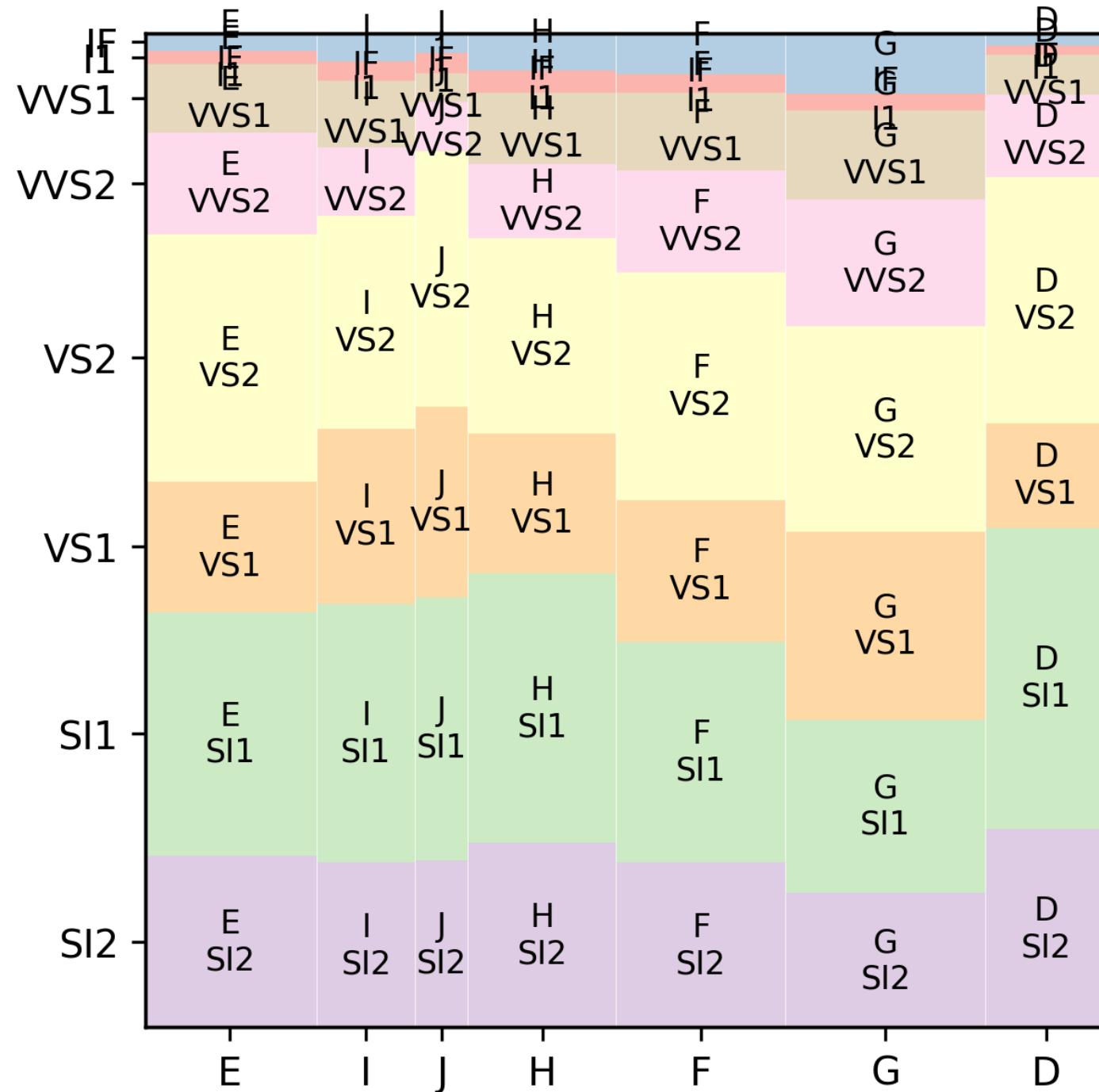
SI1 F

Diamond clarity and colour

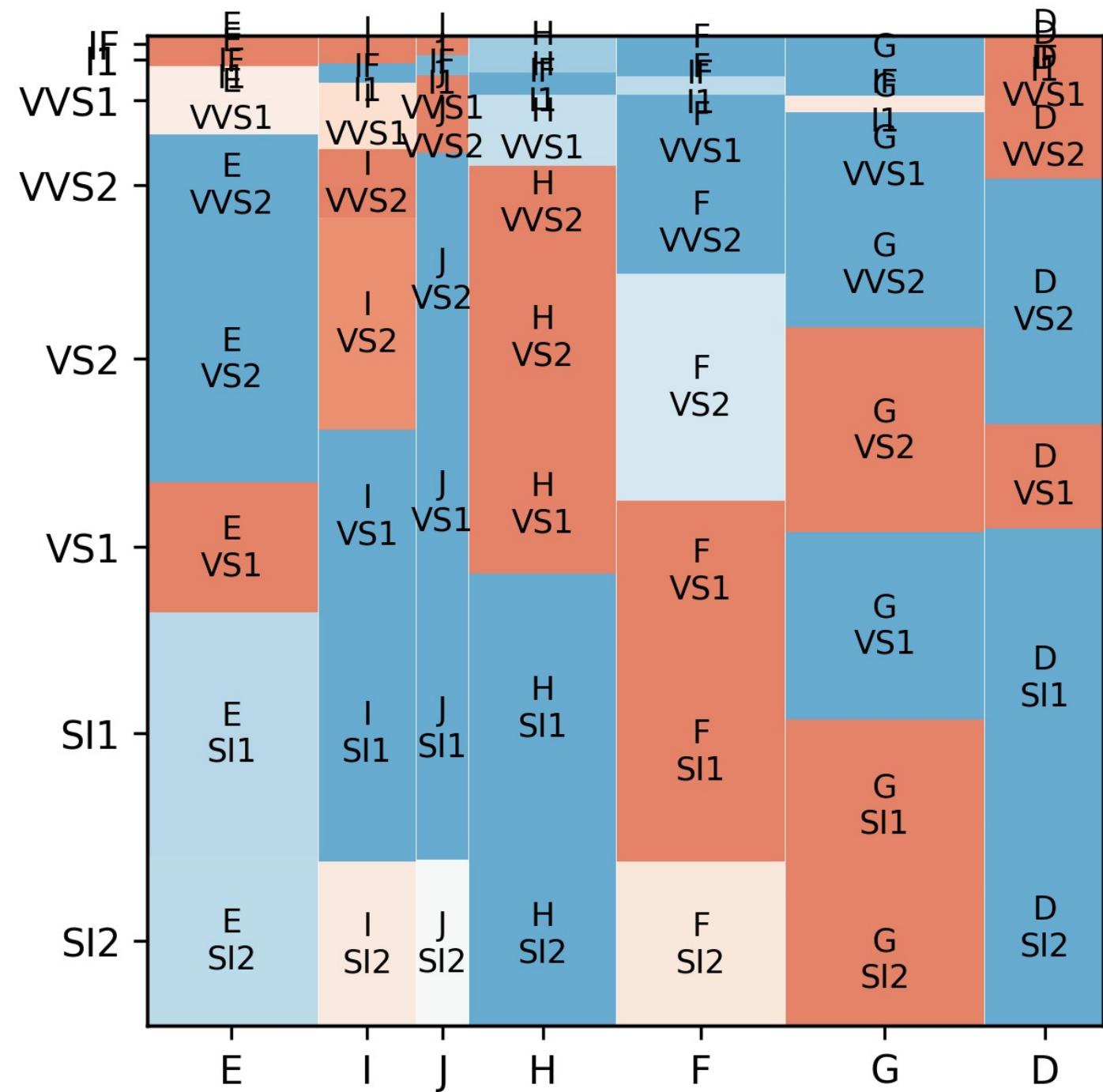
clarity	I1	IF	SI1	SI2	VS1	VS2	VVS1	VVS2
color								
D	42	73	2083	1370	705	1697	252	553
E	102	158	2426	1713	1281	2470	656	991
F	143	385	2131	1609	1364	2201	734	975
G	150	681	1976	1548	2148	2347	999	1443
H	162	299	2275	1563	1169	1643	585	608
I	92	143	1424	912	962	1169	355	365
J	50	51	750	479	542	731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

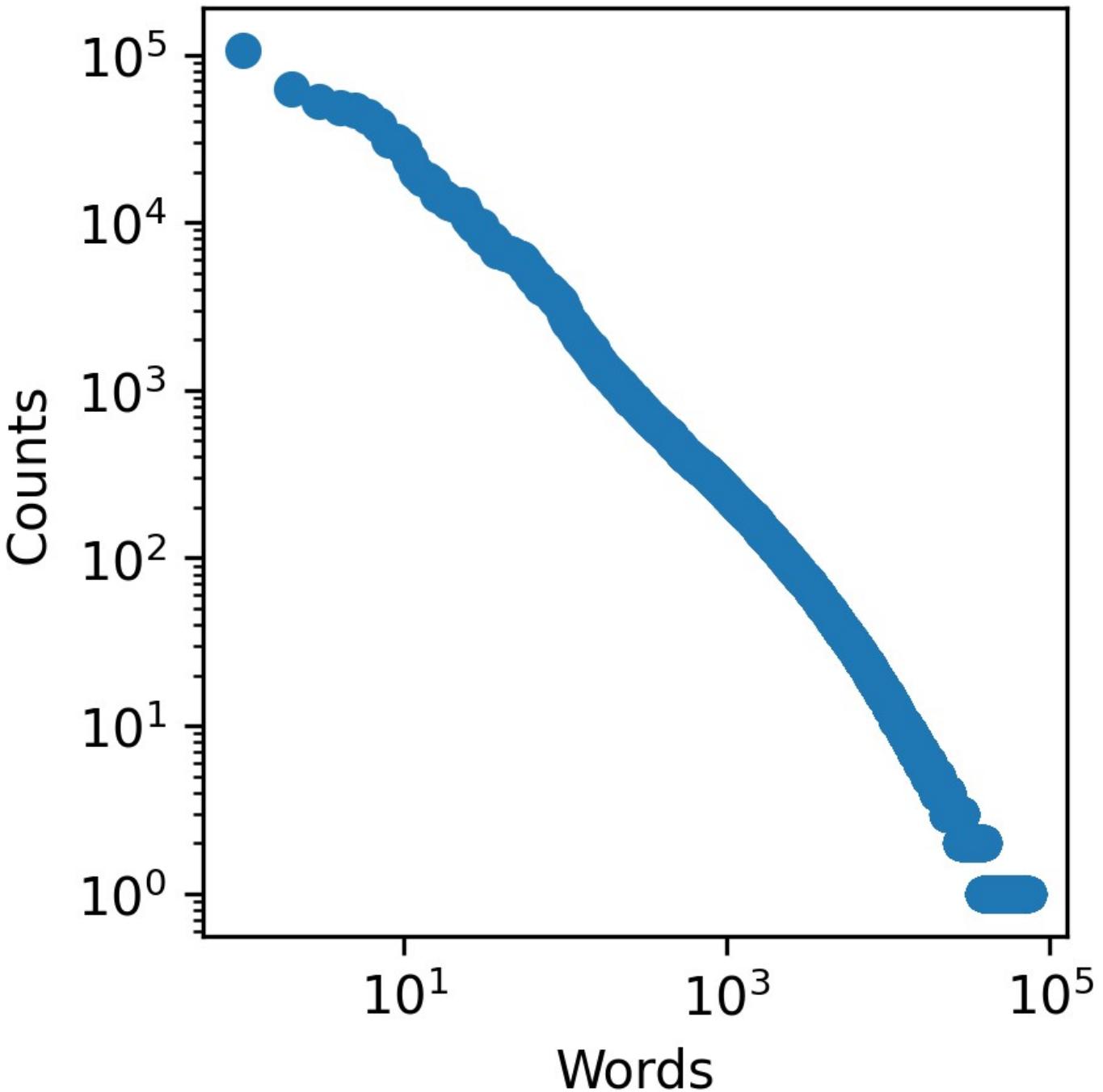
So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

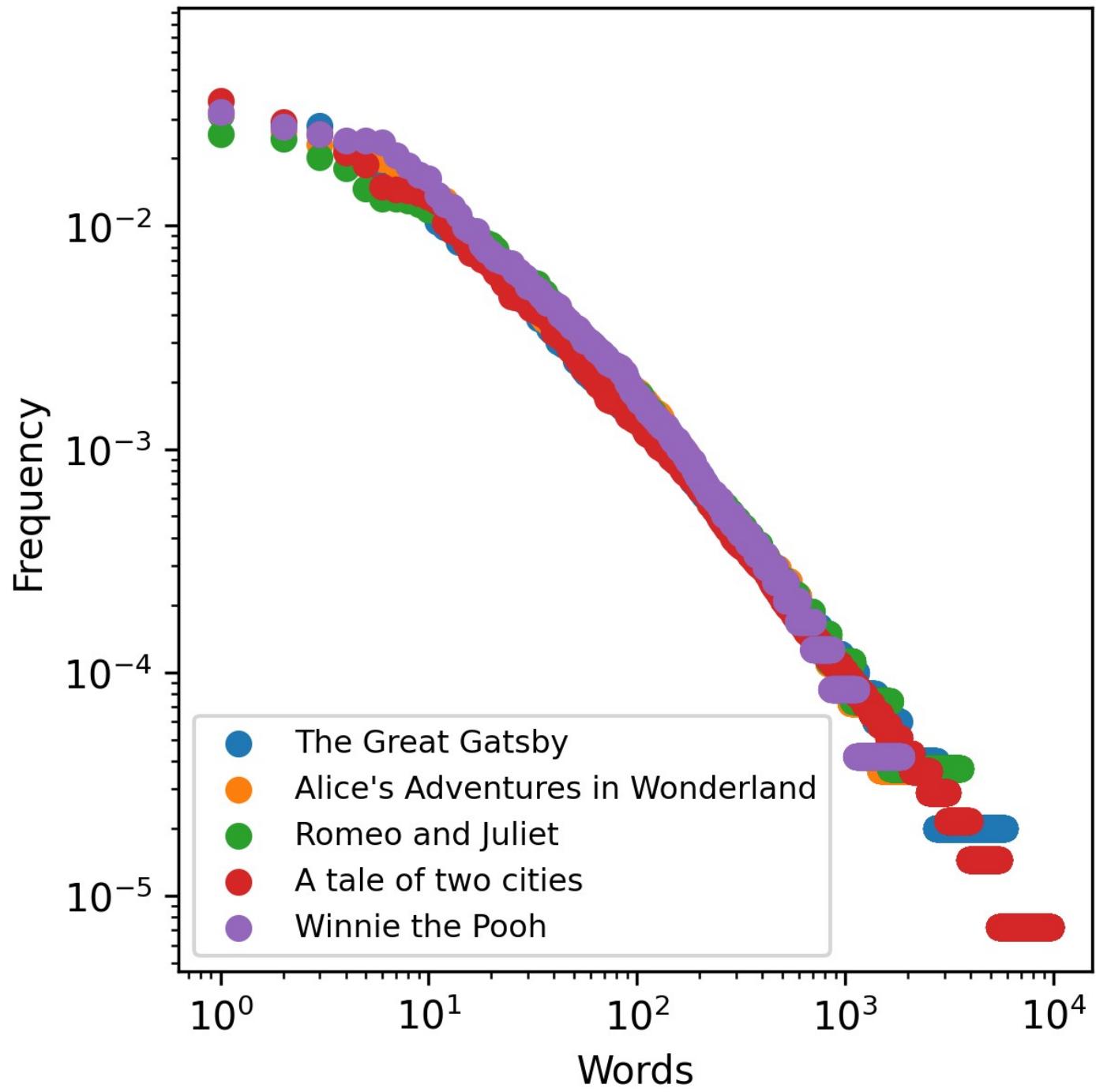
There was nothing so very remarkable in that; nor did Alice think it so very much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite 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.

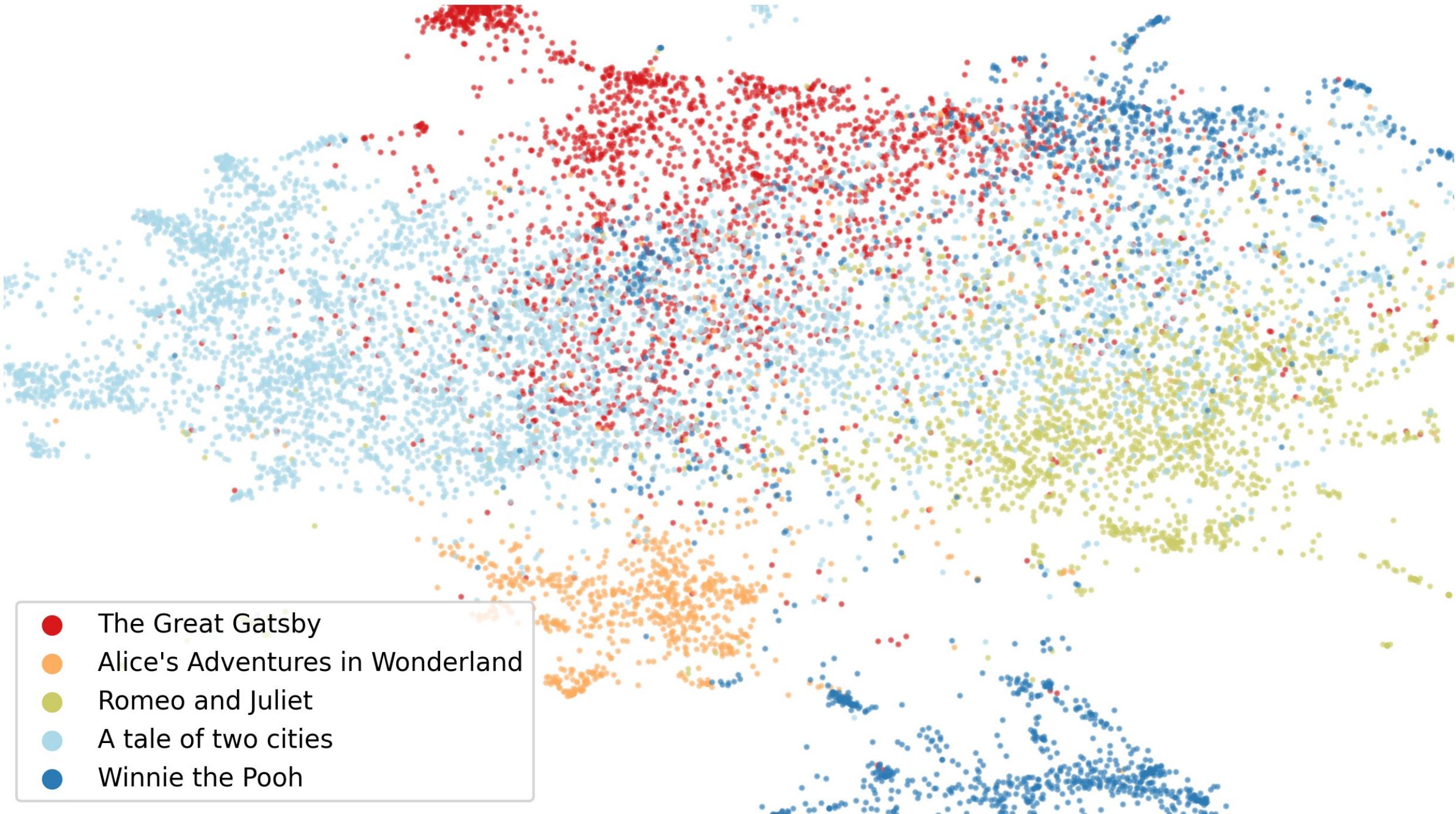
The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

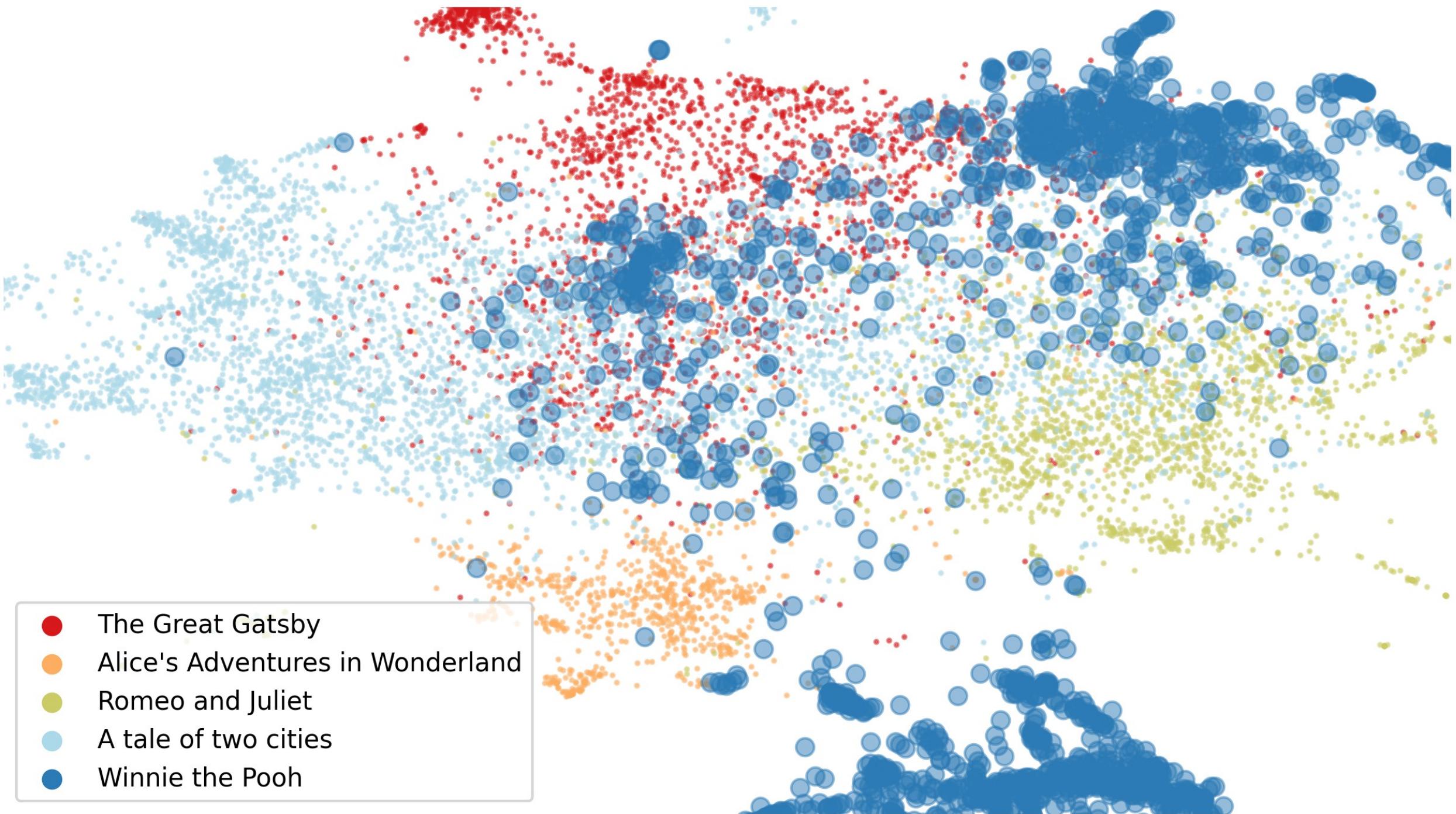
Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the 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.



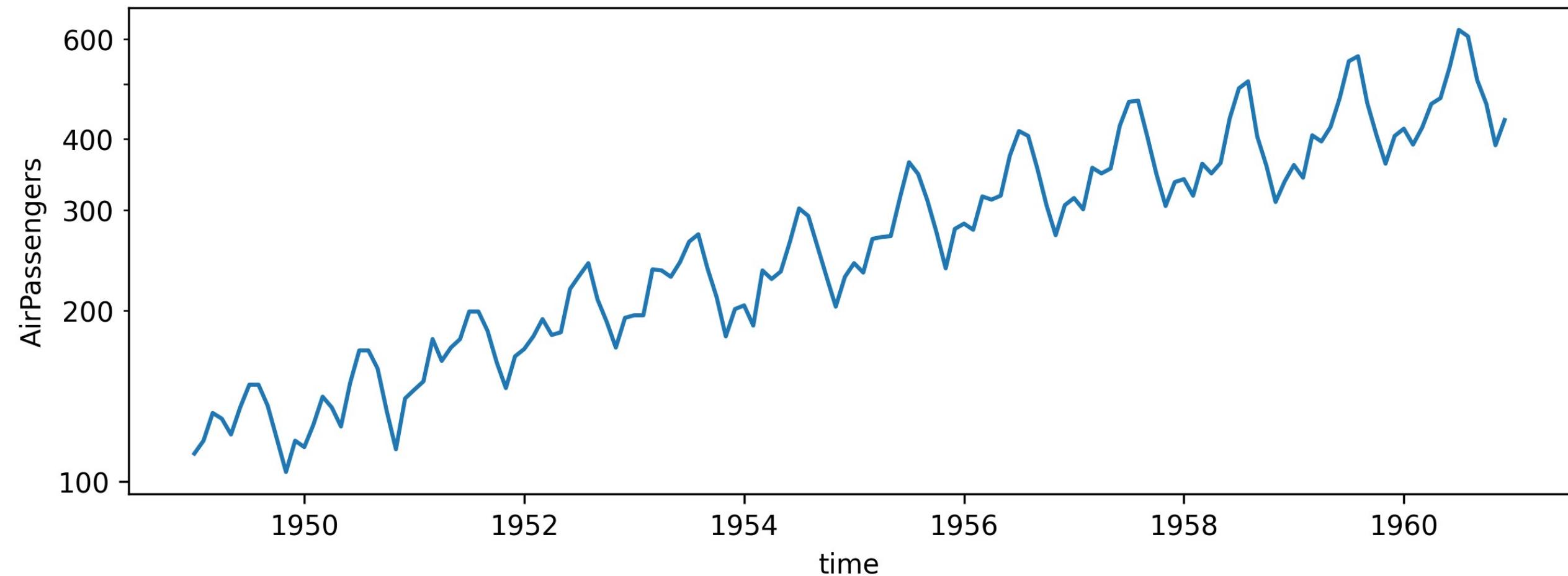


Sentences

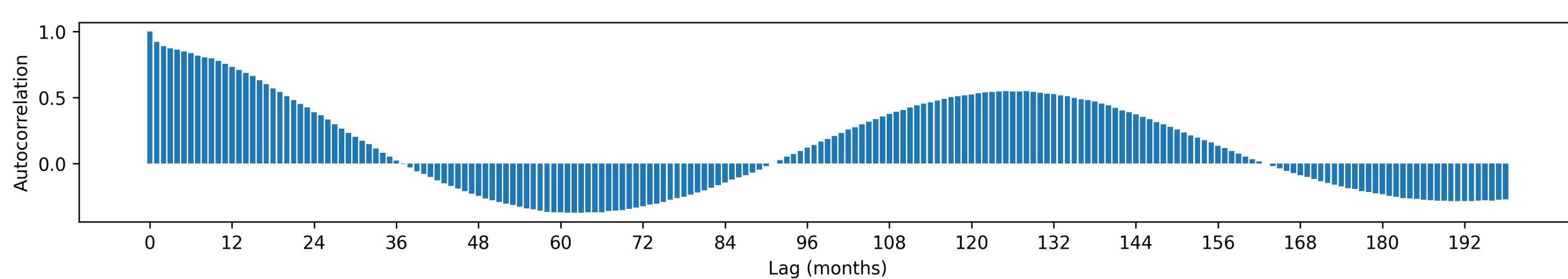
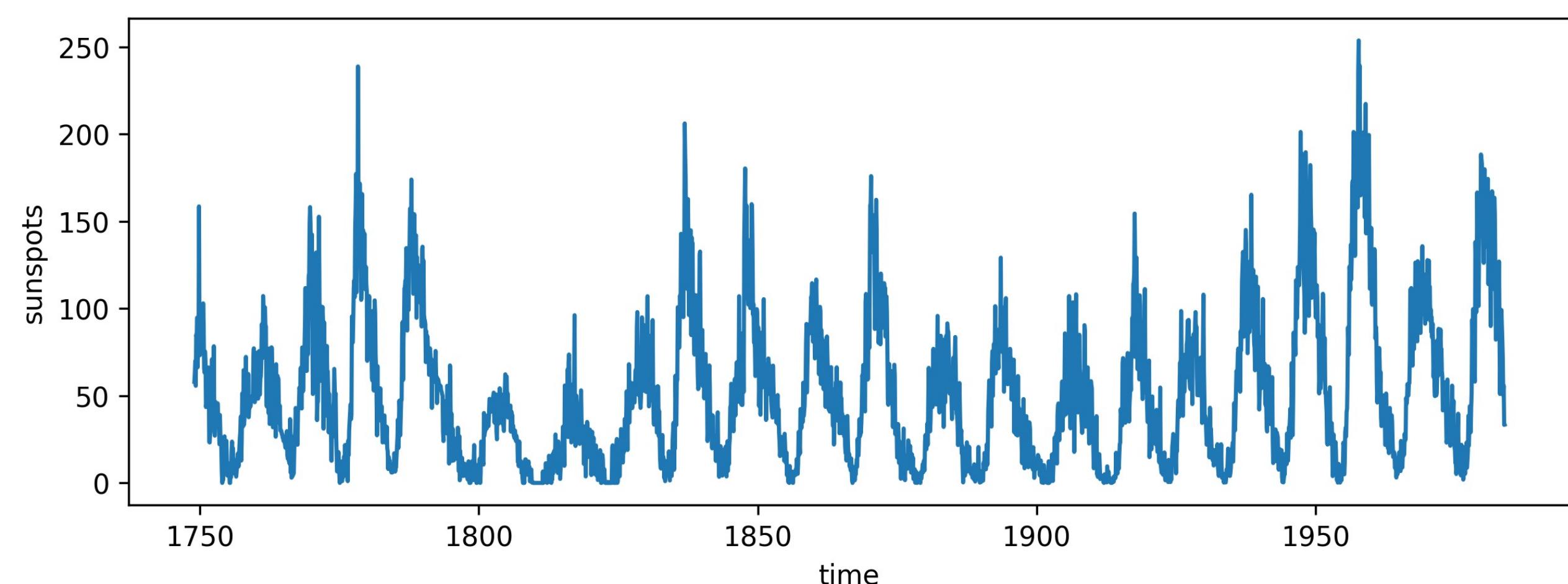




Air
passengers



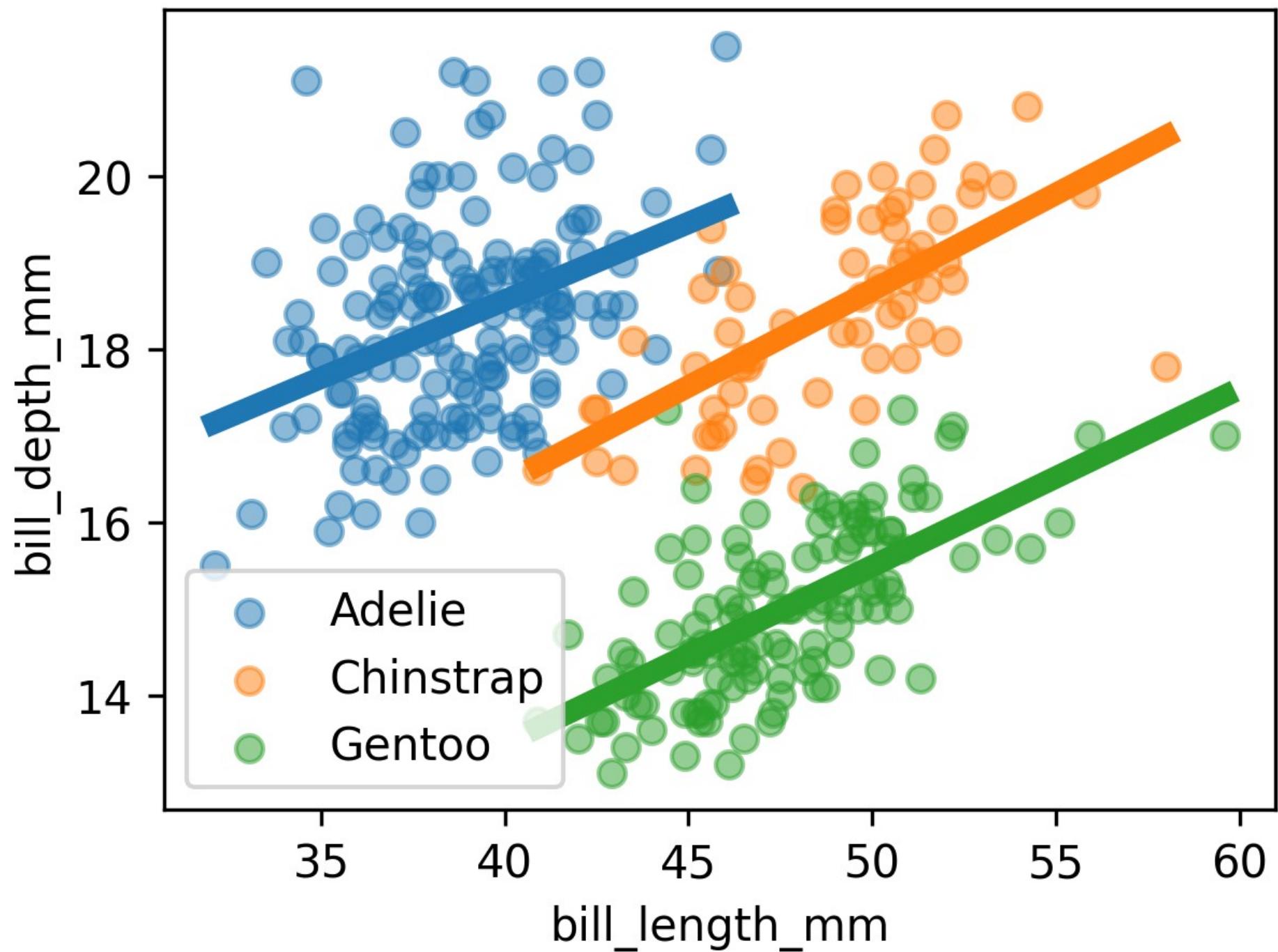
Sunspots



**Palmer
Penguins**



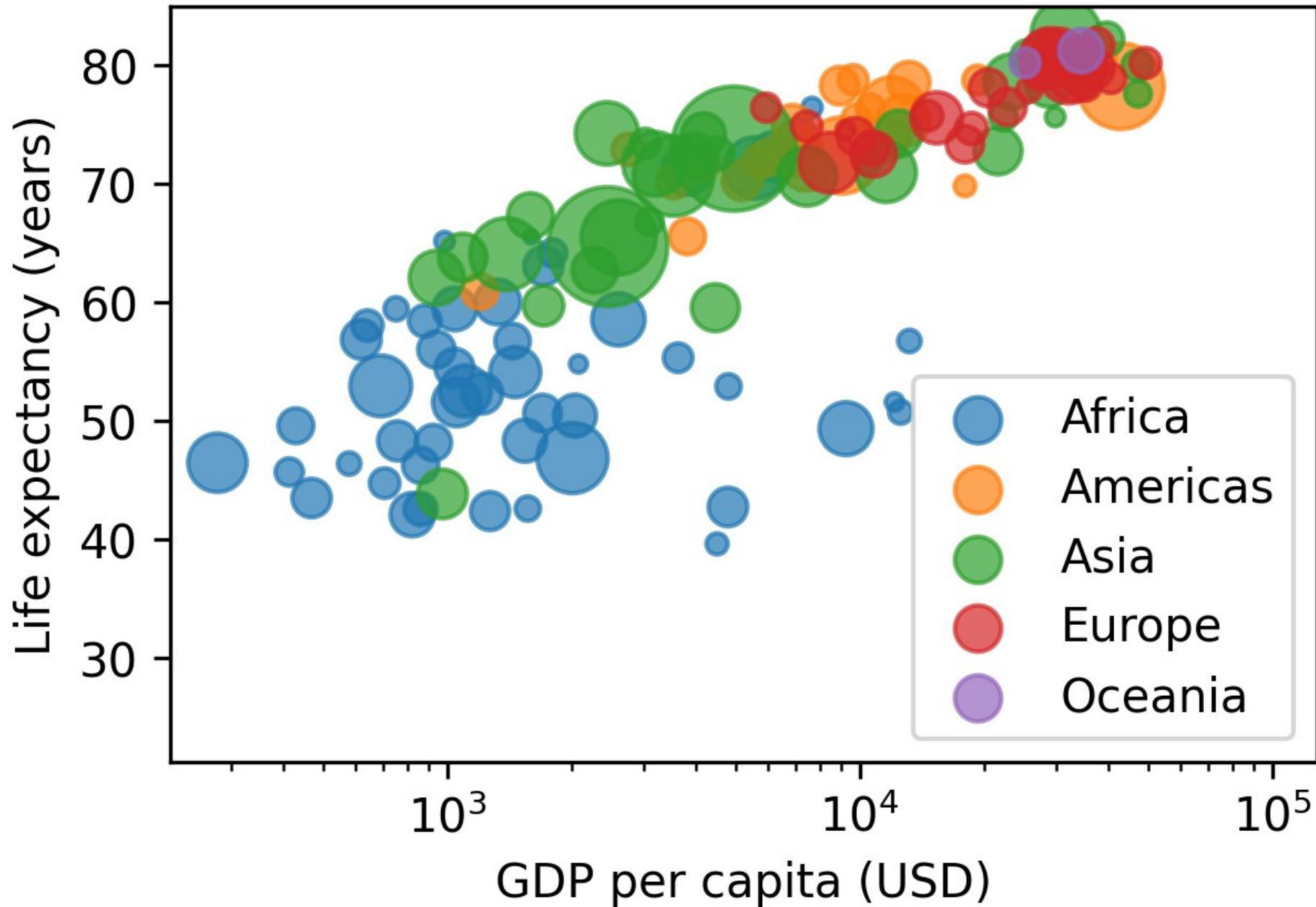
species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
Adelie	Torgersen	36.7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

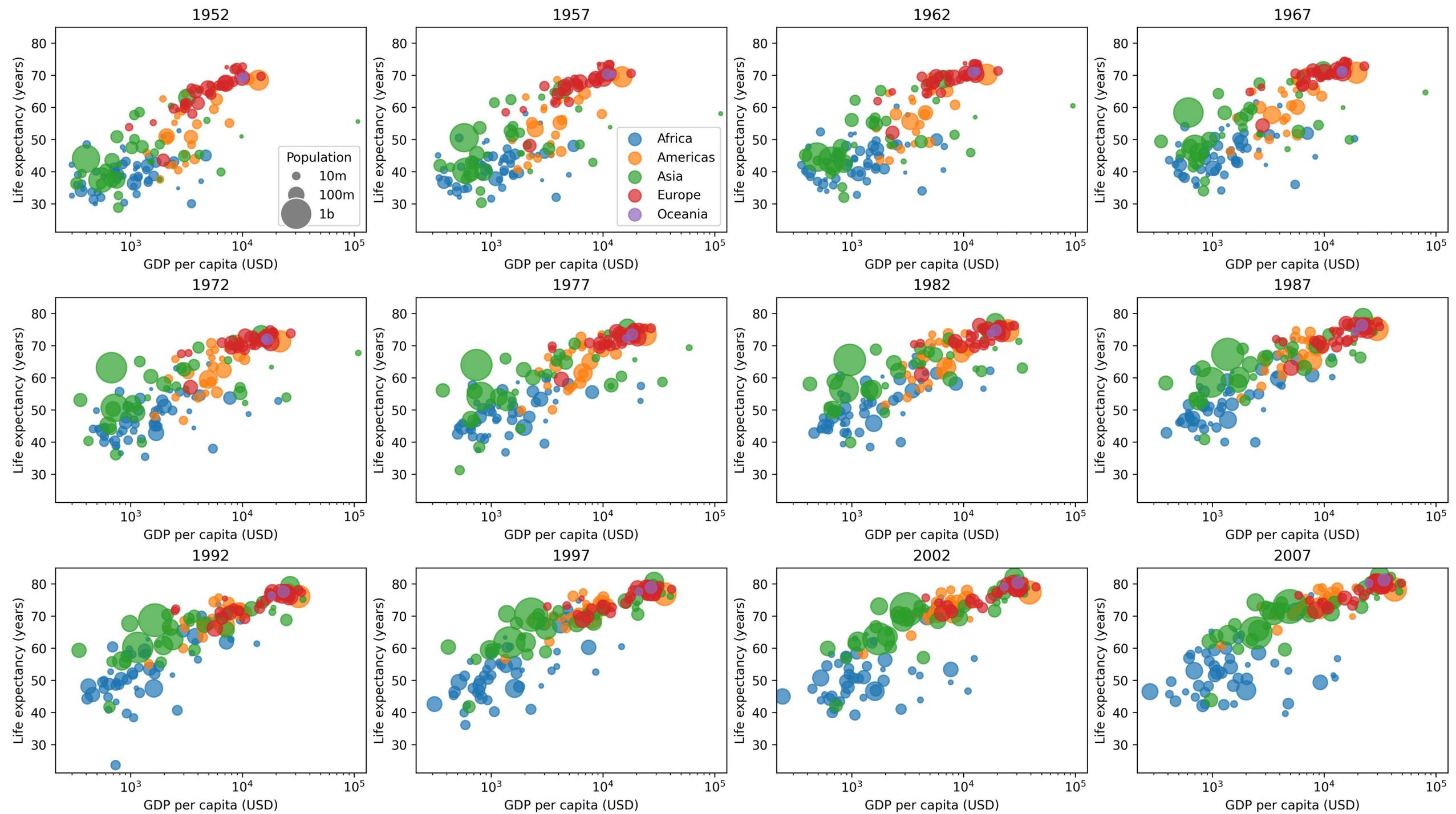


Gapminder

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





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
8 9 2 5 3
- 3 1 3 3 3
3 3 3 3 3
3 3 3 3 3

1000000000

۶۶۶۶۶۶

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

A hand-drawn diagram of a branching structure, possibly a tree or a network, with nodes labeled with numbers 1, 2, and 3. The structure consists of several main branches originating from a single point at the bottom right. One branch extends upwards and to the left, with several smaller branches extending from it. Nodes are labeled with the number 1 along this main branch. Another branch extends upwards and to the left, with nodes labeled with the number 2. A third branch extends upwards and to the left, with nodes labeled with the number 3. The drawing is done in black ink on a white background.

4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7

5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5
8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8

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

1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1

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

6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6

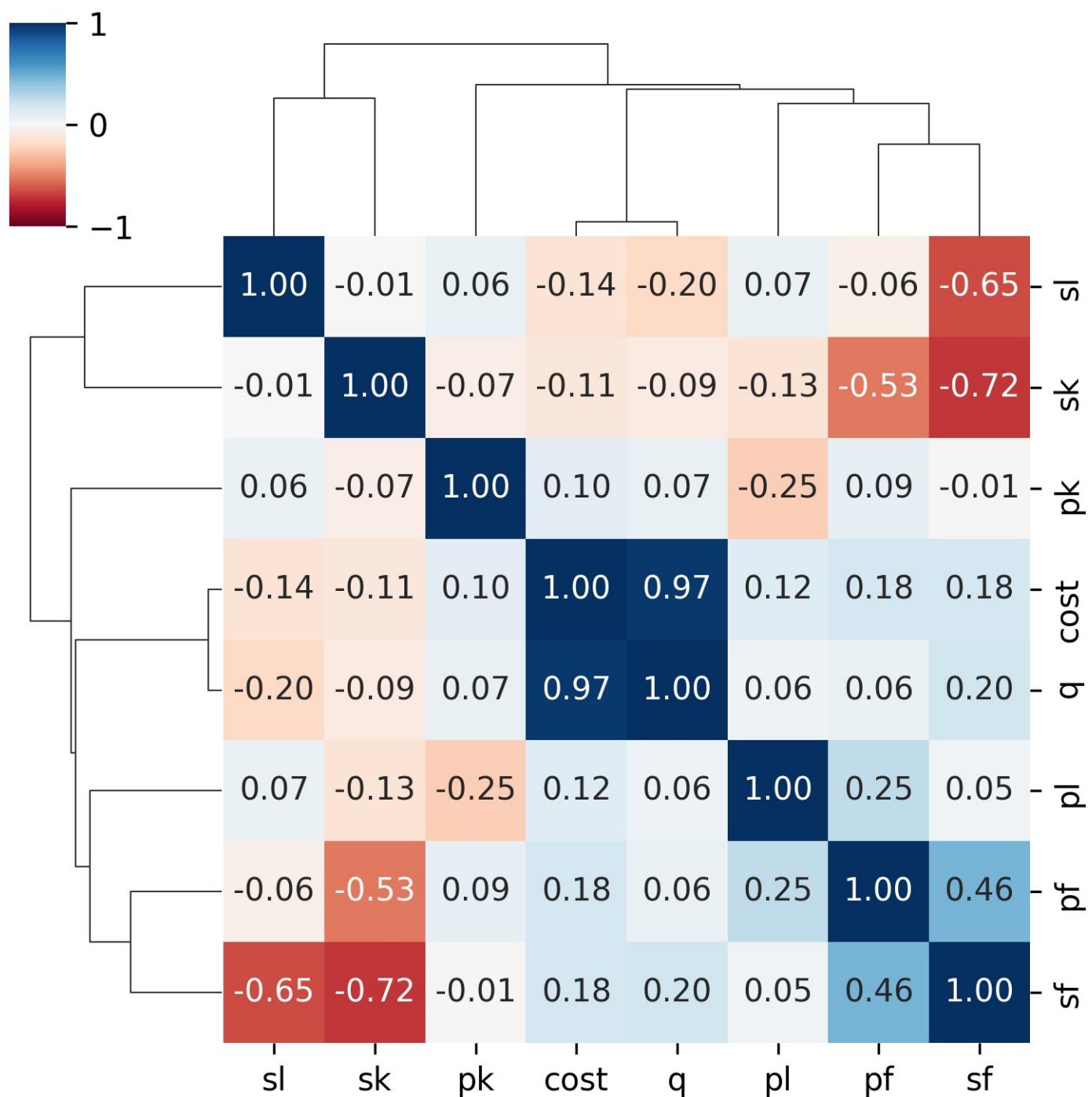
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

Images?



**Unused
Slides**

cost	1.00	0.97	0.12	-0.14	0.10	-0.11	0.18	0.18
q	0.97	1.00	0.06	-0.20	0.07	-0.09	0.06	0.20
pl	0.12	0.06	1.00	0.07	-0.25	-0.13	0.25	0.05
sl	-0.14	-0.20	0.07	1.00	0.06	-0.01	-0.06	-0.65
pk	0.10	0.07	-0.25	0.06	1.00	-0.07	0.09	-0.01
sk	-0.11	-0.09	-0.13	-0.01	-0.07	1.00	-0.53	-0.72
pf	0.18	0.06	0.25	-0.06	0.09	-0.53	1.00	0.46
sf	0.18	0.20	0.05	-0.65	-0.01	-0.72	0.46	1.00



```
x = sklearn.decomposition.PCA(2).  
    fit_transform(dd)
```

```
fig, axs = plt.subplots( 3, 3 )
```

```
for i, column in enumerate(d.columns):
```

```
    ax = axs.flatten()[i]
```

```
    ax.scatter(
```

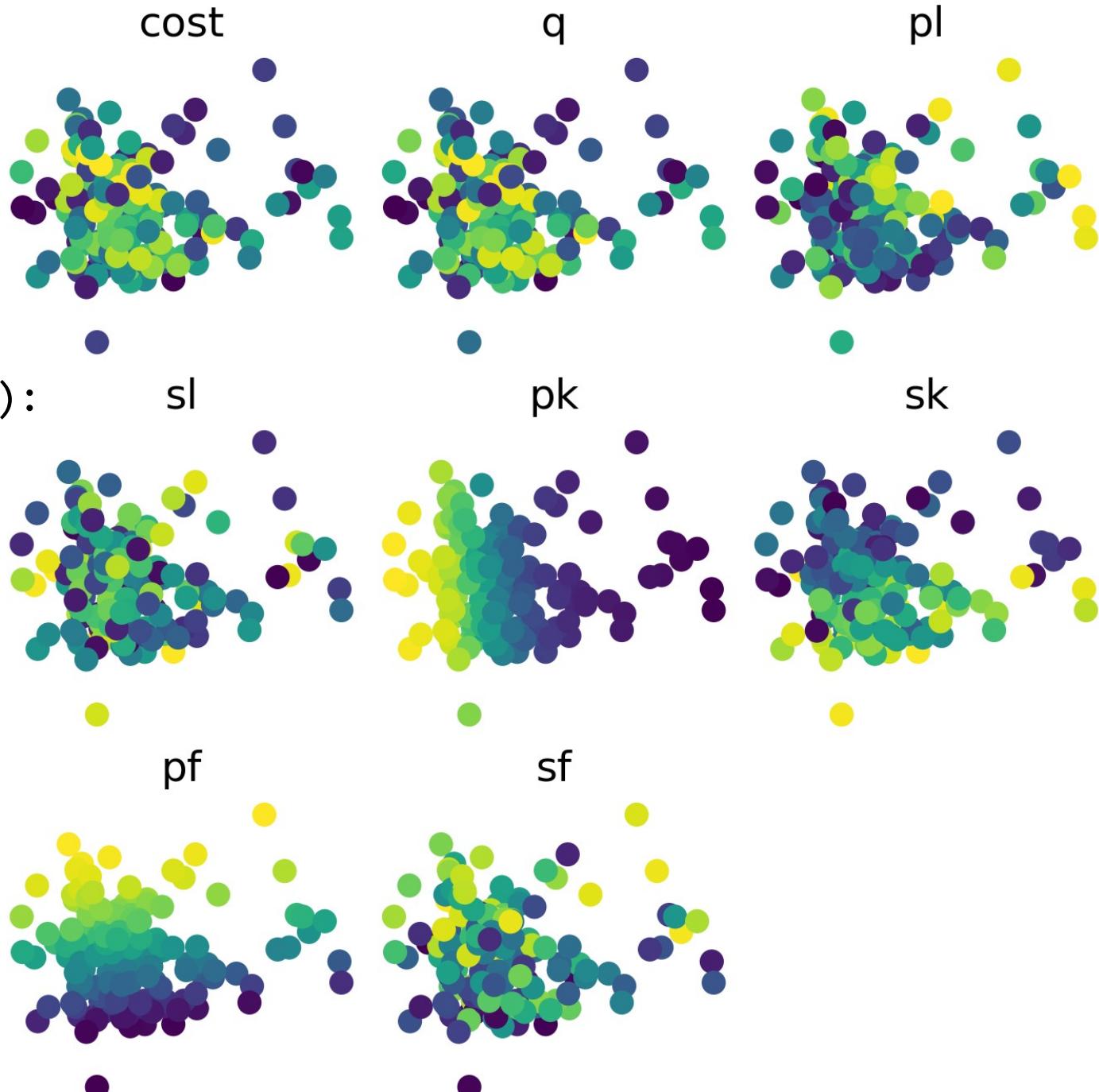
```
        X[:,0], X[:,1],  
        c = d[column],
```

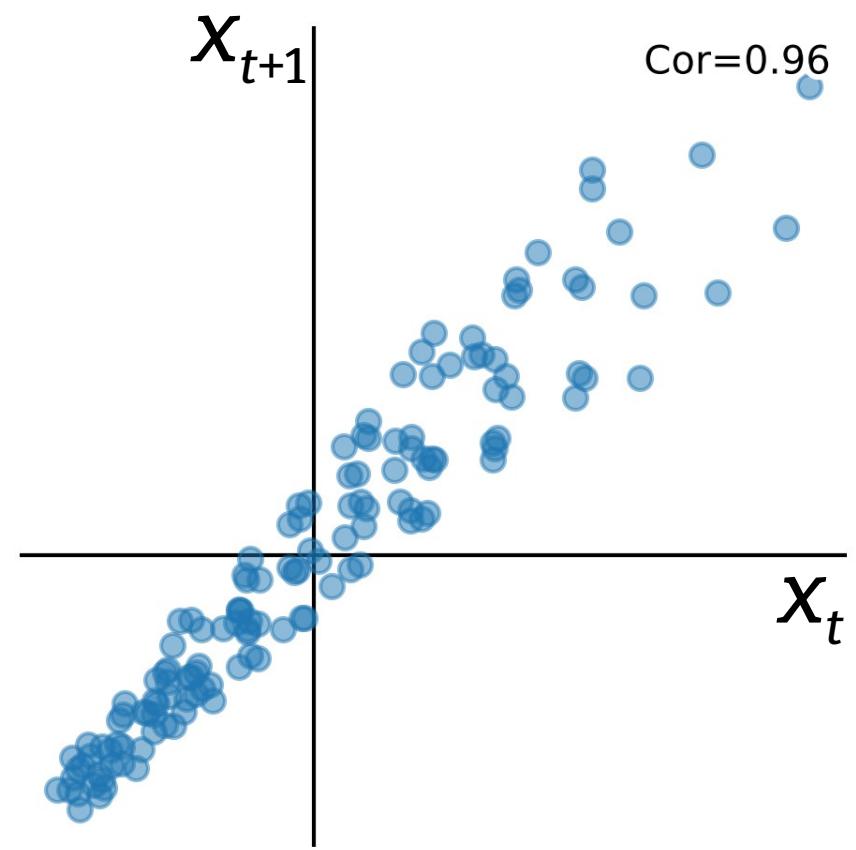
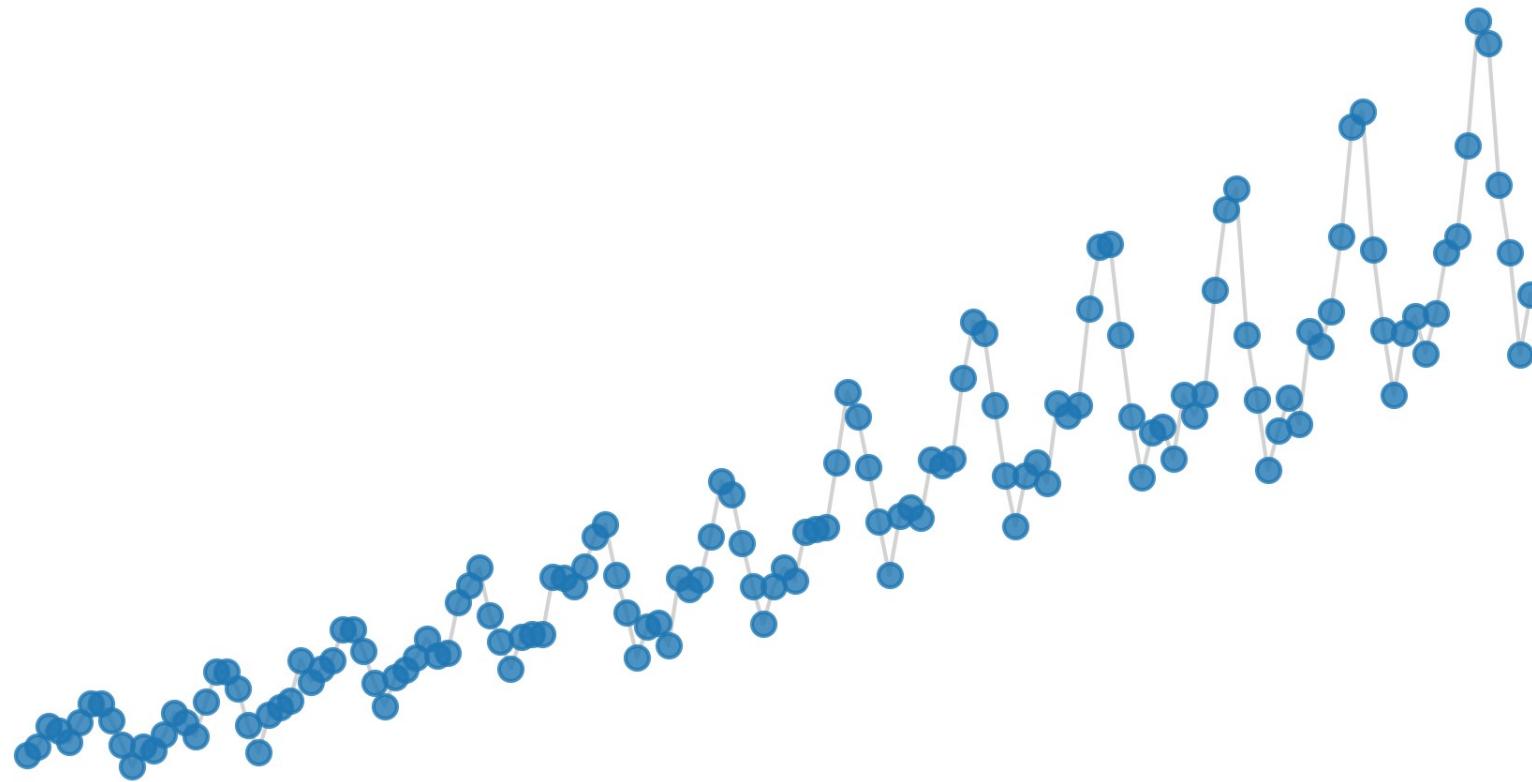
```
)
```

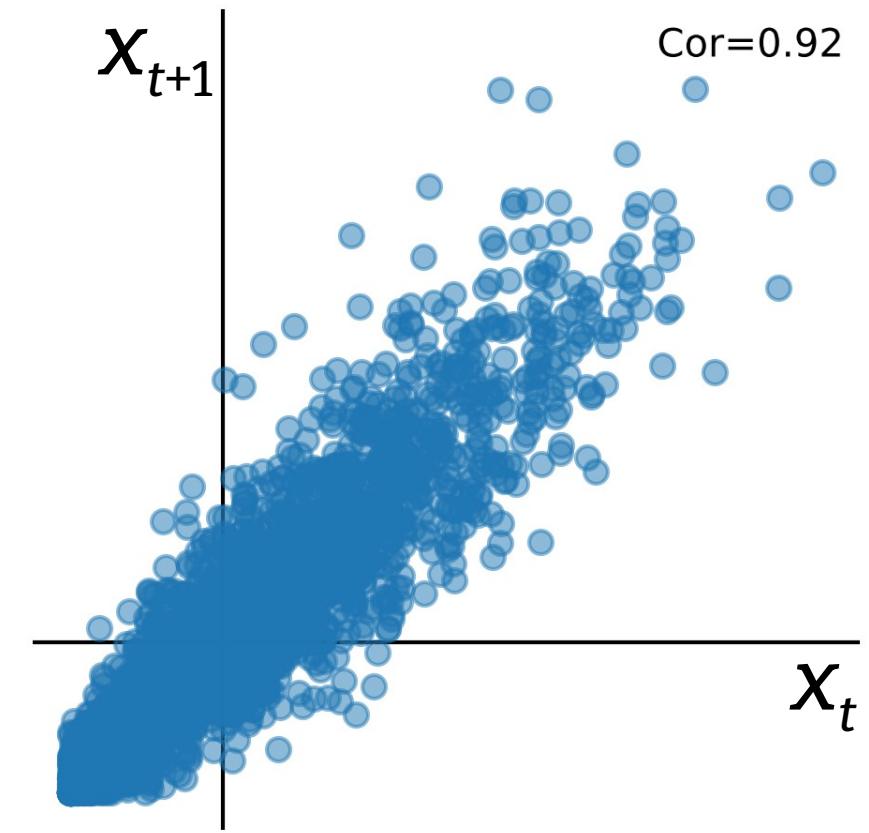
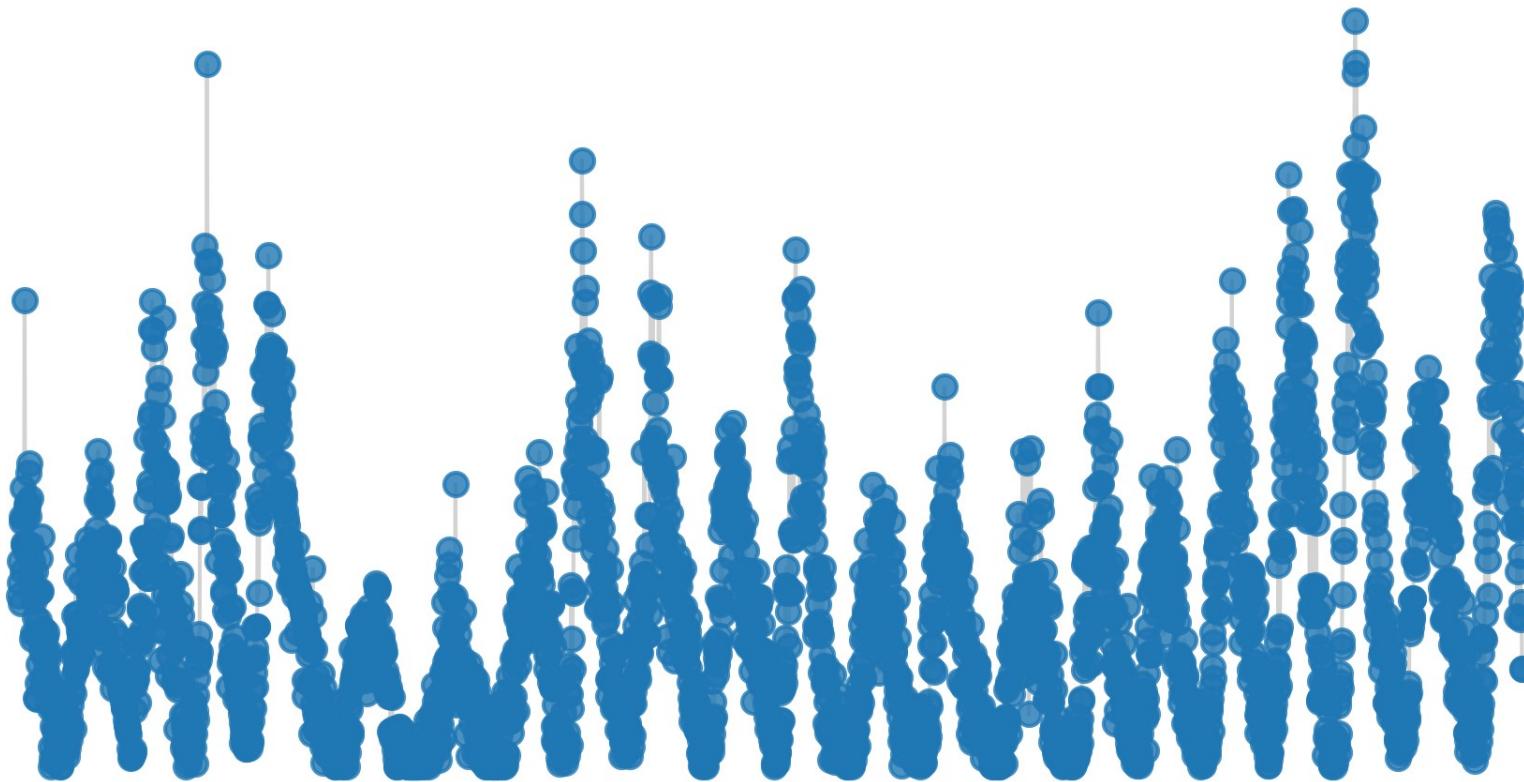
```
    ax.axis('off')
```

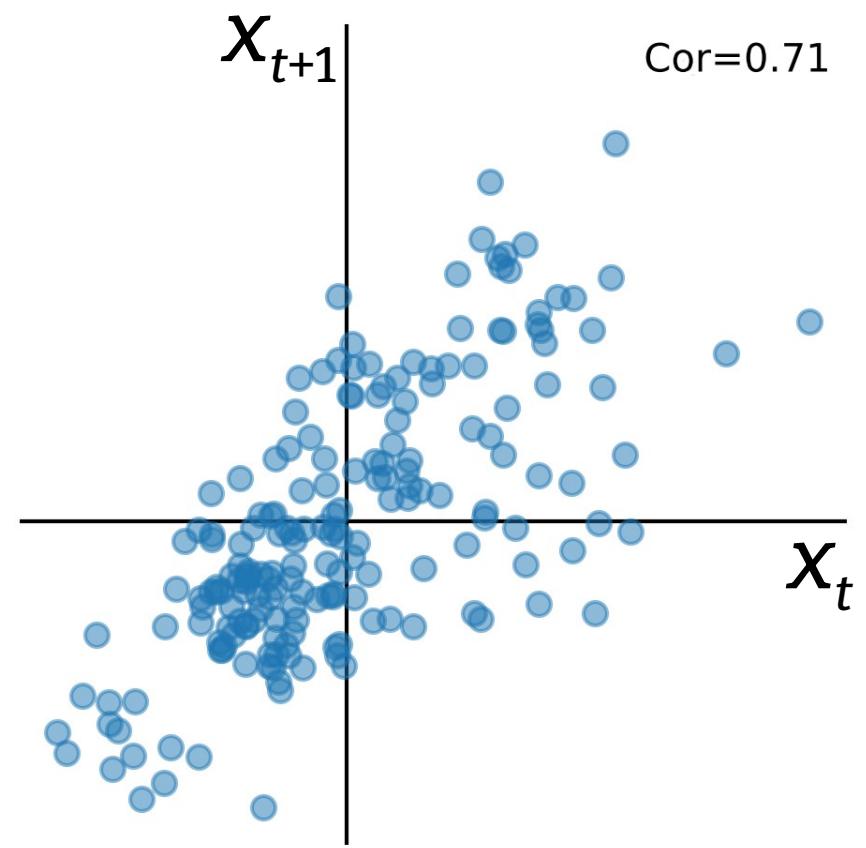
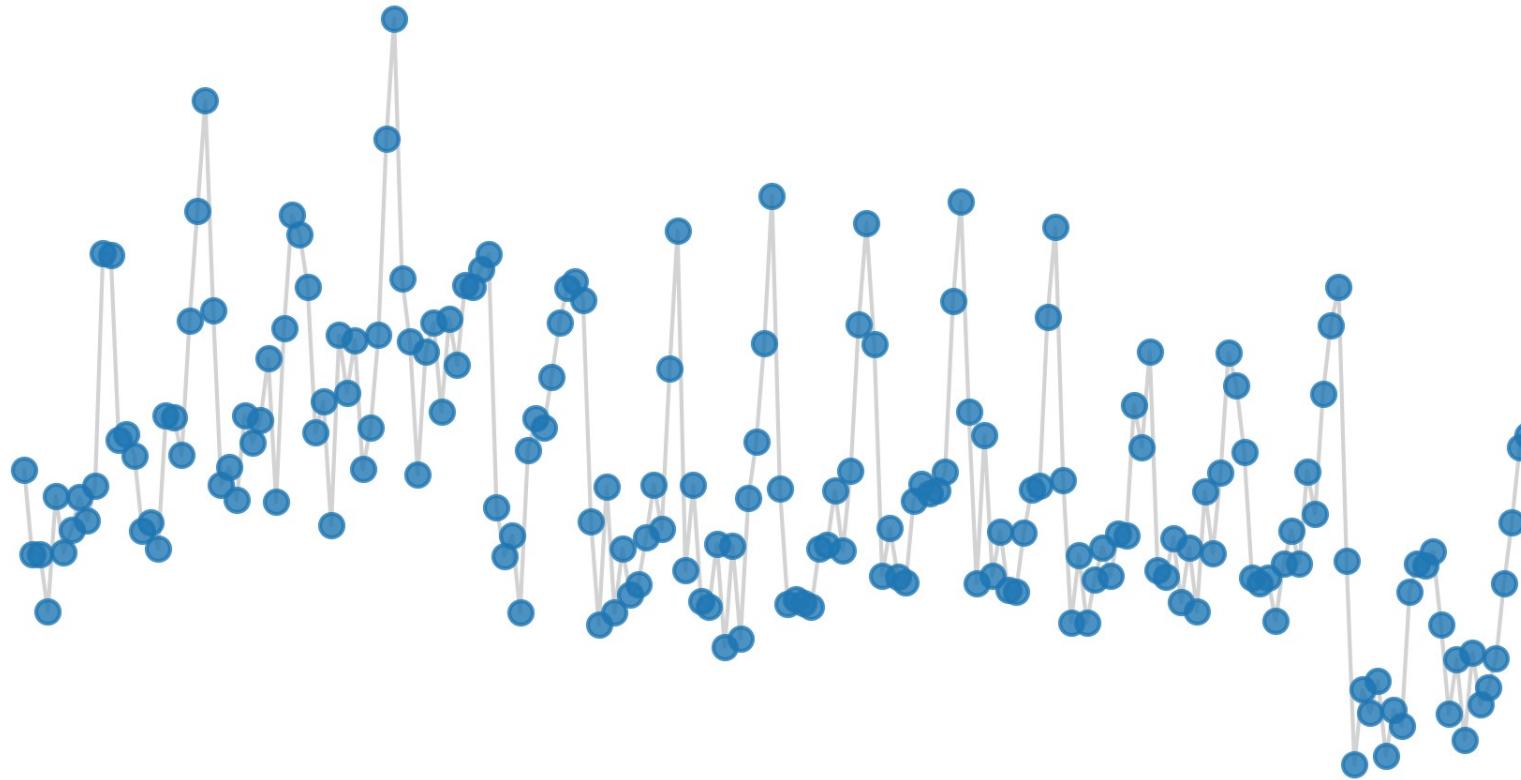
```
    ax.set_title(column)
```

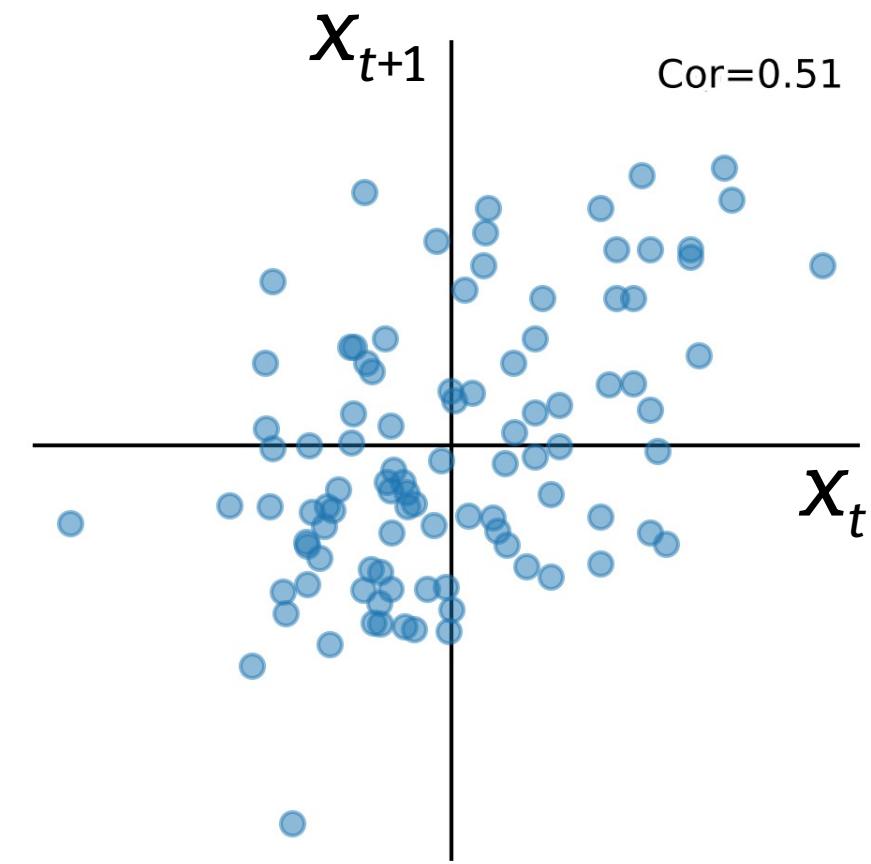
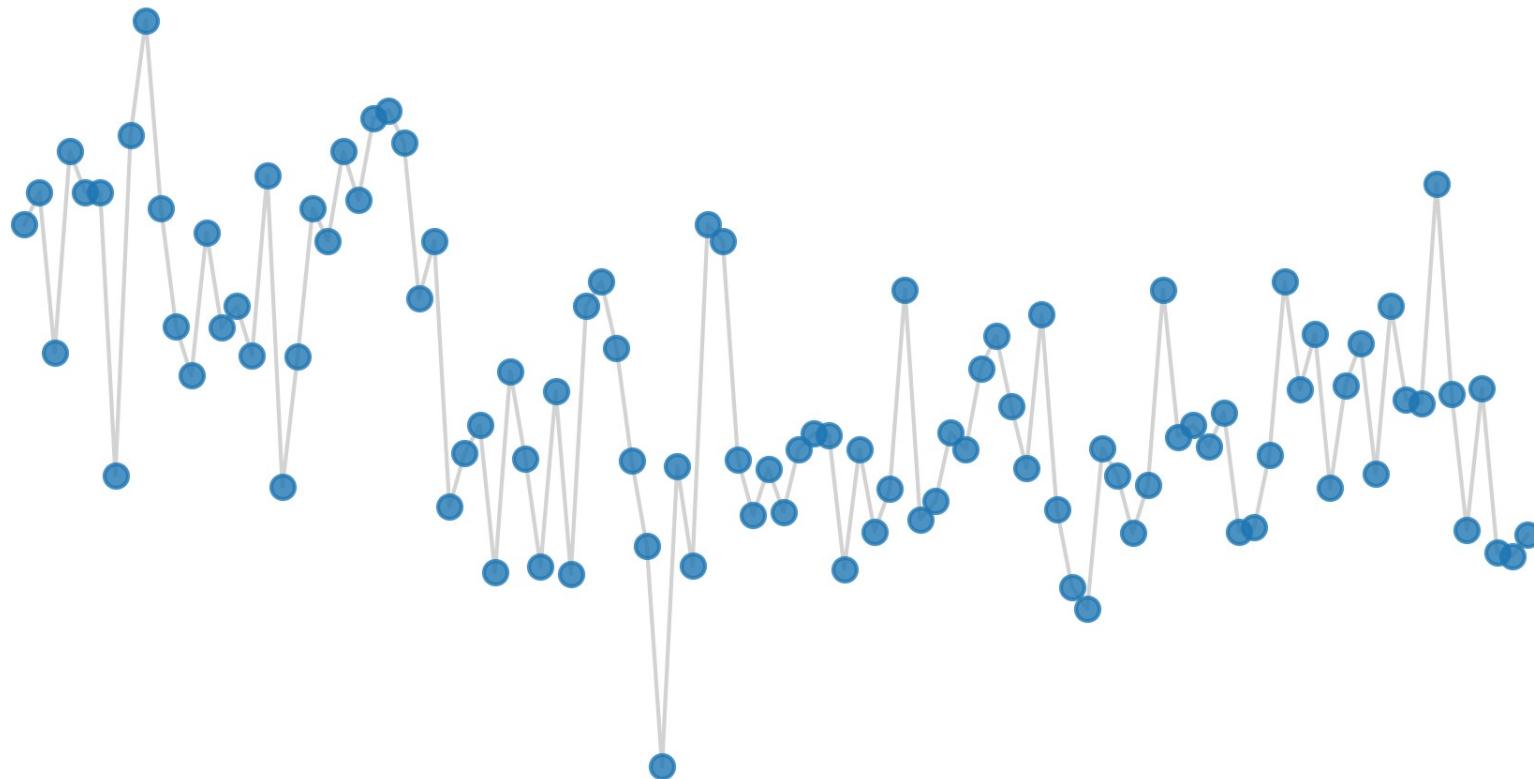
```
plt.show()
```

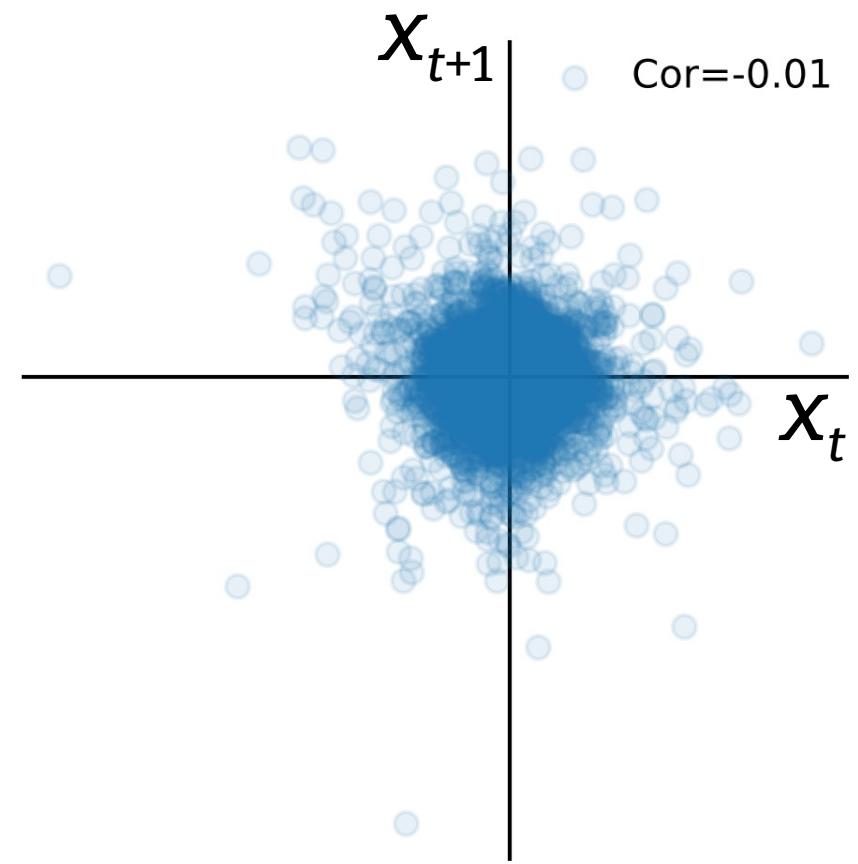
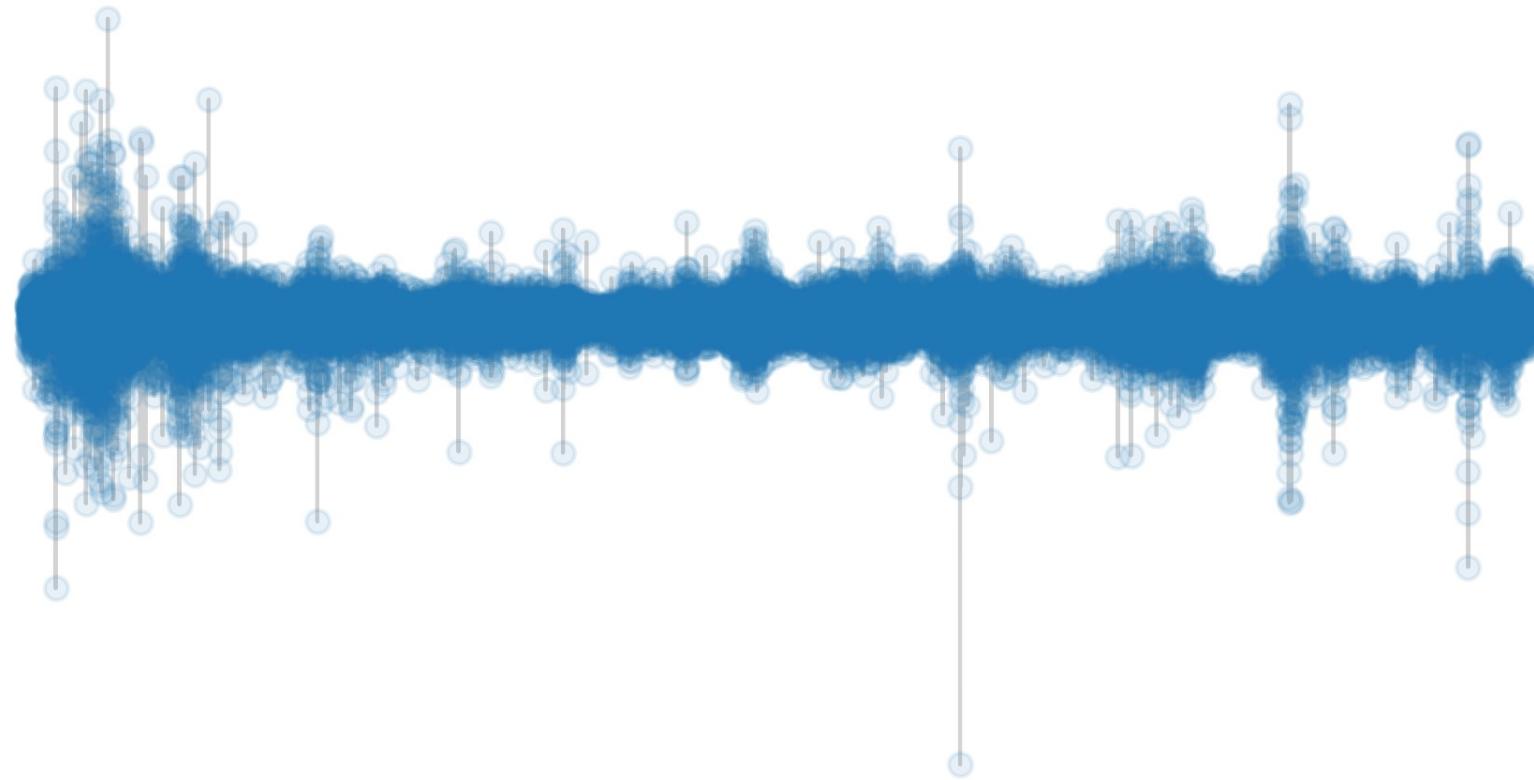


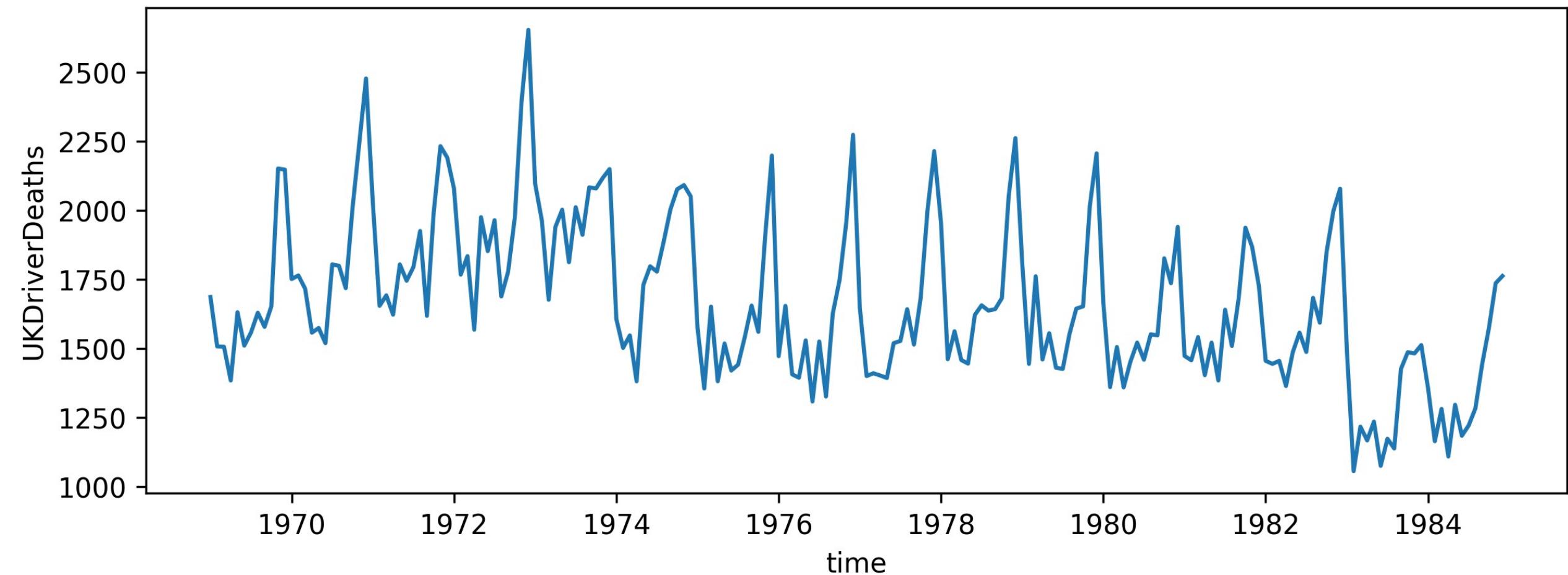


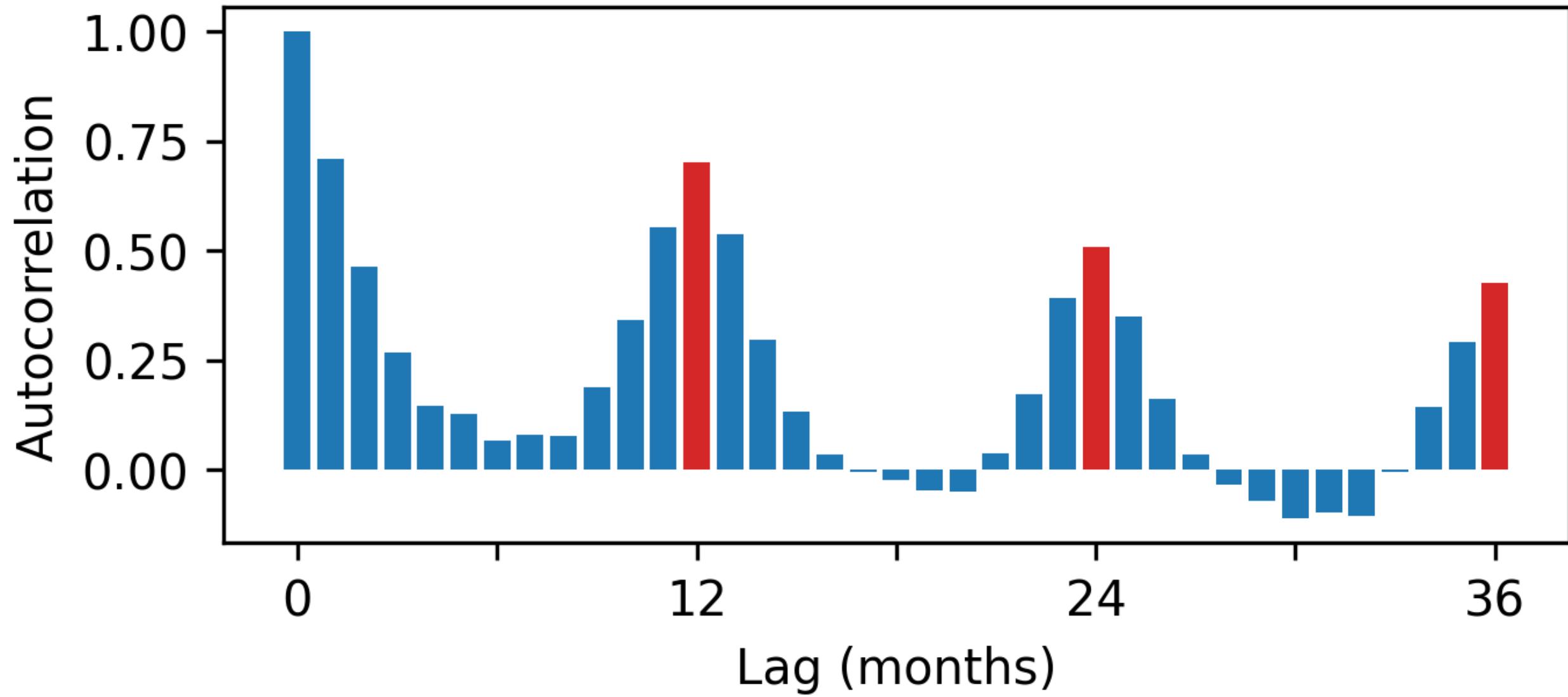


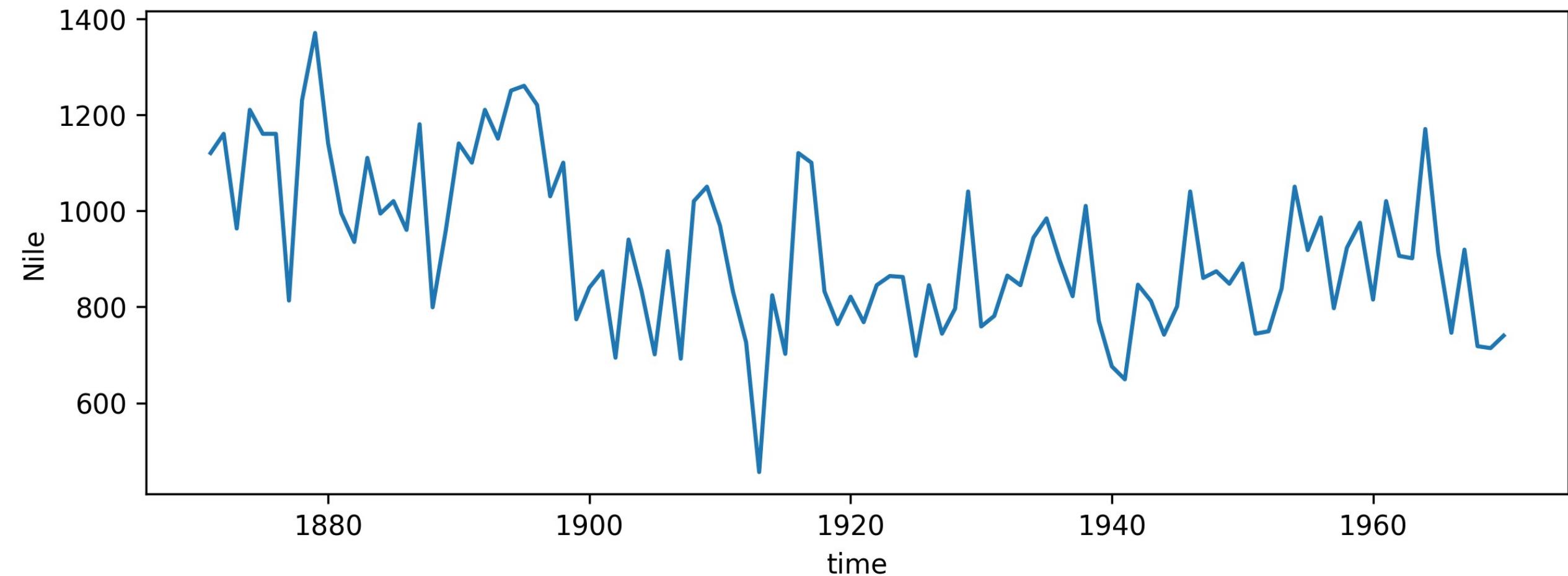


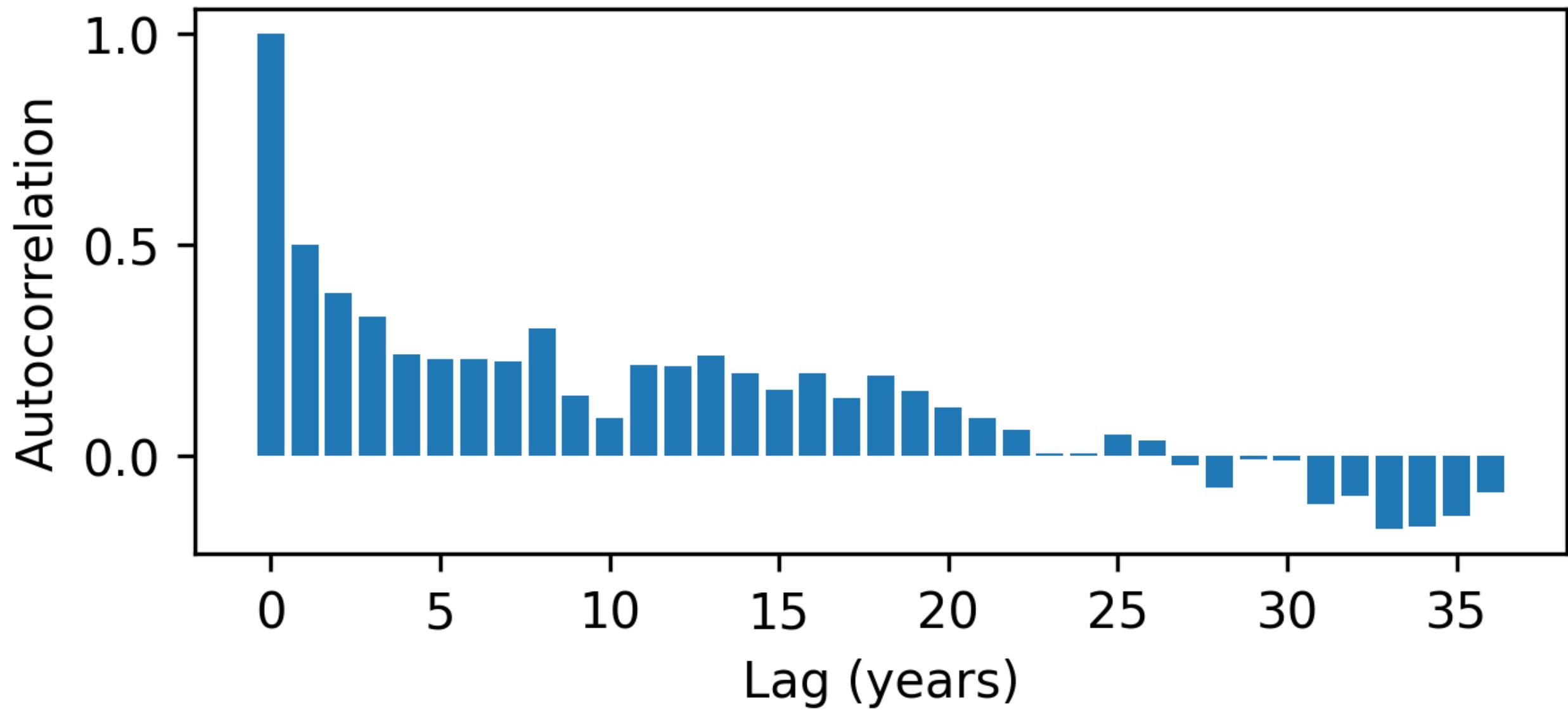




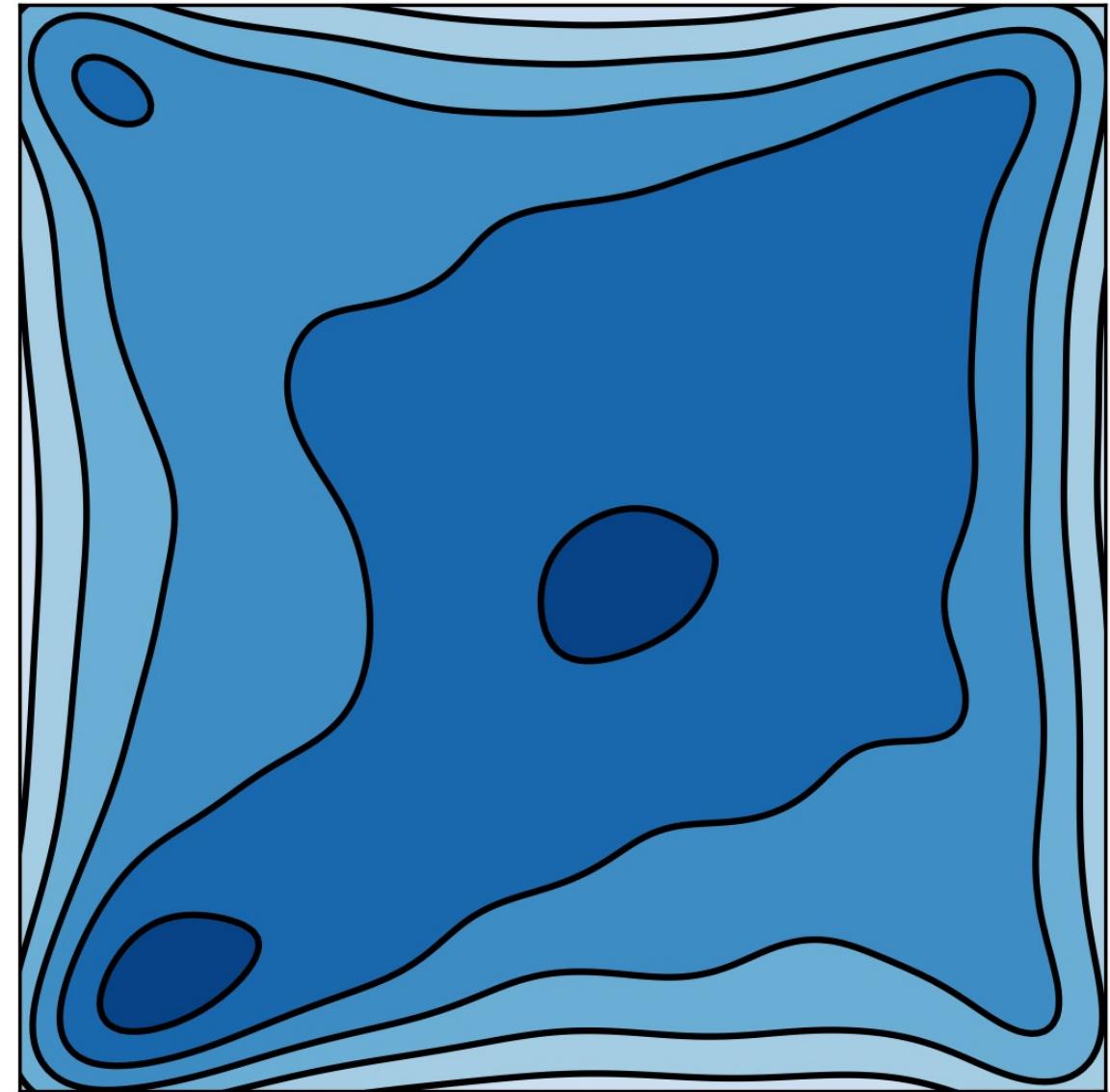
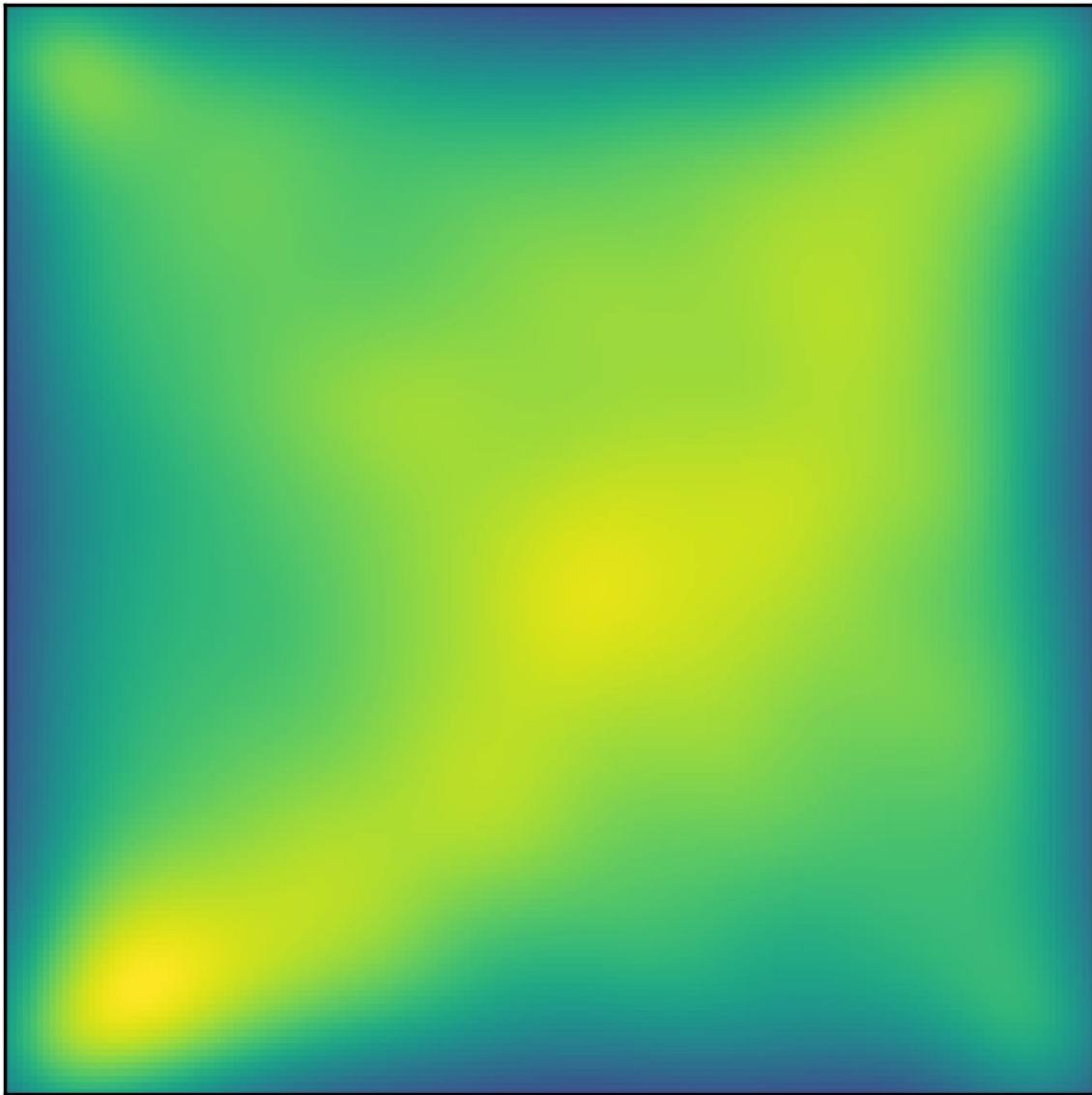


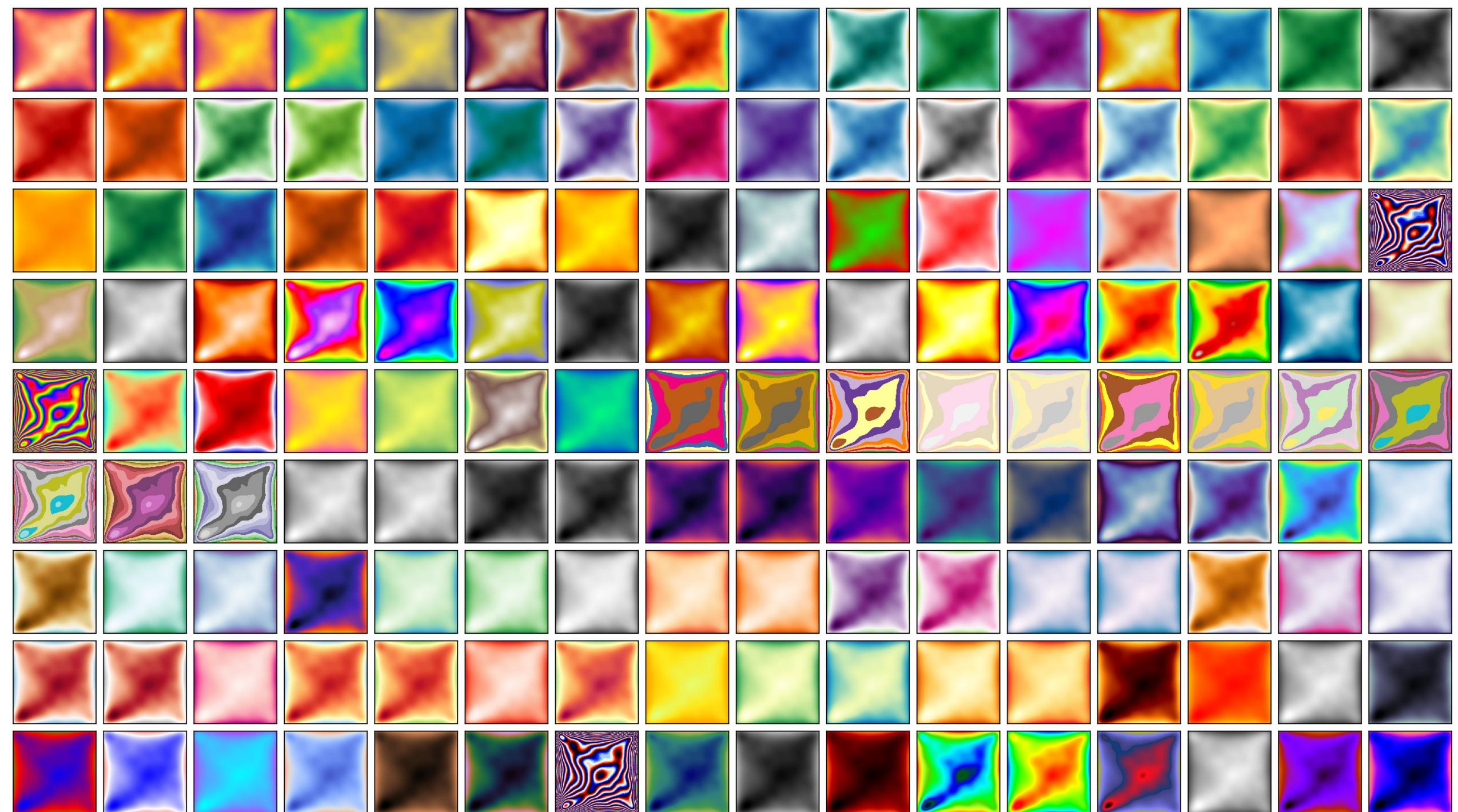


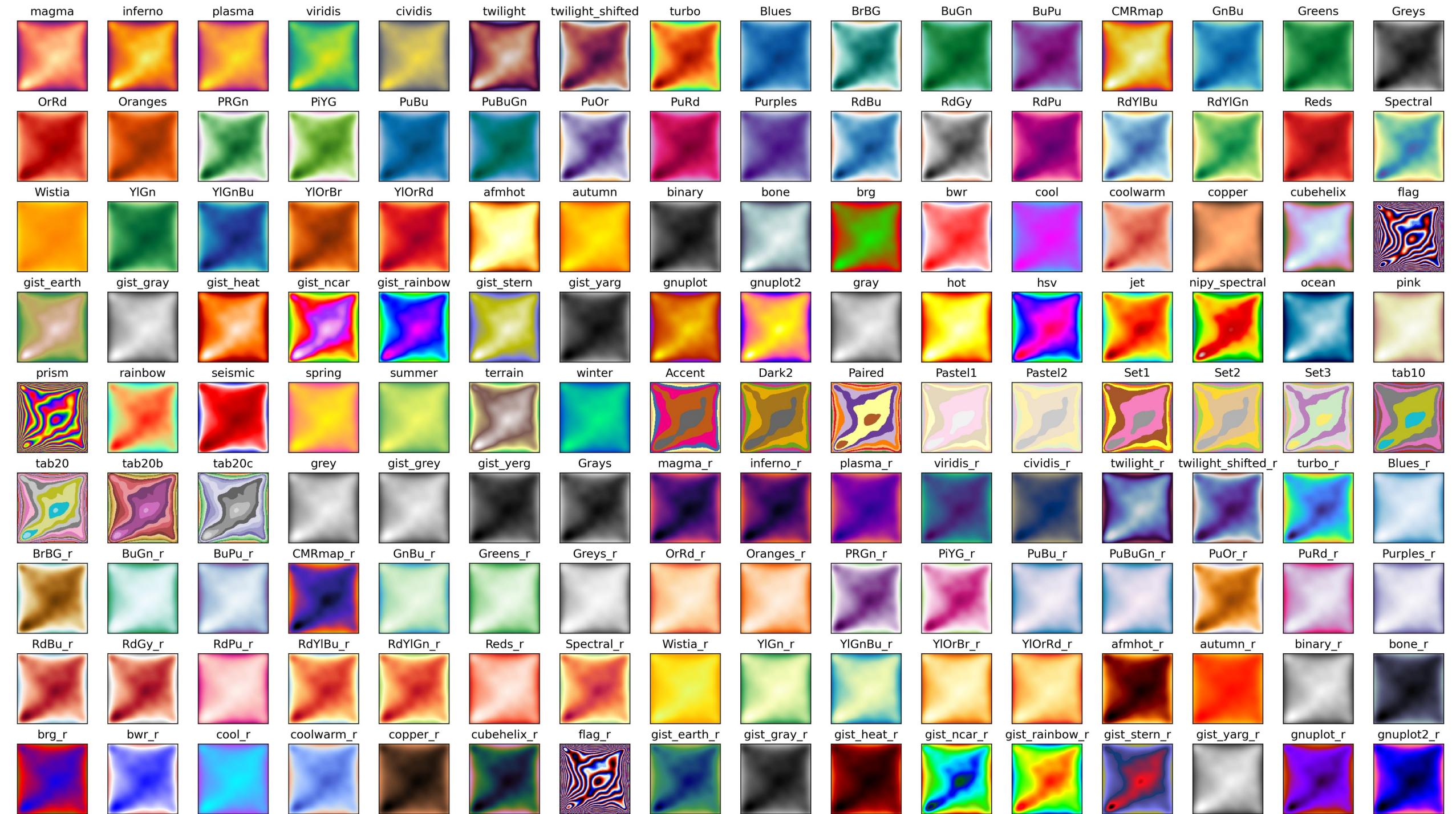


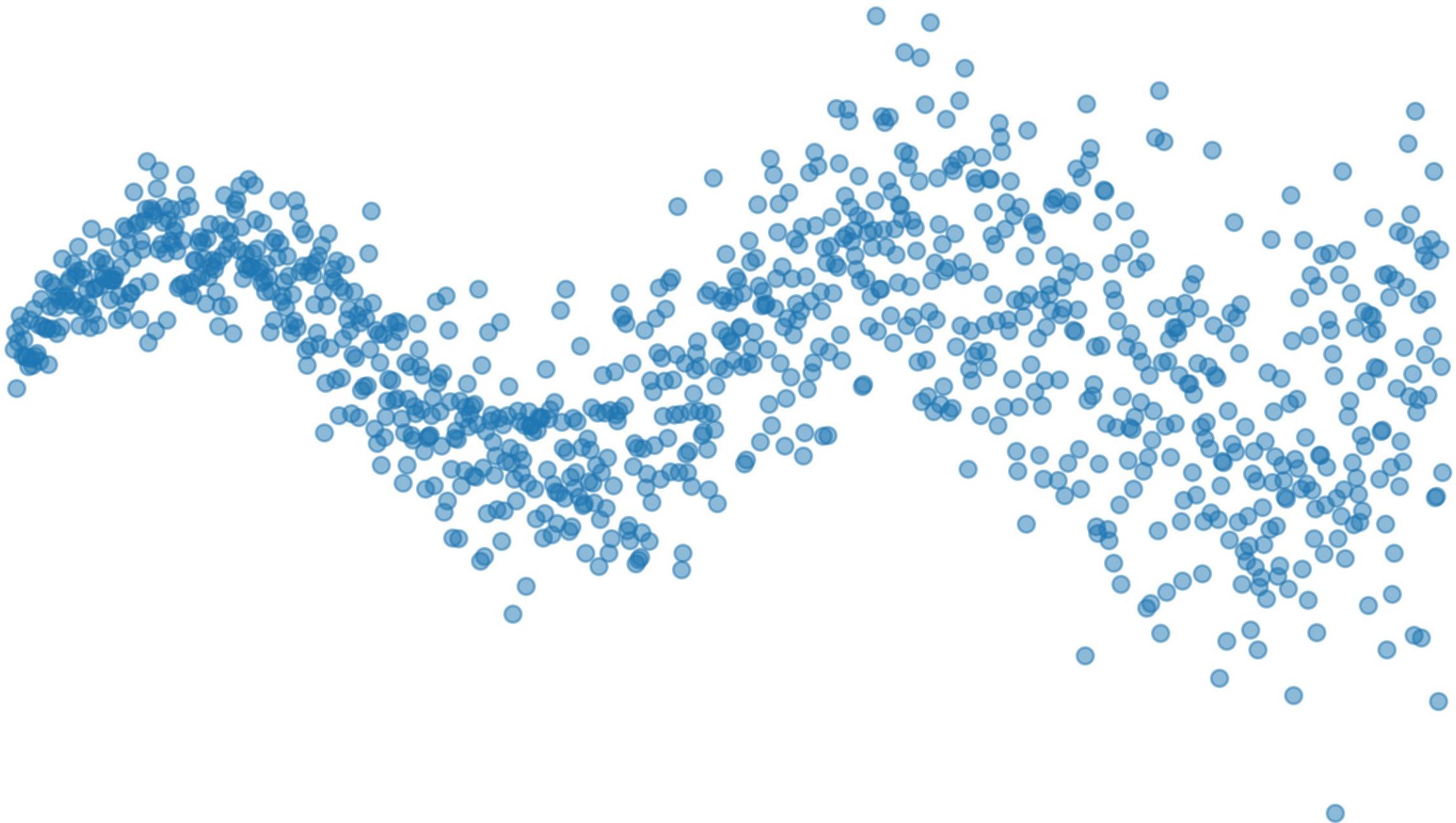


Discrete gradients









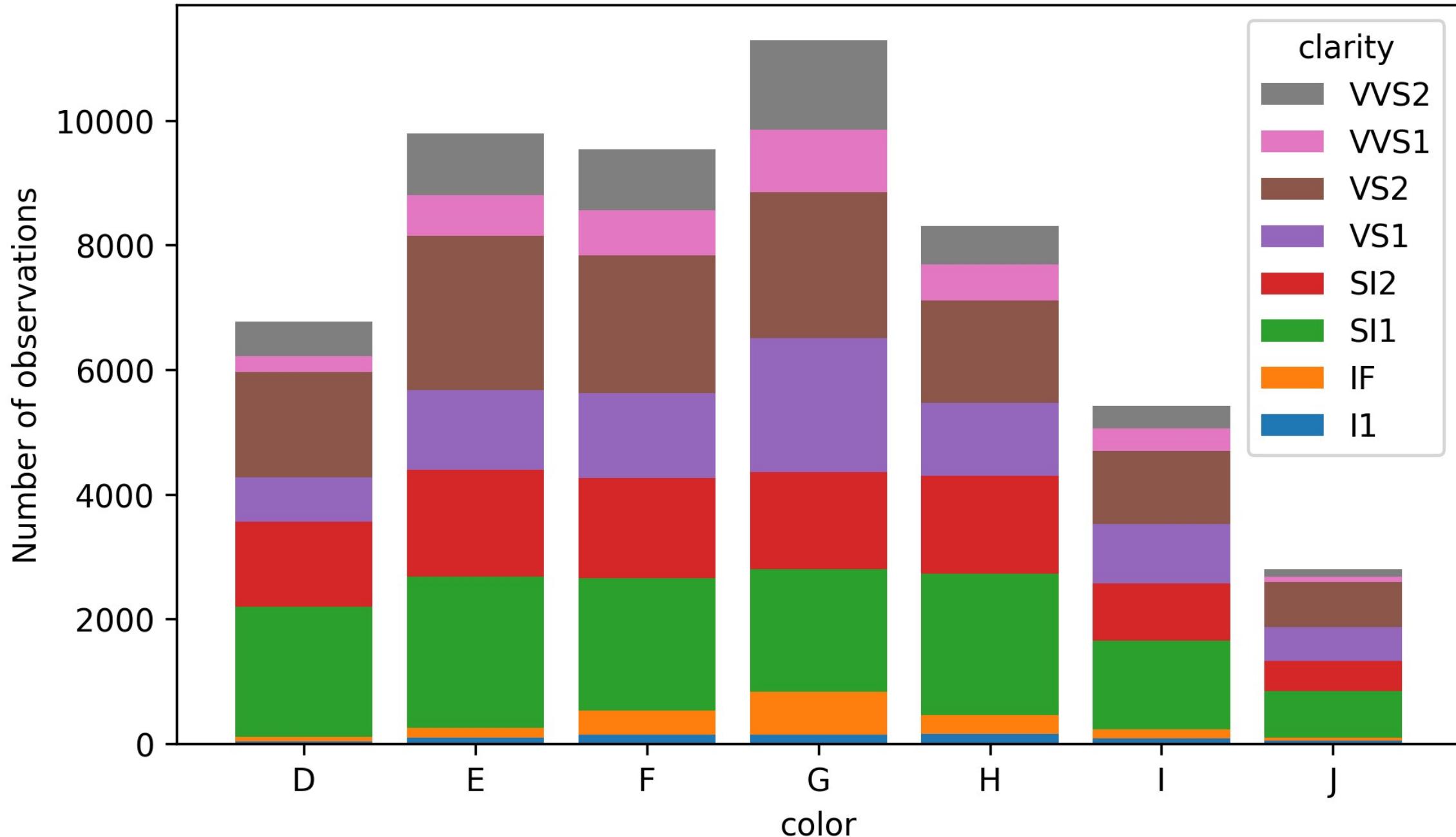
Logarithmic scale



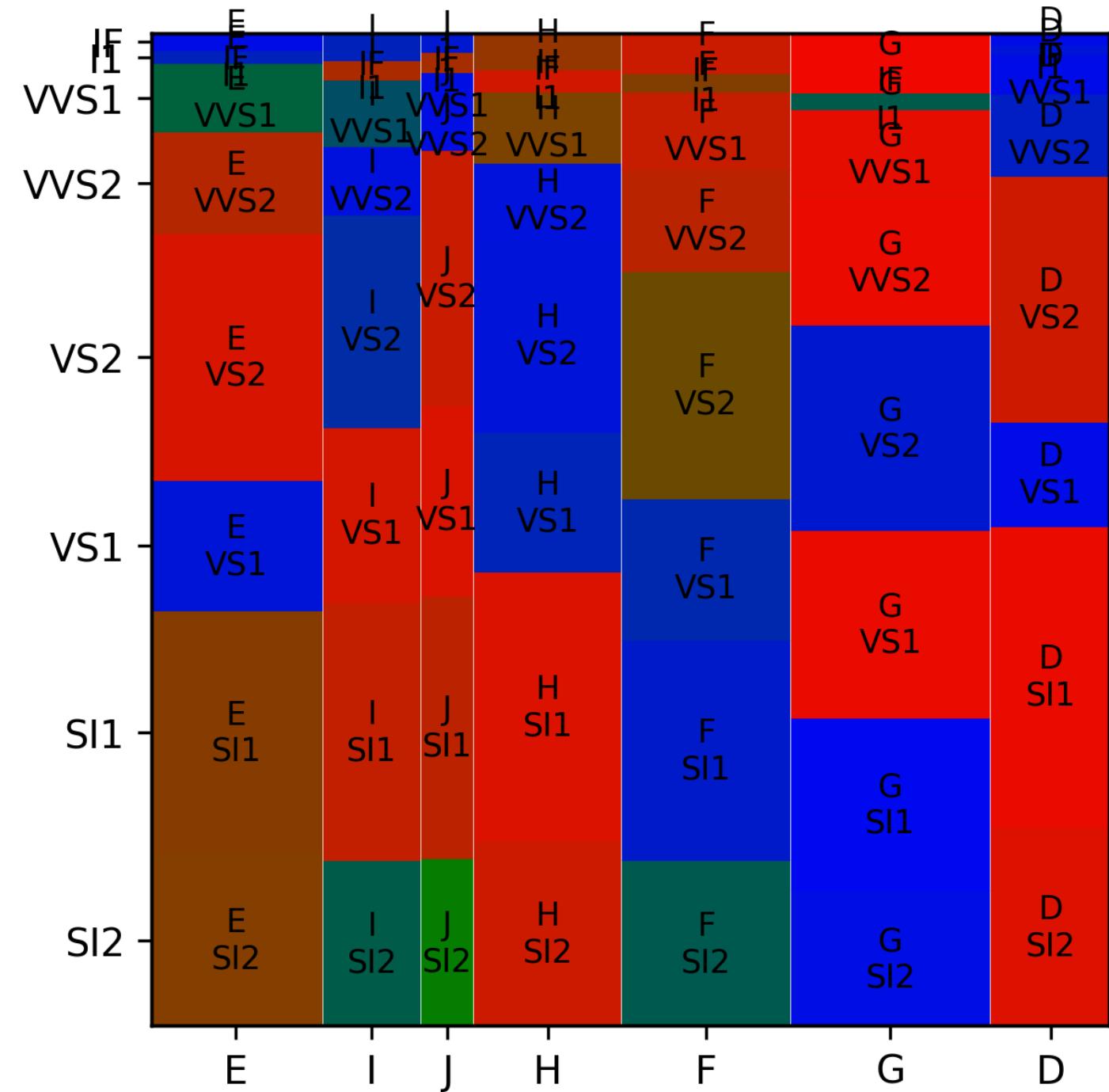
Images: Dall-E 3

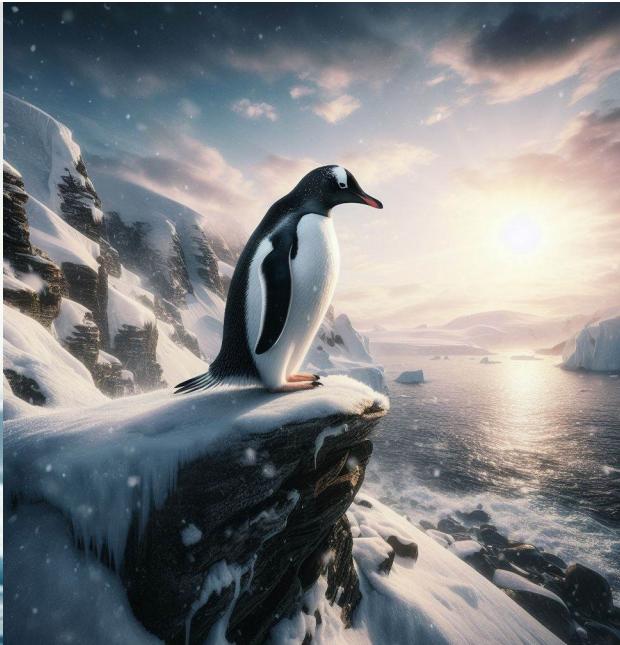
Sizes: https://www.youtube.com/watch?v=Z_1Q0XB4X0Y

Diamond clarity and colour



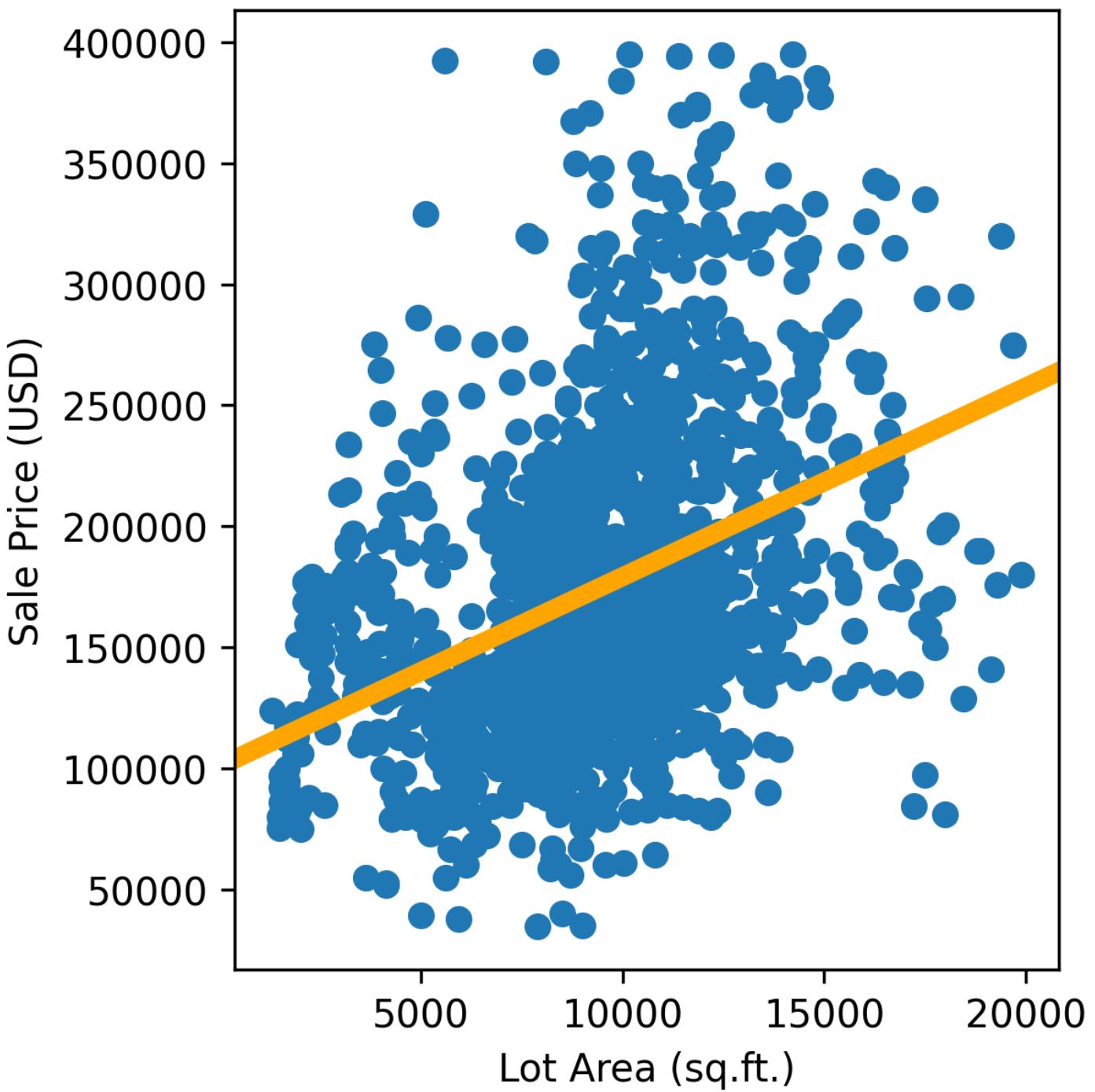
```
# Why are the default colours like that???
statsmodels.graphics.mosaicplot.mosaic(
    d,
    ['color', 'clarity'],
    statistic = True,
)
```



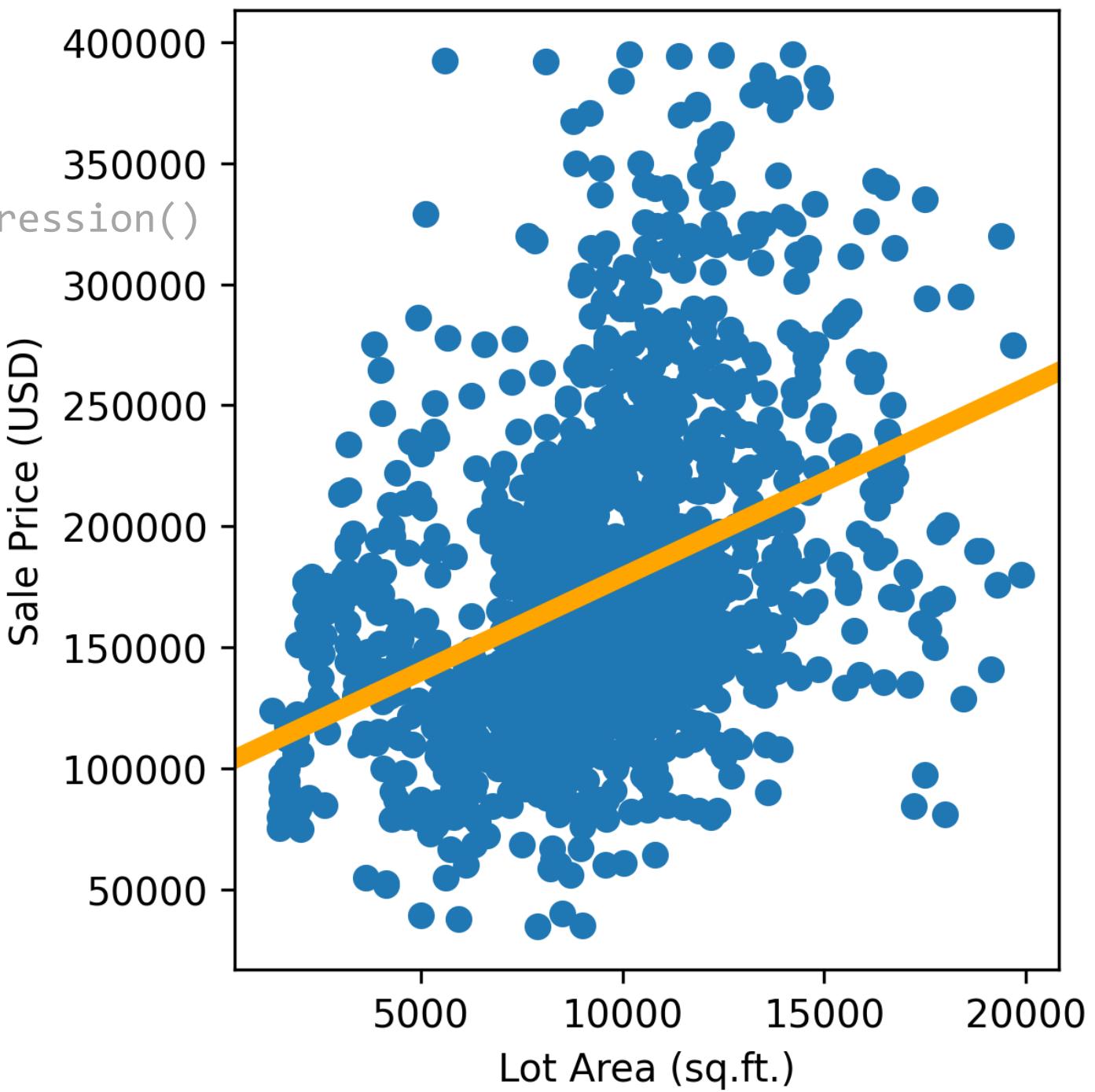


species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
Adelie	Torgersen	36.7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

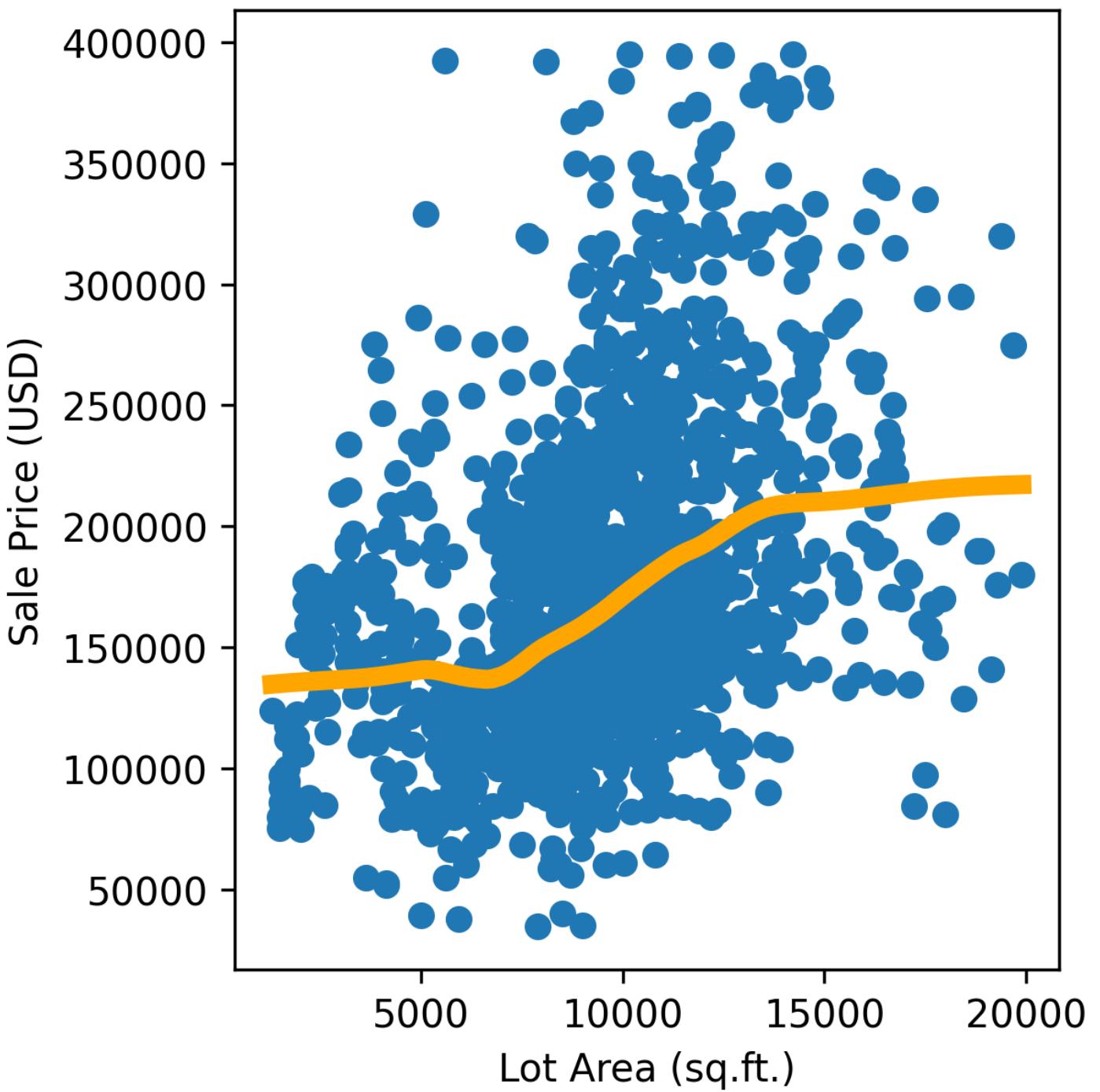
```
sns.regplot(x=x, y=y)
```



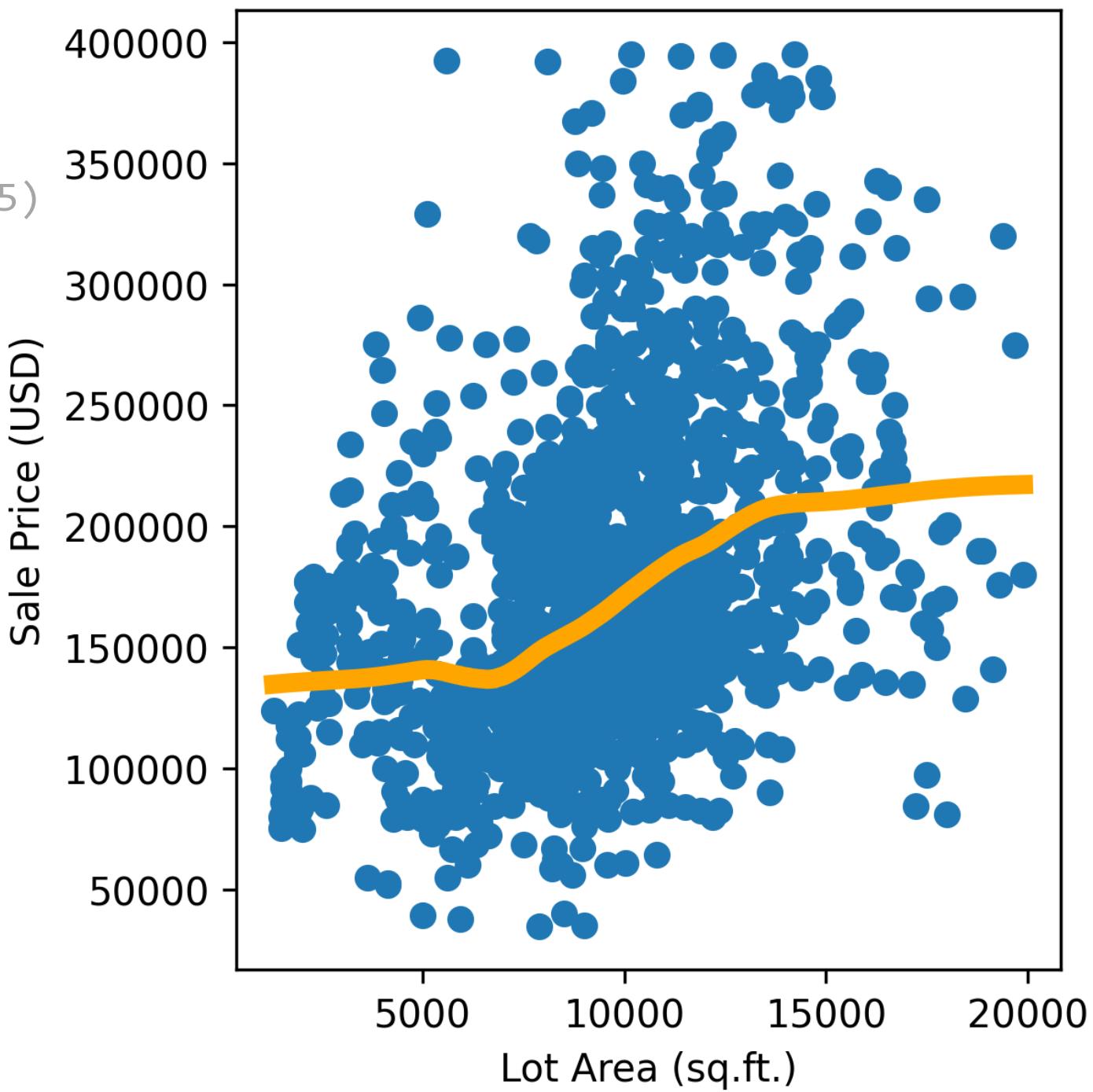
```
model = sklearn.linear_model.LinearRegression()  
model.fit( pd.DataFrame(x.values), y )  
plt.scatter( x, y )  
plt.axline(  
    (x.mean(), y.mean()),  
    slope = model.coef_[0],  
    color = 'orange',  
    linewidth = 5,  
)
```



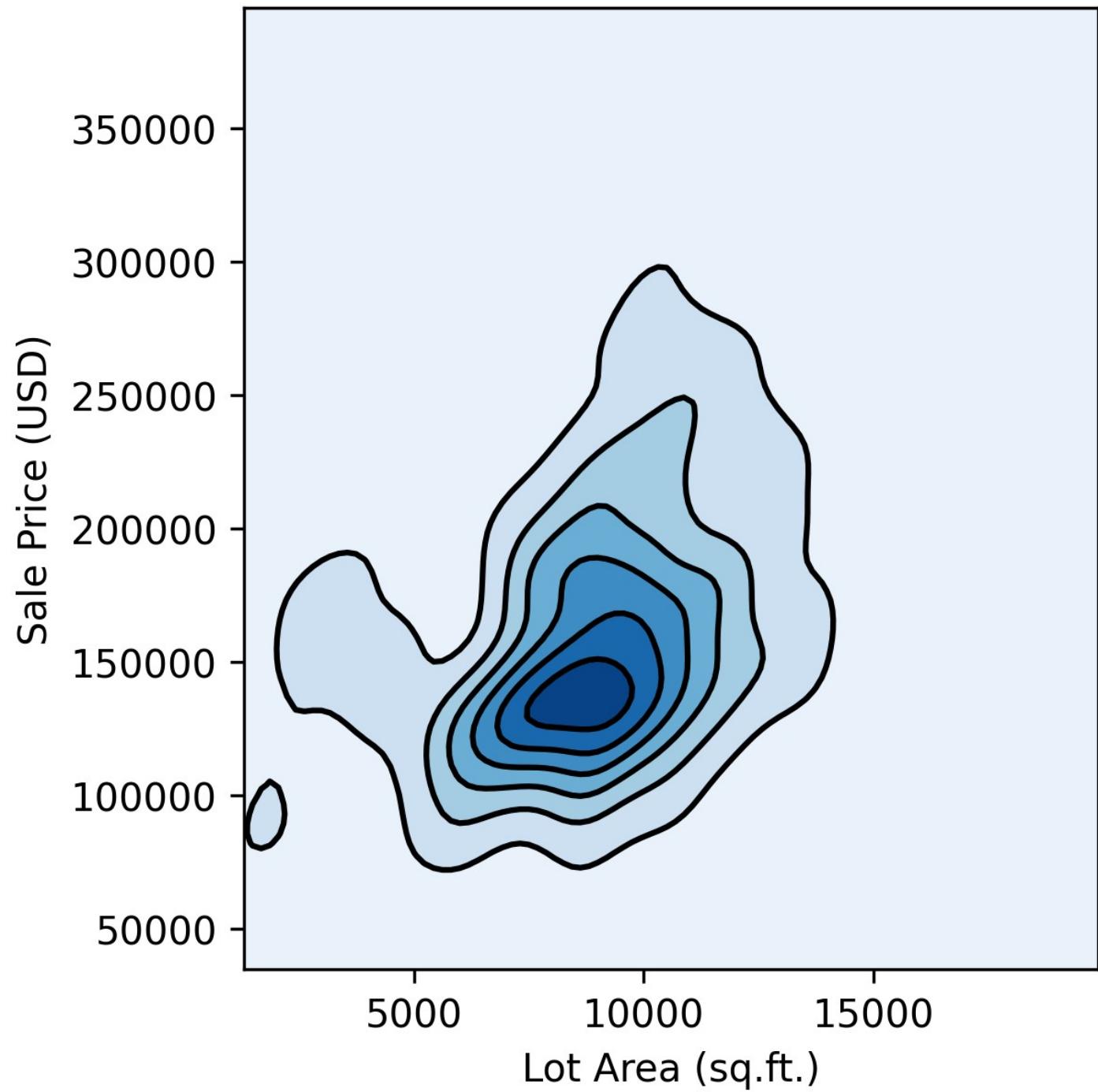
```
sns.regplot(x=x, y=y, lowess=True)
```



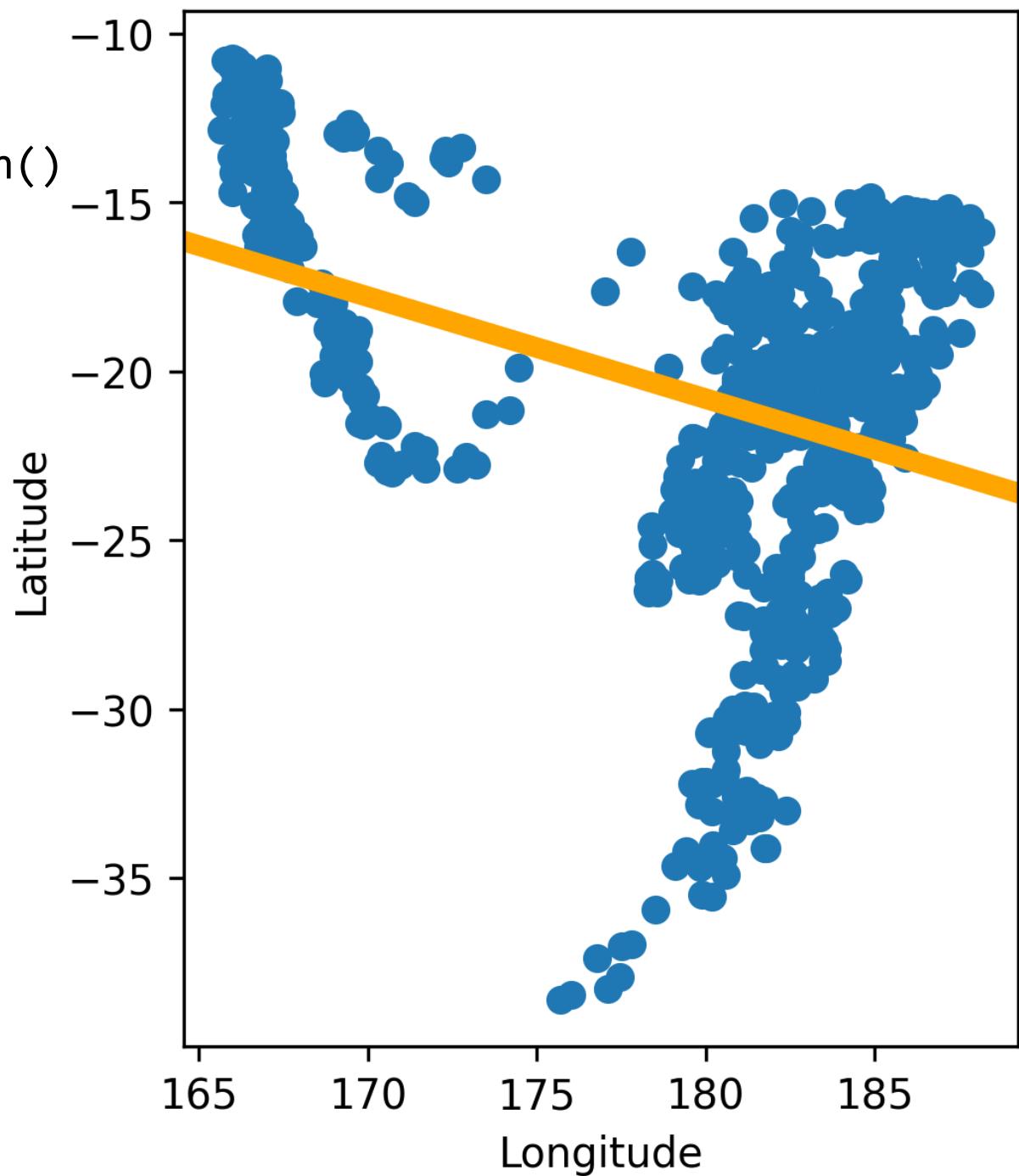
```
ys = statsmodels.nonparametric.  
    smoothers_lowess.lowess(y, x, frac=.5)  
  
plt.scatter( x, y )  
  
plt.plot(  
    ys[:,0],  
    ys[:,1],  
    color = 'orange',  
    linewidth = 5,  
)
```



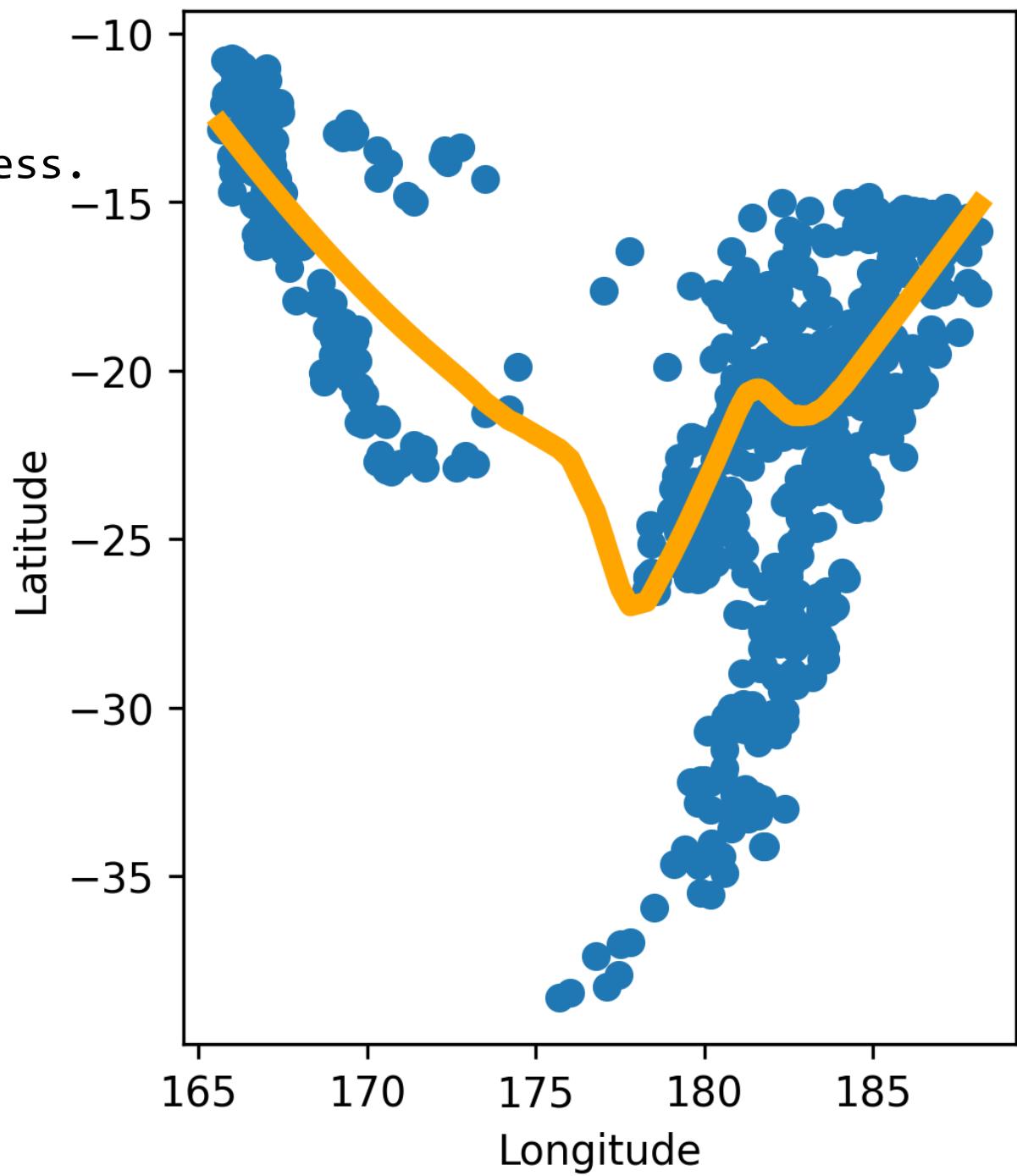
```
x, y = d.values.T
xx, yy = np.mgrid[
    min(x) : max(x) : 100j,
    min(y) : max(y) : 100j,
]
positions = np.vstack([xx.ravel(), yy.ravel()])
kernel = scipy.stats.gaussian_kde(
    np.vstack([x, y]),
    bw_method = .2,
)
f = np.reshape(kernel(positions).T, xx.shape)
fig, ax = plt.subplots(
    figsize = (4,4), layout = 'constrained',
)
ax.set_xlim( min(x), max(x) )
ax.set_ylim( min(y), max(y) )
ax.contourf(xx, yy, f, cmap='Blues')
ax.contour(xx, yy, f, colors='k')
ax.set_xlabel( d.columns[0] )
ax.set_ylabel( d.columns[1] )
ax.set_aspect(1)
plt.show()
```



```
model = sklearn.linear_model.LinearRegression()  
model.fit( pd.DataFrame(x.values), y )  
plt.scatter( x, y )  
plt.axline(  
    (x.mean(), y.mean()),  
    slope = model.coef_[0],  
    color = 'orange',  
    linewidth = 5,  
)
```



```
ys = statsmodels.nonparametric.smoothers_lowess.  
    lowess(y, x, frac=.5)  
  
plt.scatter( x, y )  
  
plt.plot(  
    ys[:,0],  
    ys[:,1],  
    color = 'orange',  
    linewidth = 5,  
)
```



```
x, y = d.values.T
xx, yy = np.mgrid[
    min(x) : max(x) : 100j,
    min(y) : max(y) : 100j,
]
positions = np.vstack([xx.ravel(), yy.ravel()])
kernel = scipy.stats.gaussian_kde(
    np.vstack([x, y]),
    bw_method = .2,
)
f = np.reshape(kernel(positions).T, xx.shape)
fig, ax = plt.subplots(
    figsize = (4,4), layout = 'constrained',
)
ax.set_xlim( min(x), max(x) )
ax.set_ylim( min(y), max(y) )
ax.contourf(xx, yy, f, cmap='Blues')
ax.contour(xx, yy, f, colors='k')
ax.set_xlabel( d.columns[0] )
ax.set_ylabel( d.columns[1] )
ax.set_aspect(1)
plt.show()
```

