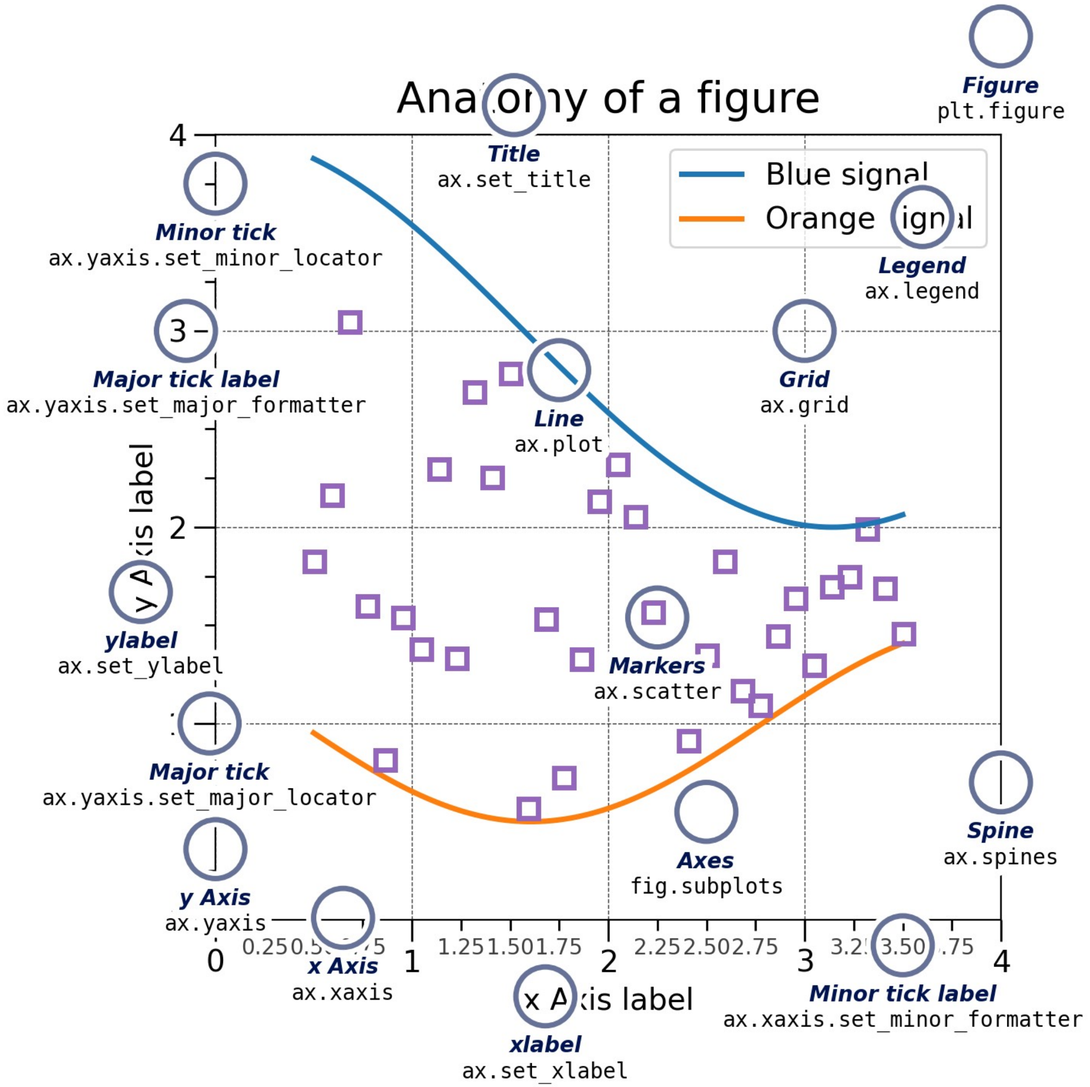


Lecture 12-1: Data Visualization

Dr. James Chen, Animal Data Scientist, School of Animal Sciences

Why Visualization?

Simplifies Complex Data

FASTAQ format

```
>NG_008679.1:5001-38170 Homo sapiens paired box 6 (PAX6)
ACCCTCTTTTCTTATCATTGACATTTAAACTCTGGGGCAGGTCCTCGCGTAGAACGCGGCTGTCAGATCT
GCCACTTCCCCTGCCGAGCGGCGGTGAGAAGTGTGGGAACCGGCGCTGCCAGGCTCACCTGCCTCCCCGC
CCTCCGCTCCCAGGTAACCGCCCGGGCTCCGGCCCCGGCCCGGCTCGGGGCCCGCGGGGCCTCTCCGCTG
CCAGCGACTGCTGTCCCCAAATCAAAGCCCGCCCCAAGTGGCCCCGGGGCTTGATTTTTGCTTTTAAAAG
GAGGCATACAAAGATGGAAGCGAGTTACTGAGGGAGGGATAGGAAGGGGGGTGGAGGAGGGACTTGTCTT
TGCCGAGTGTGCTCTTCTGCAAAAGTAGCAAAATGTTCCACTCCTAAGAGTGGACTTCCAGTCCGGCCCT
GAGCTGGGAGTAGGGGGCGGGAGTCTGCTGCTGCTGTCTGCTAAAGCCACTCGCGACCGCGAAAAATGCA
GGAGGTGGGGACGCACTTTGCATCCAGACCTCCTCTGCATCGCAGTTCACGACATCCACGCTTGGGAAAG
TCCGTACCCGCGCCTGGAGCGCTTAAAGACACCCTGCCGCGGGTCGGGCGAGGTGCAGCAGAAGTTTCCC
GCGGTTGCAAAGTGCAGATGGCTGGACCGCAACAAAGTCTAGAGATGGGGTTCGTTTCTCAGAAAGACGC
```

FIGURE 7.1: An example fasta file showing the first part of the PAX6 gene.

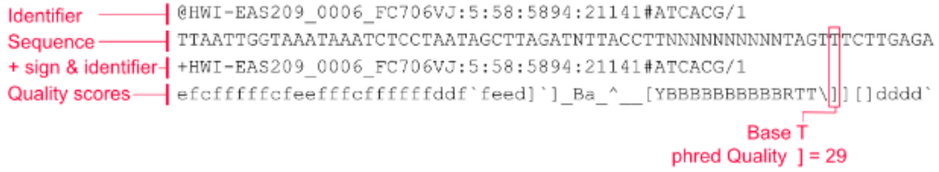
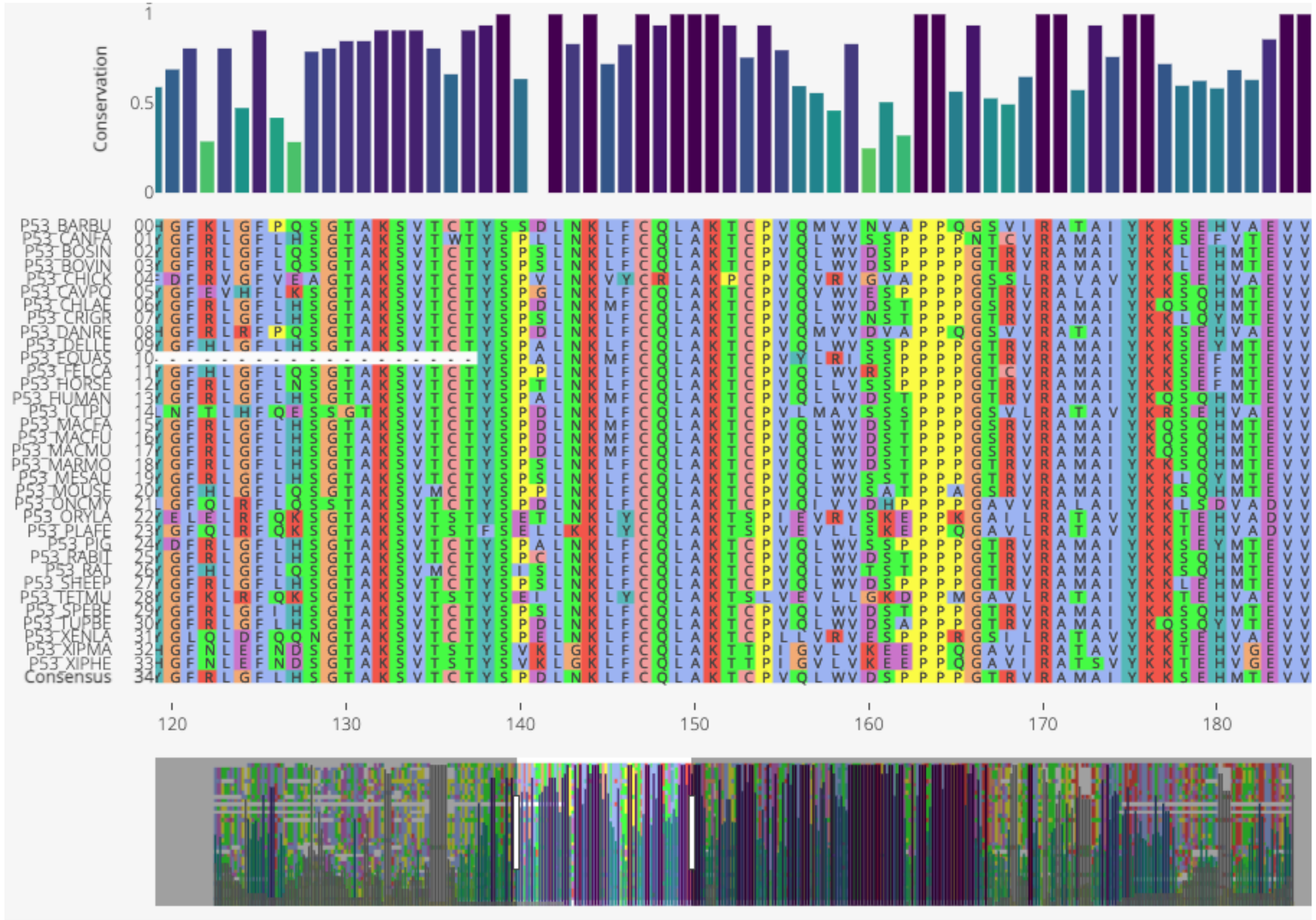


FIGURE 7.2: FASTQ format and a brief explanation of each line in the format.

Alignment Viewer

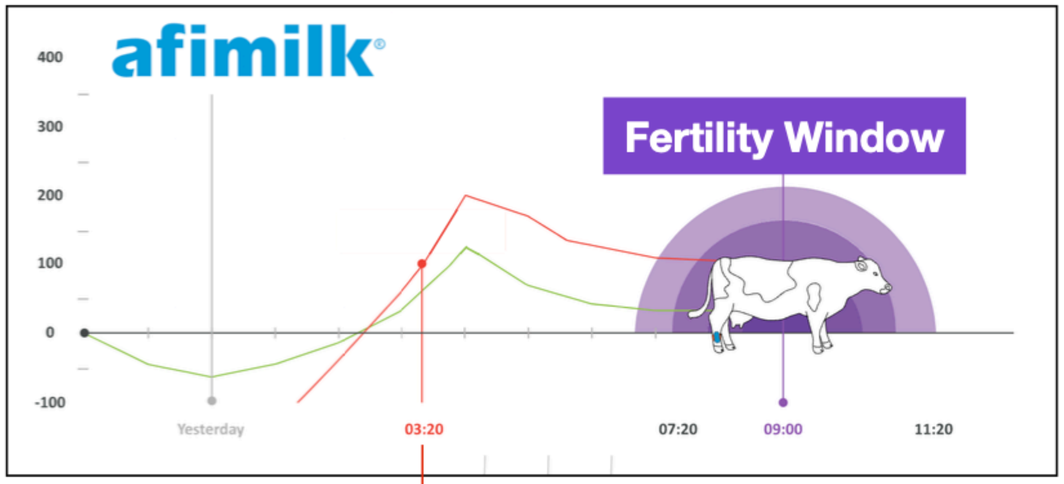


https://compgenomr.github.io/book/fasta-and-fastq-formats.html https://dash.gallery/dash-alignment-chart/?_gl=1*gjxemv*_ga*ODE5MDc3OTA1LjE2ODA4ODU1NDM.*_ga_6G7EE0JNSC*MTY4MTEzNjMzMC40LjEuMTY4MTEzNjU0MS4wLjAuMA..

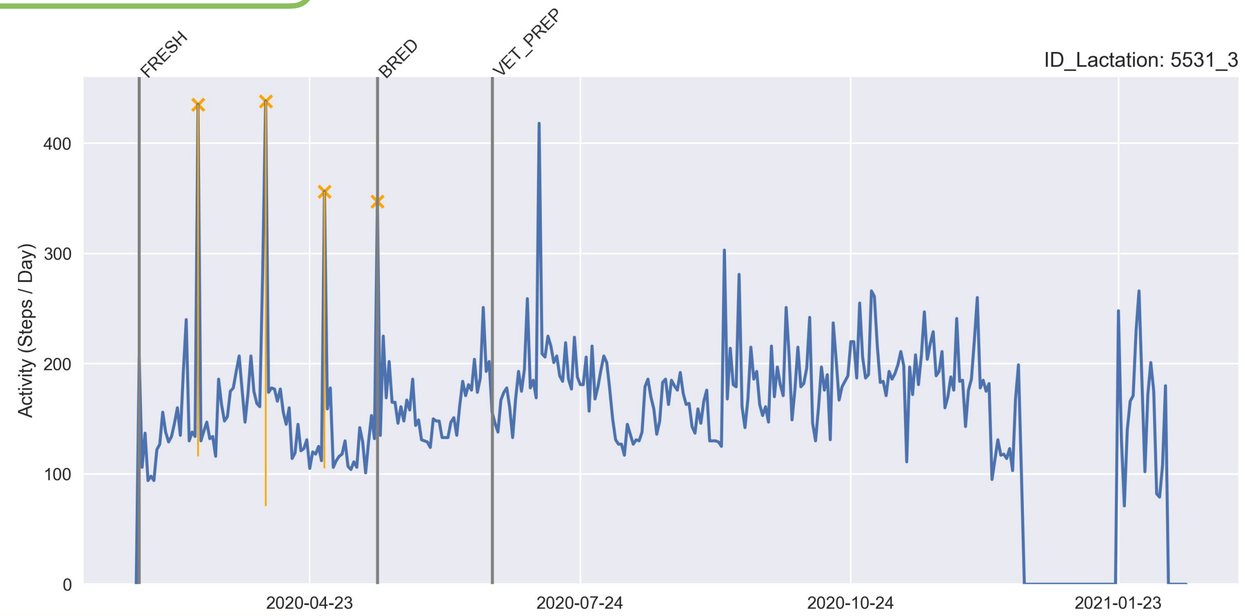
Why Visualization?

Supports hypothesis testing and discovery

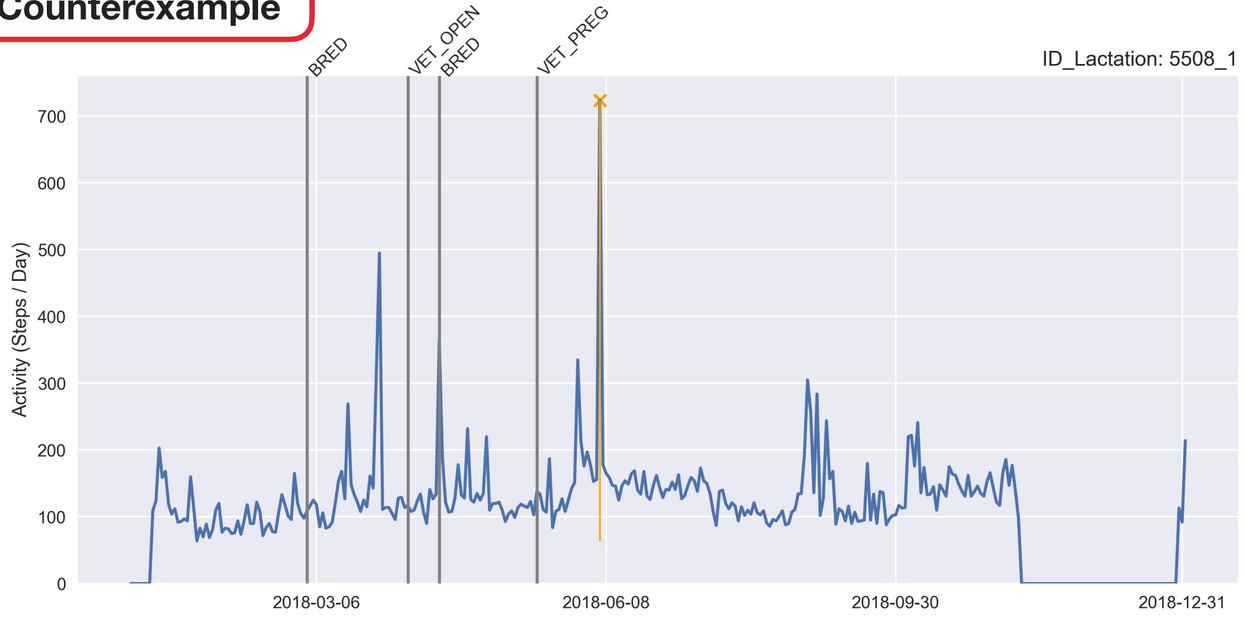
Hypothesis



Example



Counterexample



Why Visualization?

Enhances collaboration and communication

Help you to deliver messages

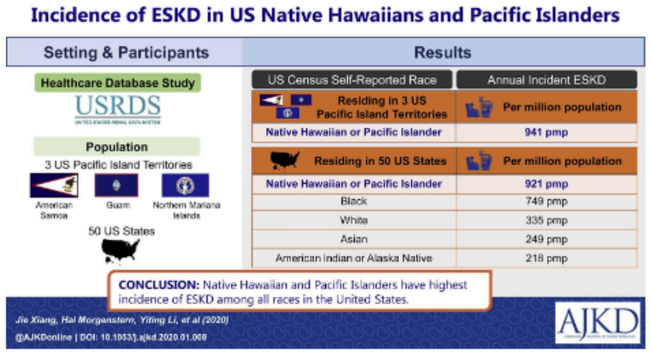
ELSEVIER

How to produce a good visual abstract

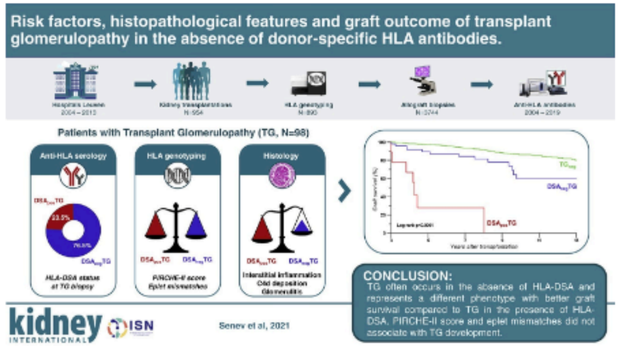
Creating a simple, accessible and visually-stimulating overview will benefit your article considerably

Introduce the context of your research or make your first point here	Here is where you showcase your methodology or make the second point	Finally this panel is for explaining the main outcome of your work or hosting the third point
		
Elsevier Author Team	Tancock, C.M., University of Oxford	elsevier.com/authors/tools-and-resources/graphical-abstract

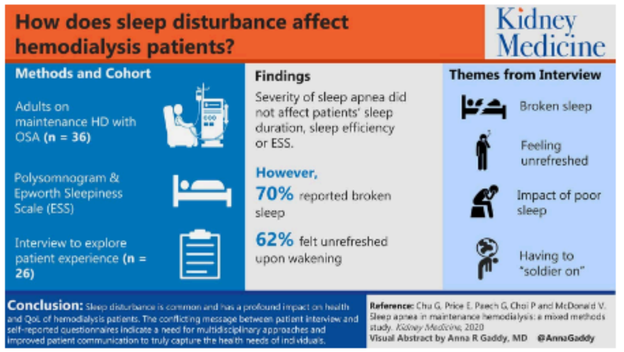
Examples of graphical abstracts



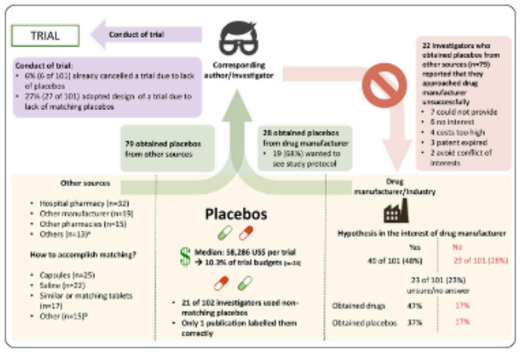
Incidence of ESKD Among Native Hawaiians and Pacific Islanders Living in the 50 US States and Pacific Island Territories ↗



Risk factors, histopathological features, and graft outcome of transplant glomerulopathy in the absence of donor-specific HLA antibodies ↗



Sleep Apnea in Maintenance Hemodialysis: A Mixed-Methods Study ↗

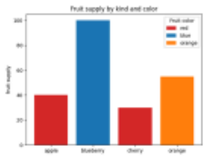


A meta-research study revealed several challenges in obtaining placebos for investigator-initiated drug trials ↗

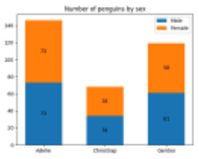
Python Community

matplotlib

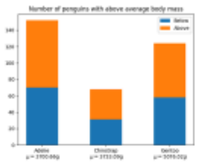
Lines, bars and markers



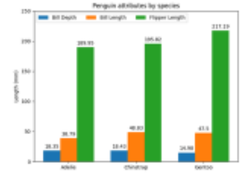
Bar color demo



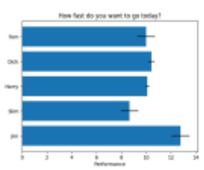
Bar Label Demo



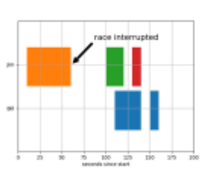
Stacked bar chart



Grouped bar chart with labels

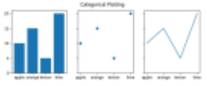


Horizontal bar chart

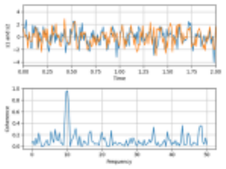


Broken Barh

CapStyle



Plotting categorical variables



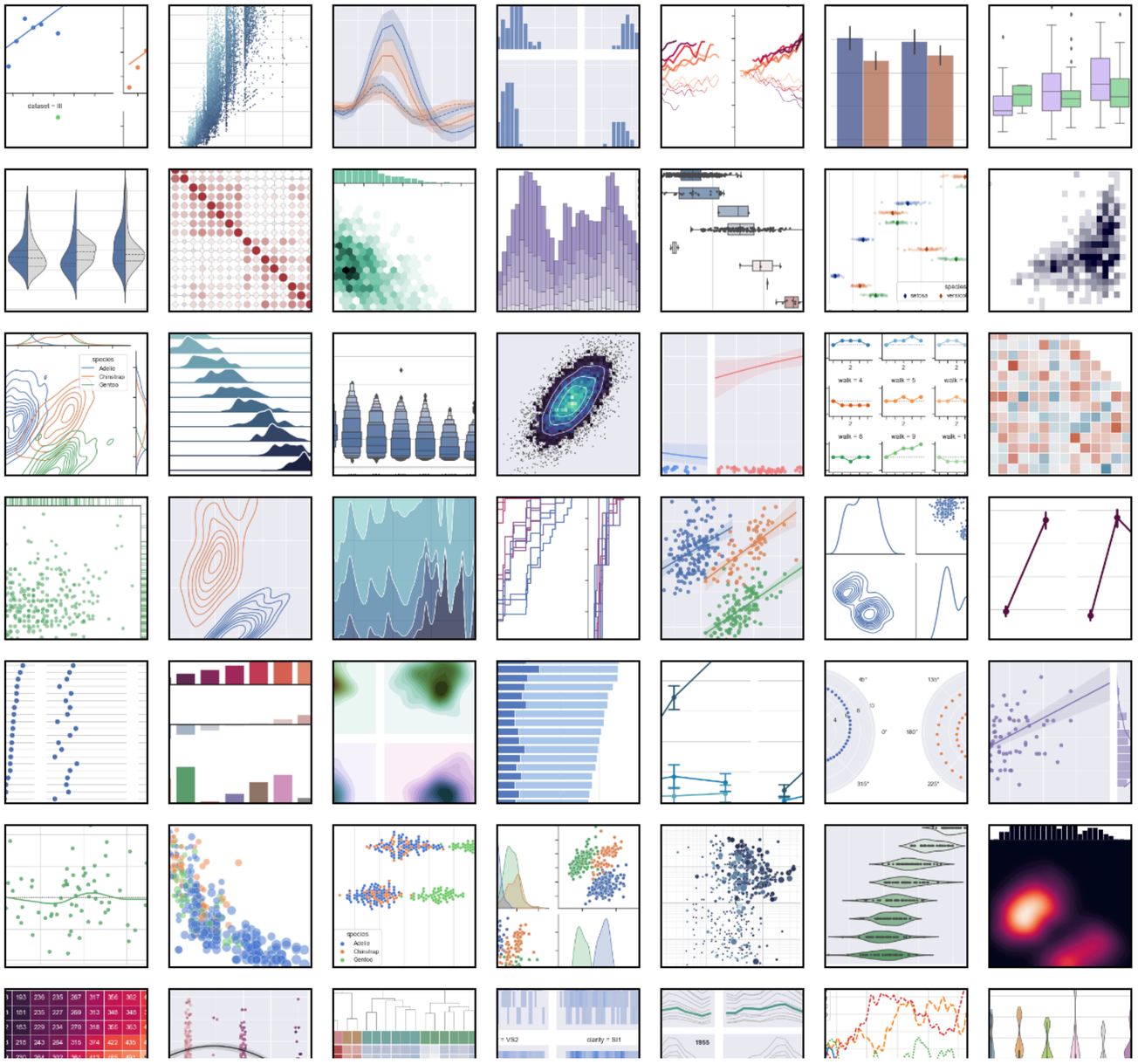
Plotting the coherence of two signals

https://matplotlib.org/stable/gallery/index.html

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

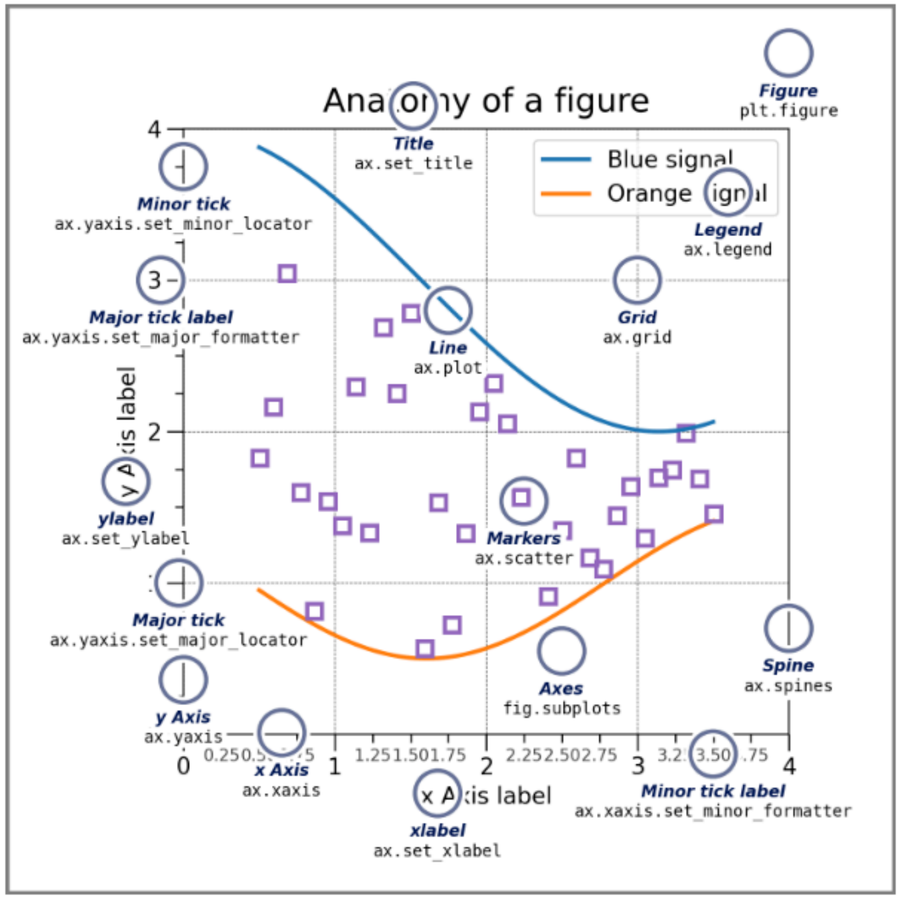
seaborn

Installing Gallery Tutorial API Releases Citing FAQ



Python Matplotlib

Matplotlib is a Python library for data visualization. We will use this library to introduce the basic concepts of data visualization. The basic components of a plot are described in the figure below:



- Title: the title of the plot
- Legend: a list of labels for each data series (variable)
- Major and minor ticks: visible marks of the axis. Major ticks are usually labeled, while minor ticks are not but still visible as small lines.
- Spines: the four lines that connect the ticks.

Scatter Plot - a Simple Example

Scatter Plot

https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.axes.Axes.scatter.html

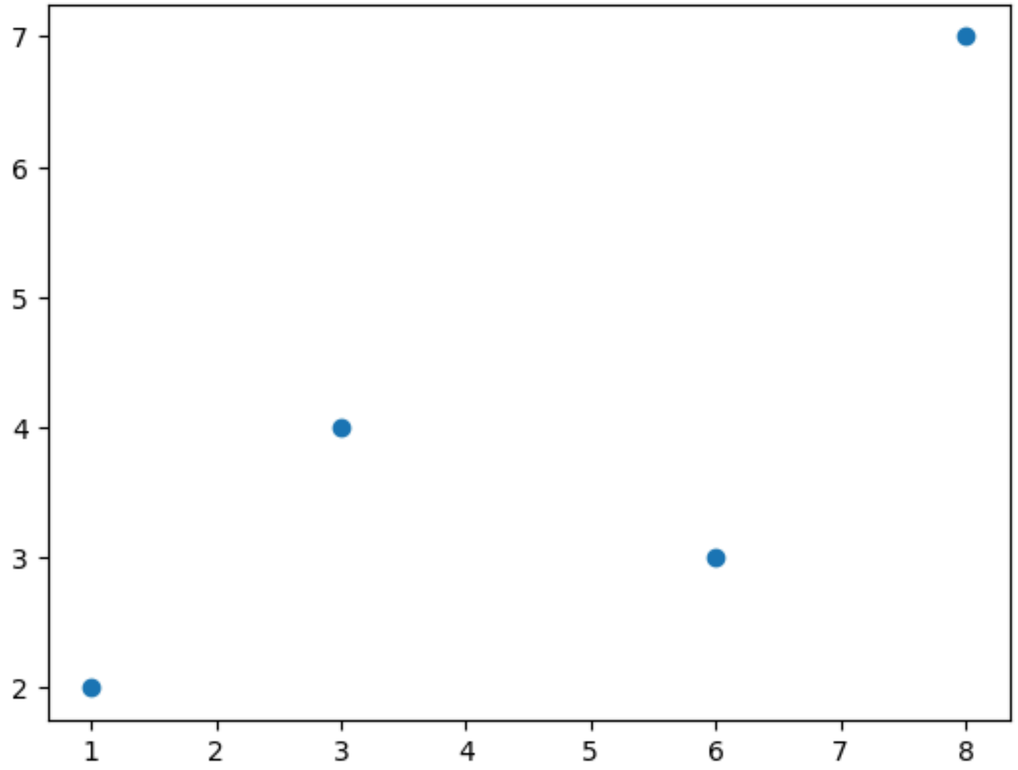
A simple figure that describes the relationship between two variables is a scatter plot. We can use the `scatter()` function to create a line plot. The `scatter()` function takes two arguments: the x-axis variable and the y-axis variable.

```
# define x and y
x = [1, 3, 6, 8]
y = [2, 4, 3, 7]
```

✓ 0.0s

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

✓ 0.2s



https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.axes.Axes.scatter.html

Scatter Plot - Arguments

matplotlib.axes.Axes.scatter

```
Axes.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None,
vmin=None, vmax=None, alpha=None, linewidths=None, *, edgecolors=None,
plotnonfinite=False, data=None, **kwargs)                    [source]
```

There are couple of optional arguments that we commonly use to customize the plot:

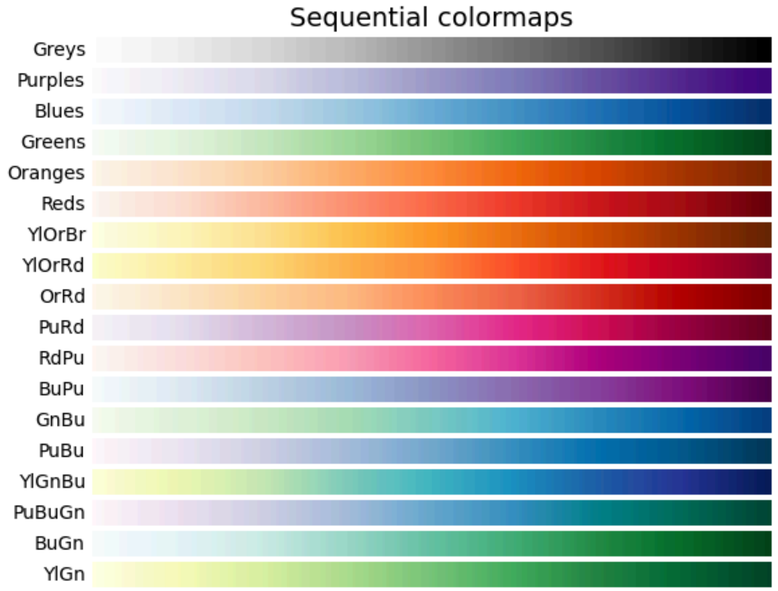
- `s`: the size of the marker
- `c`: the color of the marker
- `alpha`: the transparency of the marker
- `marker`: the shape of the marker (https://matplotlib.org/3.5.3/api/markers_api.html#module-matplotlib.markers)
- `edgecolors`: the color of the marker's edge
- `linewidths`: the width of the marker's edge

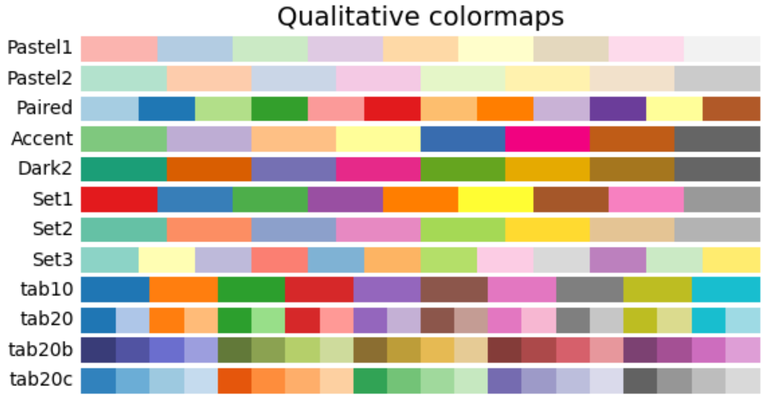
marker	symbol	description
"."	•	point
","	·	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	Y	tri_down
"2"	⅄	tri_up
"3"	≺	tri_left
"4"	≻	tri_right
"8"	●	octagon
"s"	■	square
"p"	⬟	pentagon
"P"	✚	plus (filled)
"*"	★	star

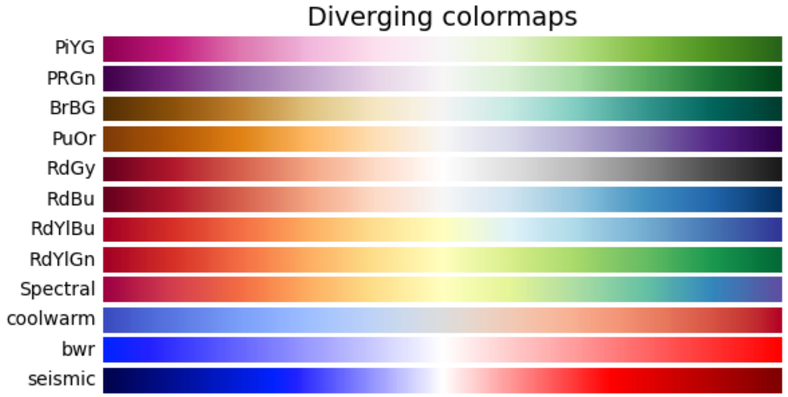
https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.axes.Axes.scatter.html

Scatter Plot - Colormap and Marker Shapes

Choosing Colormaps in Matplotlib







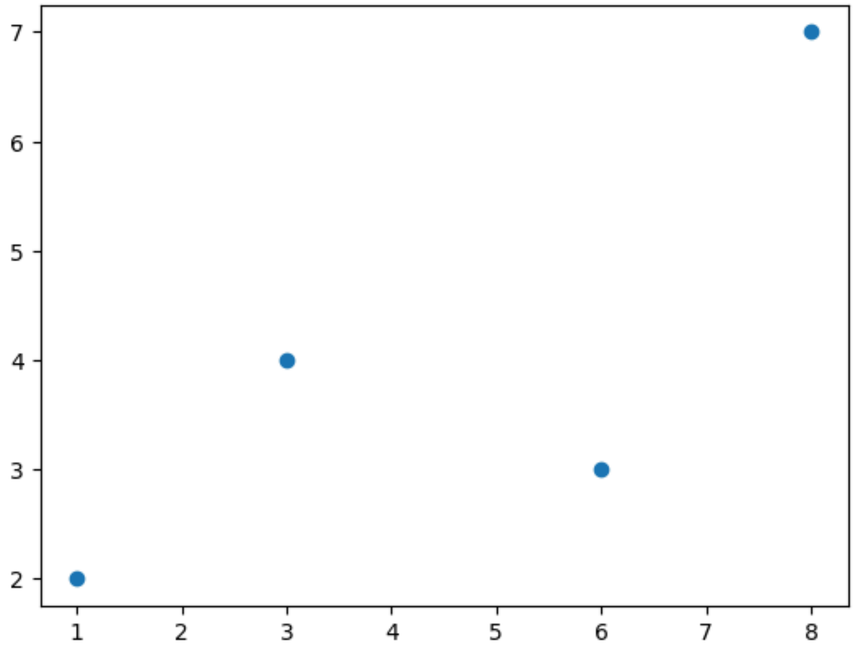
https://matplotlib.org/stable/tutorials/colors/colormaps.html

marker	symbol	description
"."	•	point
","	·	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	Y	tri_down
"2"	⅄	tri_up
"3"	≺	tri_left
"4"	≻	tri_right
"8"	●	octagon
"s"	■	square
"p"	⬟	pentagon
"P"	✚	plus (filled)
"*"	★	star

Scatter Plot - Arguments

No argument provided

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



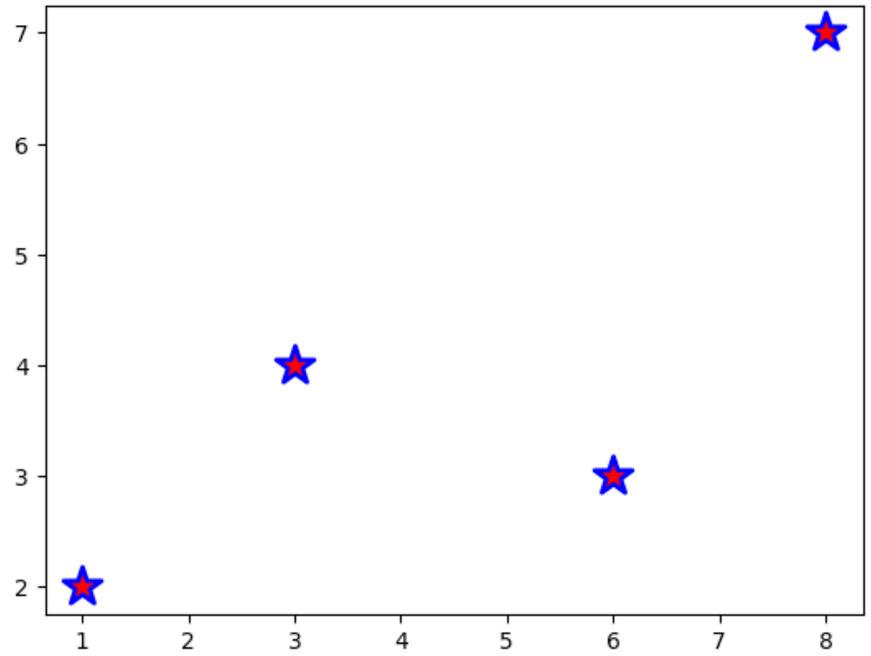
Keyword arguments

```
plt.scatter(x, y,
            s=300,
            c='red',
            marker='*',
            edgecolor='blue',
            linewidth=2)
```

size
color
shape

Dictionary as arguments

```
param = {
    "s": 300,
    "c": "red",
    "marker": "*",
    "edgecolor": "blue",
    "linewidth": 2
}
plt.scatter(x, y, **param)
```



Figure

matplotlib.figure

class matplotlib.figure.**Figure**(*figsize=None, dpi=None, facecolor=None, edgecolor=None, linewidth=0.0, frameon=None, subplotpars=None, tight_layout=None, constrained_layout=None, *, layout=None, **kwargs*)

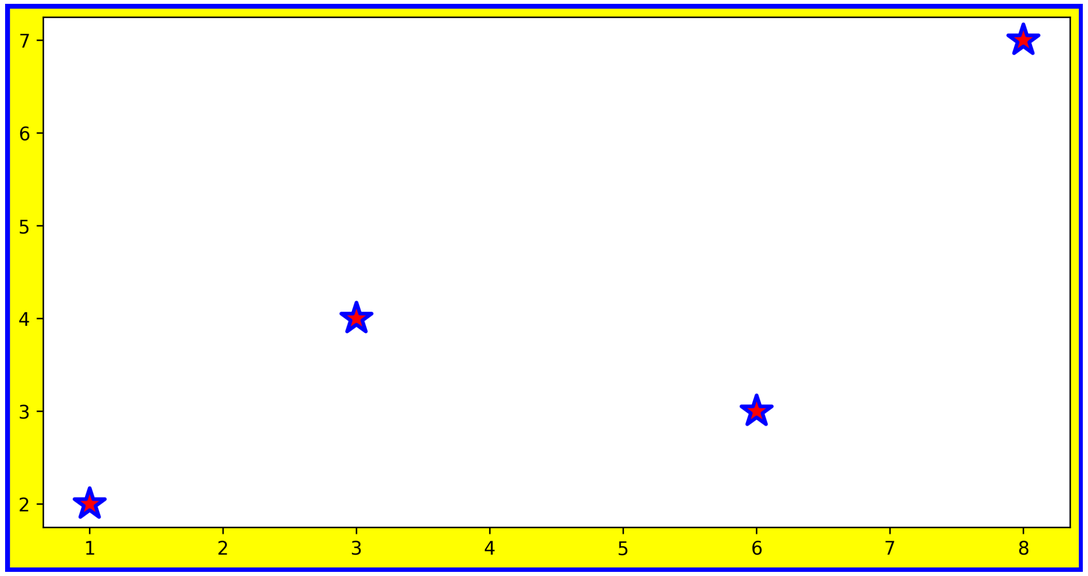
In addtion to modify the data points, we can also explicitly set the details of the figure and axes. The `figure()` function creates a figure object, which is the container of the plot. The available arguments are:

- `figsize`: the size of the figure. It takes a tuple of two numbers, which are the width and the height in inches.
- `dpi`: the resolution of the figure. It is recommended to set the resolution to 300 for publication.
- `facecolor`: the background color of the figure.
- `edgecolor`: the edge color of the figure.
- `linewidth`: the edge width of the figure.

Here is an example to make the figure even uglier:

- a width of 10 inches and a height of 5 inches
- dpi of 300
- facecolor of yellow, edgecolor of blue, and linewidth of 5

```
param_fig = {
    "figsize": (10, 5),
    "dpi": 300,
    "facecolor": "yellow",
    "edgecolor": "blue",
    "linewidth": 5
}
param_data = {
    "s": 300,
    "c": "red",
    "marker": "*",
    "edgecolor": "blue",
    "linewidth": 2
}
plt.figure(**param_fig)
plt.scatter(x, y, **param_data)
```



Axes

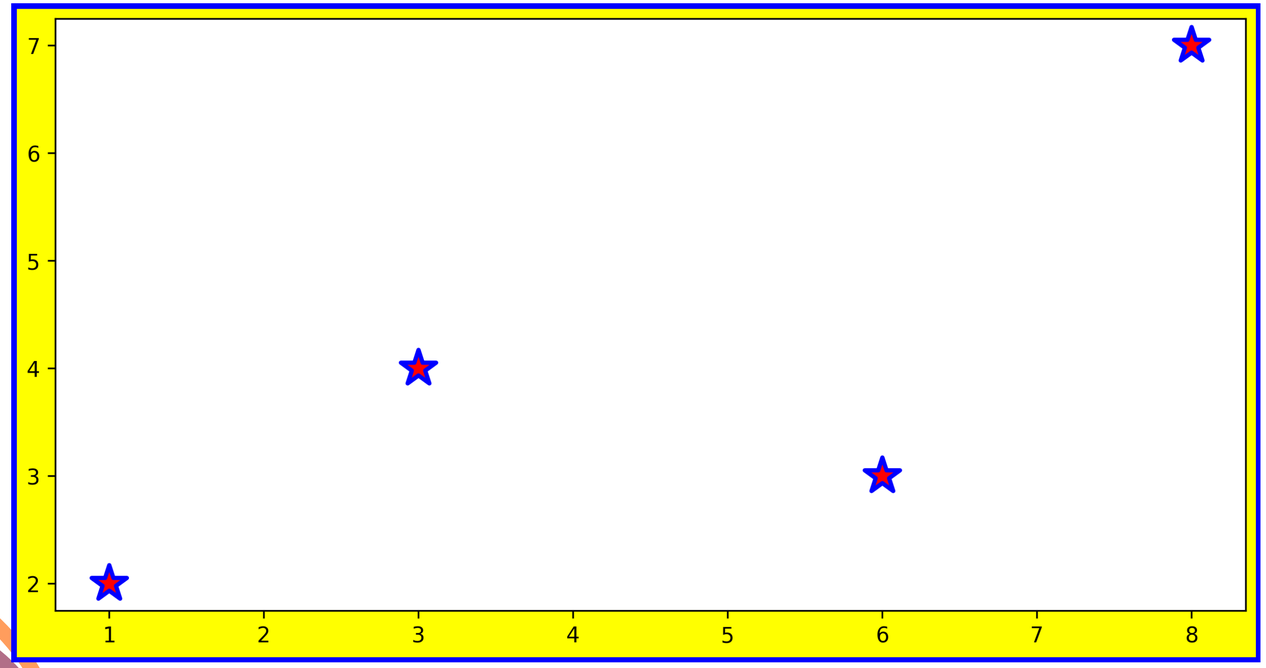
matplotlib.axes

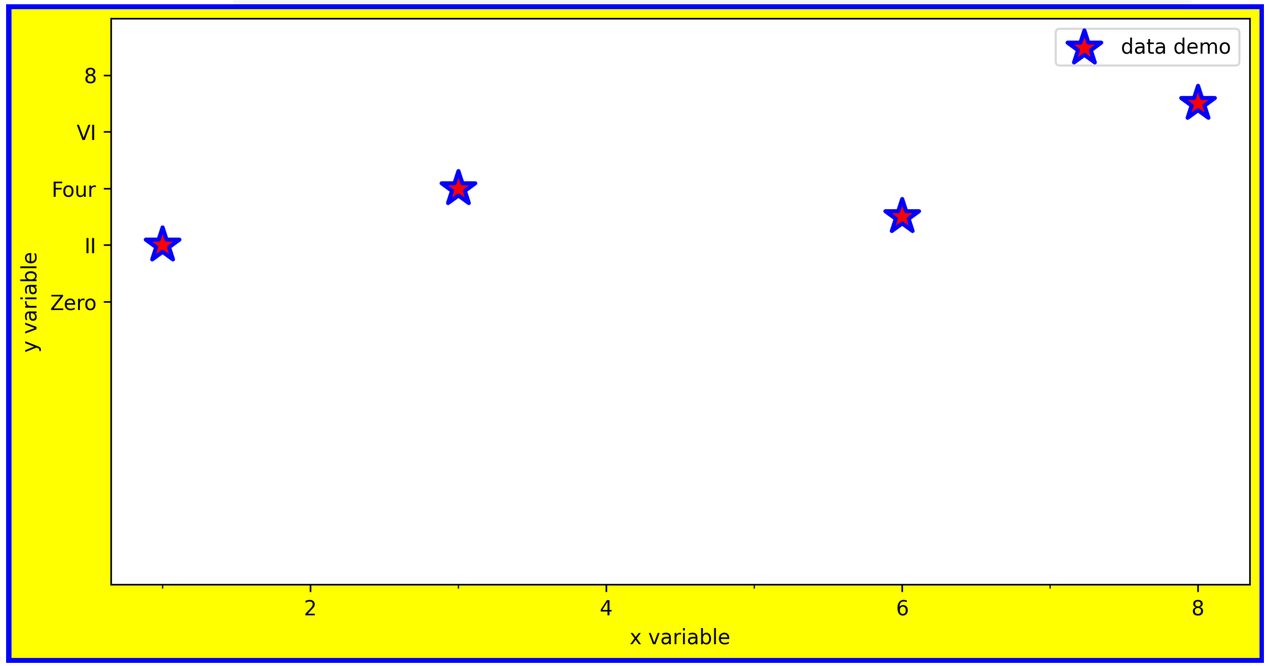
Axes are another important component of a plot. In some cases, we may only want to show a part of the data or highlight a specific range of a variable. Available arguments of the `axes()` function are:

- `xlim`: the range of the x-axis. It takes a tuple of two numbers, which are the minimum and the maximum of the x-axis.
- `xticks`: the major ticks of the x-axis. It takes a list of numbers, which are the positions of the major ticks. You can also modify the minor ticks by setting an additional argument `minor=True`.
- `xlabel`: the label of the x-axis.

Are you able to explain the following code/plot?

```
# figure
plt.figure(**param_fig)
# x axis
# major ticks (0, 2, 4, 6, 8)
plt.xticks(np.arange(0, 10, 2))
# minor ticks (0, 1, 2, ... , 9)
plt.xticks(np.arange(0, 10, 1), minor=True)
plt.xlabel("x variable")
# y axis
plt.yticks(np.arange(0, 10, 2),
           labels=["Zero", "II", "Four", "VI", "8"])
plt.ylabel("y variable")
plt.ylim(-10, 10)
# data
plt.scatter(x, y, **param_data, label="data demo")
# add label for the legend
plt.legend() # legend must be added AFTER the data
```





Axes

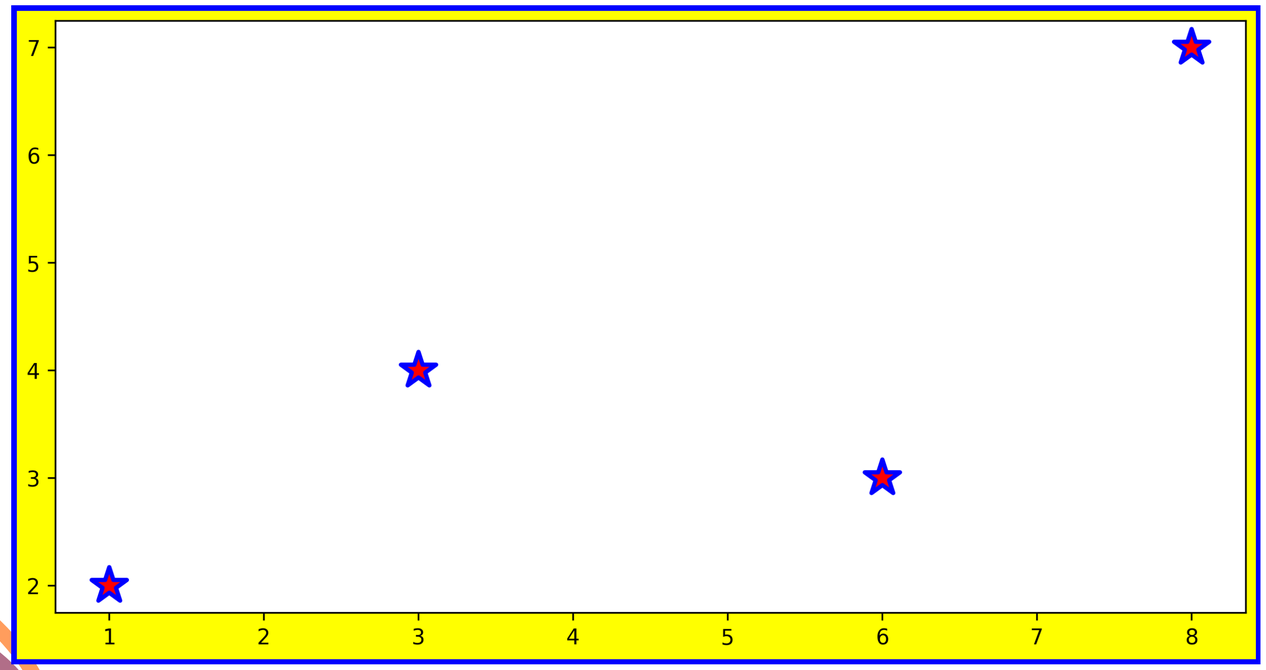
matplotlib.axes

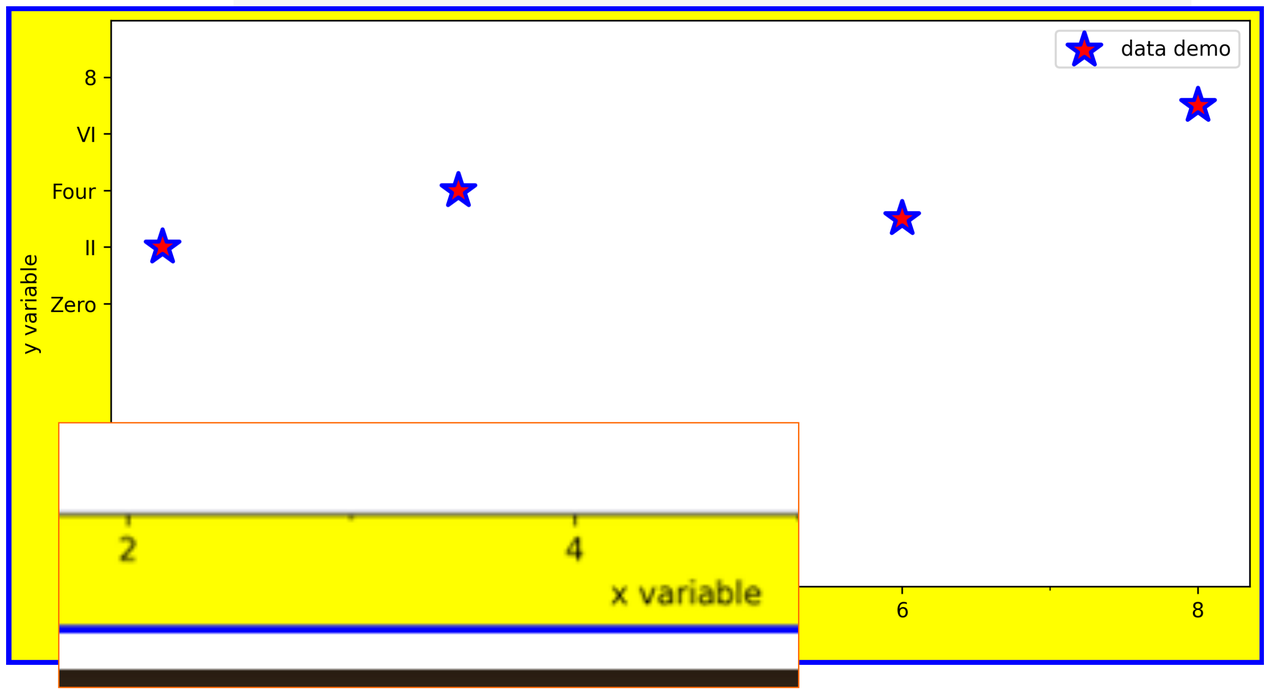
Axes are another important component of a plot. In some cases, we may only want to show a part of the data or highlight a specific range of a variable. Available arguments of the `axes()` function are:

- `xlim`: the range of the x-axis. It takes a tuple of two numbers, which are the minimum and the maximum of the x-axis.
- `xticks`: the major ticks of the x-axis. It takes a list of numbers, which are the positions of the major ticks. You can also modify the minor ticks by setting an additional argument `minor=True`.
- `xlabel`: the label of the x-axis.

Are you able to explain the following code/plot?

```
# figure
plt.figure(**param_fig)
# x axis
# major ticks (0, 2, 4, 6, 8)
plt.xticks(np.arange(0, 10, 2))
# minor ticks (0, 1, 2, ... , 9)
plt.xticks(np.arange(0, 10, 1), minor=True)
plt.xlabel("x variable")
# y axis
plt.yticks(np.arange(0, 10, 2),
           labels=["Zero", "II", "Four", "VI", "8"])
plt.ylabel("y variable")
plt.ylim(-10, 10)
# data
plt.scatter(x, y, **param_data, label="data demo")
# add label for the legend
plt.legend() # legend must be added AFTER the data
```





Axes

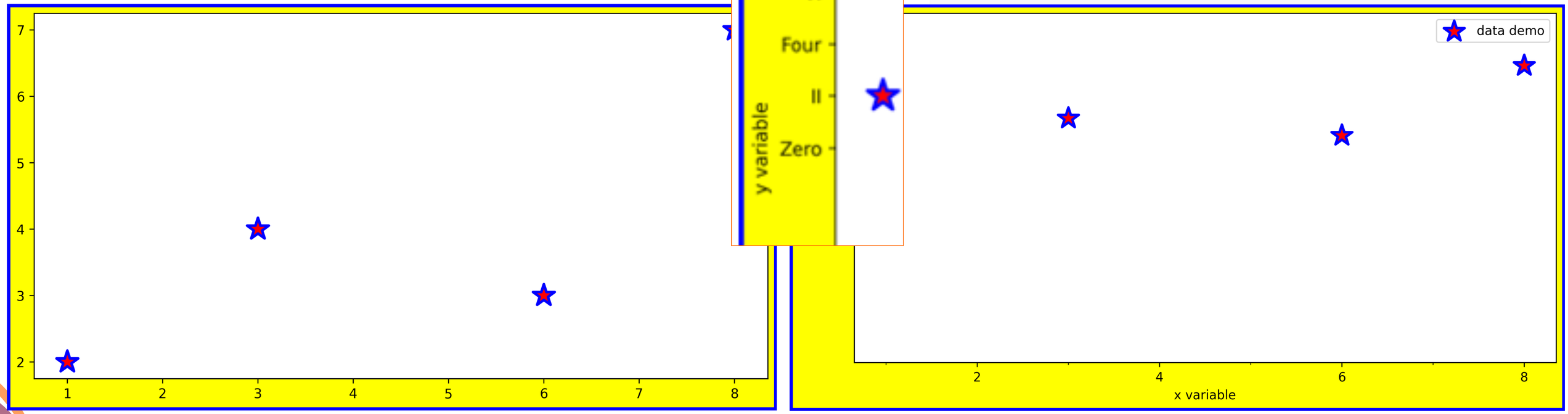
matplotlib.axes

Axes are another important component of a plot. In some cases, we may only want to show a part of the data or highlight a specific range of a variable. Available arguments of the `axes()` function are:

- `xlim`: the range of the x-axis. It takes a tuple of two numbers, which are the minimum and the maximum of the x-axis.
- `xticks`: the major ticks of the x-axis. It takes a list of numbers, which are the positions of the major ticks. You can also modify the minor ticks by setting an additional argument `minor=True`.
- `xlabel`: the label of the x-axis.

Are you able to explain the following code/plot?

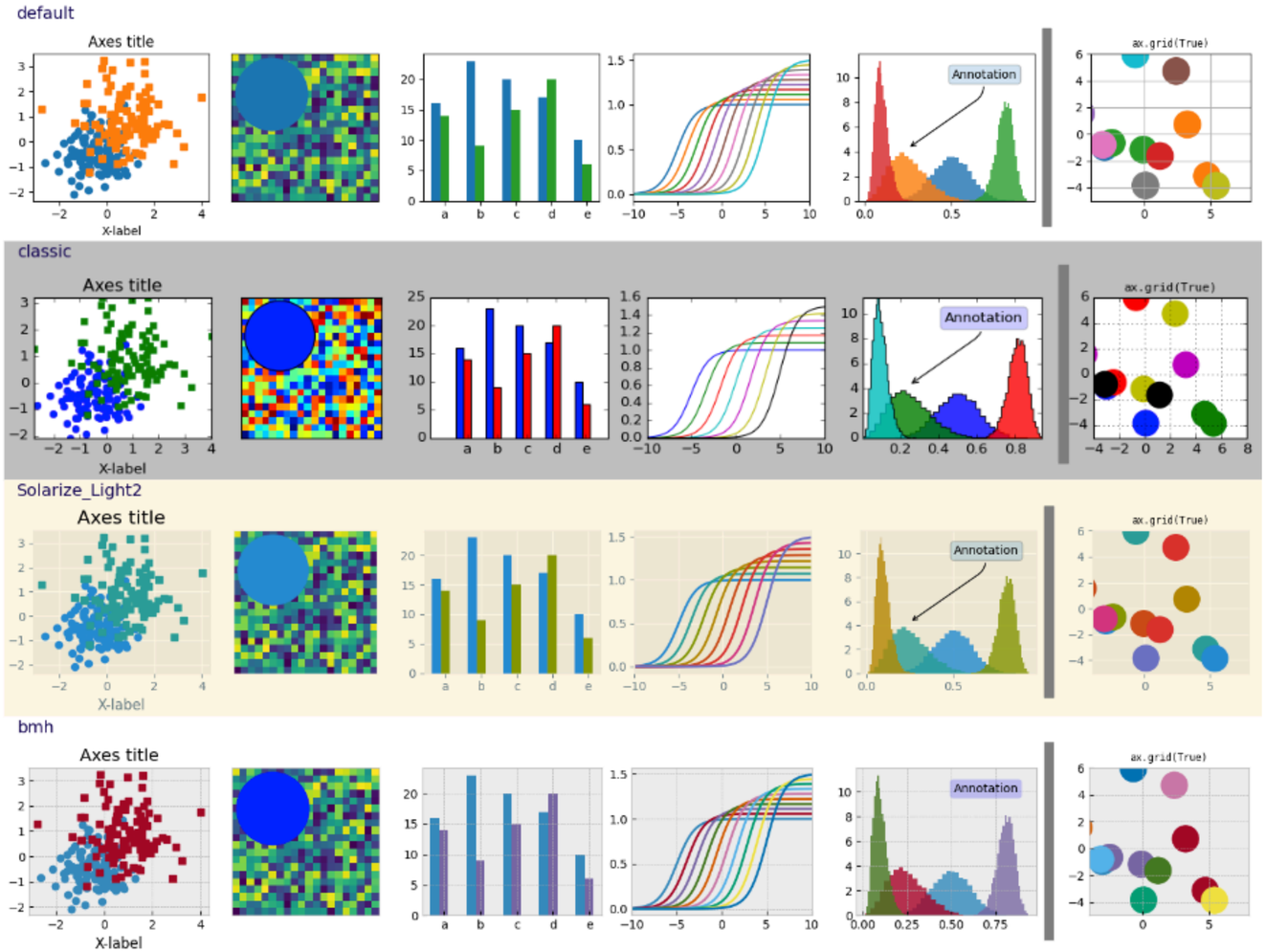
```
# figure
plt.figure(**param_fig)
# x axis
# major ticks (0, 2, 4, 6, 8)
plt.xticks(np.arange(0, 10, 2))
# minor ticks (0, 1, 2, ... , 9)
plt.xticks(np.arange(0, 10, 1), minor=True)
plt.xlabel("x variable")
# y axis
plt.yticks(np.arange(0, 10, 2),
           labels=["Zero", "II", "Four", "VI", "8"])
plt.ylabel("y variable")
plt.ylim(-10, 10)
# data
plt.scatter(x, y, **param_data, label="data demo")
# add label for the legend
plt.legend() # legend must be added AFTER the data
```

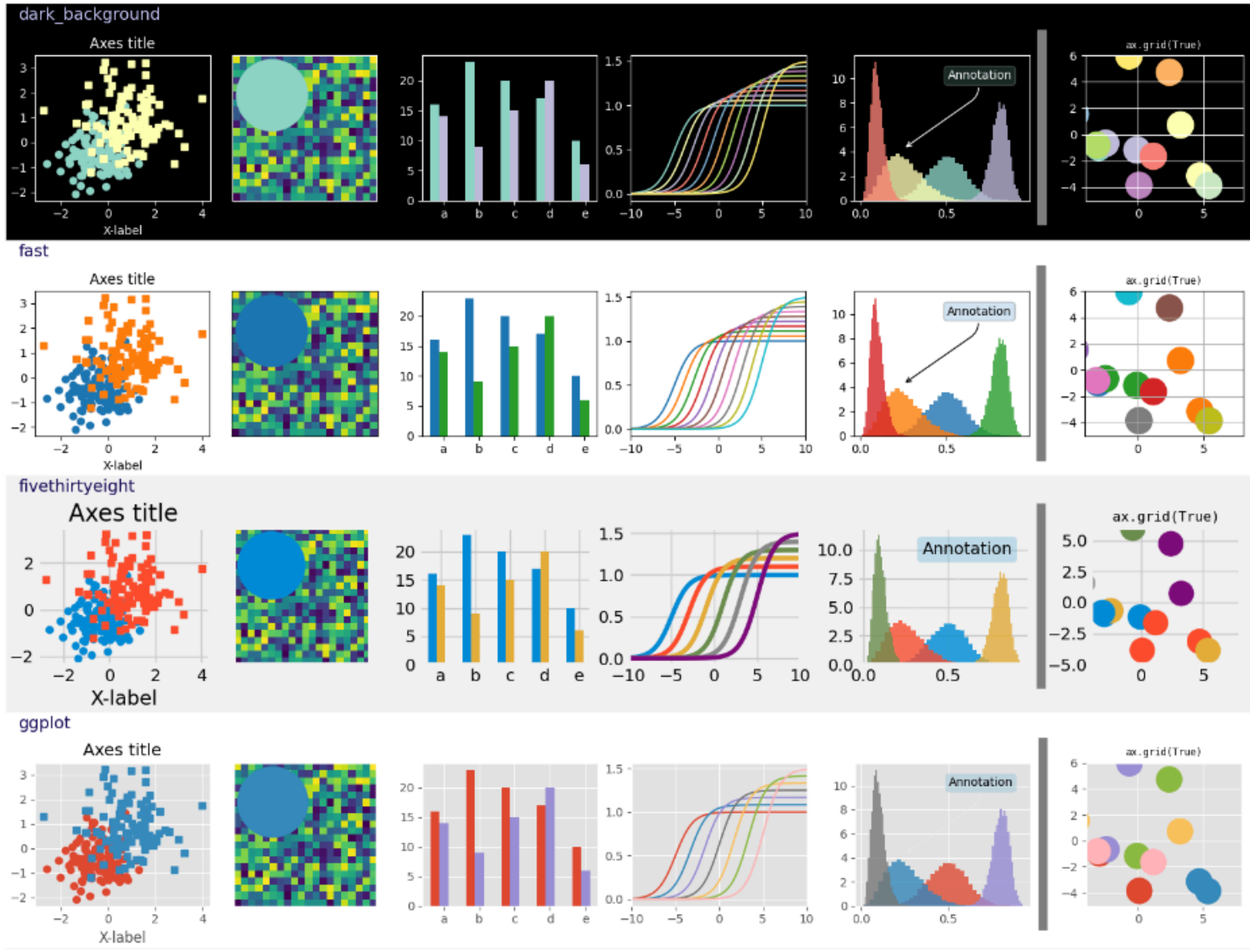


Style Sheets

Style sheets reference

This script demonstrates the different available style sheets on a common set of example plots: scatter plot, image, bar graph, patches, line plot and histogram,





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

Revisit the Tidy Dataset

This dataset records various factors related to death cases, including **sex**, **age**, **time of death**, and **cause of death.** We aim to examine the **relationship between the death rate and the three variables**

```
data = pd.read_csv("data/tidy_X.csv")
data
```

✓ 0.1s

	sex	age	yod	mod	dod	hod	cod
0	1	90	2008	1	7	20	F17
1	1	72	2008	1	13	14	I05
2	1	49	2008	1	12	20	K65
3	2	79	2008	1	20	10	I38
4	1	15	2008	1	1	15	N18
...	...	...	...	...	...	...	...
528323	1	1	2008	10	6	12	P22
528324	2	20	2008	10	18	20	Q24
528325	2	3	2008	11	11	19	P22
528326	1	24	2008	9	25	12	P22
528327	1	2	2008	9	22	16	P26

528328 rows × 7 columns

The case count for each subgroup

```
data_ct = data.\
    groupby(["sex", "age", "hod"]).\
        agg(count=("hod", "count")).\
            reset_index()
data_ct
```

✓ 0.0s

	sex	age	hod	count
0	1	1	0	171
1	1	1	1	152
2	1	1	2	185
3	1	1	3	190
4	1	1	4	197
...	...	...	...	...
4747	2	99	19	40
4748	2	99	20	40
4749	2	99	21	38
4750	2	99	22	35
4751	2	99	23	23

4752 rows × 4 columns

Revisit the Tidy Dataset

This dataset records various factors related to death cases, including **sex**, **age**, **time of death**, and **cause of death.** We aim to examine the **relationship between the death rate and the three variables**

	sex	age	hod	count
0	1	1	0	171
1	1	1	1	152
2	1	1	2	185
3	1	1	3	190
4	1	1	4	197
...	...	...	...	...
4747	2	99	19	40
4748	2	99	20	40
4749	2	99	21	38
4750	2	99	22	35
4751	2	99	23	23

4752 rows × 4 columns

Use the **sex factor** as the denominator of the **death rate**.

```
data_ct = data_ct.\
    groupby("sex").\
        agg(sum=("count", "sum")).\
            reset_index().\
                merge(data_ct)
data_ct["rate"] = data_ct["count"] / data_ct["sum"]
data_ct["per_mille"] = data_ct["rate"] * 1000
data_ct
```

✓ 0.0s

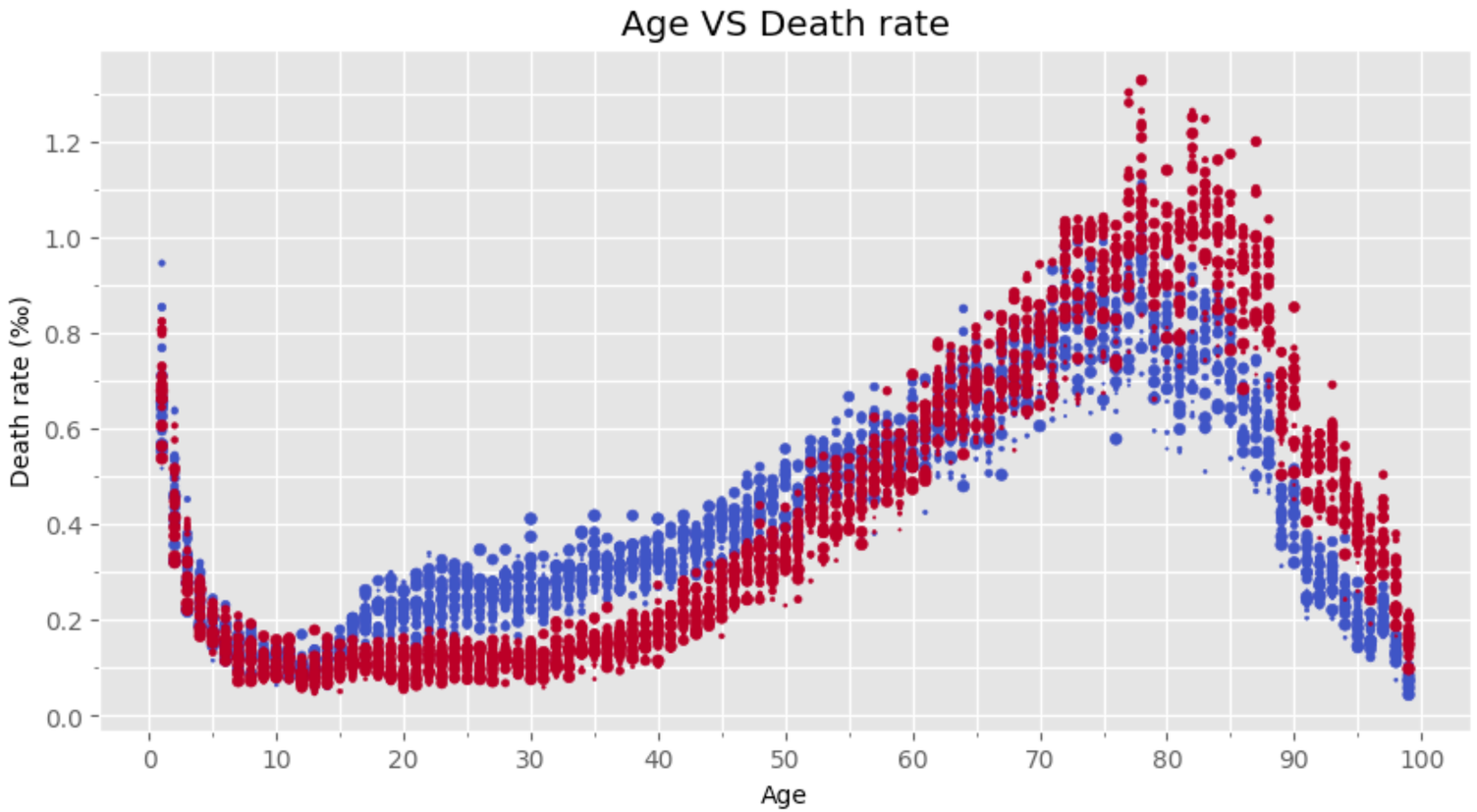
	sex	sum	age	hod	count	rate	per_mille
0	1	294035	1	0	171	0.000582	0.581563
1	1	294035	1	1	152	0.000517	0.516945
2	1	294035	1	2	185	0.000629	0.629177
3	1	294035	1	3	190	0.000646	0.646182
4	1	294035	1	4	197	0.000670	0.669988
...	...	...	...	...	...	...	...
4747	2	234293	99	19	40	0.000171	0.170726
4748	2	234293	99	20	40	0.000171	0.170726
4749	2	234293	99	21	38	0.000162	0.162190
4750	2	234293	99	22	35	0.000149	0.149386
4751	2	234293	99	23	23	0.000098	0.098168

4752 rows × 7 columns

Revisit the Tidy Dataset

	sex	sum	age	hod	count	rate	per_mille
0	1	294035	1	0	171	0.000582	0.581563
1	1	294035	1	1	152	0.000517	0.516945
2	1	294035	1	2	185	0.000629	0.629177
3	1	294035	1	3	190	0.000646	0.646182
4	1	294035	1	4	197	0.000670	0.669988
...	...	...	...	...	...	...	...

```
# figure
plt.style.use("ggplot")
plt.figure(figsize=(10, 5))
plt.grid(True, which="both")
plt.title("Age VS Death rate")
# axis
plt.xlabel("Age")
plt.xticks(np.arange(0, 110, 10))
plt.xticks(np.arange(0, 110, 5), minor=True)
plt.ylabel("Death rate (‰)")
plt.yticks(np.arange(0, 1.6, 0.1), minor=True)
# data
plt.scatter("age", "per_mille",
            s="hod",
            c="sex",
            data=data_ct,
            cmap="coolwarm")
```



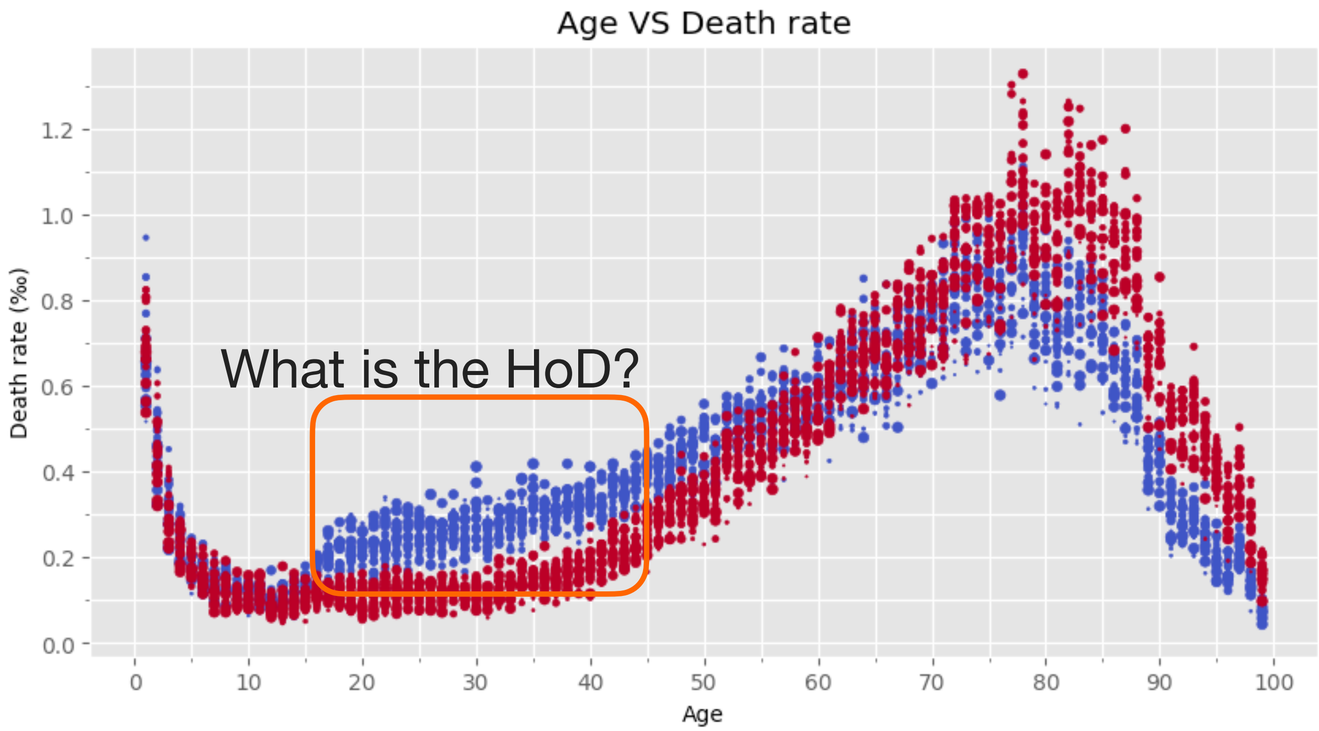
Revisit the Tidy Dataset - Categorize a Variable

Age and Death Rate by Gender by Hour of the Day

The figure above shows the relationship between four variables: sex, age, death, and hod. To further explore this relationship, we can break down the death rate by hour of the day. Specifically, we can categorize the hour of the day into four groups:

- 5 am to 11 am
- 11 am to 5 pm
- 5 pm to 11 pm
- 11 pm to 5 am

```
def cat_hod(x):
    if x ⩾ 5 and x < 11:
        return "5 am - 11 am"
    elif x ⩾ 11 and x < 17:
        return "11 am - 5 pm"
    elif x ⩾ 17 and x < 23:
        return "5 pm - 11 pm"
    else:
        return "11 pm - 5 am"
data_ct["hod_cat"] = data_ct["hod"].apply(cat_hod)
data_ct
```



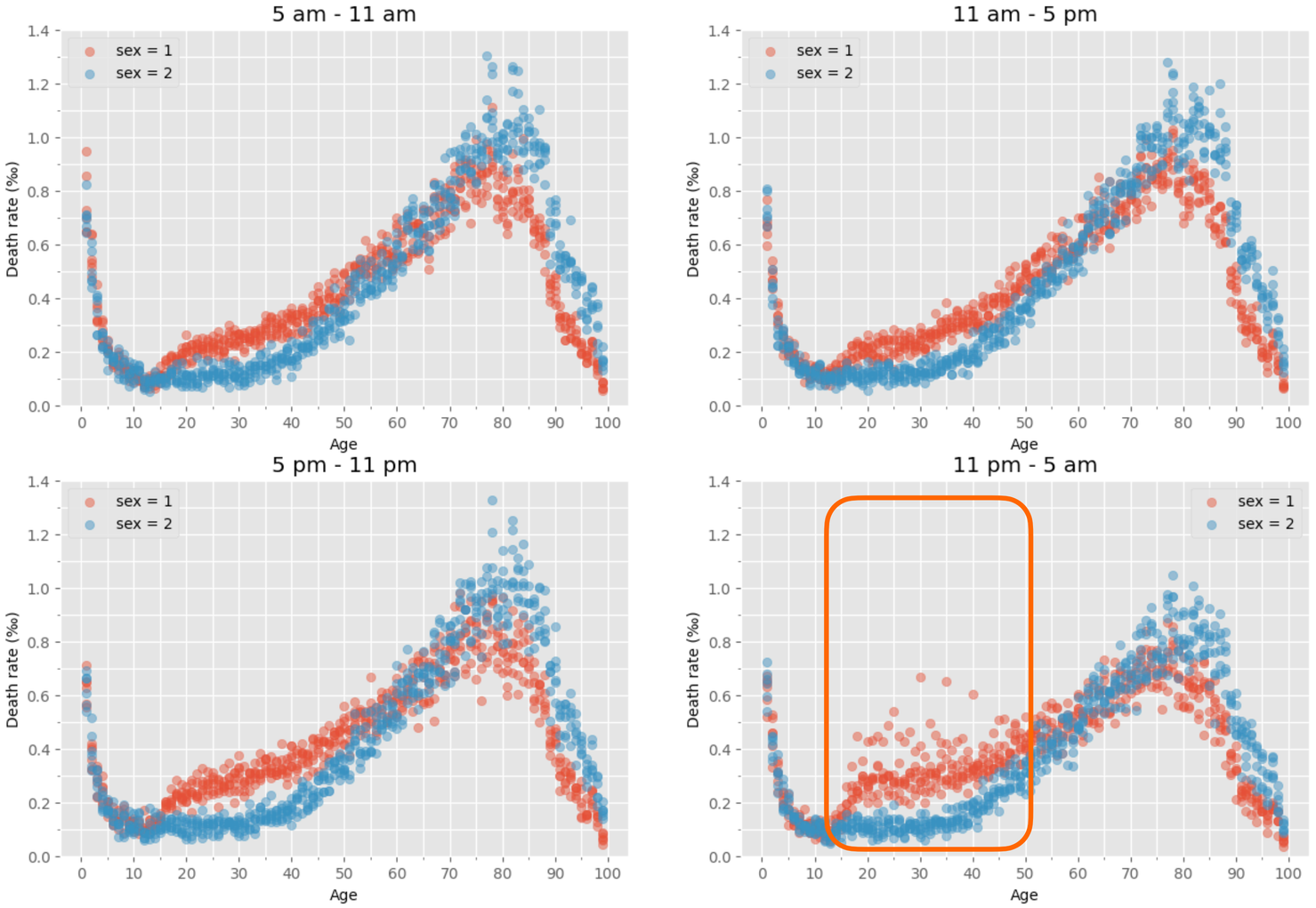
	sex	sum	age	hod	count	rate	per_mille
0	1	294035	1	0	171	0.000582	0.581563
1	1	294035	1	1	152	0.000517	0.516945
2	1	294035	1	2	185	0.000629	0.629177
3	1	294035	1	3	190	0.000646	0.646182
4	1	294035	1	4	197	0.000670	0.669988
...	...	...	...	...	...	...	...

	sex	sum	age	hod	count	rate	per_mille	hod_cat
0	1	294035	1	0	171	0.000582	0.581563	11 pm - 5 am
1	1	294035	1	1	152	0.000517	0.516945	11 pm - 5 am
2	1	294035	1	2	185	0.000629	0.629177	11 pm - 5 am
3	1	294035	1	3	190	0.000646	0.646182	11 pm - 5 am
4	1	294035	1	4	197	0.000670	0.669988	11 pm - 5 am
...	...	...	...	...	...	...	...	...

Revisit the Tidy Dataset - Subplots

Two-row, two-column plot

```
# same available parameters as plt.figure()
figure, axes = plt.subplots(2, 2, figsize=(15, 10))
axes = axes.flatten() # from 2D array to 1D array
for i, hod_cat in enumerate(["5 am - 11 am",
                             "11 am - 5 pm",
                             "5 pm - 11 pm",
                             "11 pm - 5 am"]):
    ax = axes[i]
    # subplot config
    ax.set_title(hod_cat)
    ax.grid(True, which="both")
    # axis
    ax.set_xlabel("Age")
    ax.set_xticks(np.arange(0, 110, 10))
    ax.set_xticks(np.arange(0, 110, 5), minor=True)
    ax.set_ylabel("Death rate (‰)")
    ax.set_yticks(np.arange(0, 1.6, 0.1), minor=True)
    ax.set_ylim(0, 1.4)
    # data
    data_sub = data_ct.query("hod_cat == @hod_cat")
    for s in [1, 2]:
        data_sub_s = data_sub.query("sex == @s")
        ax.scatter("age", "per_mille", alpha=.5,
                   data=data_sub_s, label="sex = " + str(s))
    ax.legend()
```



	sex	sum	age	hod	count	rate	per_mille	hod_cat
0	1	294035	1	0	171	0.000582	0.581563	11 pm - 5 am
1	1	294035	1	1	152	0.000517	0.516945	11 pm - 5 am
2	1	294035	1	2	185	0.000629	0.629177	11 pm - 5 am
3	1	294035	1	3	190	0.000646	0.646182	11 pm - 5 am
4	1	294035	1	4	197	0.000670	0.669988	11 pm - 5 am

Revisit the Tidy Dataset - Histogram

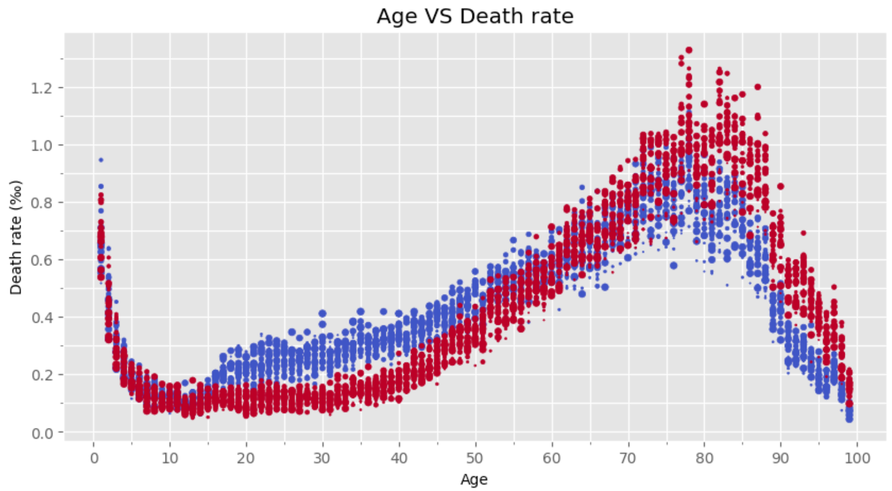
Histogram

Scatter plots are commonly used to visualize the relationship between two variables. However, in some cases, we may be interested in visualizing the distribution of a single variable. For example, we demonstrated above how to show the distribution of the death rate over different age groups. In this case, a histogram or a density plot may be more appropriate than a scatter plot.

Let's re-load the data and only do the categorization of the hour of the day.

```
# density plot
data = pd.read_csv("data/tidy_X.csv")
data["hod_cat"] = data["hod"].apply(cat_hod)
data
```

	sex	age	yod	mod	dod	hod	cod	hod_cat
0	1	90	2008	1	7	20	F17	5 pm - 11 pm
1	1	72	2008	1	13	14	I05	11 am - 5 pm
2	1	49	2008	1	12	20	K65	5 pm - 11 pm
3	2	79	2008	1	20	10	I38	5 am - 11 am
4	1	15	2008	1	1	15	N18	11 am - 5 pm
...	...	...	...	...	...	...	...	...
528323	1	1	2008	10	6	12	P22	11 am - 5 pm
528324	2	20	2008	10	18	20	Q24	5 pm - 11 pm
528325	2	3	2008	11	11	19	P22	5 pm - 11 pm
528326	1	24	2008	9	25	12	P22	11 am - 5 pm
528327	1	2	2008	9	22	16	P26	11 am - 5 pm

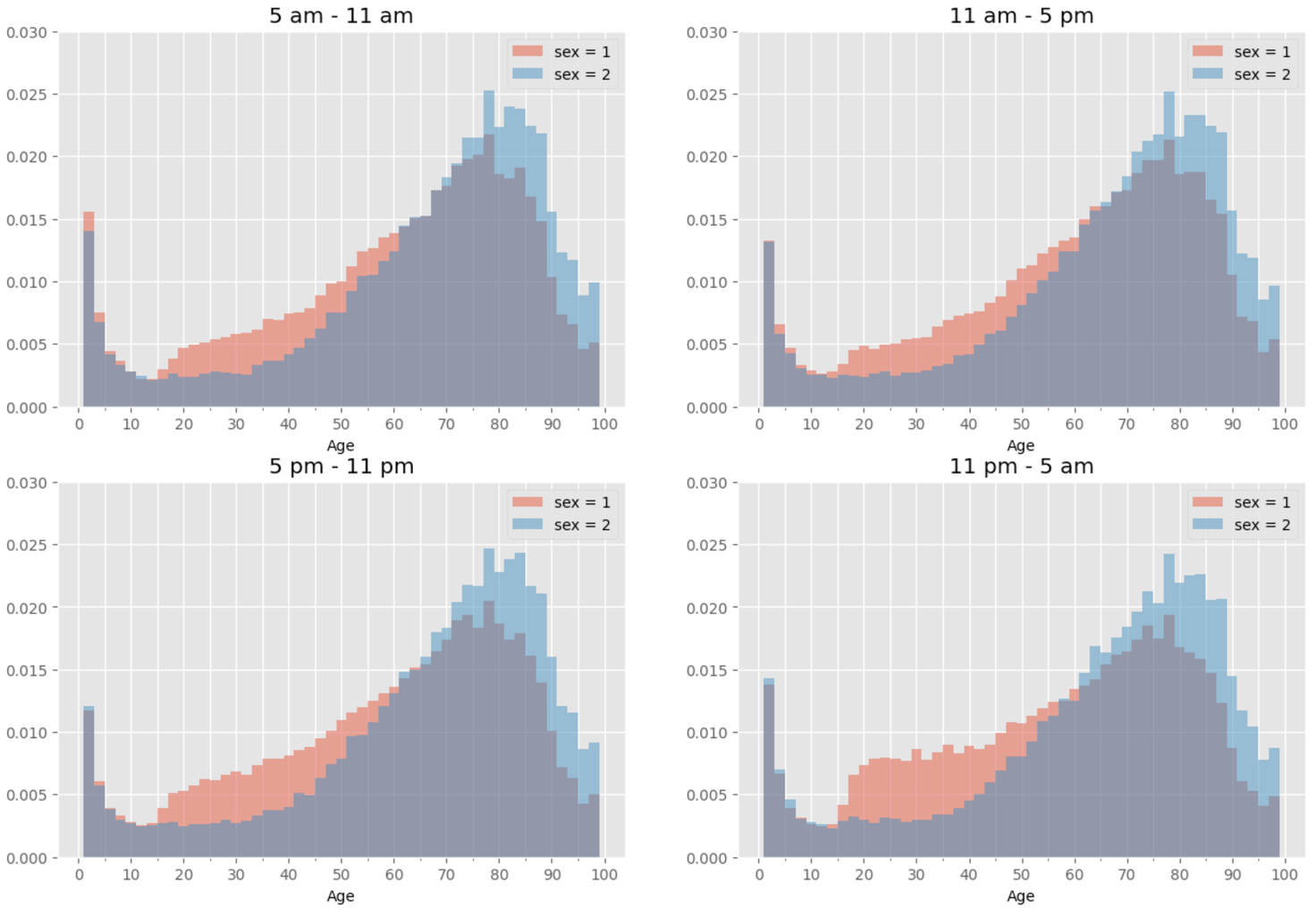


Can we turn it into a histogram?

Revisit the Tidy Dataset - Histogram

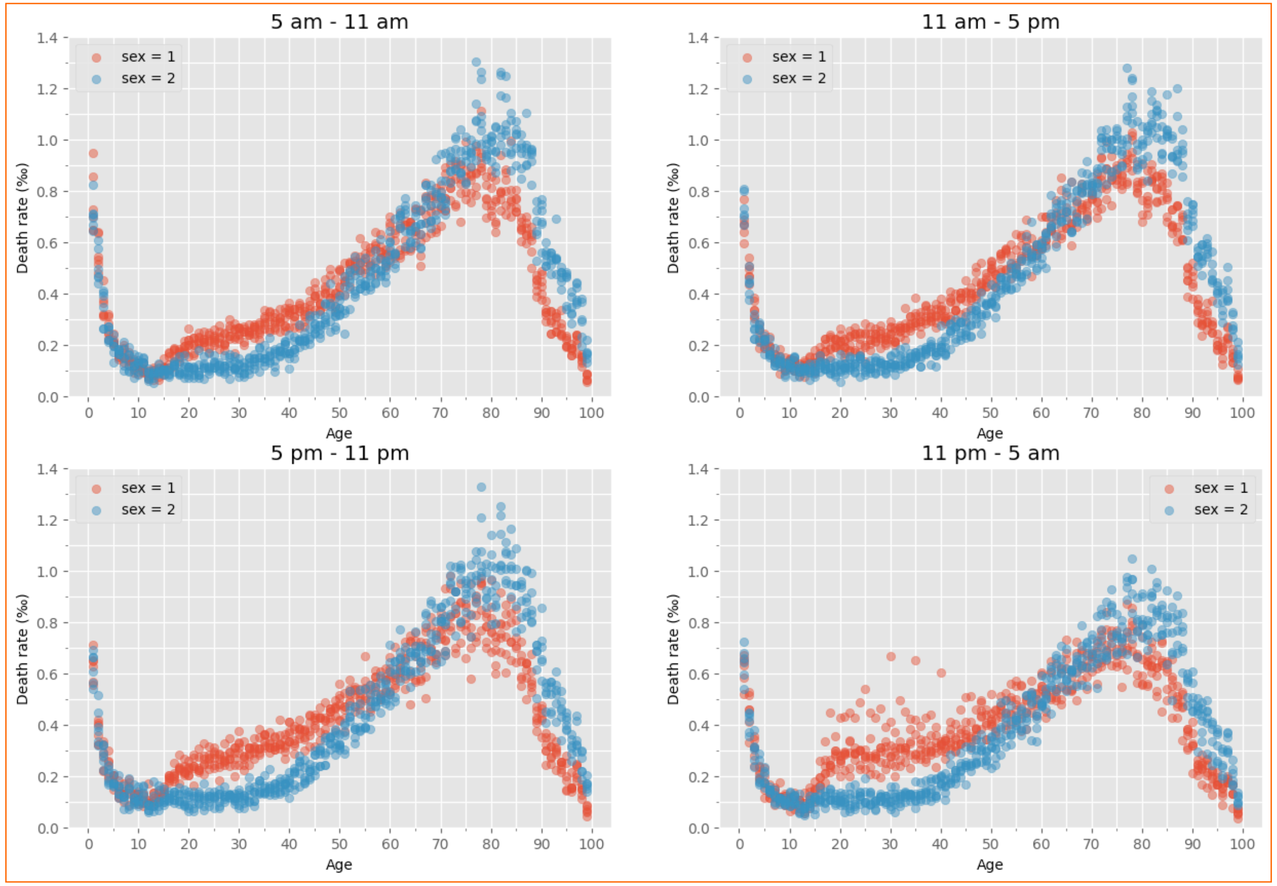
	sex	age	yod	mod	dod	hod	cod	hod_cat
0	1	90	2008	1	7	20	F17	5 pm - 11 pm
1	1	72	2008	1	13	14	I05	11 am - 5 pm
2	1	49	2008	1	12	20	K65	5 pm - 11 pm
3	2	79	2008	1	20	10	I38	5 am - 11 am
4	1	15	2008	1	1	15	N18	11 am - 5 pm

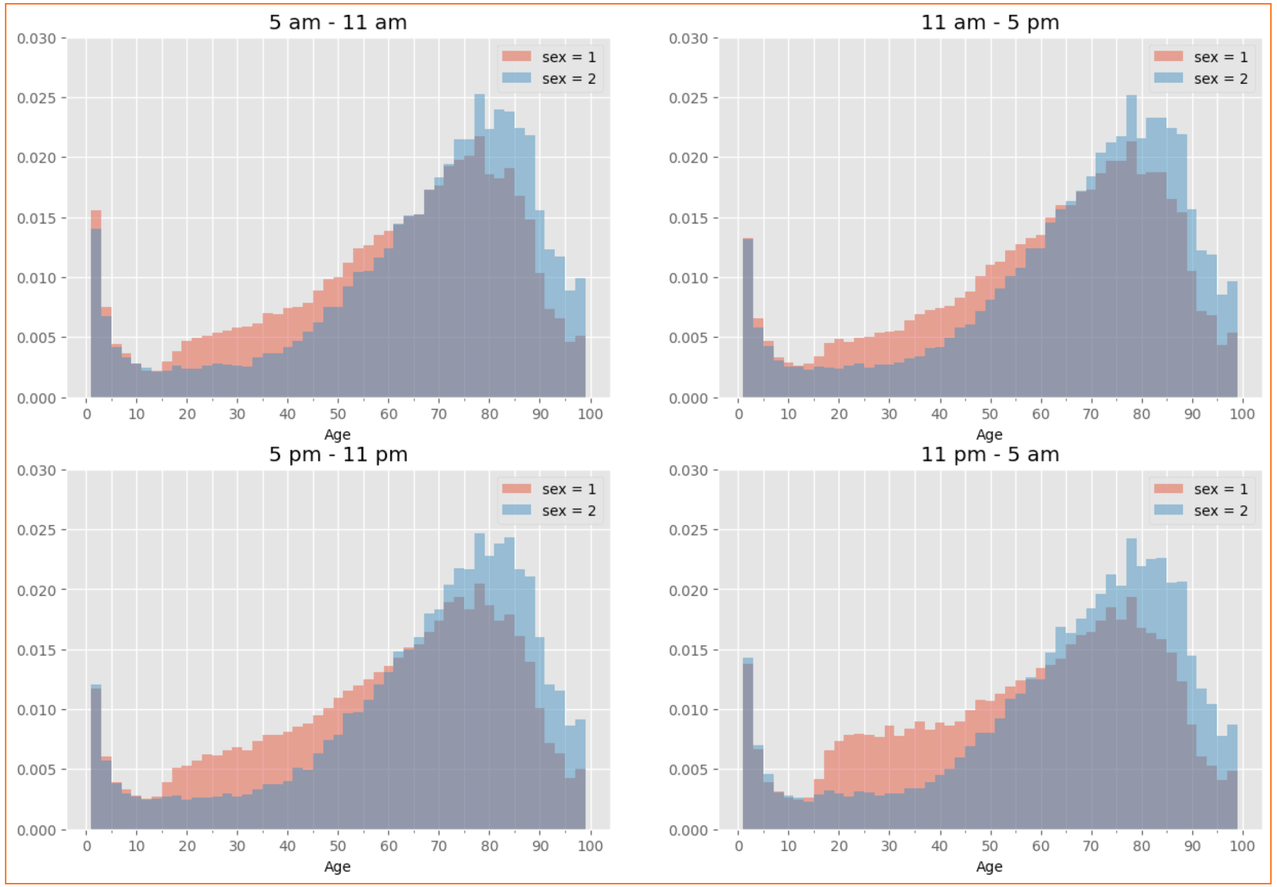
```
# same available parameters as plt.figure()
figure, axes = plt.subplots(2, 2, figsize=(15, 10))
axes = axes.flatten() # from 2D array to 1D array
for i, hod_cat in enumerate([
                             "5 am - 11 am",
                             "11 am - 5 pm",
                             "5 pm - 11 pm",
                             "11 pm - 5 am"]):
    ax = axes[i]
    # subplot config
    ax.set_title(hod_cat)
    ax.grid(True, which="both")
    # axis
    ax.set_xlabel("Age")
    ax.set_xticks(np.arange(0, 110, 10))
    ax.set_xticks(np.arange(0, 110, 5), minor=True)
    ax.set_ylim(0, 0.03)
    # data
    data_sub = data.query("hod_cat == @hod_cat")
    for s in [1, 2]:
        data_sub_s = data_sub.query("sex == @s")
        ax.hist("age", data=data_sub_s, label="sex = " + str(s),
                bins=49, alpha=.5, density=True)
    ax.legend()
```



Scatter Plot VS. Histogram

Which way is better?





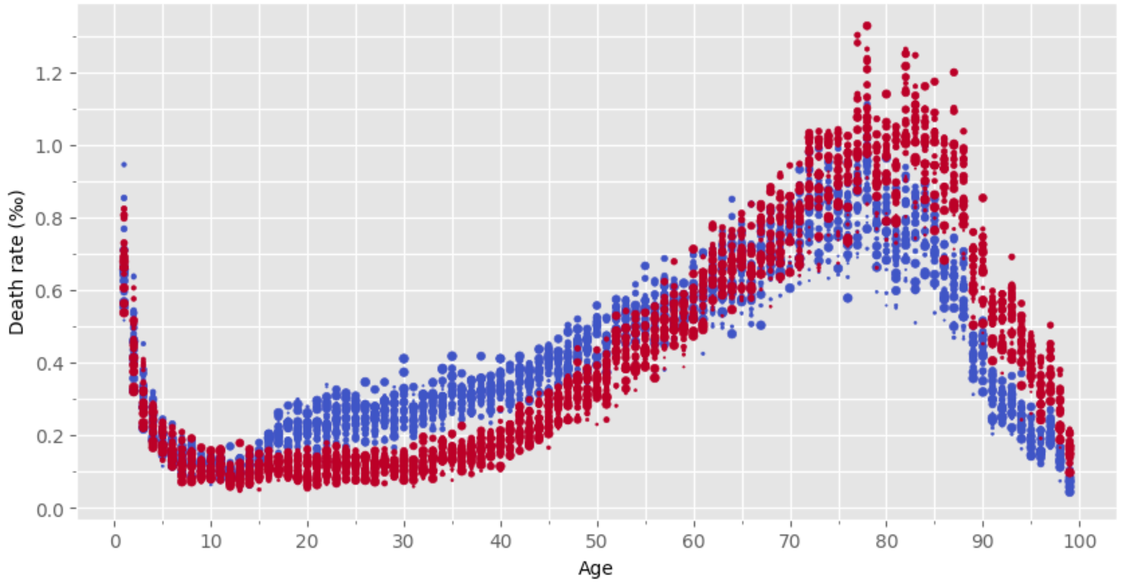
Boxplot

A Scatter plot might not always be a good way to represent the data.

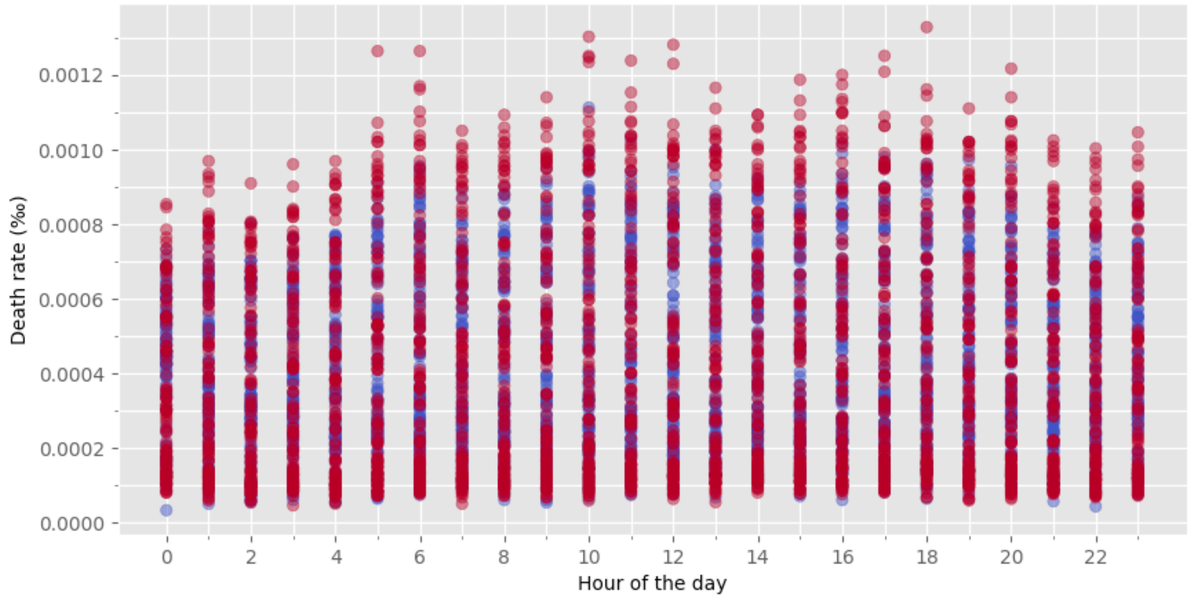
	sex	sum	age	hod	count	rate	per_mille
0	1	294035	1	0	171	0.000582	0.581563
1	1	294035	1	1	152	0.000517	0.516945
2	1	294035	1	2	185	0.000629	0.629177
3	1	294035	1	3	190	0.000646	0.646182
4	1	294035	1	4	197	0.000670	0.669988

```
plt.scatter("age",
            "per_mille",
            c="sex",
            cmap="coolwarm",
            alpha=.5,
            data=data_ct)
```

Age VS Death rate

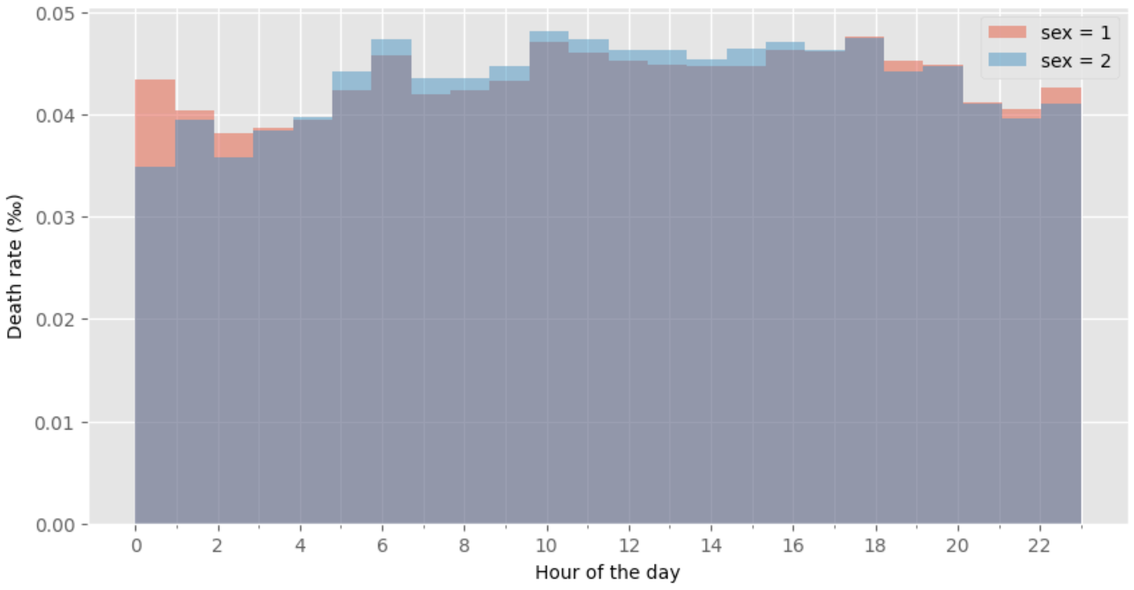



```
plt.scatter("hod",
            "per_mille",
            c="sex",
            cmap="coolwarm",
            alpha=.5,
            data=data_ct)
```

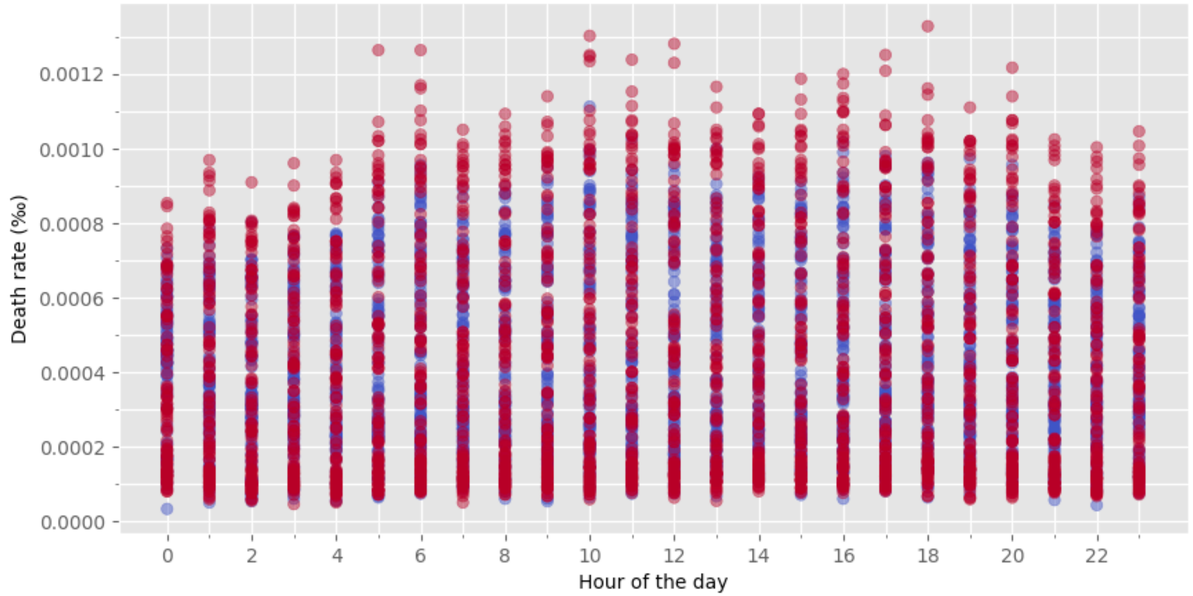


Boxplot

Histogram

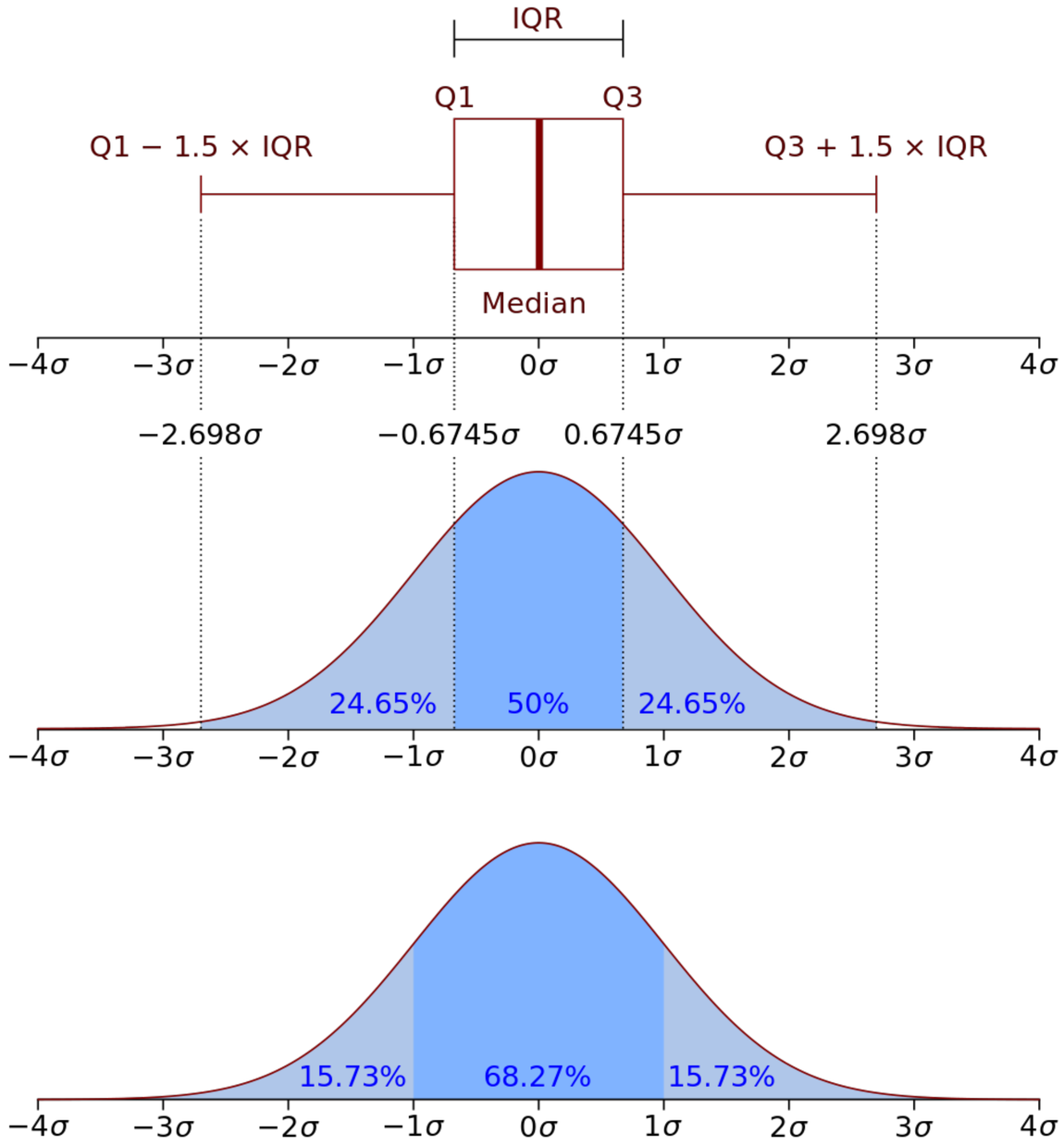


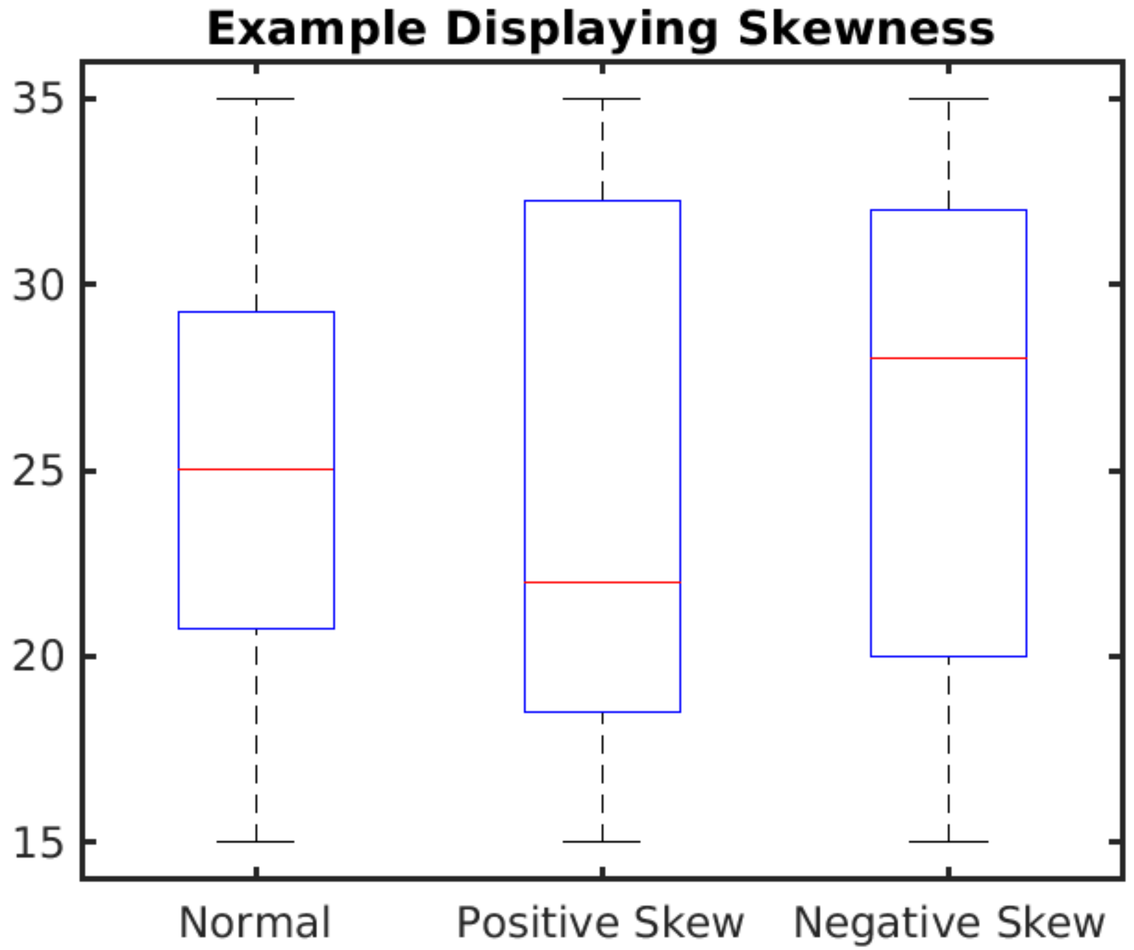
Scatter plot



It may be difficult to observe the distribution of each “x” value in the plot

Boxplot

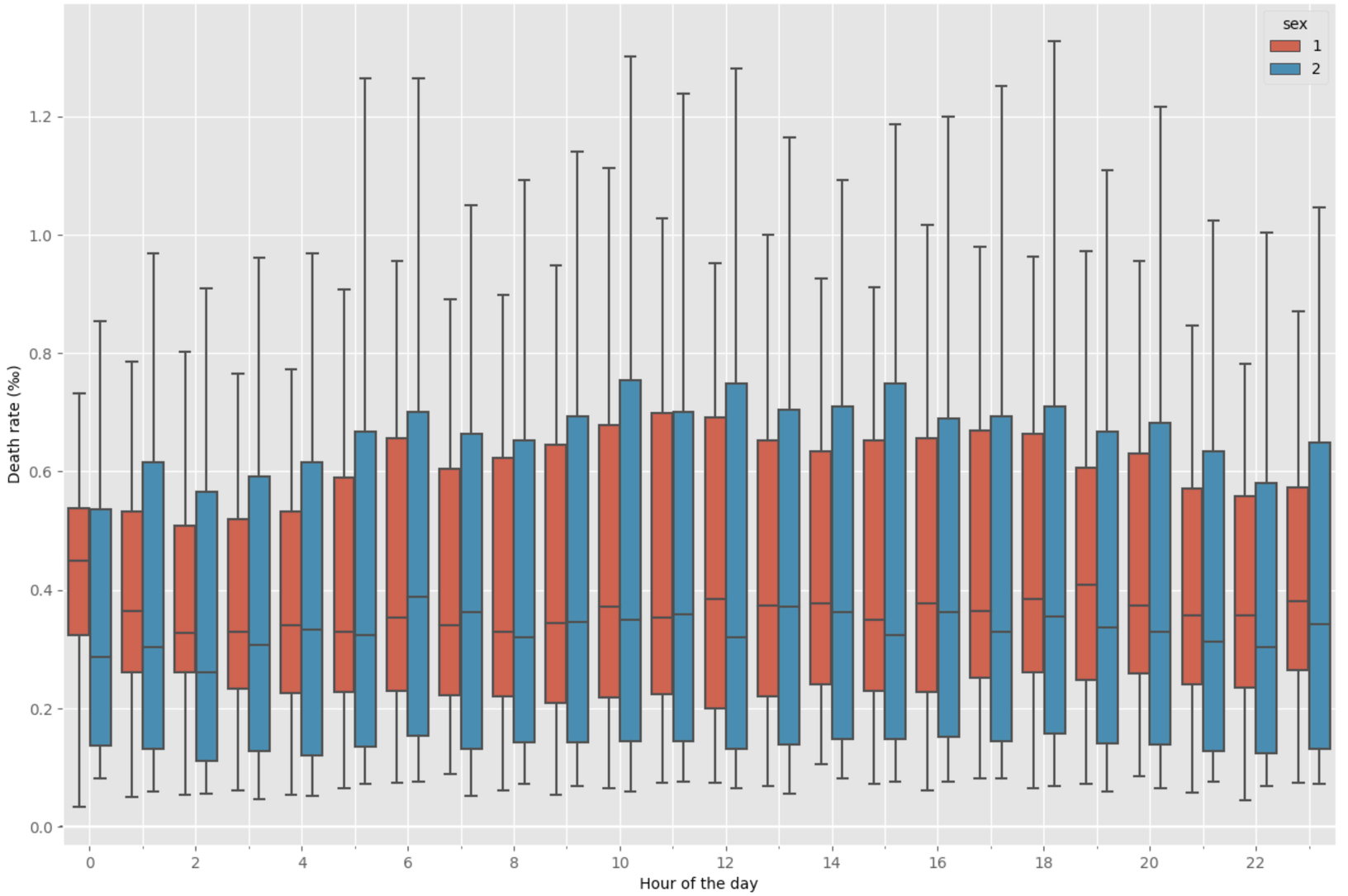




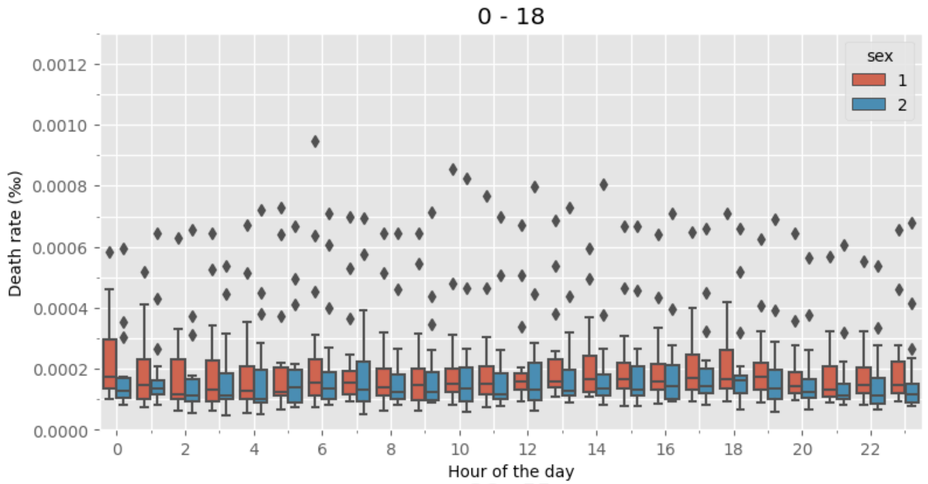
https://en.wikipedia.org/wiki/Box_plot

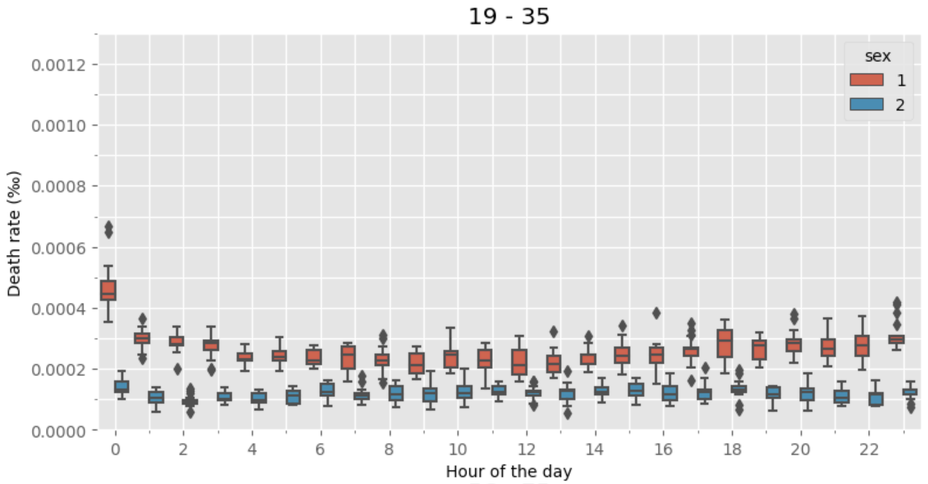
Boxplot

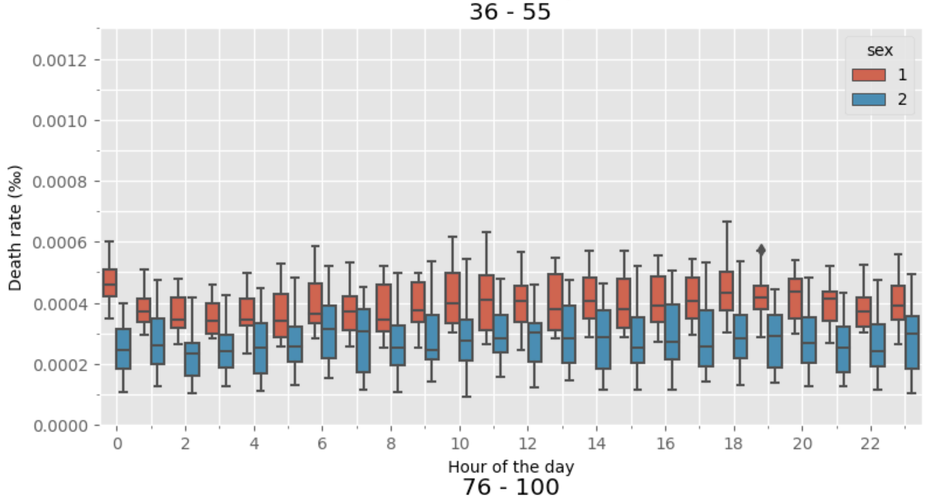

```
import seaborn as sns

# figure
plt.figure(figsize=(15, 10))
sns.boxplot(x="hod",
            y="per_mille",
            hue="sex",
            data=data_ct)
plt.grid(True, which="both")
# axis
plt.xlabel("Hour of the day")
plt.xticks(np.arange(0, 24, 2))
plt.xticks(np.arange(0, 24, 1), minor=True)
plt.ylabel("Death rate (‰)")
plt.yticks(np.arange(0, 0.0014, 0.0001), minor=True)
```

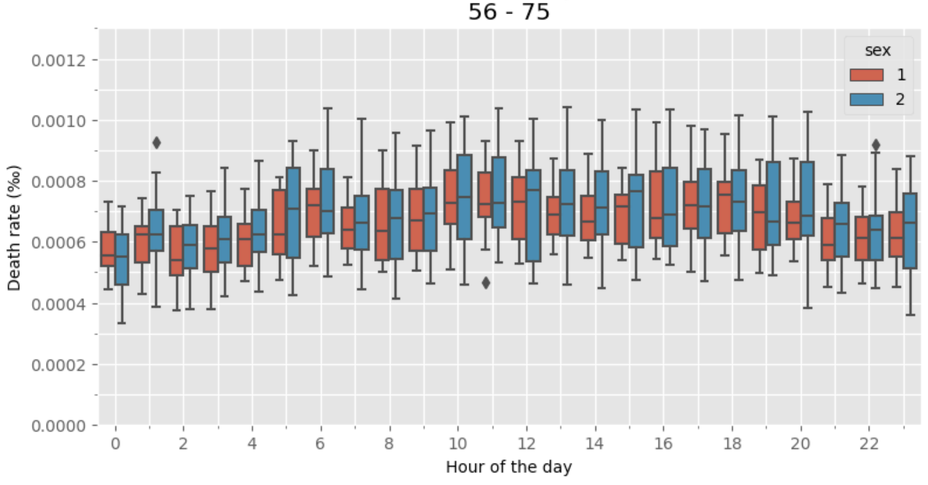


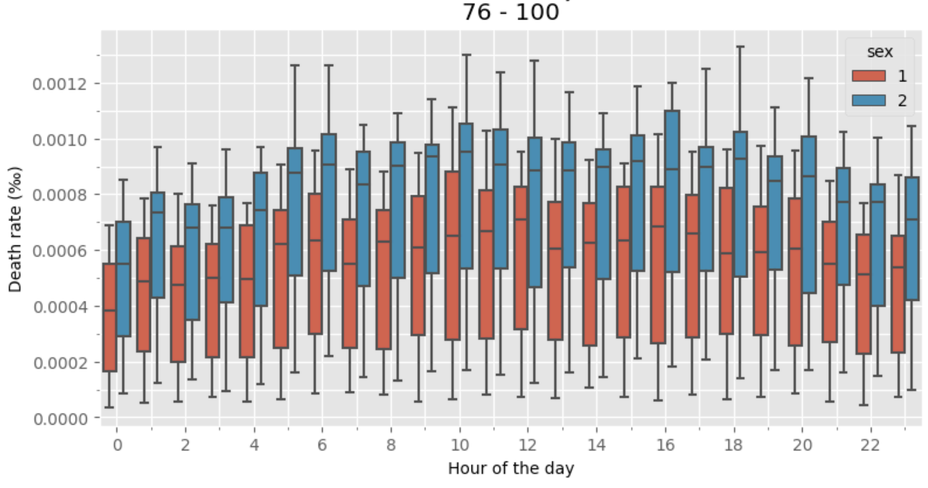
Boxplot


0 - 18
sex
1
2
Death rate (‰)
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0 2 4 6 8 10 12 14 16 18 20 22
Hour of the day


19 - 35
sex
1
2
Death rate (‰)
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0 2 4 6 8 10 12 14 16 18 20 22
Hour of the day

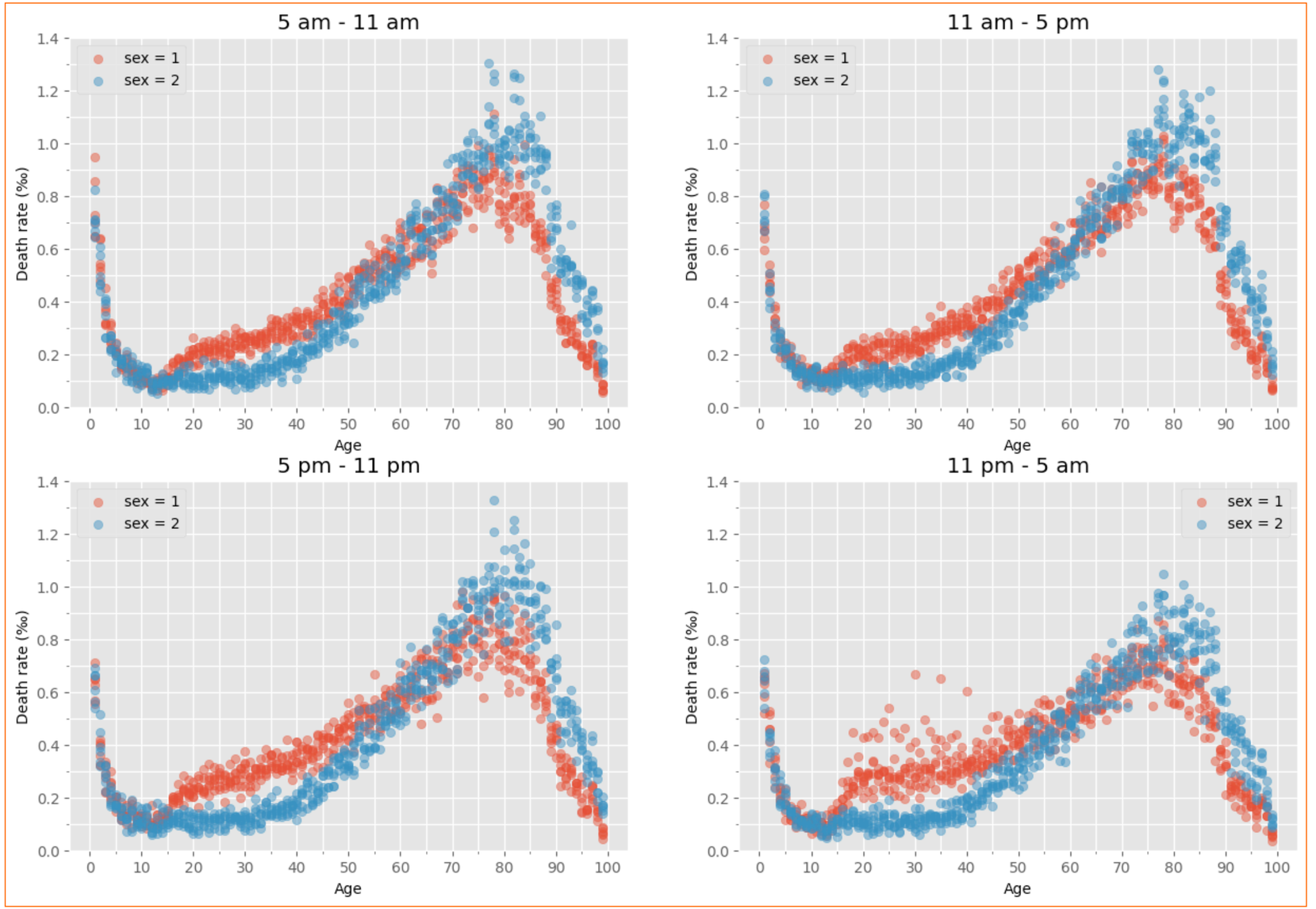

36 - 55
sex
1
2
Death rate (‰)
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0 2 4 6 8 10 12 14 16 18 20 22
Hour of the day


56 - 75
sex
1
2
Death rate (‰)
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0 2 4 6 8 10 12 14 16 18 20 22
Hour of the day


76 - 100
sex
1
2
Death rate (‰)
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0 2 4 6 8 10 12 14 16 18 20 22
Hour of the day

Heatmap

Can we visualize the correlation relationship between “hod” groups?



Heatmap

Need more data wrangling

sex	age	yod	mod	dod	hod	cod	hod_cat
1	90	2008	1	7	20	F17	5 pm - 11 pm
1	72	2008	1	13	14	I05	11 am - 5 pm
1	49	2008	1	12	20	K65	5 pm - 11 pm
2	79	2008	1	20	10	I38	5 am - 11 am
1	15	2008	1	1	15	N18	11 am - 5 pm
...	...	...	...	...	...	...	...
1	1	2008	10	6	12	P22	11 am - 5 pm
2	20	2008	10	18	20	Q24	5 pm - 11 pm
2	3	2008	11	11	19	P22	5 pm - 11 pm
1	24	2008	9	25	12	P22	11 am - 5 pm
1	2	2008	9	22	16	P26	11 am - 5 pm

pd.pivot()

sex	age	hid	11 am - 5 pm	11 pm - 5 am	5 am - 11 am	5 pm - 11 pm
1	1	0	0.669988	0.581563	0.945466	0.710800
1	1	1	0.686993	0.516945	0.697196	0.625776
1	1	2	0.595167	0.629177	0.646182	0.646182
1	1	3	0.666587	0.646182	0.646182	0.567960
1	1	4	0.639380	0.669988	0.853640	0.554356
...	...	...	...	...	...	...
2	99	1	0.162190	0.123777	0.145117	0.170726
2	99	2	0.145117	0.136581	0.132313	0.170726
2	99	3	0.209140	0.093900	0.166458	0.162190
2	99	4	0.183531	0.119508	0.170726	0.149386
2	99	5	0.153654	0.098168	0.162190	0.204872

Heatmap

sex	age	hid	11 am - 5 pm	11 pm - 5 am	5 am - 11 am	5 pm - 11 pm
1	1	0	0.669988	0.581563	0.945466	0.710800
1	1	1	0.686993	0.516945	0.697196	0.625776
1	1	2	0.595167	0.629177	0.646182	0.646182
1	1	3	0.666587	0.646182	0.646182	0.567960
1	1	4	0.639380	0.669988	0.853640	0.554356

```
fig, axes = plt.subplots(1, 2, figsize=(20, 10))
param_map = {"annot": True,
             "cmap": "coolwarm",
             "vmin": 0.9,
             "vmax": 1}
axes[0].set_title("sex = 1")
sns.heatmap(data_ht.query("sex == 1").iloc[:, 1:].corr(), ax=axes[0], **param_map)
axes[1].set_title("sex = 2")
sns.heatmap(data_ht.query("sex == 2").iloc[:, 1:].corr(), ax=axes[1], **param_map)
```