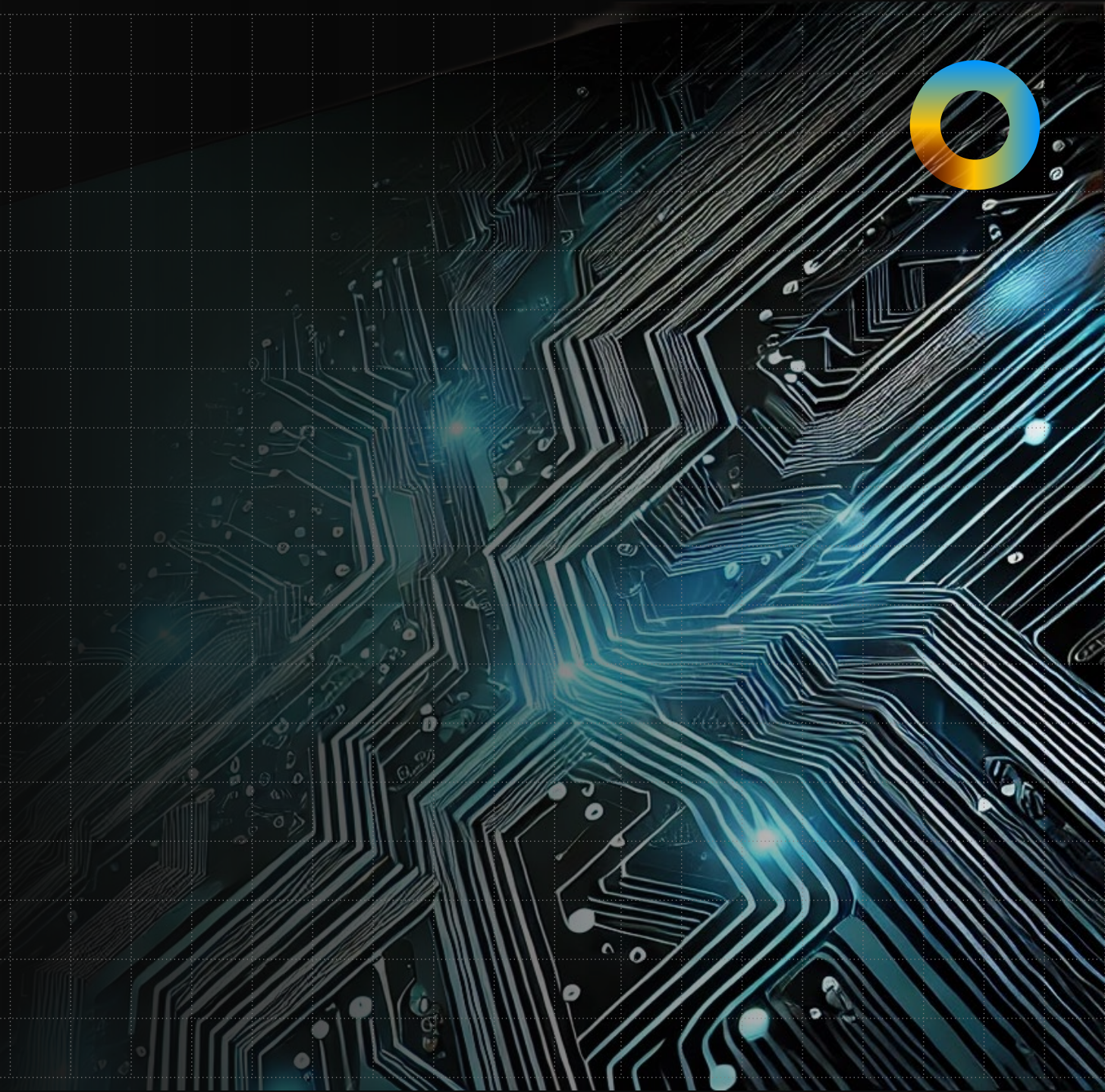
Python Programming

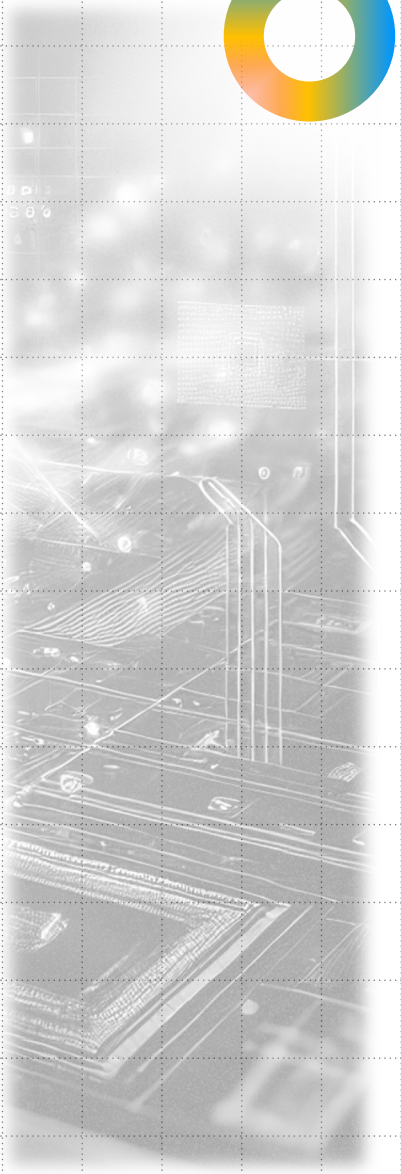
Visualization

Dr. Chun-Hsiang Chan
Department of Geography
National Taiwan Normal University



Outlines

- Visualization Methods
- Gallery
- Matplotlib
- Colormap
- Line Plot
- Export Figure
- Subplot & Style Adjustment
- Scatter & Bar Chart
- Fill Between & Stack Plot
- Histogram & Accumulated Histogram
- Hexbin & Hist2D
- Box & Violin Plot
- Quiver
- Contour & Contourf
- Surface & Trisurf
- Interpolation for imshow
- Seaborn
- Pairplot
- Plotly
- Sankey Diagram



Visualization – Methods

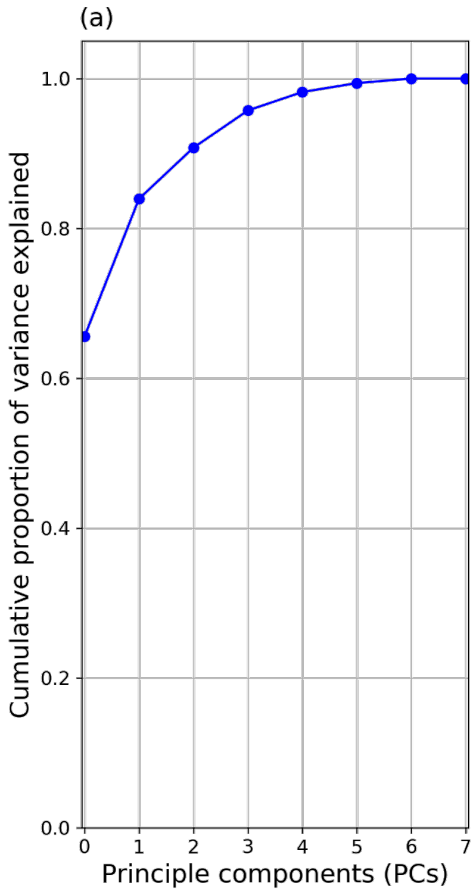
- Several visualization functions were developed in built-in package, such as line, scatter, boxplot, and histogram.

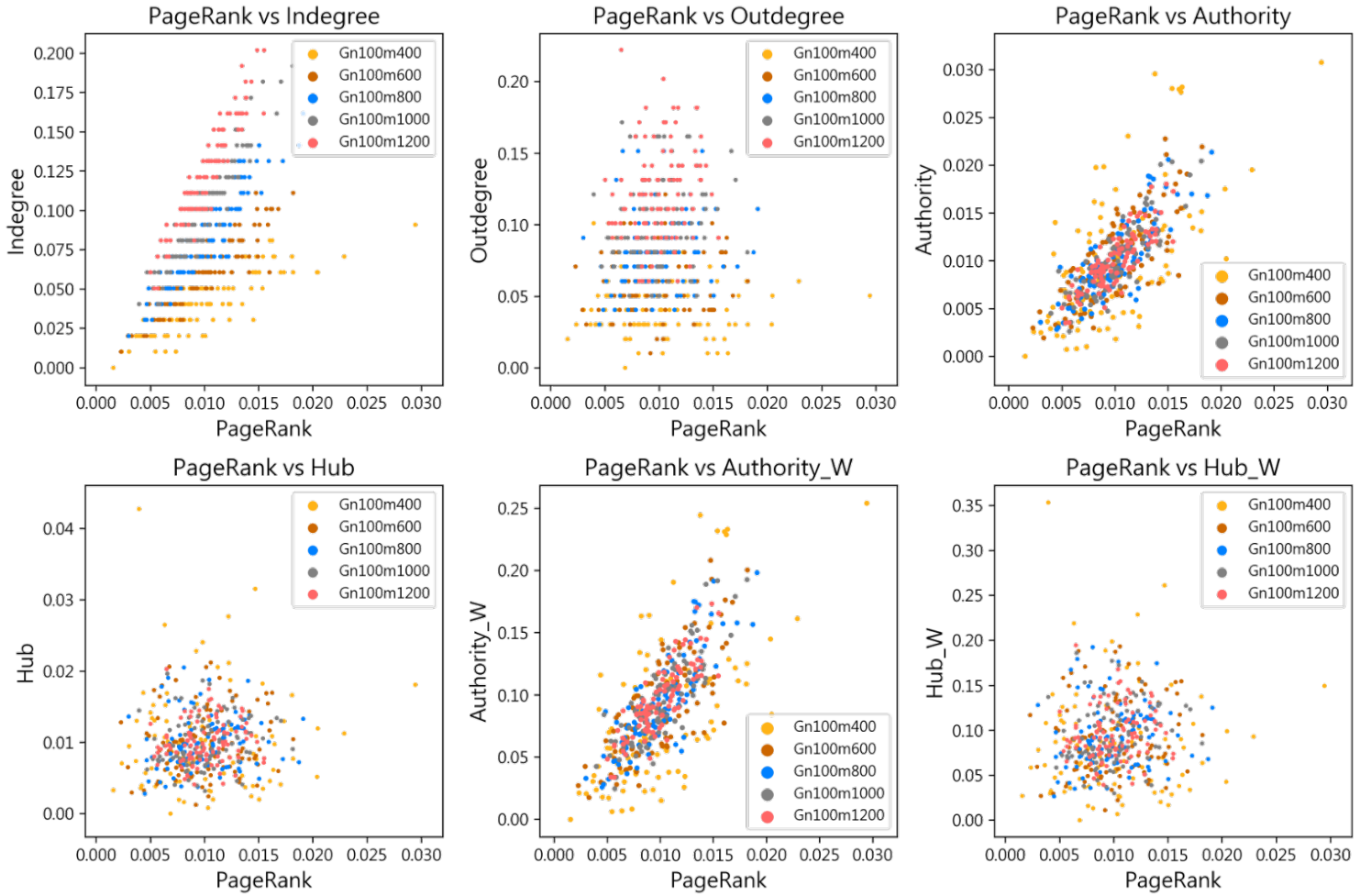
Distribution			
Continuous Data	**Discrete Data**	**Ordered/ Categorical Data**	**Proportional Data**
Scatter Bubble Histogram Violin Plot Box Plot Heatmap Density Map	Scatter Bubble Histogram Violin Plot Box Plot Heatmap Density Map	Group/ Stacked Bar Pie	Group/ Stacked Bar Pie

Gallery

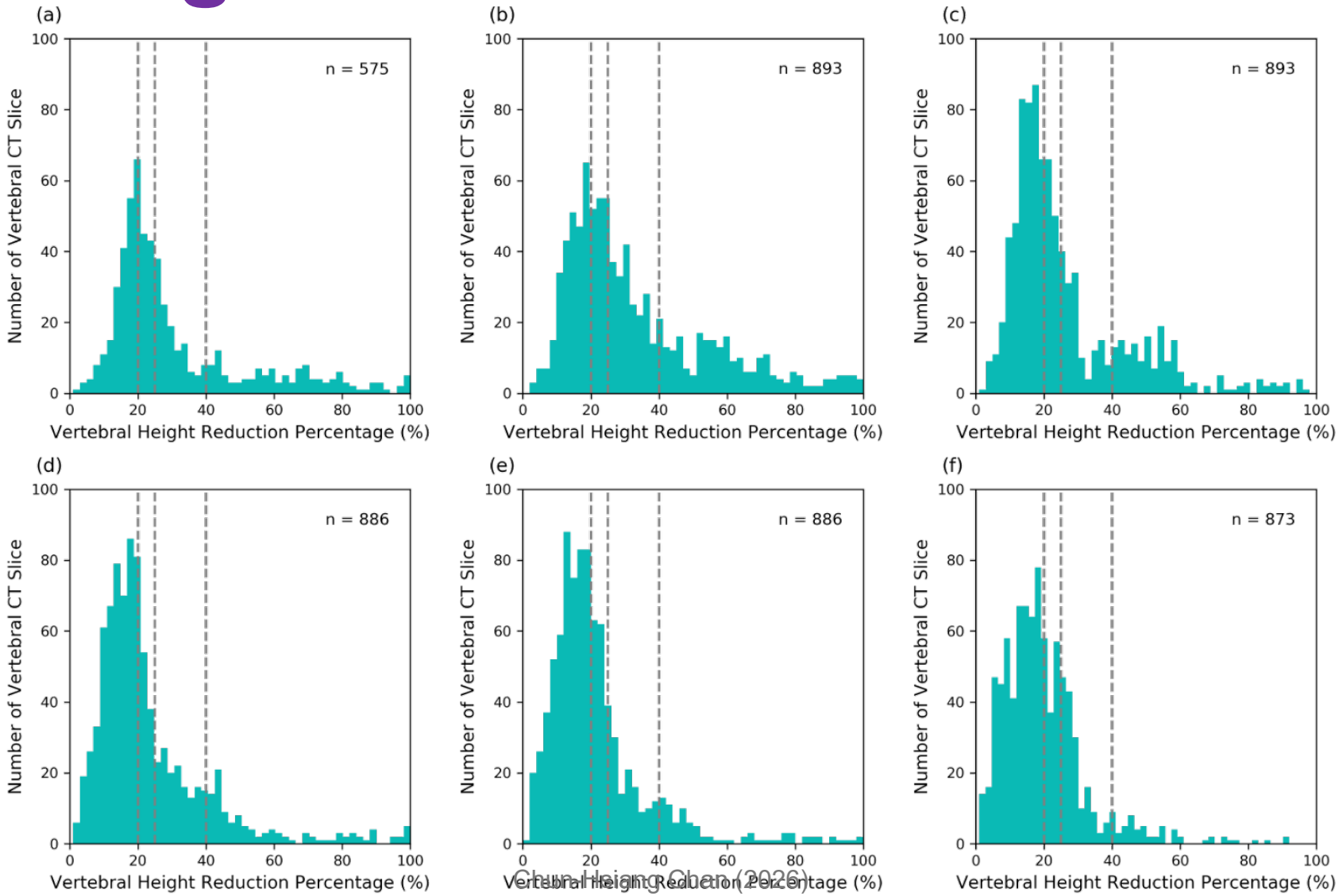
Demonstration Example

Line Plot & Scatter Plot

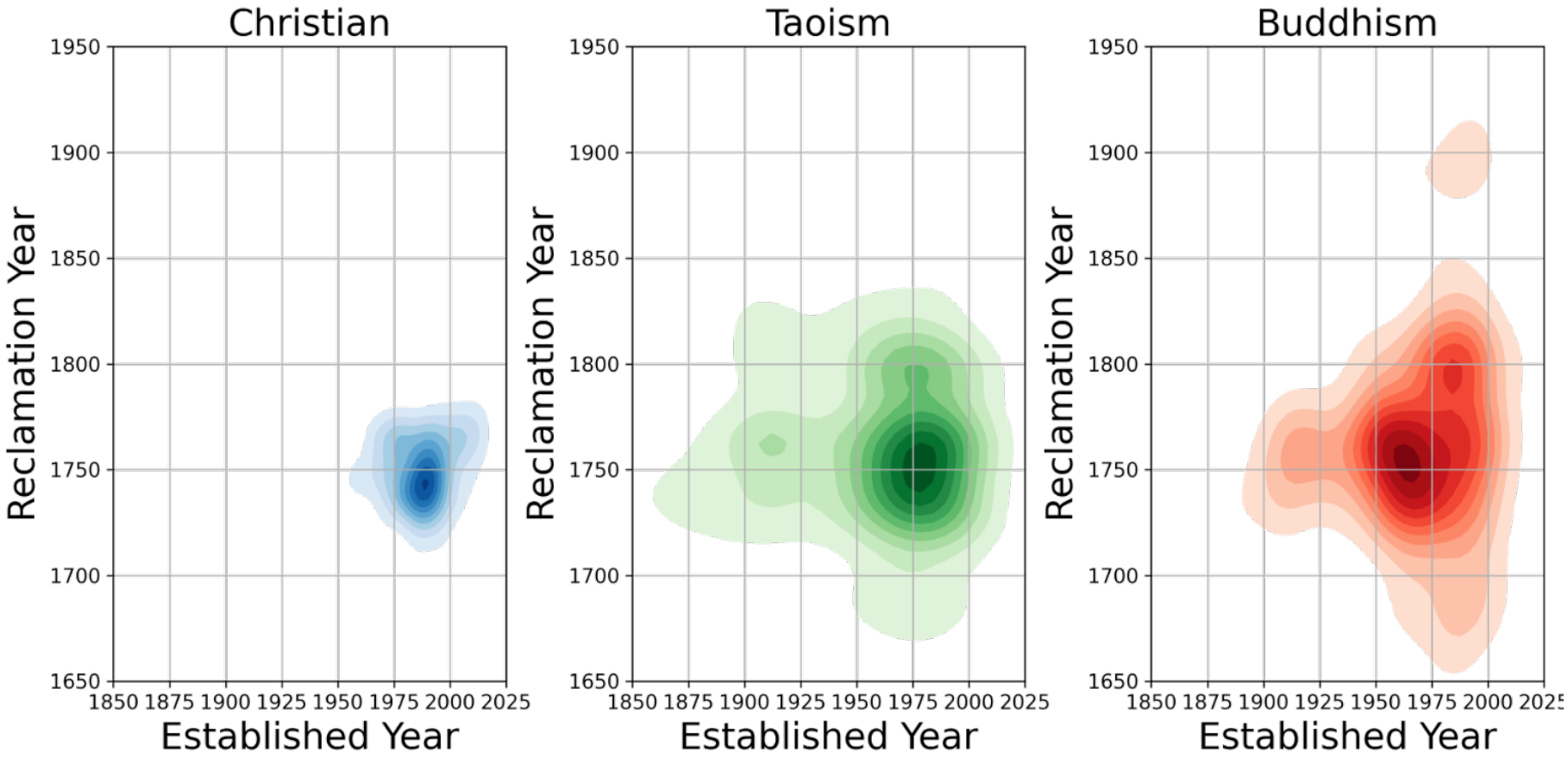




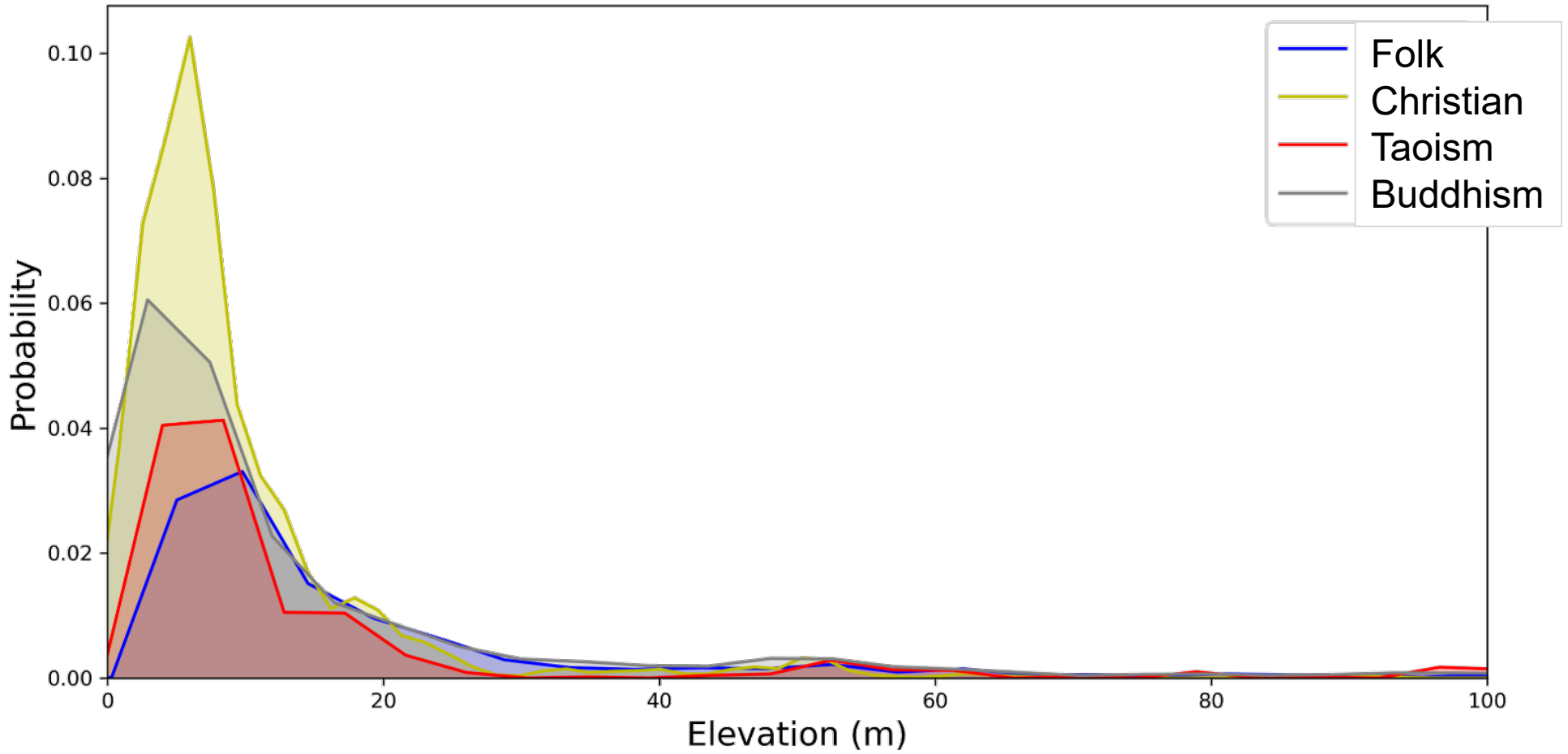
1D Histogram



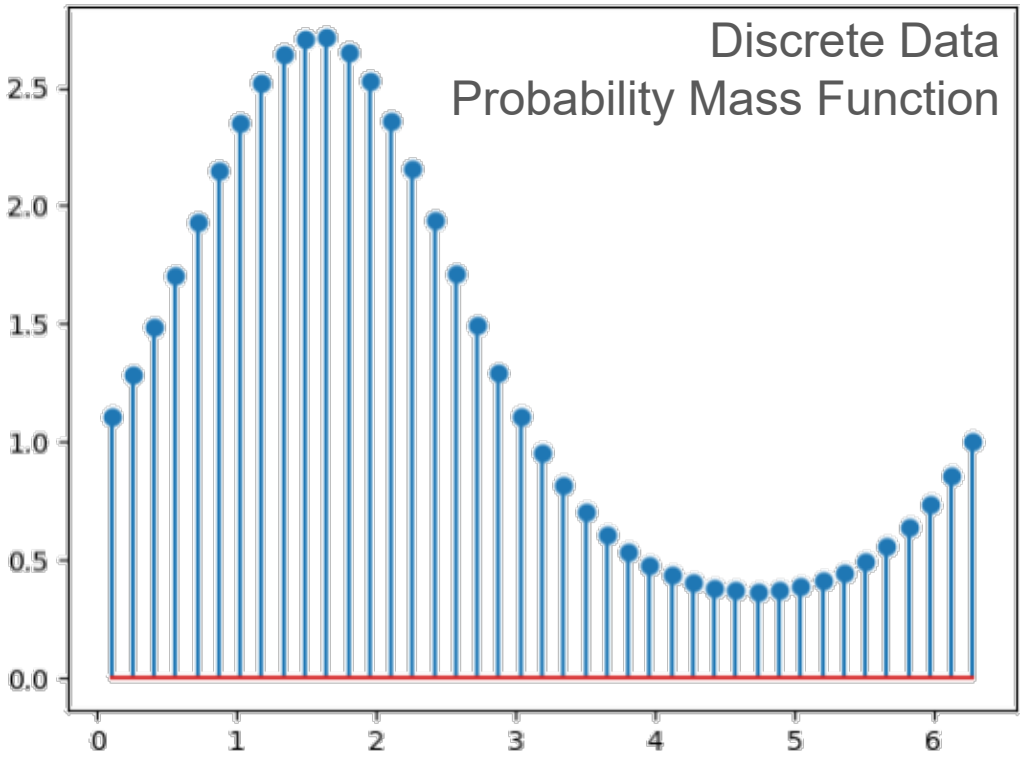
2D Histogram



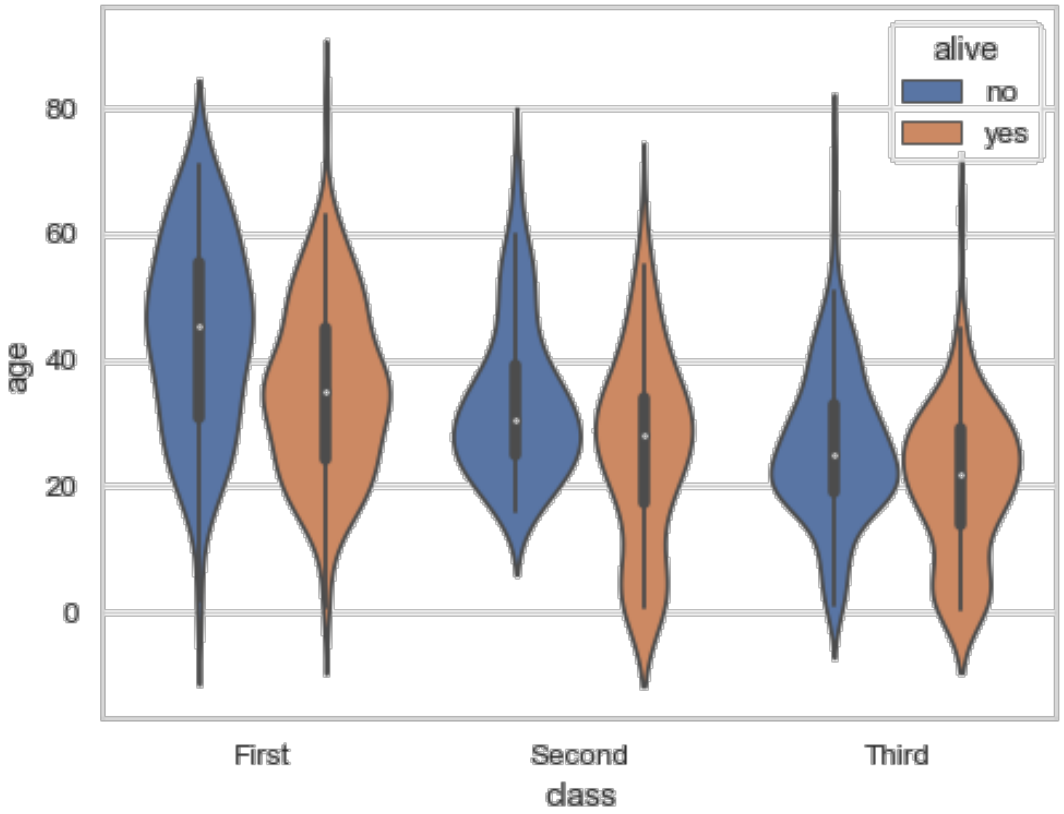
Area Plot


Folk
Christian
Taoism
Buddhism
Probability
0.00
0.02
0.04
0.06
0.08
0.10
0
20
40
60
80
100
Elevation (m)

Stem Plot & Violin Plot



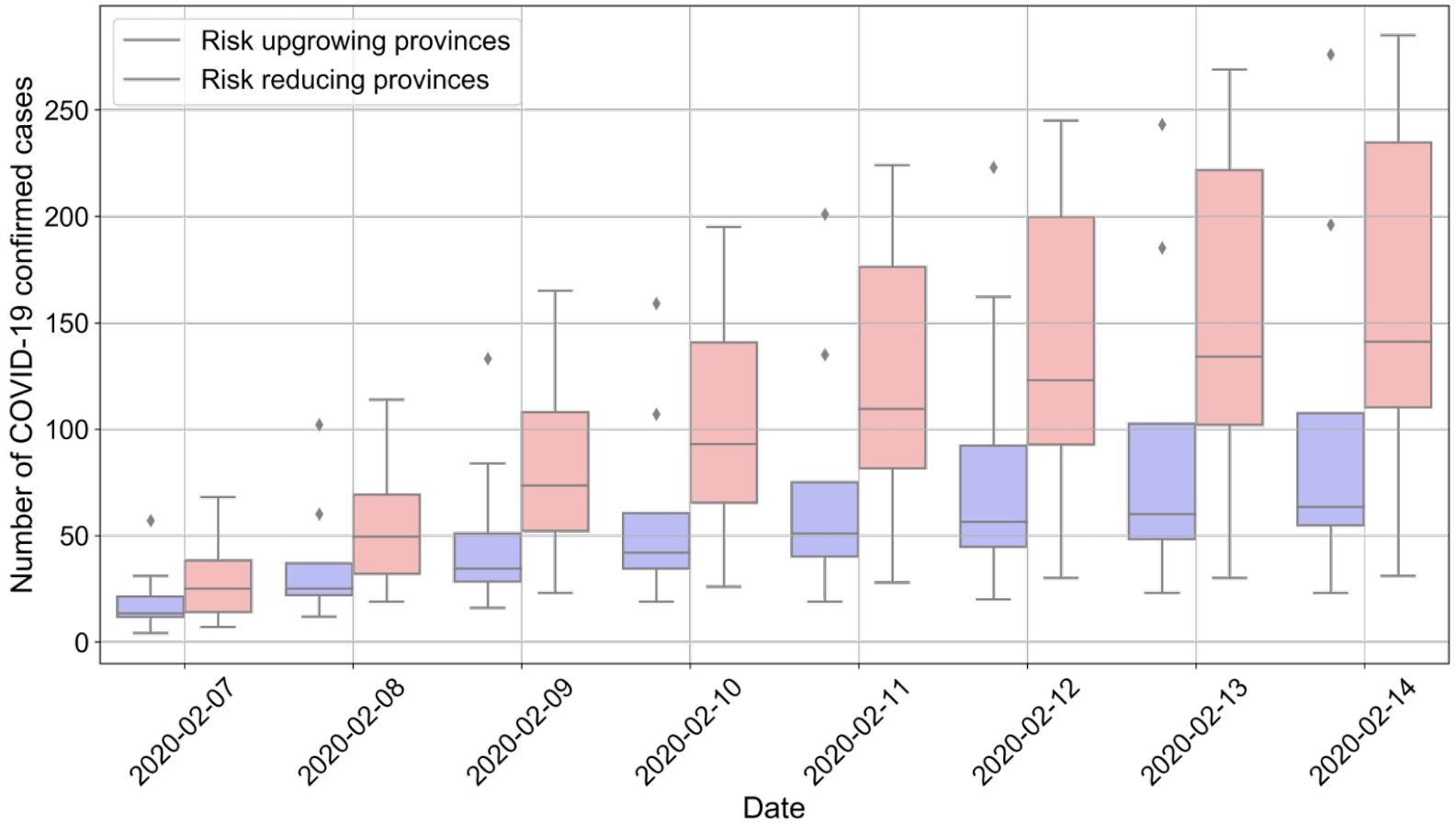
Continuous Data

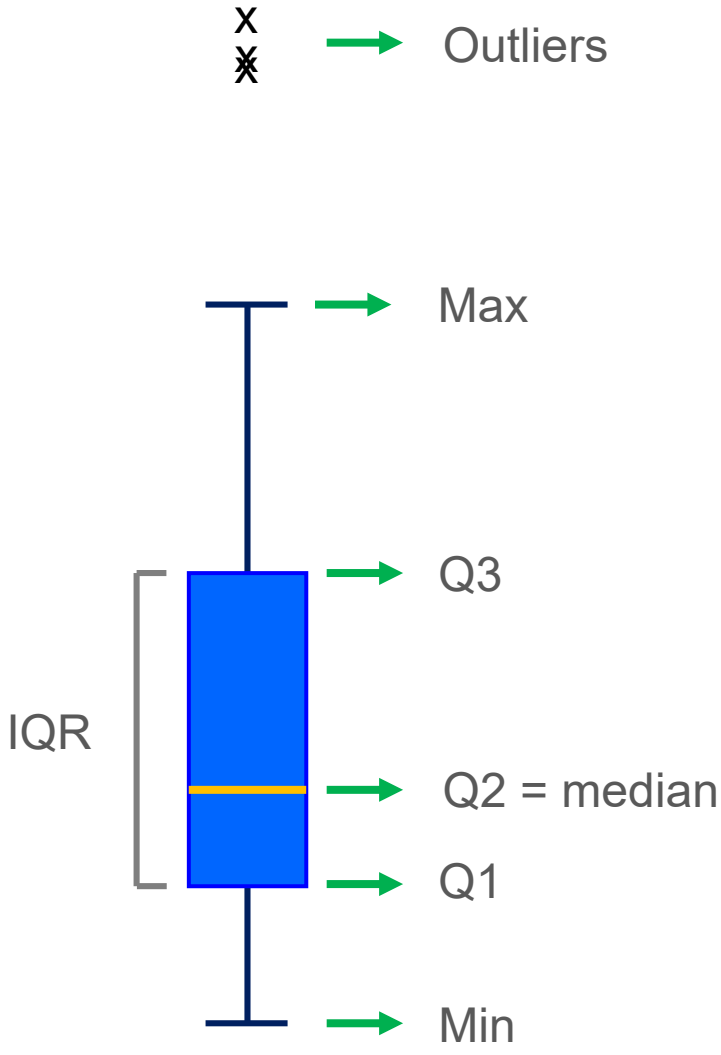


Source: https://matplotlib.org/stable/gallery/lines_bars_and_markers/stem_plot.html#sphx-glr-gallery-lines-bars-and-markers-stem-plot-py

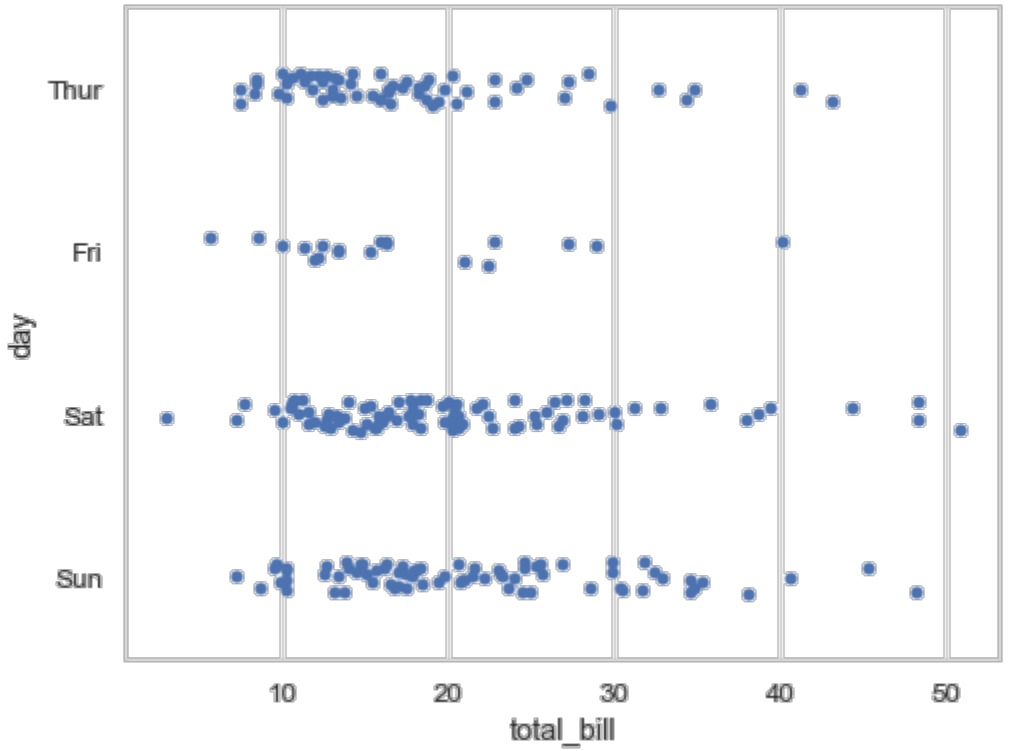
Source: https://seaborn.pydata.org/generated/seaborn.violinplot.html

Box Plot

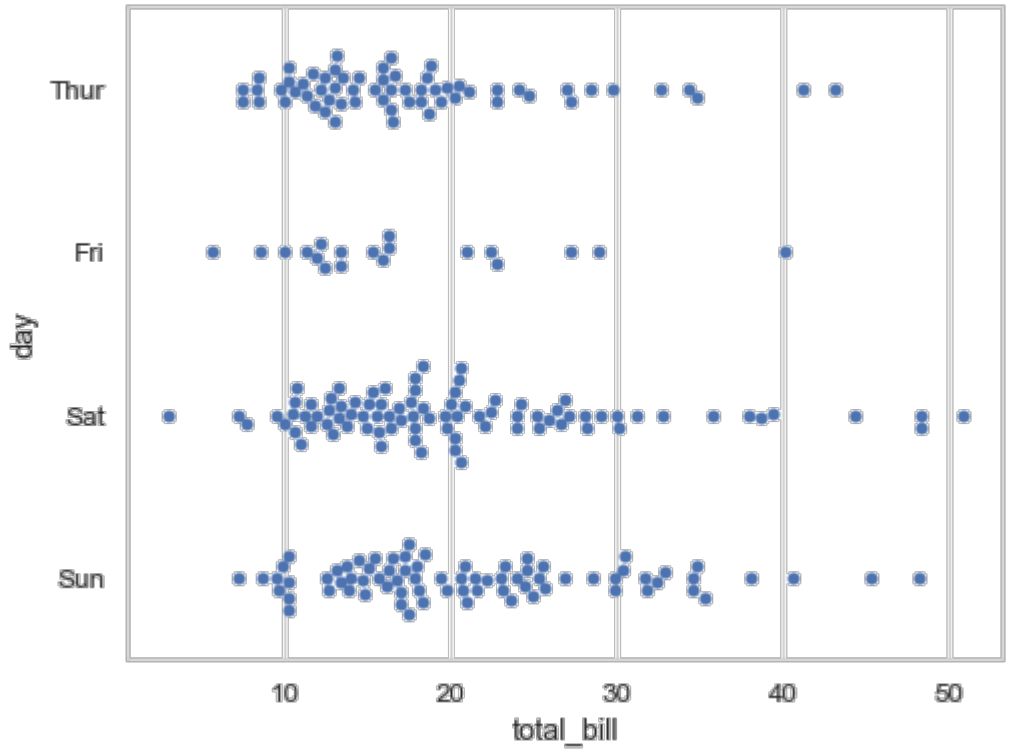

Risk upgrowing provinces
Risk reducing provinces
Number of COVID-19 confirmed cases
0
50
100
150
200
250
2020-02-07
2020-02-08
2020-02-09
2020-02-10
2020-02-11
2020-02-12
2020-02-13
2020-02-14
Date


Outliers
Max
Q3
IQR
Q2 = median
Q1
Min

Strip Plot and Swarm Plot

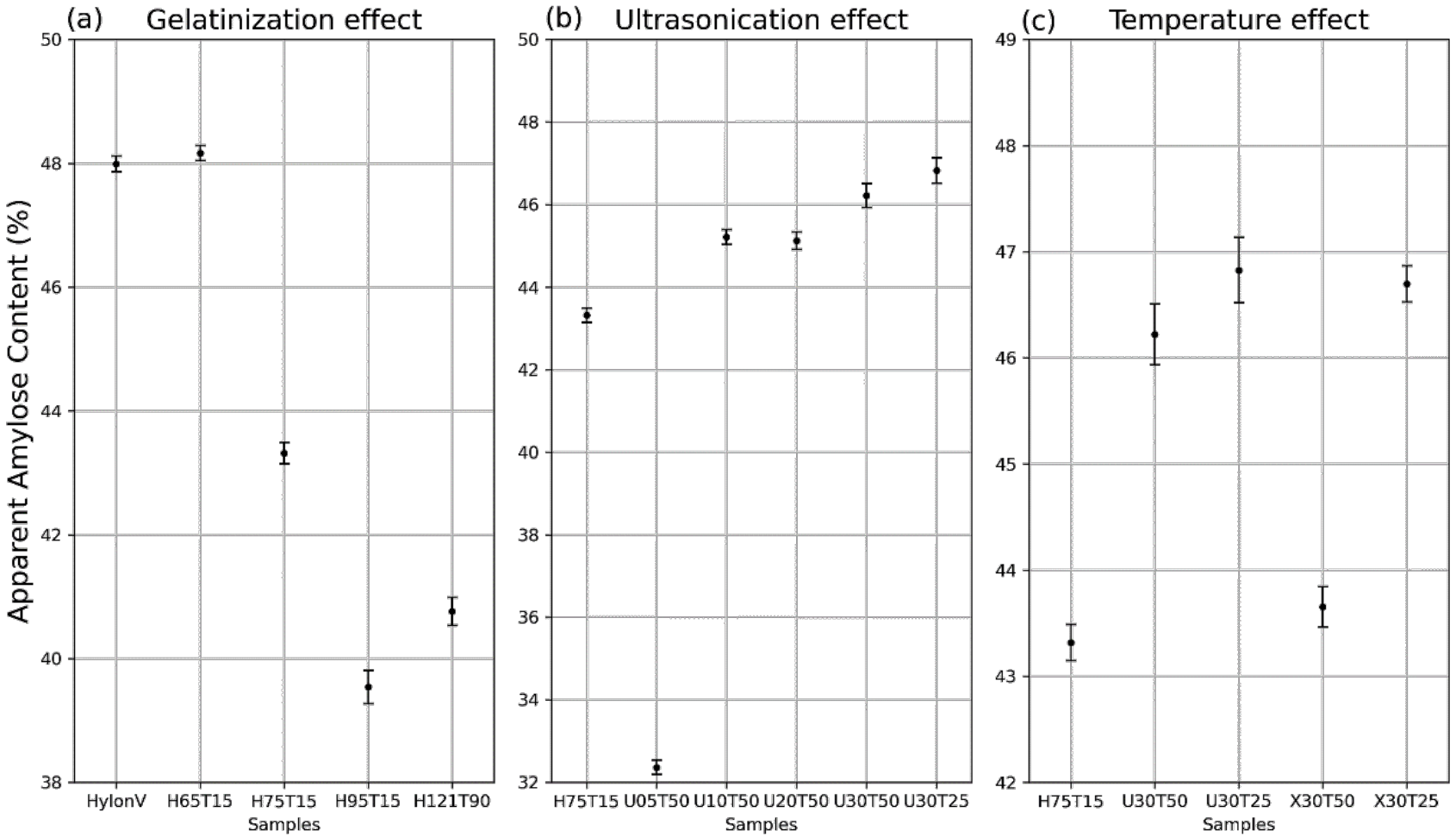


Source: https://seaborn.pydata.org/generated/seaborn.stripplot.html

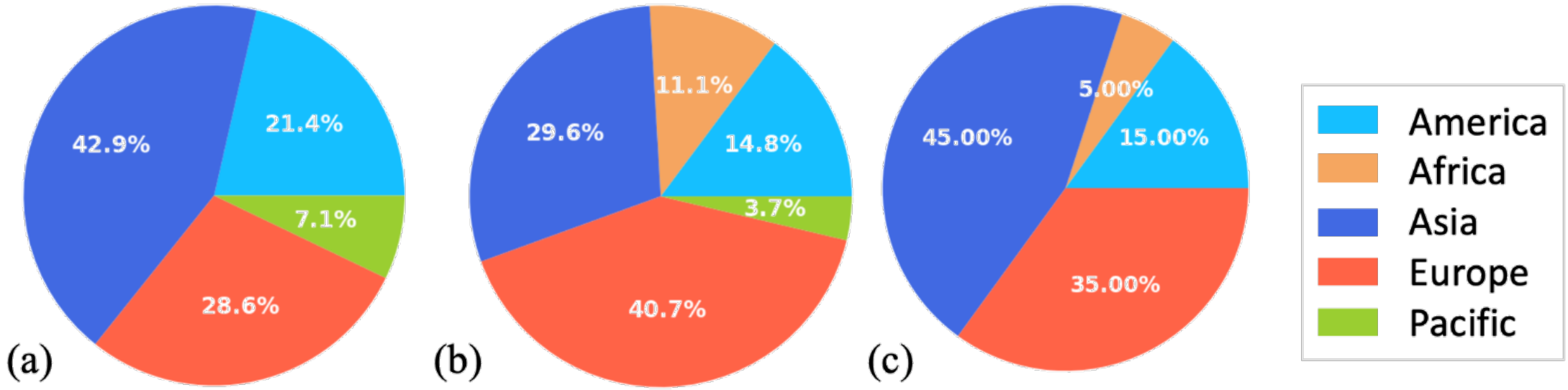


Source: https://seaborn.pydata.org/generated/seaborn.swarmplot.html

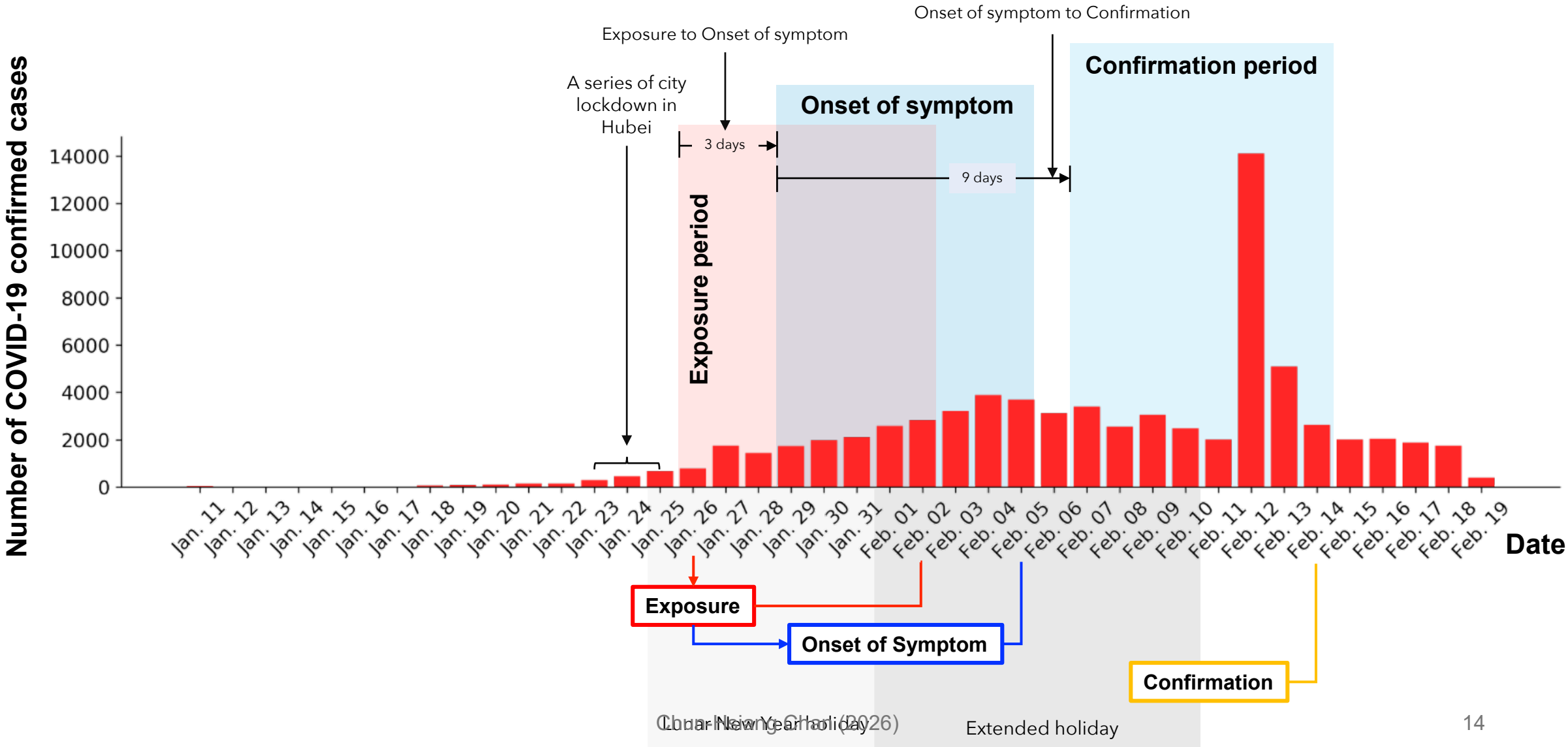
Error Bar


(a) Gelatinization effect
(b) Ultrasonication effect
(c) Temperature effect
Apparent Amylose Content (%)
HylonV H65T15 H75T15 H95T15 H121T90
H75T15 U05T50 U10T50 U20T50 U30T50 U30T25
H75T15 U30T50 U30T25 X30T50 X30T25
Samples

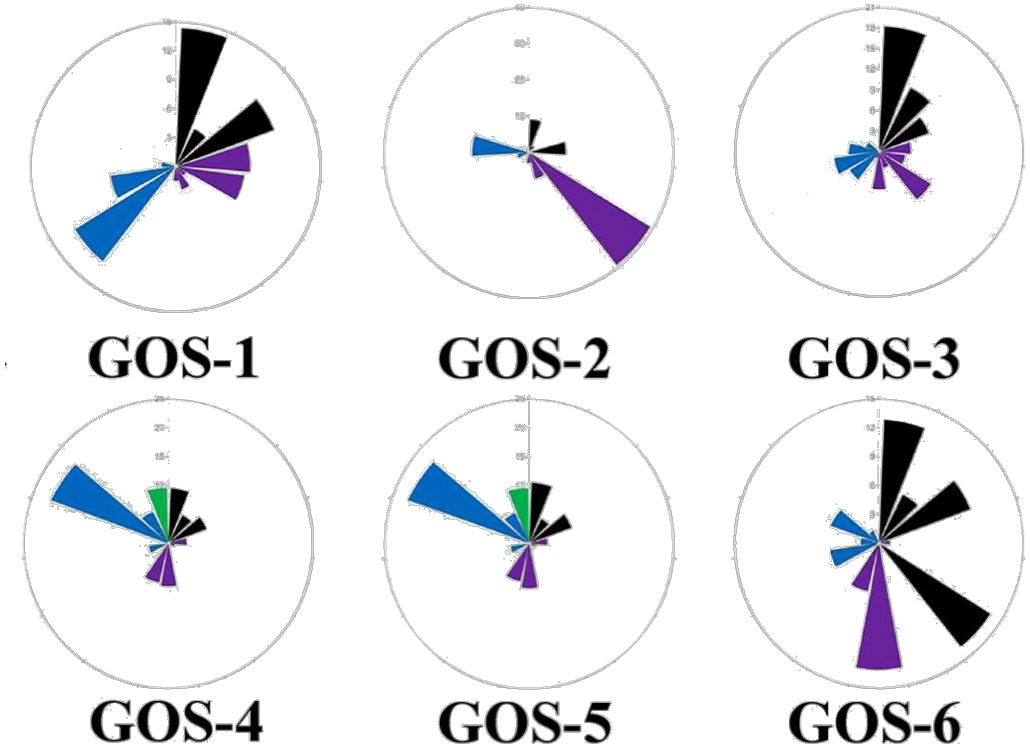
Pie Chart


(a)
21.4%
42.9%
7.1%
28.6%
(b)
11.1%
29.6%
14.8%
3.7%
40.7%
(c)
5.00%
45.00%
15.00%
35.00%
America
Africa
Asia
Europe
Pacific

Bar Chart

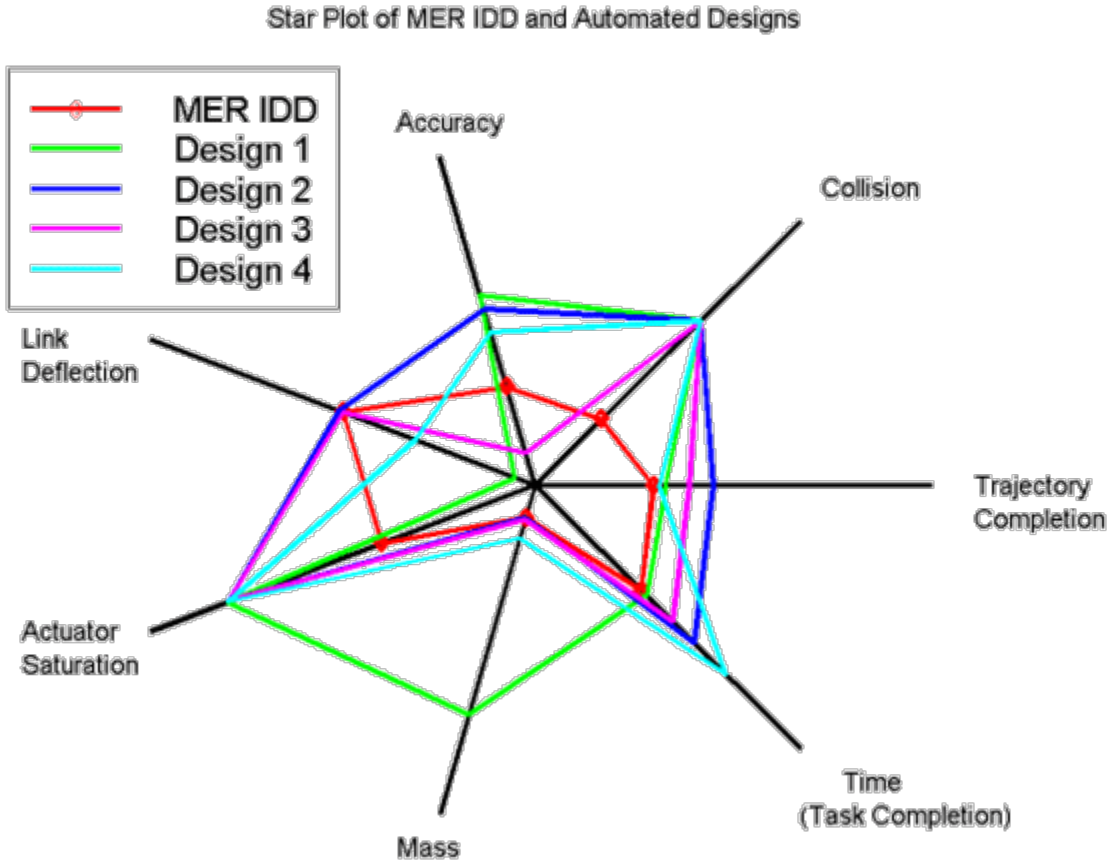

Number of COVID-19 confirmed cases
14000
12000
10000
8000
6000
4000
2000
0
Jan. 11
Jan. 12
Jan. 13
Jan. 14
Jan. 15
Jan. 16
Jan. 17
Jan. 18
Jan. 19
Jan. 20
Jan. 21
Jan. 22
Jan. 23
Jan. 24
Jan. 25
Jan. 26
Jan. 27
Jan. 28
Jan. 29
Jan. 30
Jan. 31
Feb. 01
Feb. 02
Feb. 03
Feb. 04
Feb. 05
Feb. 06
Feb. 07
Feb. 08
Feb. 09
Feb. 10
Feb. 11
Feb. 12
Feb. 13
Feb. 14
Feb. 15
Feb. 16
Feb. 17
Feb. 18
Feb. 19
Date
A series of city lockdown in Hubei
Exposure to Onset of symptom
Onset of symptom to Confirmation
3 days
9 days
Exposure period
Onset of symptom
Confirmation period
Exposure
Onset of Symptom
Confirmation
Lunar New Year holiday
Extended holiday

Rose Plot & Radar Plot



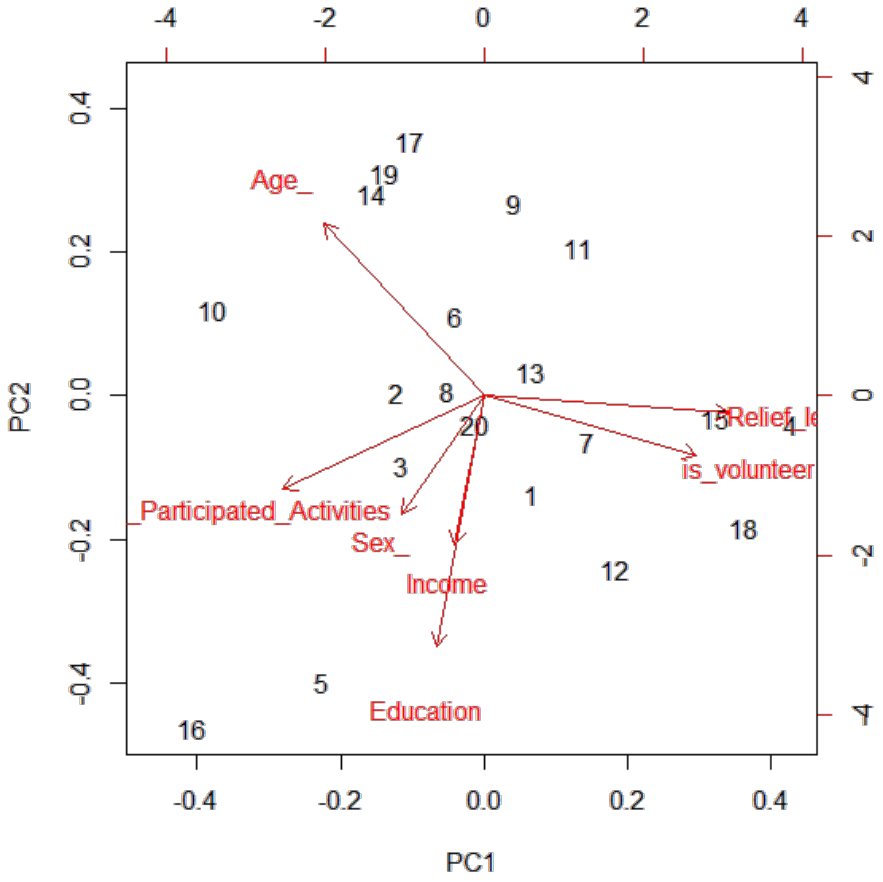
Profile diversity in various GOS samples

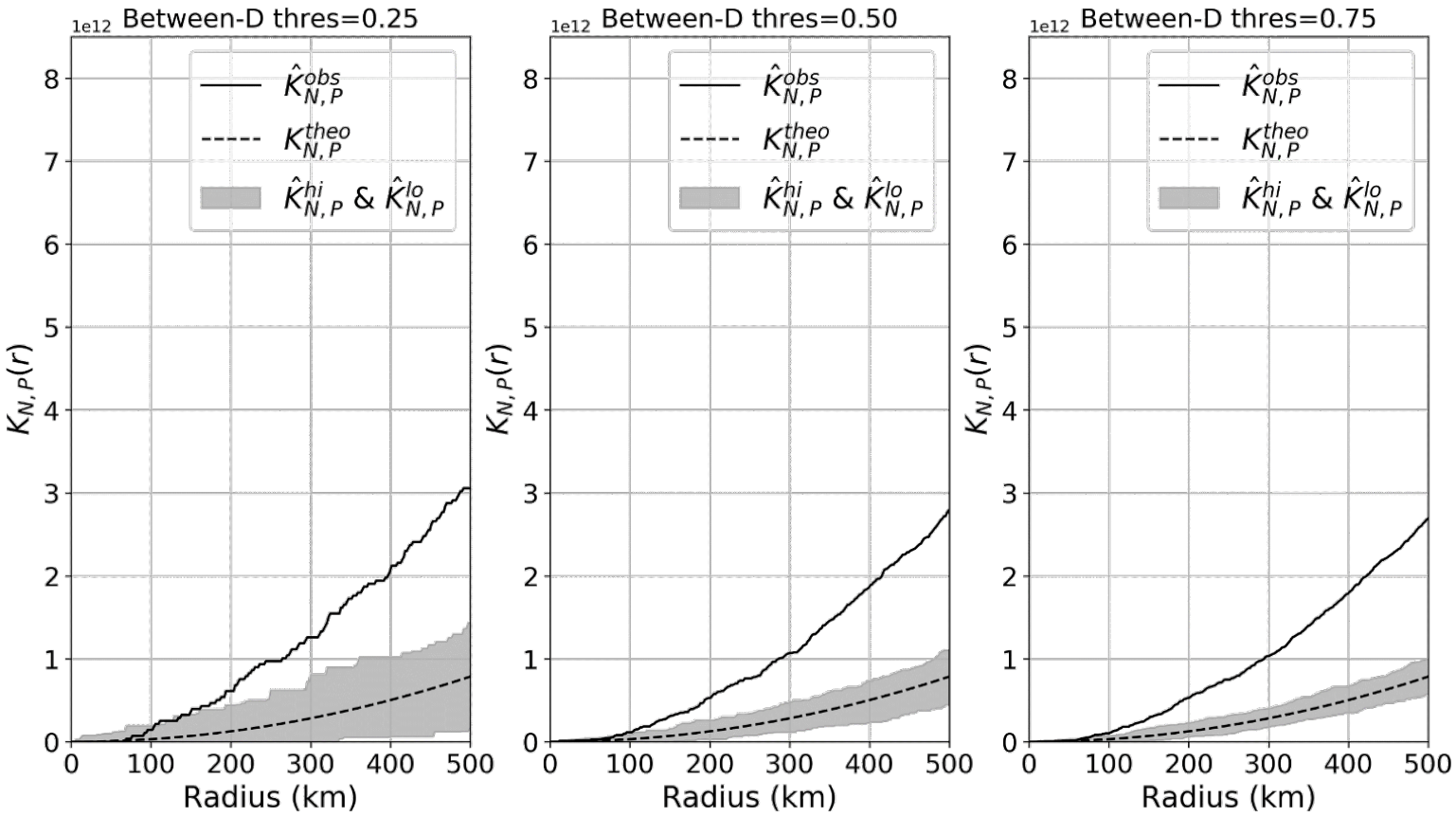
Lin et al. (2022) Profile diversity of galacto-oligosaccharides from disaccharides to hexasaccharides by porous graphitic carbon liquid chromatography-orbitrap tandem mass spectrometry. Food Chem. Vol. 390. 133151.



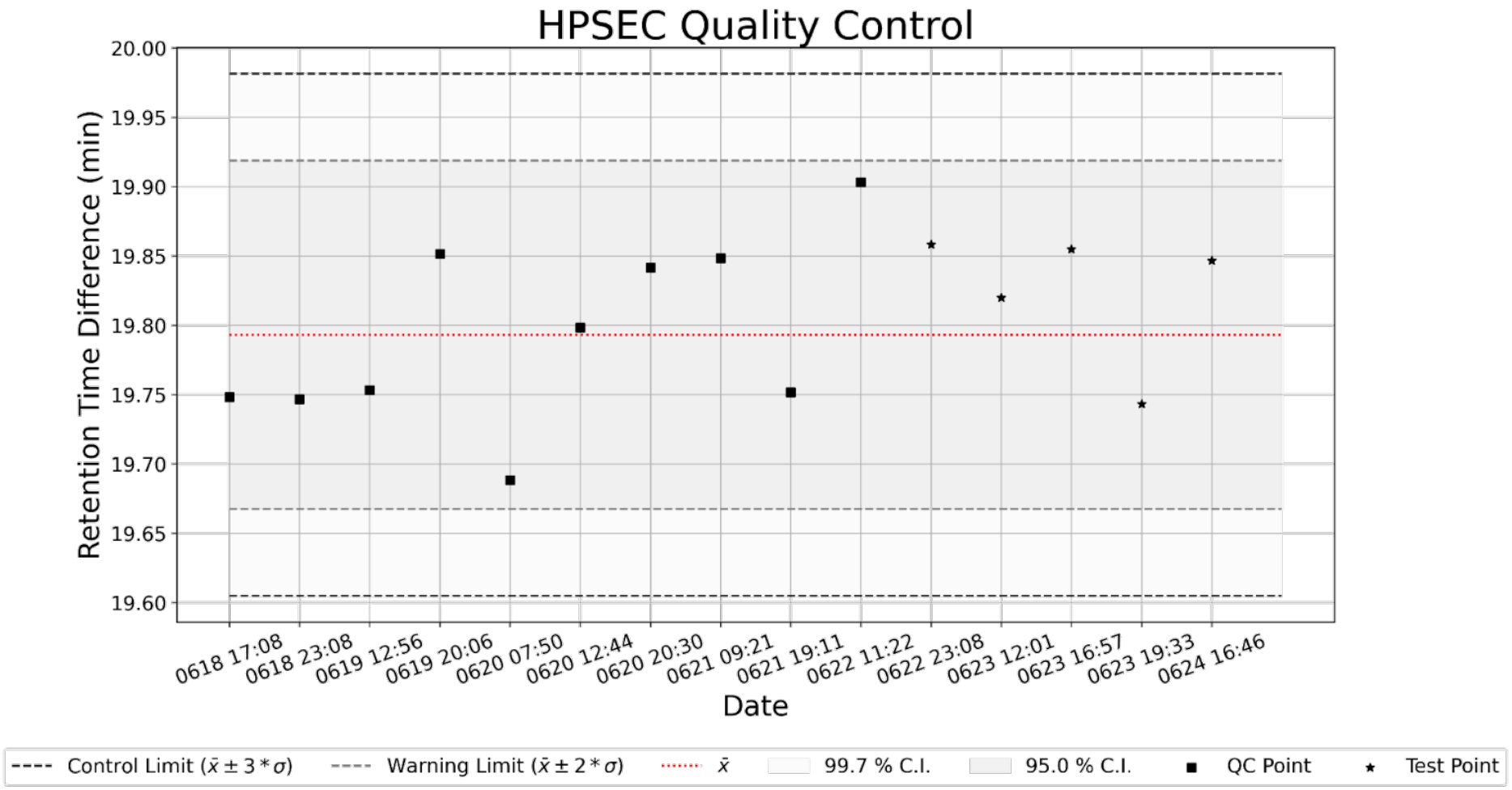
Source: https://en.wikipedia.org/wiki/Radar_chart#/media/File:MER_Star_Plot.gif

Biplot & Control Chart (I)

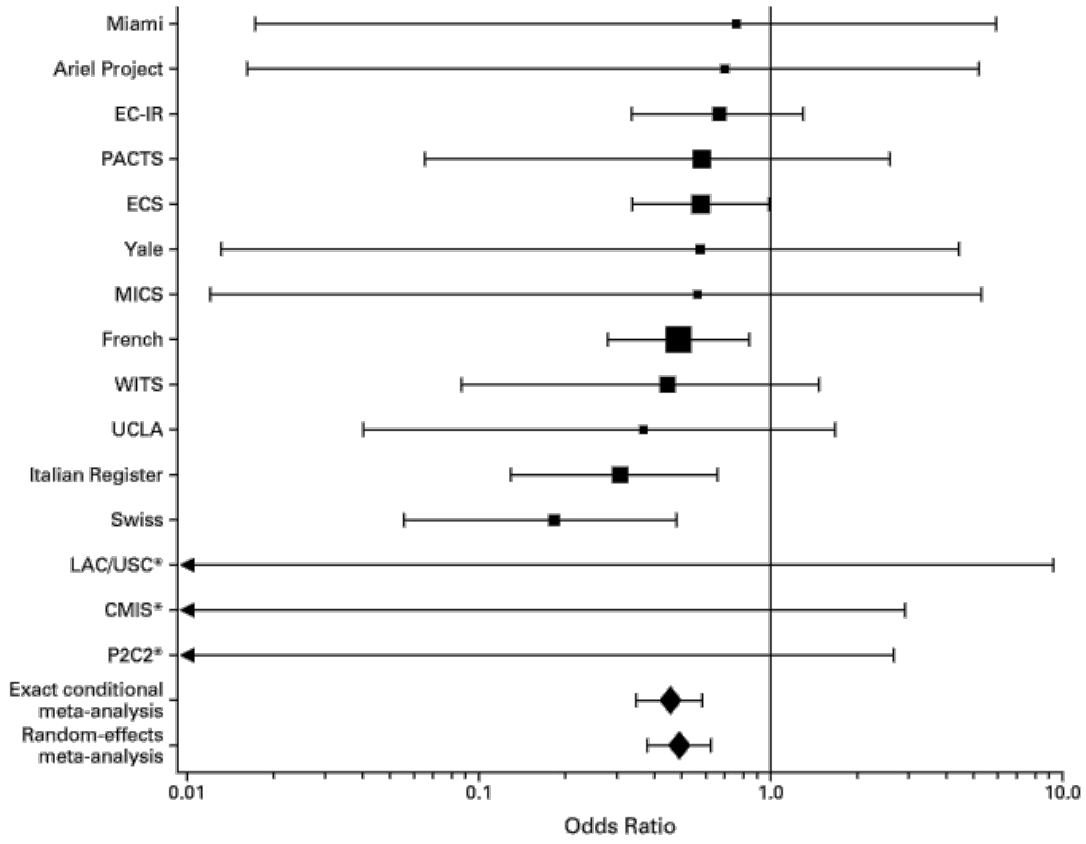




Control Chart (II)



Forest Plot



	Diseased	Healthy
Exposed	20	380
Not Exposed	10	490

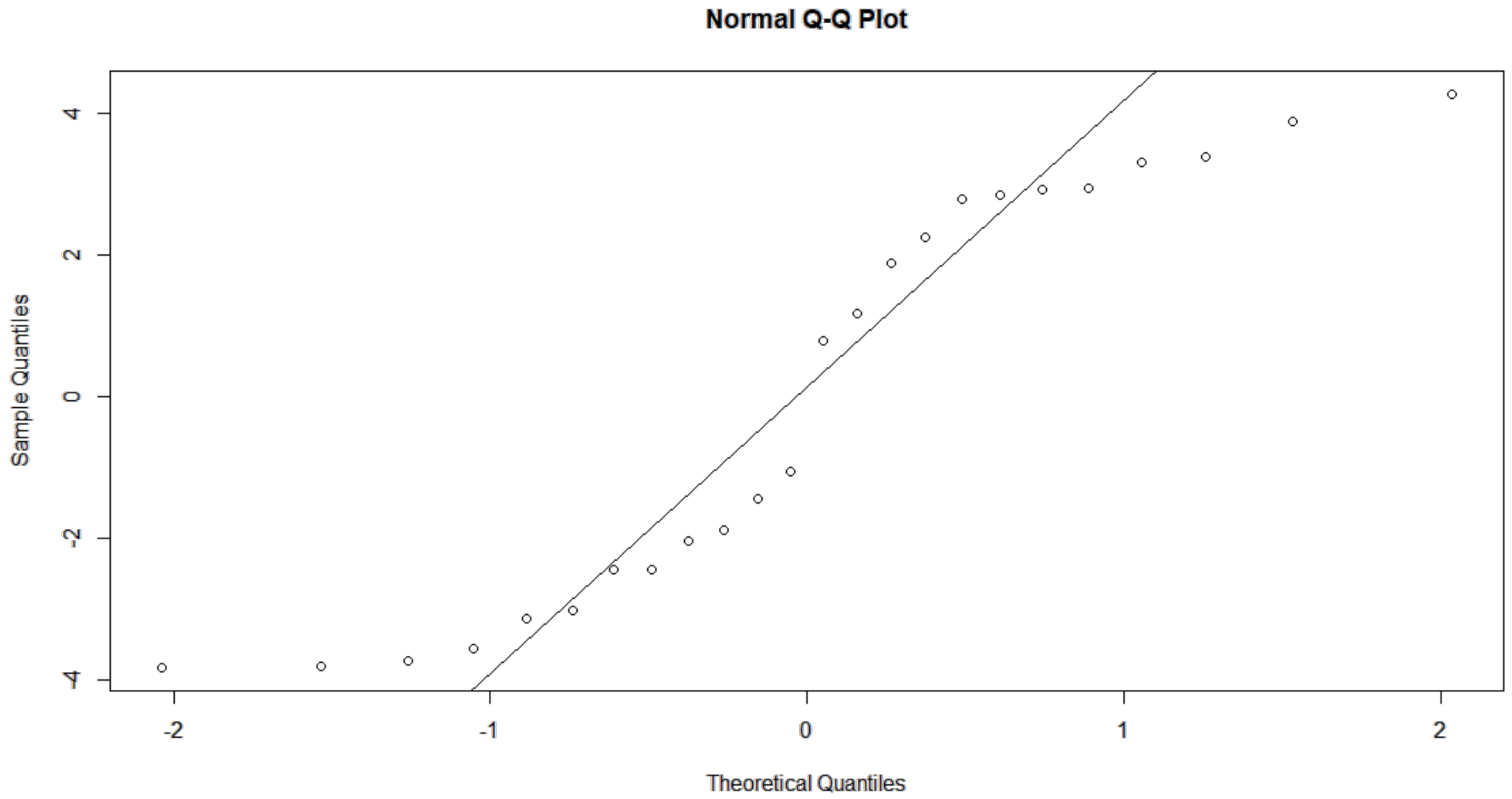
$$risk\ of\ developing\ the\ disease\ given\ exposure = \frac{DE}{VE} = \frac{20}{400}$$

$$risk\ of\ developing\ the\ disease\ given\ non-exposure = \frac{DN}{VN} = \frac{10}{500}$$

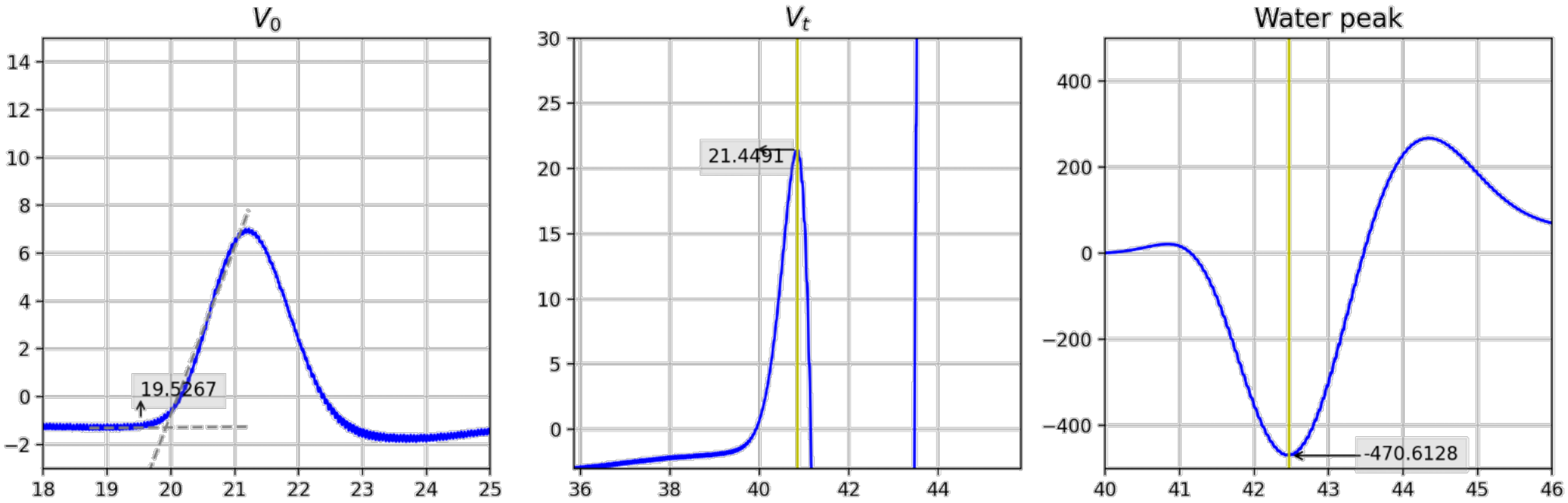
$$relative\ risk = \frac{\frac{DE}{(DE+HE)}}{\frac{DN}{(DN+HN)}} = \frac{DE/VE}{DN/VN} = \frac{20/400}{10/500}$$

$$odds\ ratio = \frac{DE/HE}{DN/HN} = \frac{20/380}{10/500}$$

Quantile-quantile Plot (Q-Q Plot)



Combination Chart



Matplotlib

- Matplotlib is a useful package for visualization; however, we cannot demonstrate all functions in this class. But we still introduce the most useful functions.

```
# import python packages
import matplotlib.pyplot as plt
```

matplotlib.pyplot.scatter

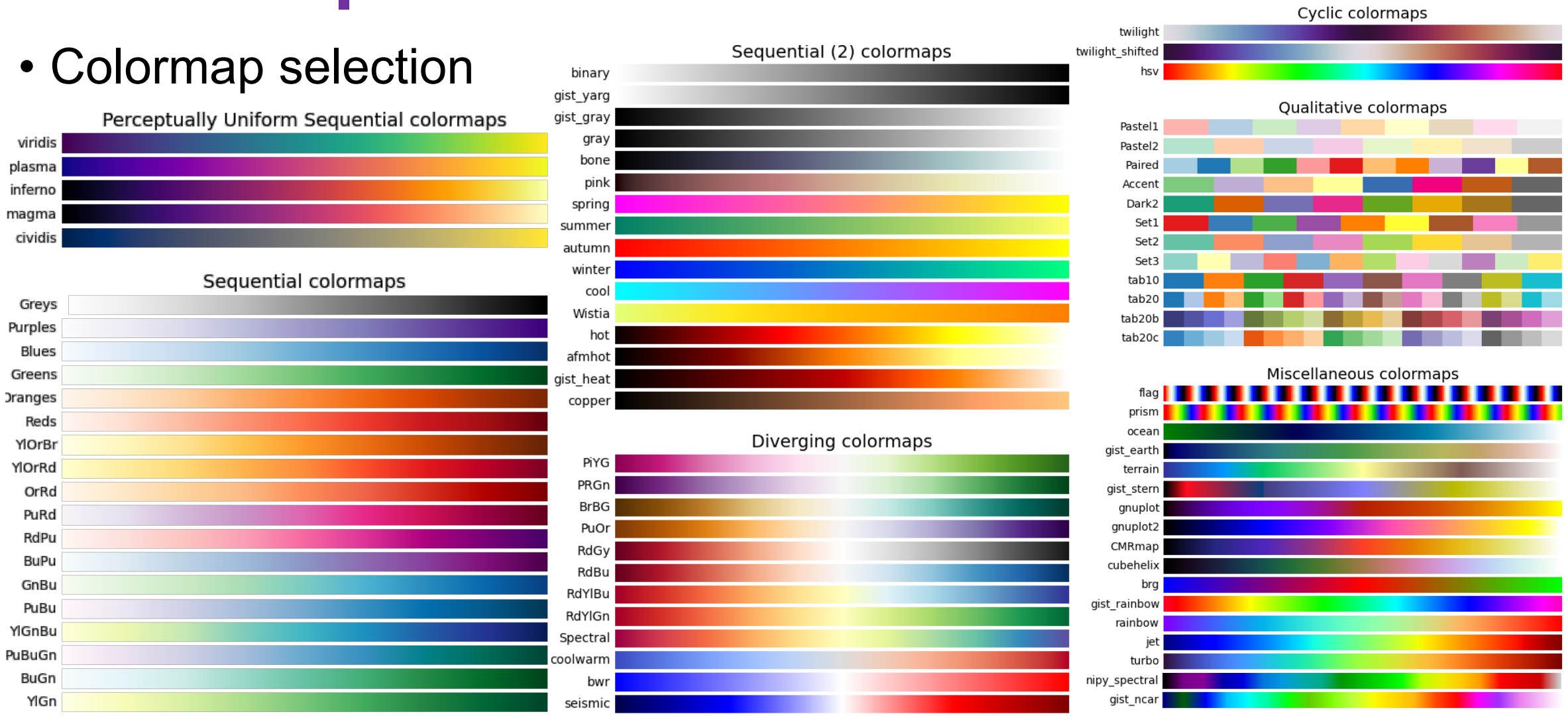
```
matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None,
norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, *,
edgecolors=None, plotnonfinite=False, data=None, **kwargs)        [source]
```

A scatter plot of *y* vs. *x* with varying marker size and/or color.

Colormap

- Colormap selection

Figure source:
https://matplotlib.org/stable/gallery/color/colormap_reference.html#sphx-glr-gallery-color-colormap-reference-py



Colormap

- Colormap selection
  - Do not pick more than 8 colors from graduated colormap
  - Graduated colormaps are for continuous values
  - Discrete colormaps are for categorical values
  - … (**think about it!)**

- To better the understanding of figures…
  - Use different sizes or symbols to represent different data
  - Plot different data into the same subplot/ figure
  - Use subplot with fixed x and y ranges

Import Packages

```
# import packages
import numpy as np                  # numerical analysis
import pandas as pd                 # table data analysis
import matplotlib.pyplot as plt # plotting library
from matplotlib import cm           # color adjustment
import seaborn as sns               # plotting library
```

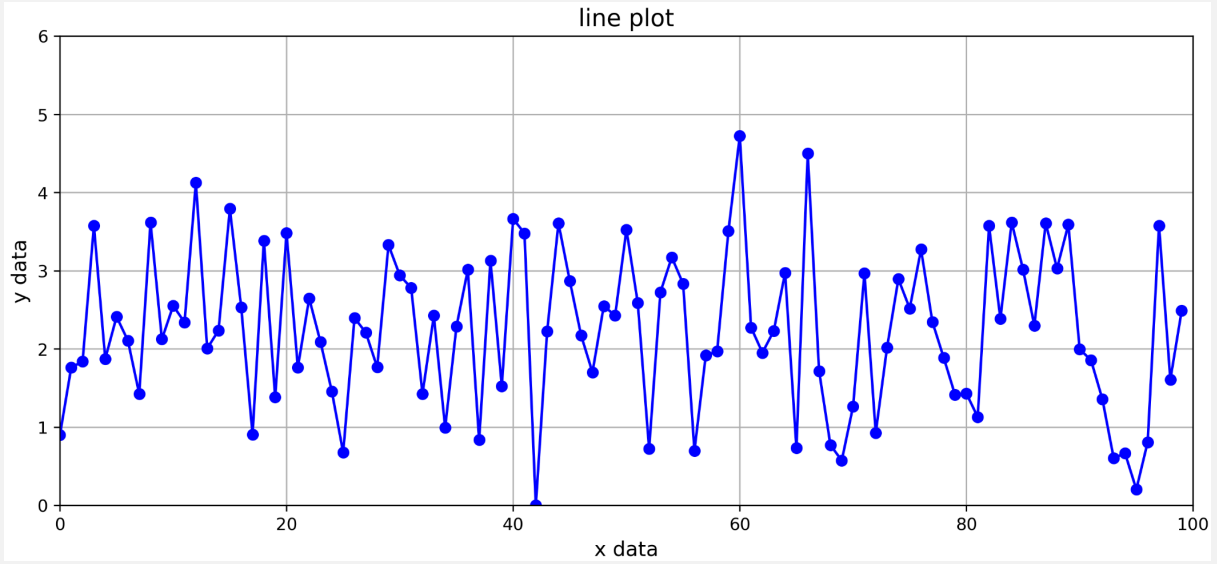
Create Data

```
# generate a normal distributed data
a1 = np.random.randn(100)
a2 = 3*np.random.randn(100)
a3 = np.random.randn(100)*np.sqrt(abs(np.random.randn(100)))
       *np.random.randn(100)
# generate a normal distributed data
a1 += abs(min(a1))
a2 += abs(min(a2))
a3 += abs(min(a3))

xdata = np.arange(100)
```

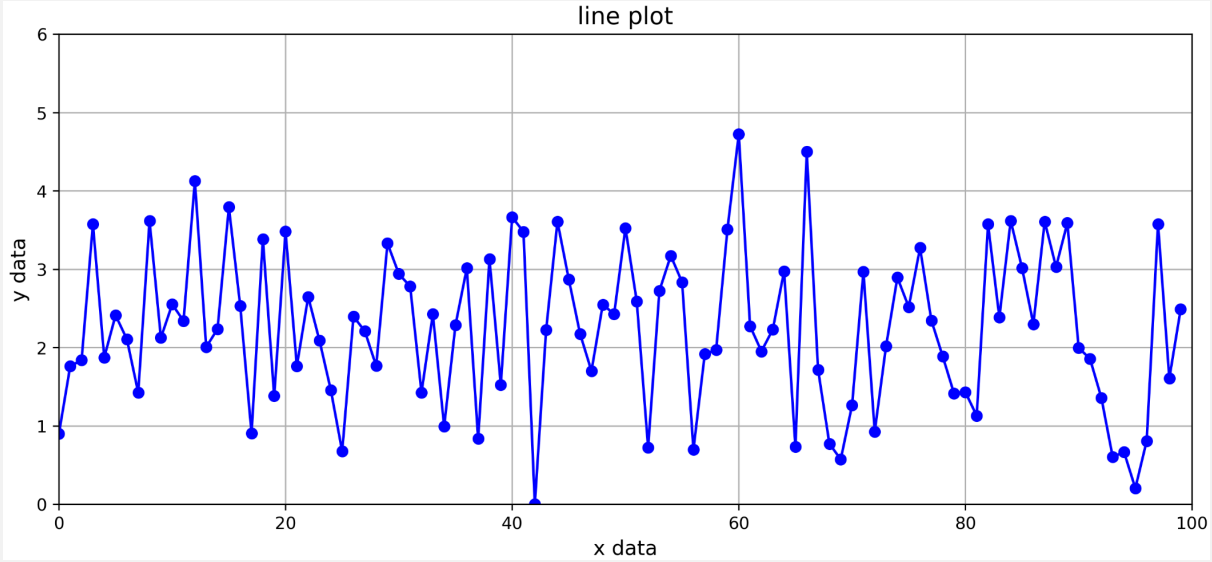
Line Plot

```
# line plot
plt.figure(figsize=[12,5], dpi=300)
plt.plot(xdata, a1, 'b-o')
plt.title('line plot', fontsize=14)
plt.xlabel('x data', fontsize=12)
plt.ylabel('y data', fontsize=12)
plt.xlim([0, 100])
plt.ylim([0, 6])
plt.grid(True)
plt.show()
```



Export Figure

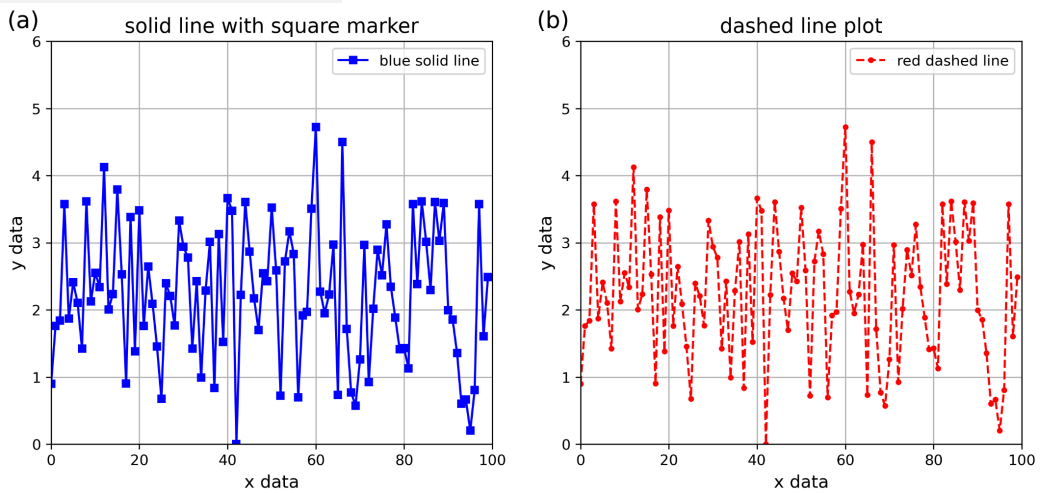
```
# line plot
plt.figure(figsize=[12,5], dpi=300)
plt.plot(xdata, a1, 'b-o')
plt.title('line plot', fontsize=14)
plt.xlabel('x data', fontsize=12)
plt.ylabel('y data', fontsize=12)
plt.xlim([0, 100])
plt.ylim([0, 6])
plt.grid(True)
plt.savefig('line plot.png', dpi=300, bbox_inches=0, pad_inches=0)
plt.show()
```



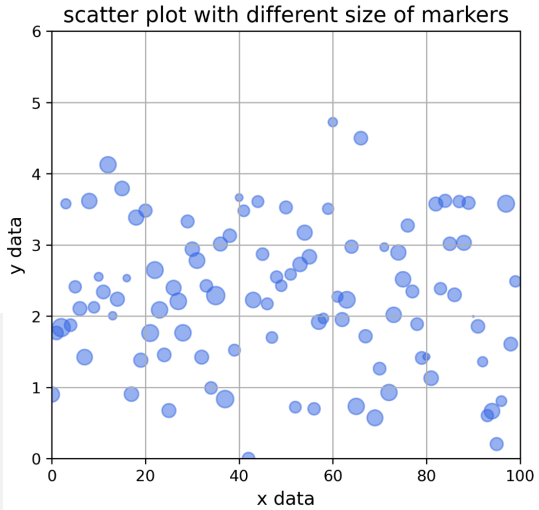
Subplot & Style Adjustment

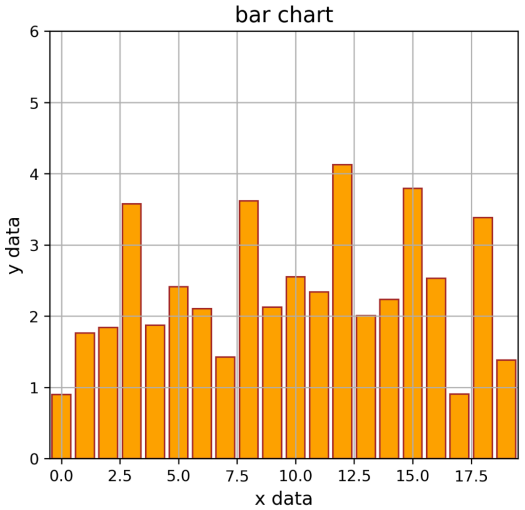
```
# subplot plot and style adjustment
fig, axe = plt.subplots(1, 2, figsize=[12,5], dpi=300)
axe[0].plot(xdata, a1, 'b-s', markersize=5, label='blue solid line')
axe[0].set_title('solid line with square marker', fontsize=14)
axe[0].set_xlabel('x data', fontsize=12)
axe[0].set_ylabel('y data', fontsize=12)
axe[0].axis([0, 100, 0, 6])
axe[0].text(-10, 6.2, '(a)', fontsize=16)
axe[0].legend()
axe[0].grid(True)

axe[1].plot(xdata, a1, 'r.--', label='red dashed line')
axe[1].set_title('dashed line plot', fontsize=14)
axe[1].set_xlabel('x data', fontsize=12)
axe[1].set_ylabel('y data', fontsize=12)
axe[1].axis([0, 100, 0, 6])
axe[1].text(-10, 6.2, '(b)', fontsize=16)
axe[1].legend()
axe[1].grid(True)
plt.show()
```



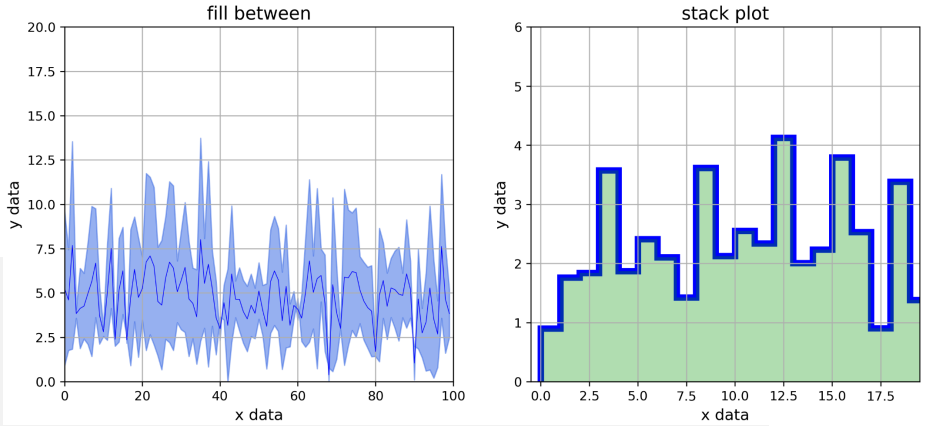
Scatter & Bar Chart






```
# scatter plot and bar chart
fig, axe = plt.subplots(1, 2, figsize=[12,5], dpi=300)
axe[0].scatter(xdata, a1, a2*10, color=(0.2,0.4,0.9,0.5))
axe[0].set_title('scatter plot with different size of markers', fontsize=14)
axe[0].set_xlabel('x data', fontsize=12)
axe[0].set_ylabel('y data', fontsize=12)
axe[0].axis([0, 100, 0, 6])
axe[0].grid(True)
axe[1].bar(xdata[:20], a1[:20], facecolor='orange', edgecolor='brown')
axe[1].set_title('bar chart', fontsize=14)
axe[1].set_xlabel('x data', fontsize=12)
axe[1].set_ylabel('y data', fontsize=12)
axe[1].axis([-.5, 19.5, 0, 6])
axe[1].grid(True)
plt.show()
```

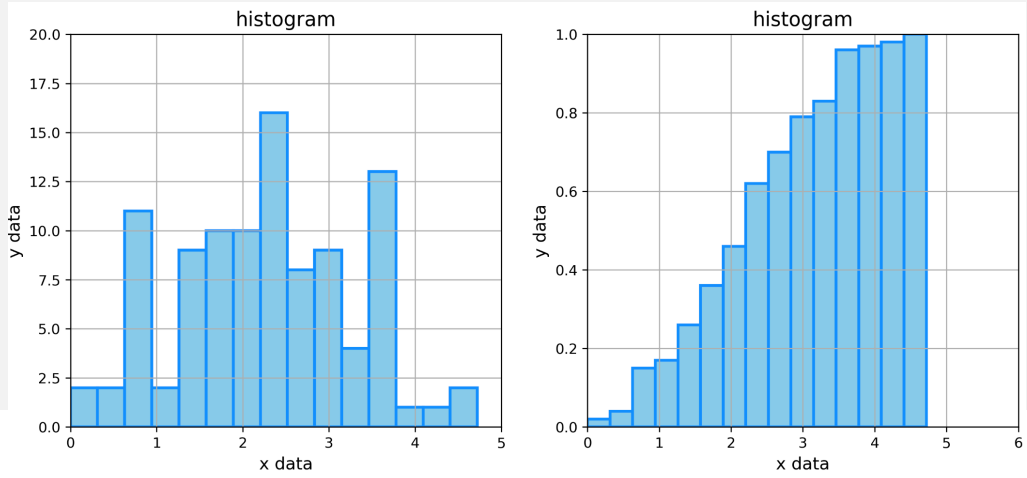
Fill Between & Stack Plot




```
# fill_between plot and stackplot
fig, axe = plt.subplots(1, 2, figsize=[12,5], dpi=300)
axe[0].fill_between(xdata, a1, a2, color=(0.2,0.4,0.9,0.5))
axe[0].plot(xdata, np.mean([a1, a2], axis=0), color='b', linewidth=0.5)
axe[0].set_title('fill between', fontsize=14)
axe[0].set_xlabel('x data', fontsize=12)
axe[0].set_ylabel('y data', fontsize=12)
axe[0].axis([0, 100, 0, 20])
axe[0].grid(True)
axe[1].stairs(a1[:20], orientation='vertical', fill=False, color='b', linewidth=5)
axe[1].stairs(a1[:20], orientation='vertical', fill=True, color=(0,0.6,0,0.3))
axe[1].set_title('stack plot', fontsize=14)
axe[1].set_xlabel('x data', fontsize=12)
axe[1].set_ylabel('y data', fontsize=12)
axe[1].axis([-.5, 19.5, 0, 6])
axe[1].grid(True)
plt.show()
```

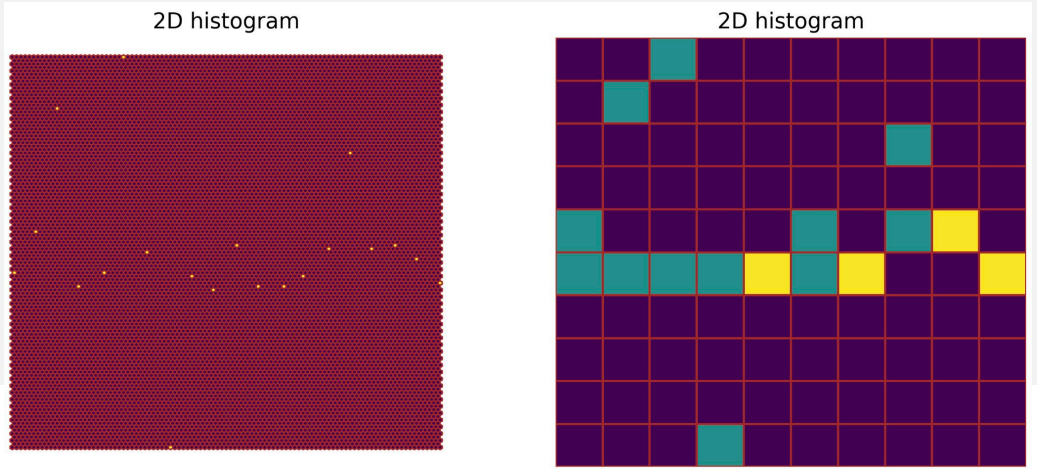
Histogram & Accumulated Histogram

```
# histogram and accumulated histogram
fig, axe = plt.subplots(1, 2, figsize=[12,5], dpi=300)
axe[0].hist(a1, bins=15, color='skyblue', edgecolor='dodgerblue', linewidth=2)
axe[0].set_title('histogram', fontsize=14)
axe[0].set_xlabel('x data', fontsize=12)
axe[0].set_ylabel('y data', fontsize=12)
axe[0].axis([0, 5, 0, 20])
axe[0].grid(True)
axe[1].hist(a1, bins=15, color='skyblue', cumulative=True, density=True,
            edgecolor='dodgerblue', linewidth=2)
axe[1].set_title('histogram', fontsize=14)
axe[1].set_xlabel('x data', fontsize=12)
axe[1].set_ylabel('y data', fontsize=12)
axe[1].axis([0, 6, 0, 1])
axe[1].grid(True)
plt.show()
```



Hexbin & Hist2D

```
# hexbin and hist2D
fig, axe = plt.subplots(1, 2, figsize=[12,5], dpi=300)
axe[0].hexbin(xdata[:20], a3[:20], facecolor='orange', edgecolor='brown')
axe[0].set_title('2D histogram', fontsize=14)
axe[0].set_xlabel('x data', fontsize=12)
axe[0].set_ylabel('y data', fontsize=12)
axe[0].axis('off')
axe[1].hist2d(xdata[:20], a3[:20], facecolor='orange', edgecolor='brown')
axe[1].set_title('2D histogram', fontsize=14)
axe[1].set_xlabel('x data', fontsize=12)
axe[1].set_ylabel('y data', fontsize=12)
axe[1].axis('off')
plt.show()
```



Add Extreme Values

We add some extreme values to demonstrate the utility of distribution or pattern charts.

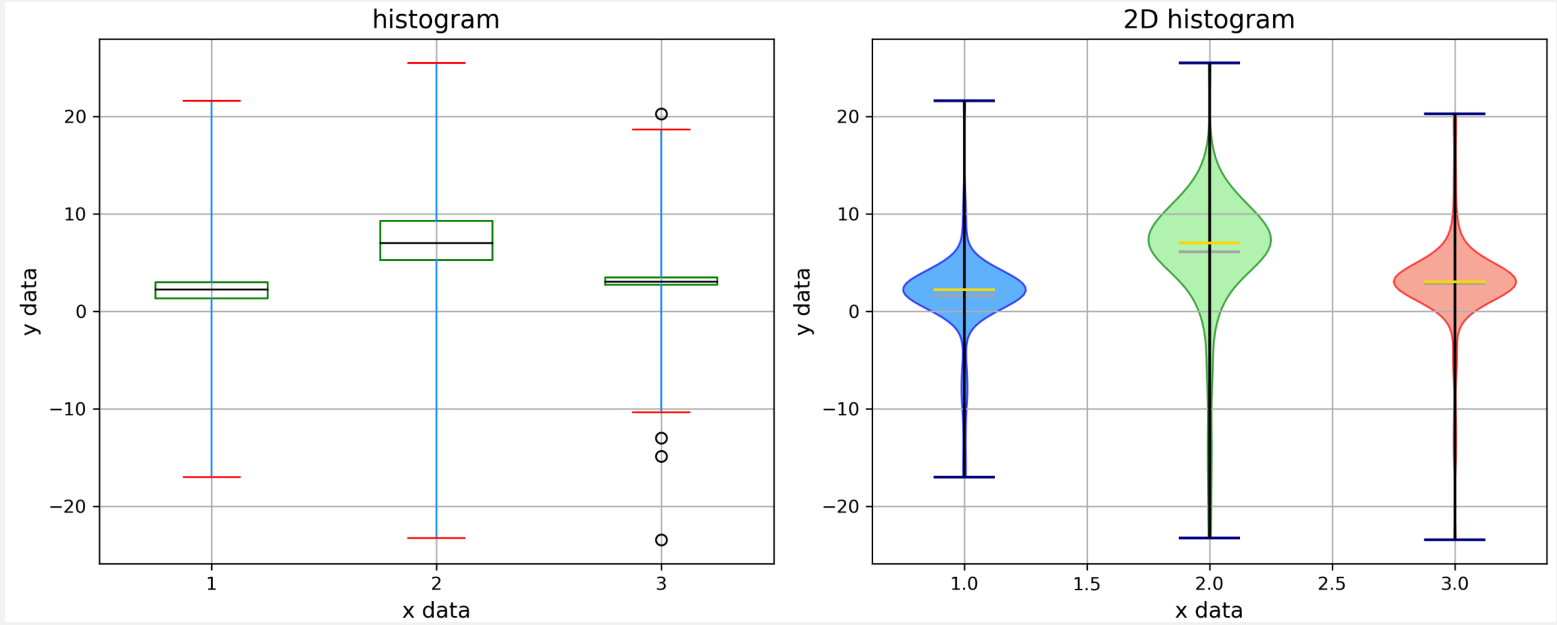
```
# add extreme values
a11 = np.append(a1, np.random.randn(25)*10)
a21 = np.append(a2, np.random.randn(25)*10)
a31 = np.append(a3, np.random.randn(25)*10)
```

Box & Violin Plot

```
fcls = ['dodgerblue', 'lightgreen', 'salmon']
ecls = ['blue', 'green', 'red']
# boxplot and violinplot
fig, axe = plt.subplots(1, 2, figsize=[12,5], dpi=300)
bp = axe[0].boxplot([a11, a21, a31], widths=0.5, whis=20)
plt.setp(bp['boxes'], color='green')
plt.setp(bp['whiskers'], color='dodgerblue')
plt.setp(bp['means'], color='darkgray')
plt.setp(bp['medians'], color='black')
plt.setp(bp['caps'], color='red')
axe[0].set_title('histogram', fontsize=14)
axe[0].set_xlabel('x data', fontsize=12)
axe[0].set_ylabel('y data', fontsize=12)
axe[0].grid(True)
```

Box & Violin Plot

```
parts = axe[1].violinplot(np.column_stack([a11, a21, a31]), [1,2,3], showmeans=True, showmedians=True, showextrema=True)
for i in range(3):
    parts['bodies'][i].set_facecolor(fcls[i])
    parts['bodies'][i].set_edgecolor(ecls[i])
    parts['bodies'][i].set_alpha(0.7)
parts['cmeans'].set_color('darkgray')
parts['cmedians'].set_color('gold')
parts['cbars'].set_color('black')
parts['cmins'].set_color('navy')
parts['cmaxes'].set_color('navy')
axe[1].set_title('2D histogram', fontsize=14)
axe[1].set_xlabel('x data', fontsize=12)
axe[1].set_ylabel('y data', fontsize=12)
axe[1].grid(True)
plt.tight_layout()
plt.show()
```



Create a Mesh Grid Data

Create a new data for demonstrating mesh grid data

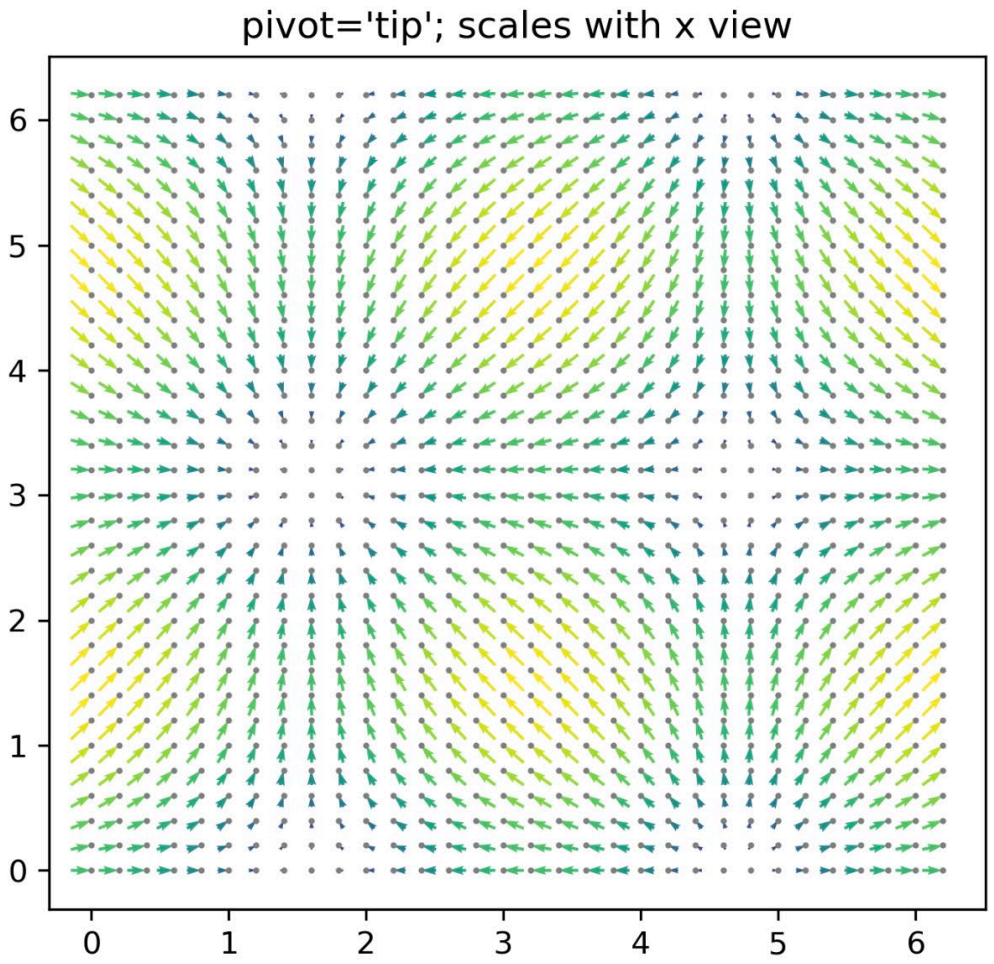
```
# create data
X, Y = np.meshgrid(np.arange(0, 2 * np.pi, .2), np.arange(0, 2 * np.pi, .2))
U = np.cos(X)
V = np.sin(Y)
```

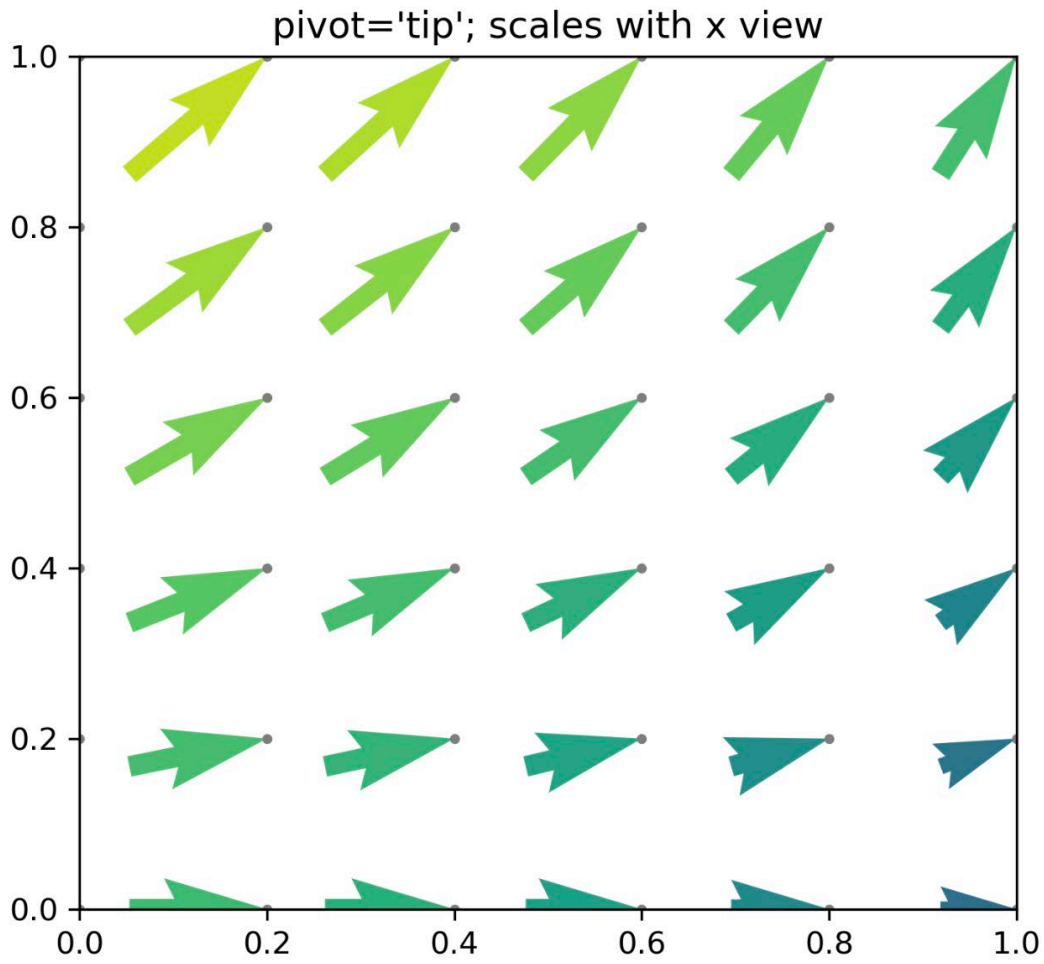
Quiver

```
fig, axe = plt.subplots(1, 2, figsize=[12,5], dpi=300)
axe[0].set_title("pivot='tip'; scales with x view")
M = np.hypot(U, V)
Q = axe[0].quiver(X, Y, U, V, M, units='x', pivot='tip', width=0.022, scale=1/0.15)
axe[0].scatter(X, Y, color='0.5', s=1)
plt.axis([0,1,0,1])

axe[1].set_title("pivot='tip'; scales with x view")
M = np.hypot(U, V)
Q = axe[1].quiver(X, Y, U, V, M, units='x', pivot='tip', width=0.022, scale=1/0.15)
axe[1].scatter(X, Y, color='0.5', s=5)
plt.show()
```

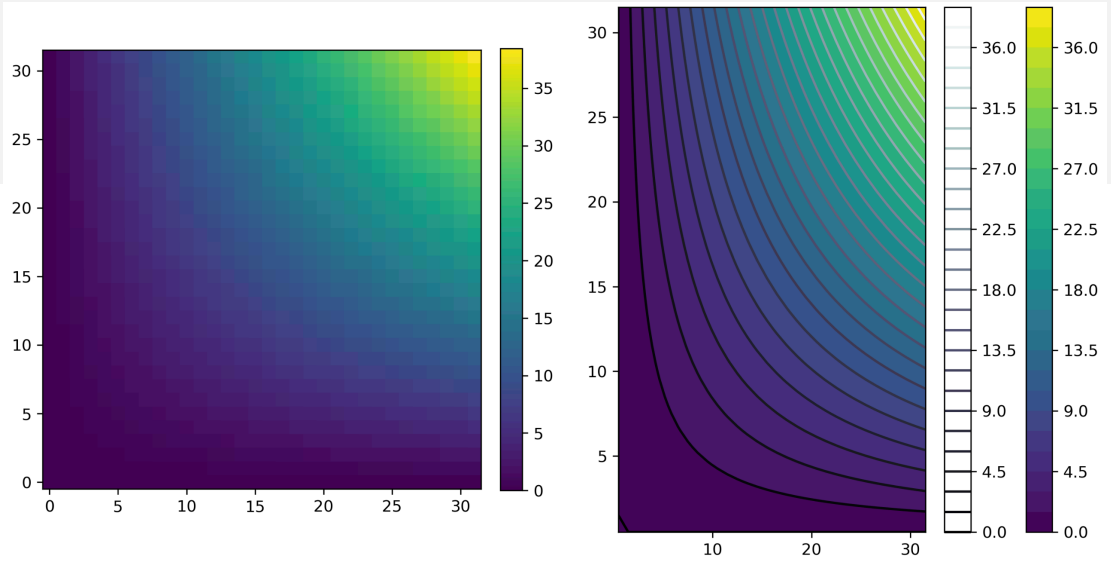
Quiver


pivot='tip'; scales with x view
0
1
2
3
4
5
6


pivot='tip'; scales with x view
0.0
0.2
0.4
0.6
0.8
1.0

Contour & Contourf

```
# contour and contourf
fig, axe = plt.subplots(1, 2, figsize=[12,6], dpi=300)
im = axe[0].imshow(X*Y, origin='lower')
plt.colorbar(im, fraction=0.046, pad=0.04)
ct1 = axe[1].contourf(X*Y, origin='lower', levels=30)
ct2 = axe[1].contour(X*Y, origin='lower', levels=30, cmap='bone')
plt.colorbar(ct1)
plt.colorbar(ct2)
plt.show()
```

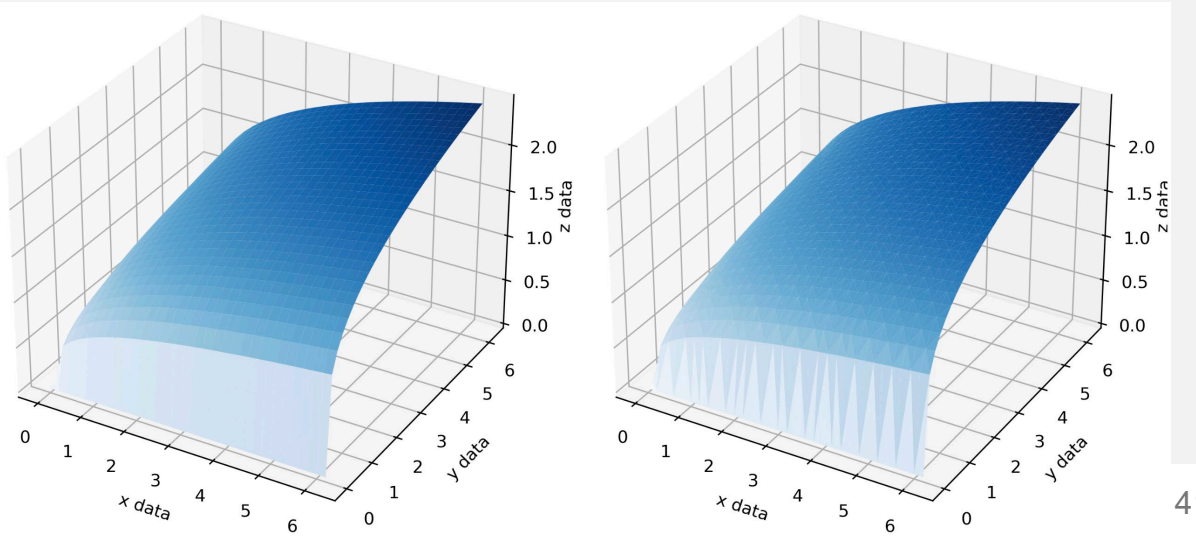


Reshape for Further Charts

```
# reshape matrix
Z = np.sqrt(np.sqrt(X*Y))
x = X.reshape(1, X.shape[0]**2)[0]
y = Y.reshape(1, Y.shape[0]**2)[0]
z = Z.reshape(1, Z.shape[0]**2)[0]
```

Surface & Trisurf

```
fig, axe = plt.subplots(1, 2, figsize=[10,10], dpi=300, subplot_kw={"projection": "3d"})
axe[0].plot_surface(X, Y, Z, vmin=Z.min() * 2, cmap=cm.Blues)
axe[0].set(xlabel='x data', ylabel='y data', zlabel='z data')
axe[0].set_zlabel('z data', labelpad=0)
axe[0].view_init(30, 300, 0)
axe[1].plot_trisurf(x, y, z, vmin=z.min() * 2, cmap=cm.Blues)
axe[1].set(xlabel='x data', ylabel='y data', zlabel='z data')
axe[1].set_zlabel('z data', labelpad=0)
axe[1].view_init(30, 300, 0)
fig.tight_layout(pad=2, w_pad=2)
plt.show()
```

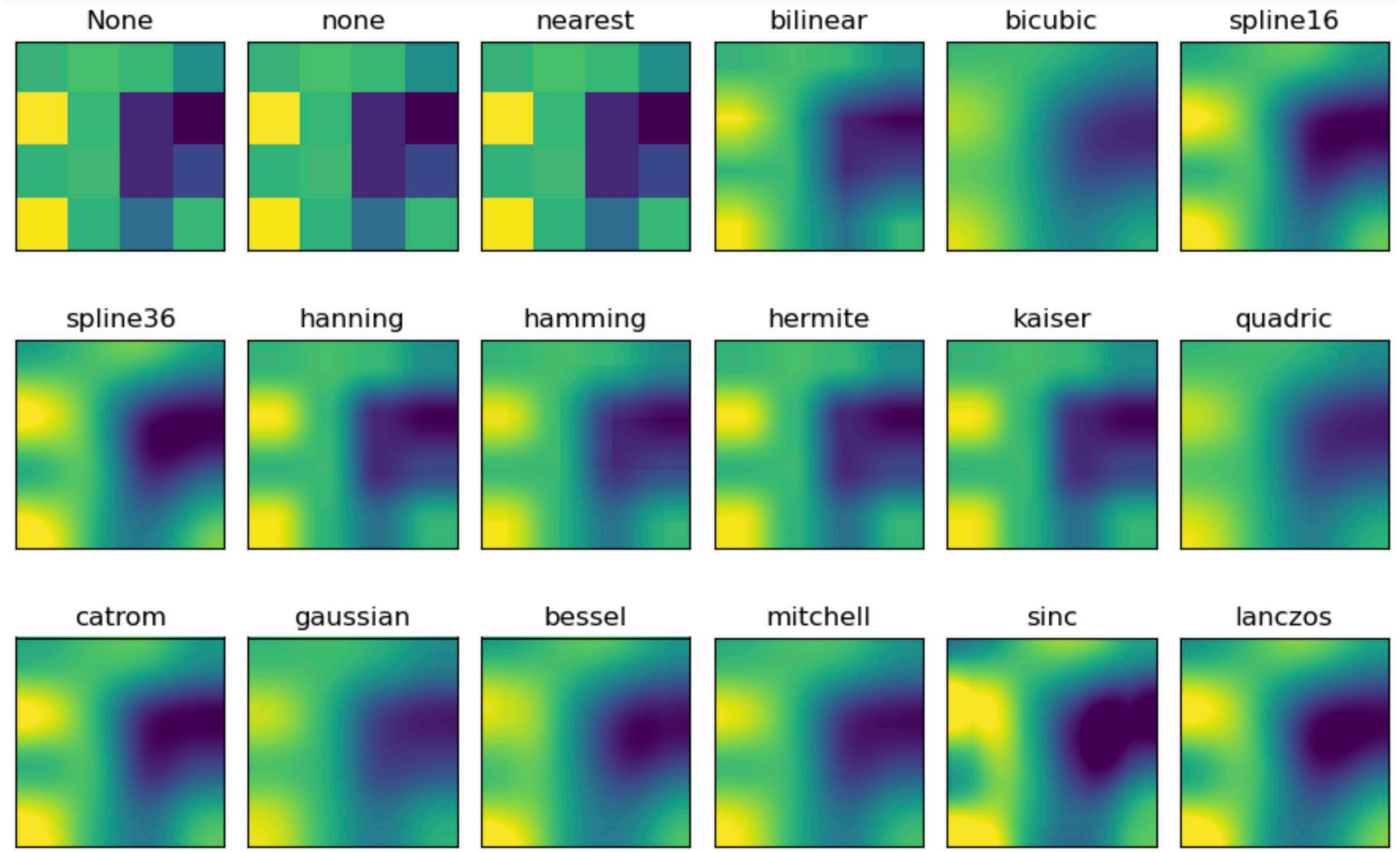


Interpolation for imshow

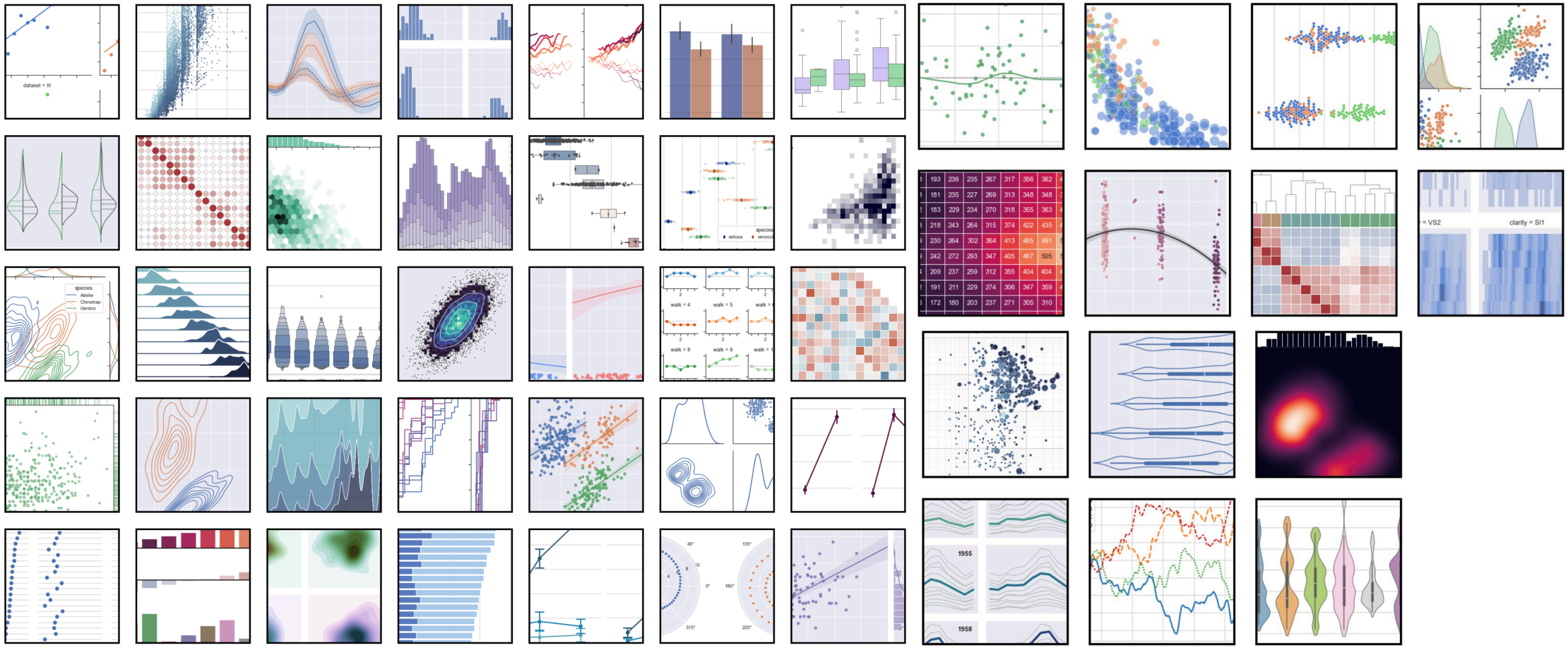
```
methods = [None, 'none', 'nearest', 'bilinear', 'bicubic', 'spline16', 'spline36', 'hanning',
           'hamming', 'hermite', 'kaiser', 'quadric', 'catrom', 'gaussian', 'bessel', 'mitchell',
           'sinc', 'lanczos']
# Fixing random state for reproducibility
np.random.seed(19680801)
grid = np.random.rand(4, 4)

fig, axs = plt.subplots(nrows=3, ncols=6, figsize=(9, 6), subplot_kw={'xticks': [], 'yticks': []})
for ax, interp_method in zip(axs.flat, methods):
    ax.imshow(grid, interpolation=interp_method, cmap='viridis')
    ax.set_title(str(interp_method))
plt.tight_layout()
plt.show()
```

Interpolation for imshow

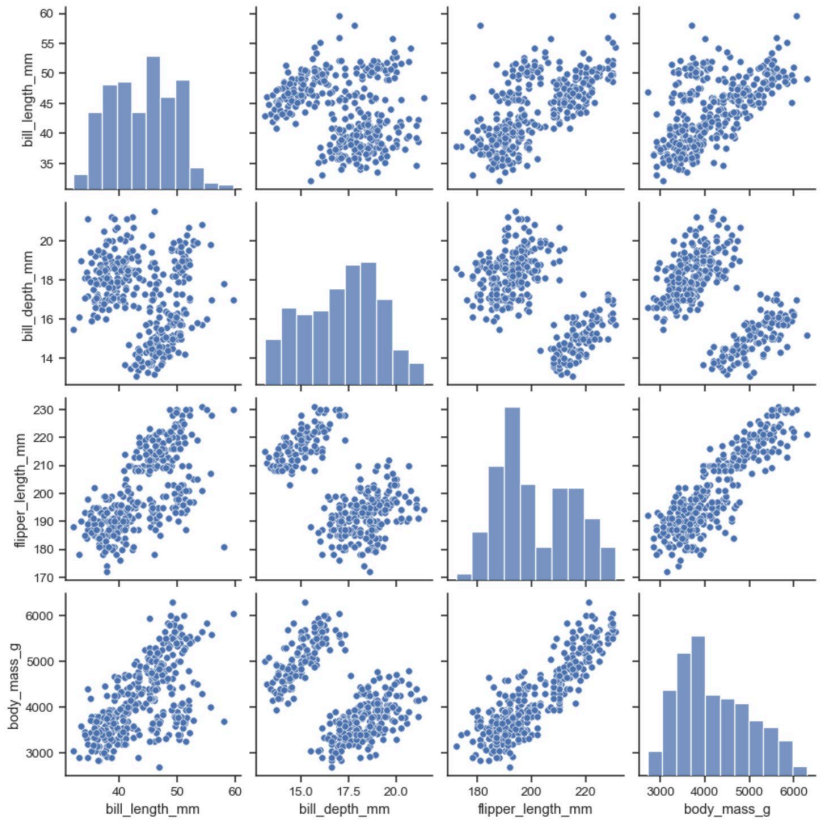


Seaborn

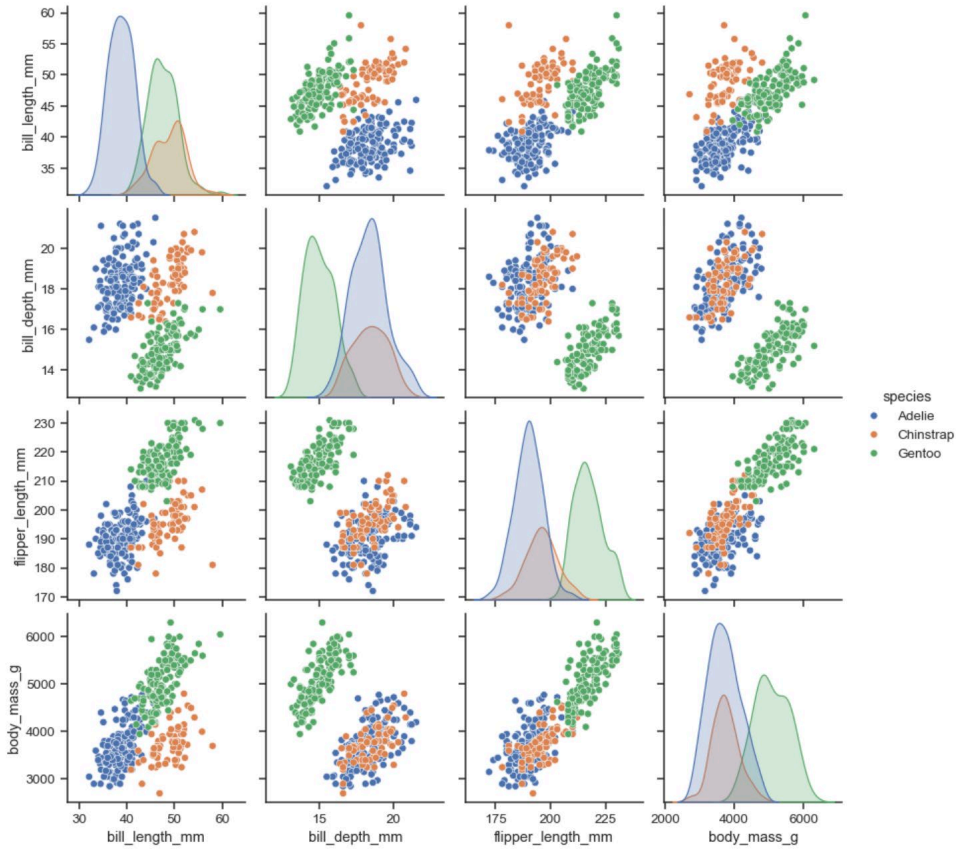


Pairplot

```
penguins = sns.load_dataset("penguins")
sns.pairplot(penguins)
```

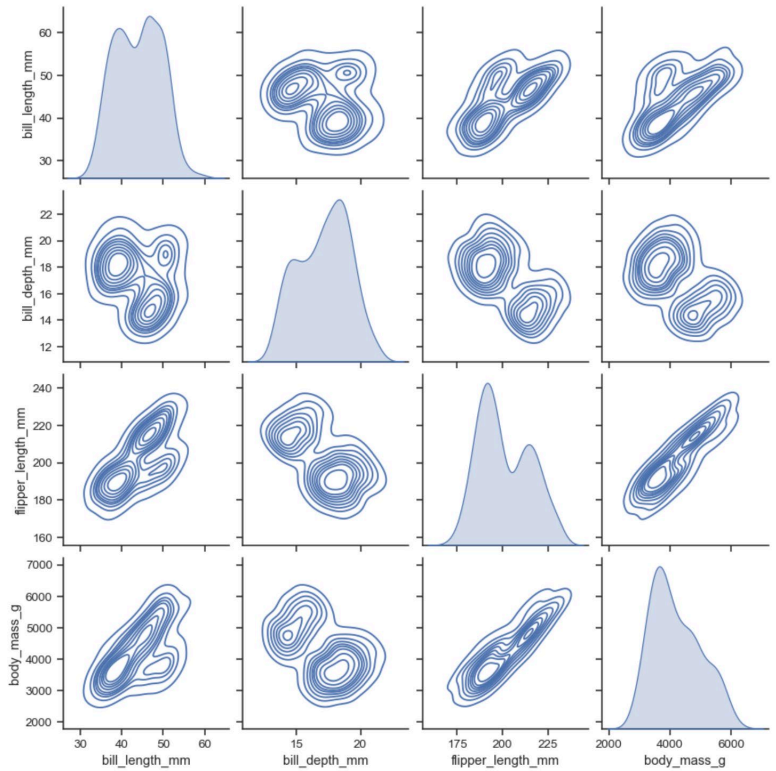



```
sns.pairplot(penguins, hue="species")
```

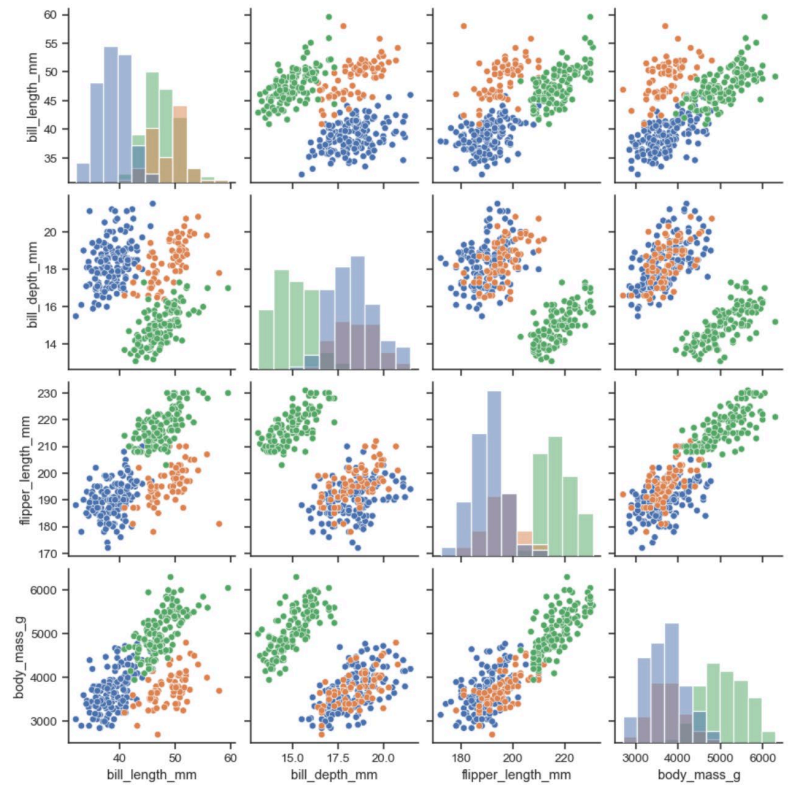


Pairplot

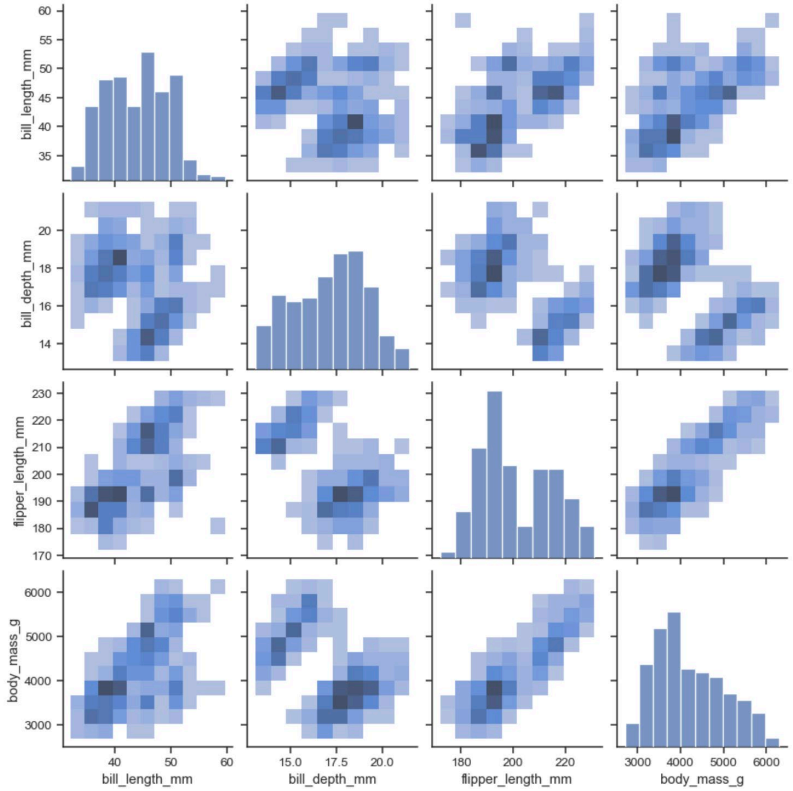
```
sns.pairplot(penguins, kind="kde")
```




```
sns.pairplot(penguins, hue="species", diag_kind="hist")
```




```
sns.pairplot(penguins, kind="hist")
```



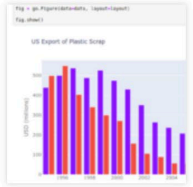
Plotly

Fundamentals

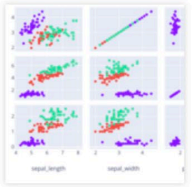
More Fundamentals »

The Figure Data Structure

Creating and Updating Figures



Displaying Figures

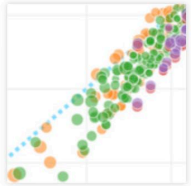


Plotly Express

Analytical Apps with Dash

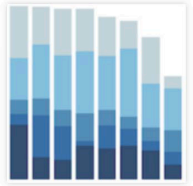
Basic Charts

More Basic Charts »

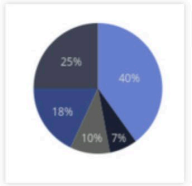


Scatter Plots

Line Charts



Bar Charts

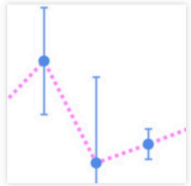


Pie Charts

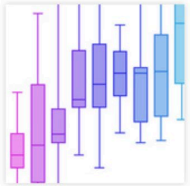
Bubble Charts

Statistical Charts

More Statistical Charts »



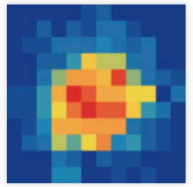
Error Bars



Box Plots

Histograms

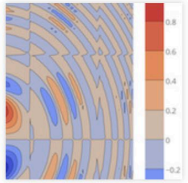
Distplots



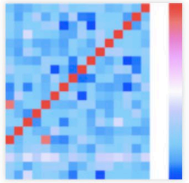
2D Histograms

Scientific Charts

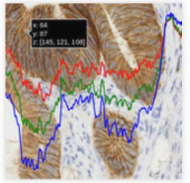
More Scientific Charts »



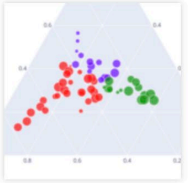
Contour Plots



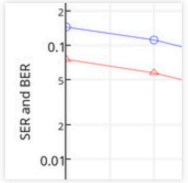
Heatmaps



Imshow



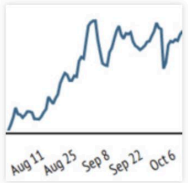
Ternary Plots



Log Plots

Financial Charts

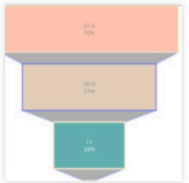
More Financial Charts »



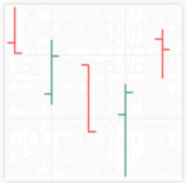
Time Series and Date Axes

Candlestick Charts

Waterfall Charts



Funnel Chart



OHLC Charts

Maps

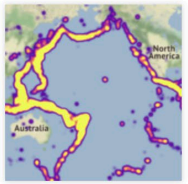
More Maps »

Mapbox Choropleth Maps

Lines on Mapbox

Filled Area on Maps

Bubble Maps

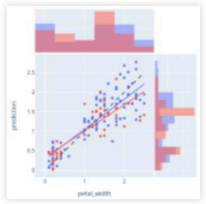


Mapbox Density Heatmap

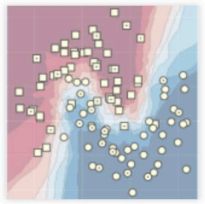
Plotly

Artificial Intelligence and Machine Learning

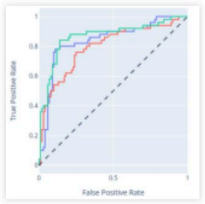
More AI and ML »



ML Regression

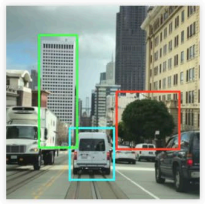


kNN Classification



ROC and PR Curves

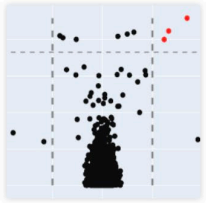
PCA Visualization



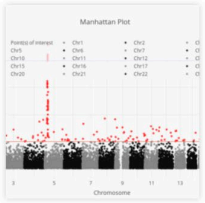
AI/ML Apps with Dash

Bioinformatics

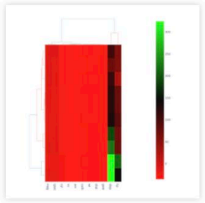
More Bioinformatics »



Volcano Plot



Manhattan Plot

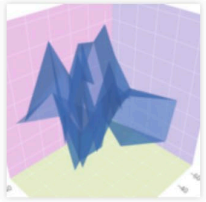


Clustergram

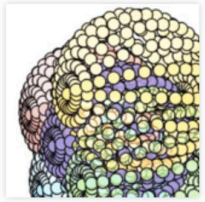
Alignment Chart

3D Charts

More 3D Charts »

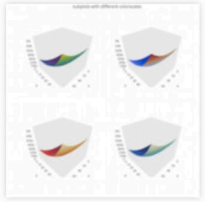


3D Axes

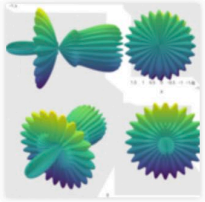


3D Scatter Plots

3D Surface Plots



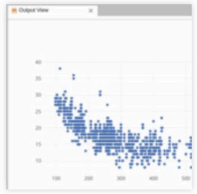
3D Subplots



3D Camera Controls

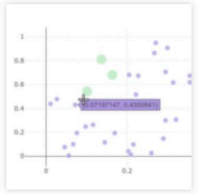
Jupyter Widgets Interaction

Plotly FigureWidget Overview



Jupyter Lab with FigureWidget

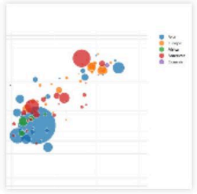
Interactive Data Analysis with FigureWidget ipywidgets



Click Events

Transforms

Filter



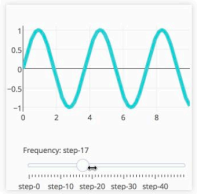
Group By

Aggregations

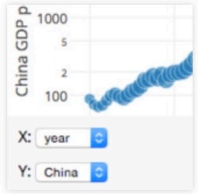
Multiple Transforms

Add Custom Controls

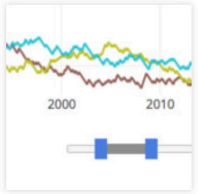
Custom Buttons



Sliders



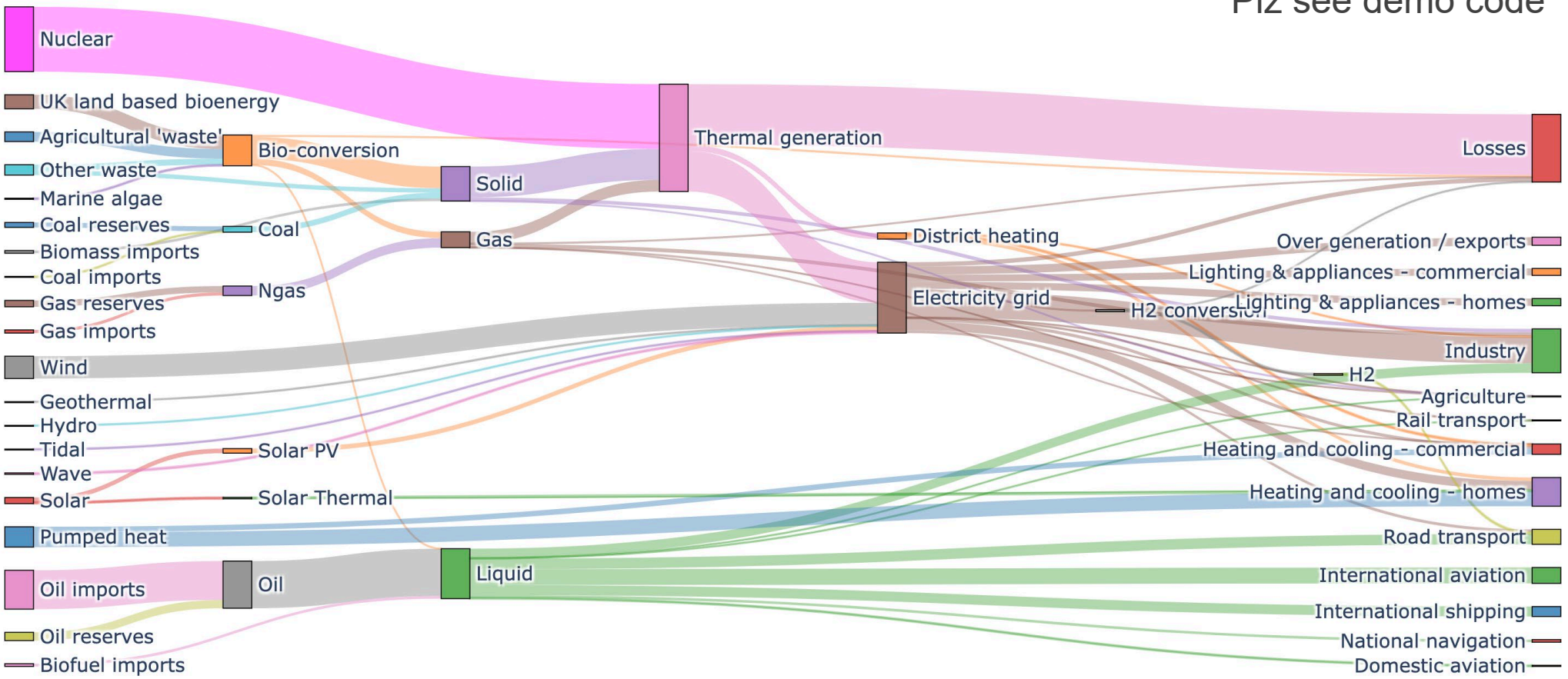
Dropdown Menus



Range Slider and Selector

Sankey Diagram

Plz see demo code



The End

Thank you for your attention!

Email: chchan@ntnu.edu.tw

Website: https://toodou.github.io/