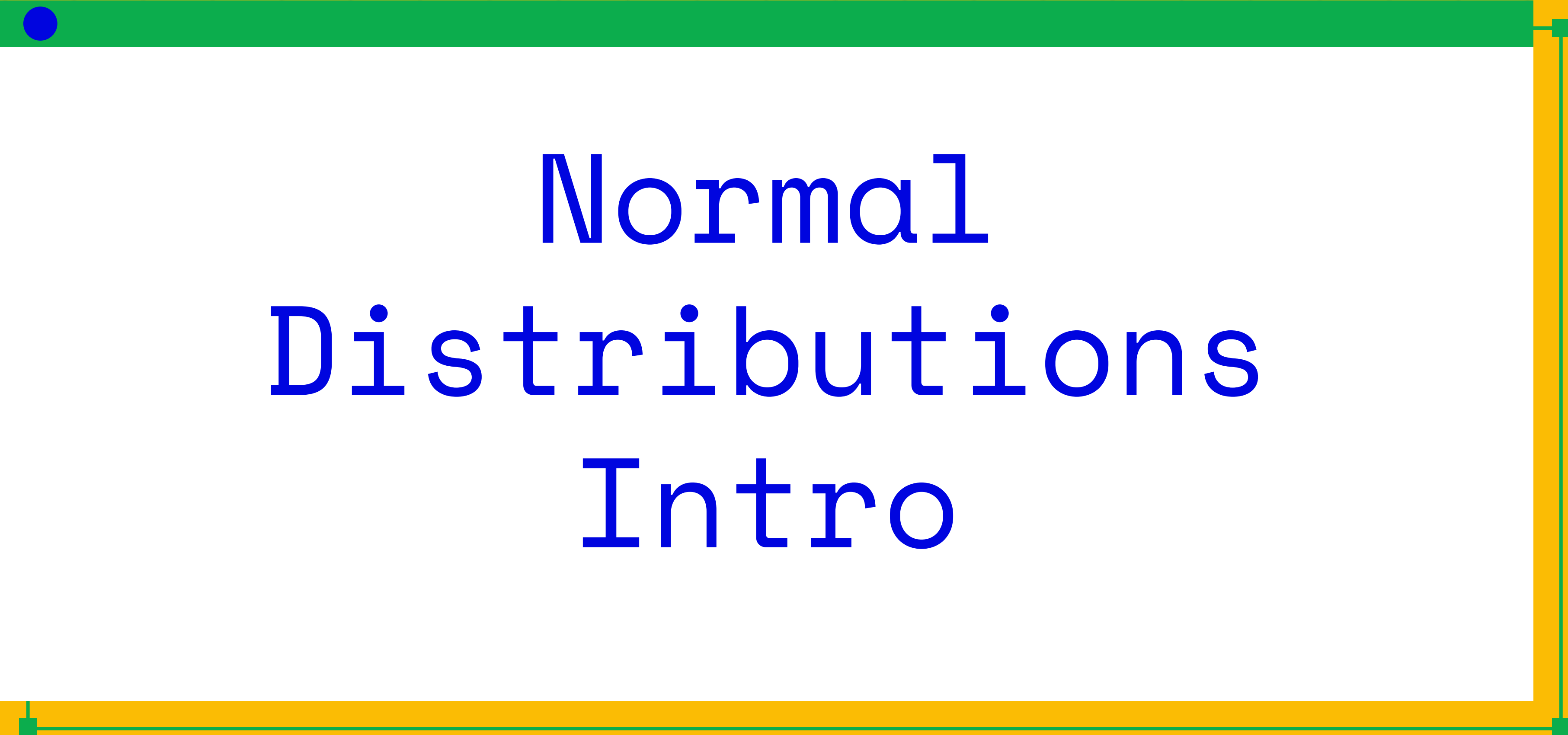




Data Visualization

Agenda

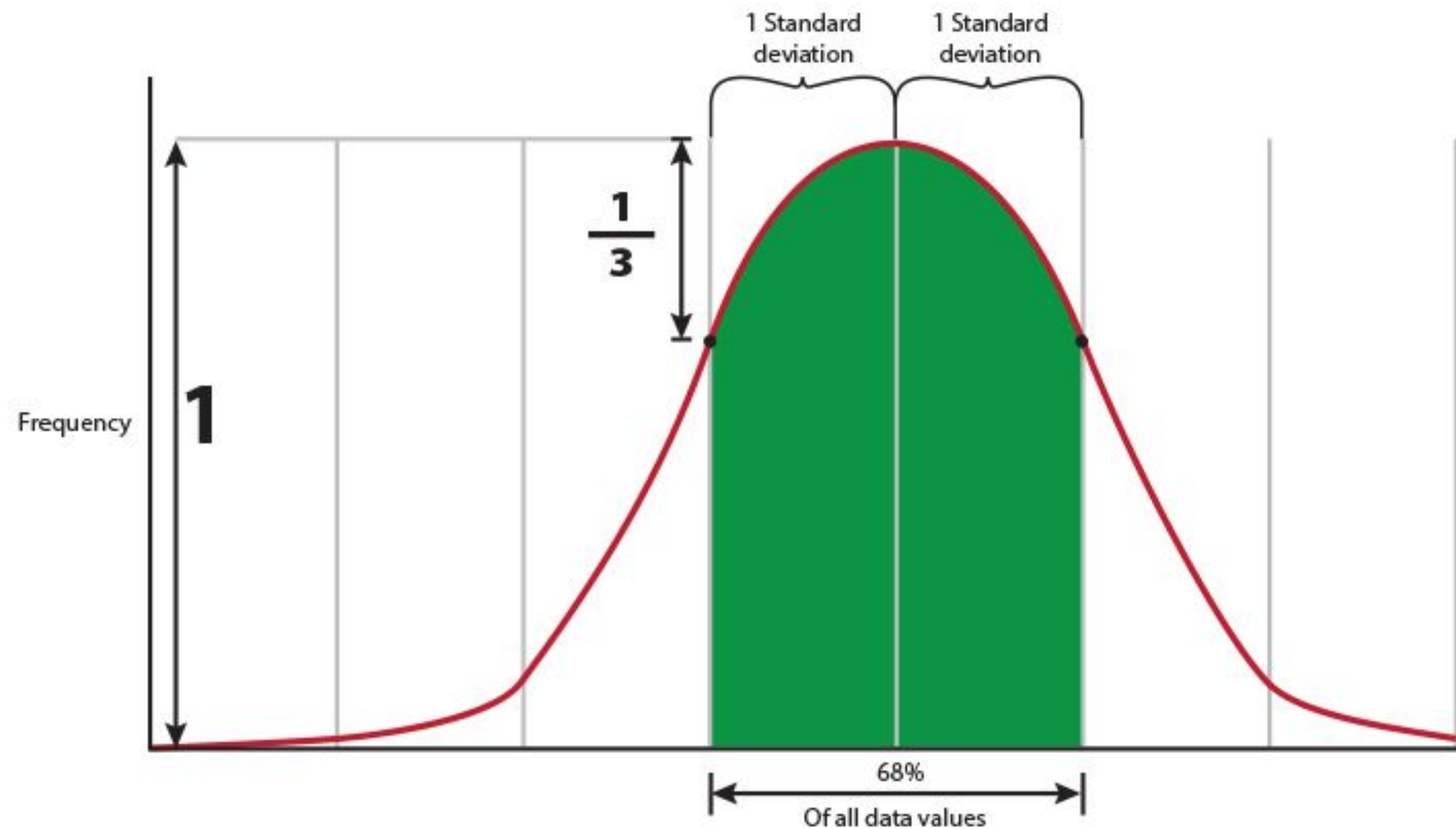
1. Normal Distributions Intro
2. Histogram
3. Matplotlib Artist Layer Examples
4. Questions



Normal Distributions Intro

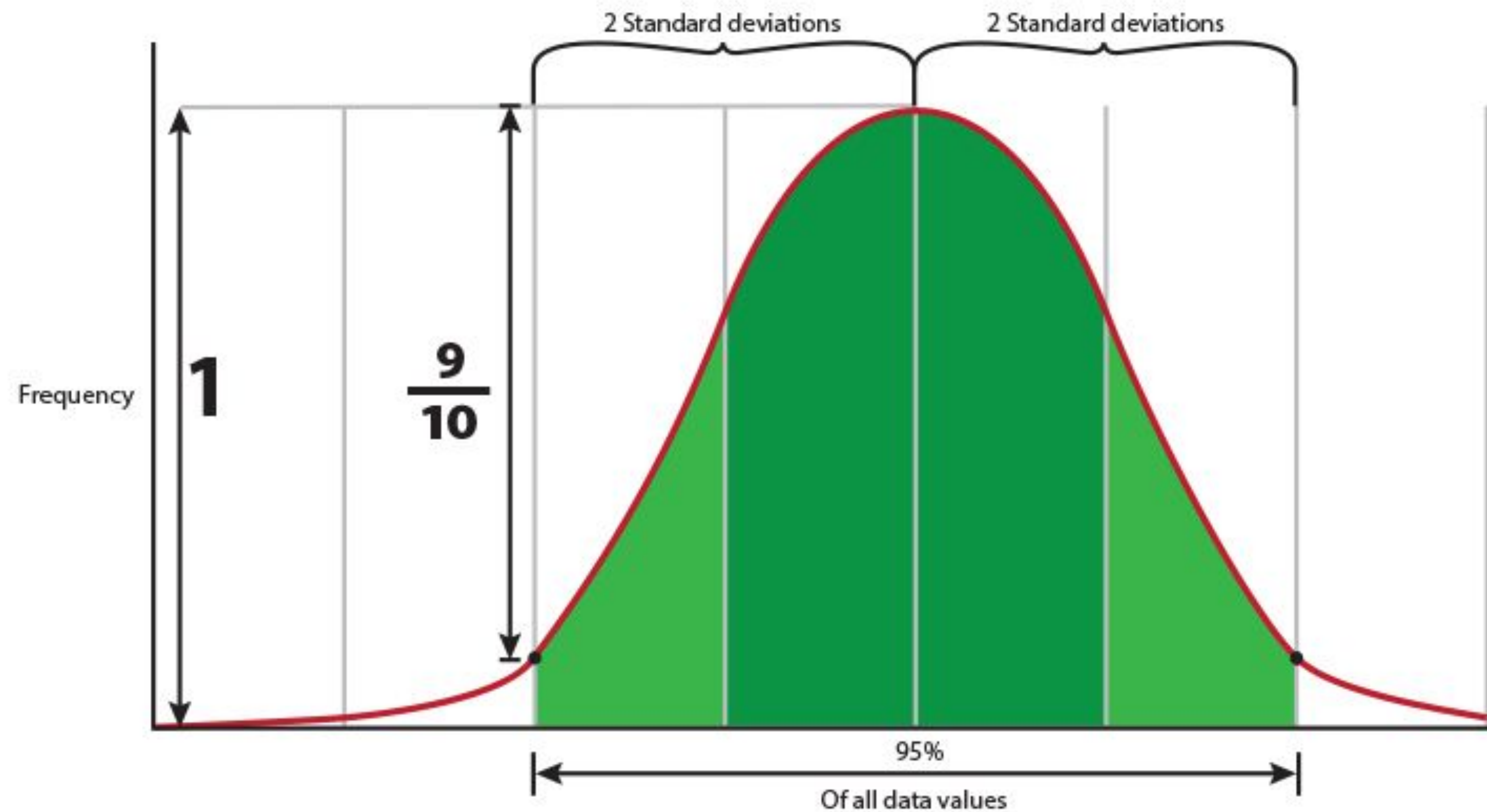
Normal Distributions (1 Standard Deviation)

68% of data is ≤ 1 standard deviation away from the mean



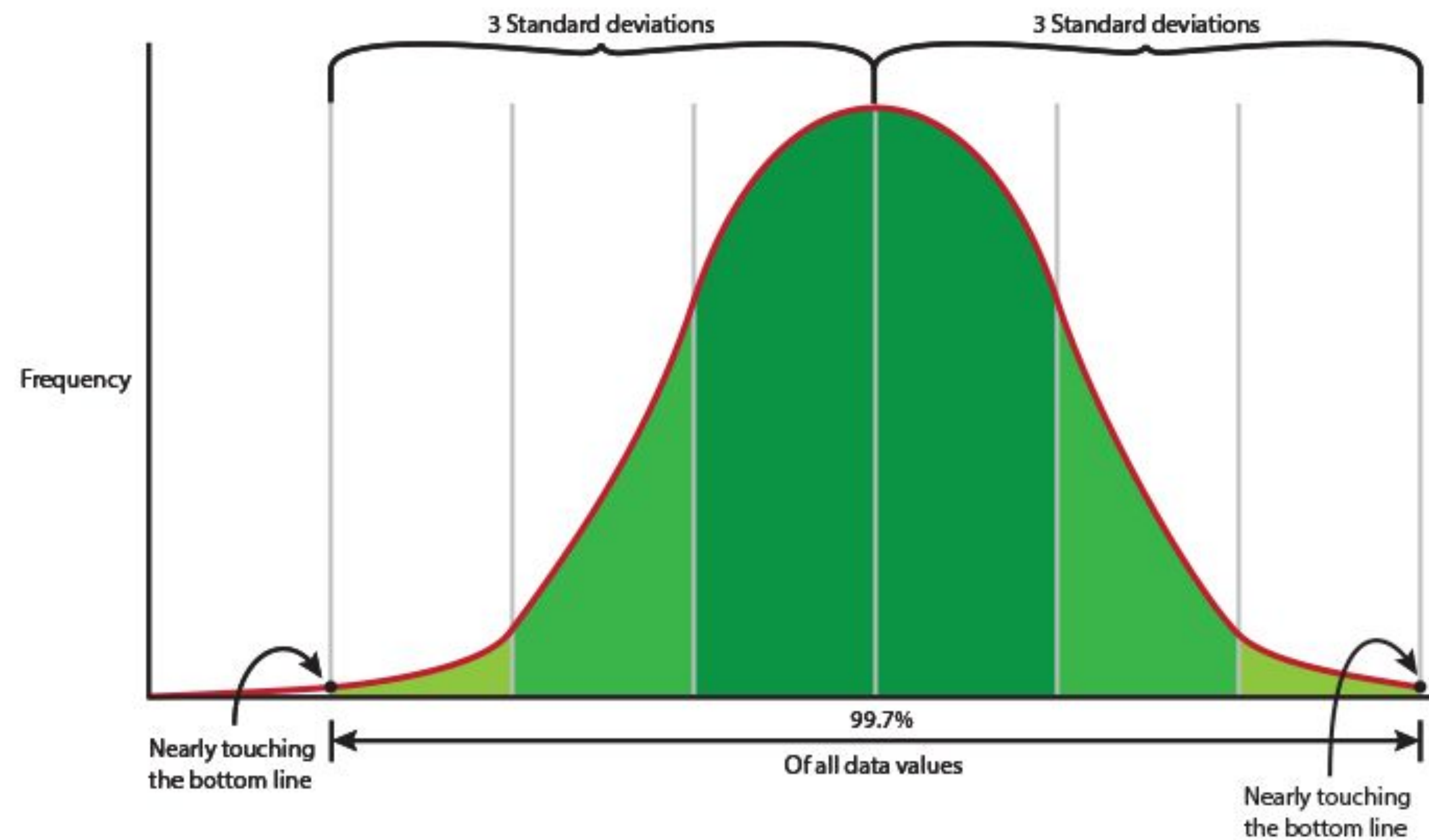
Normal Distributions (2 Standard Deviation)

95% of data is ≤ 2 standard deviations away from the mean

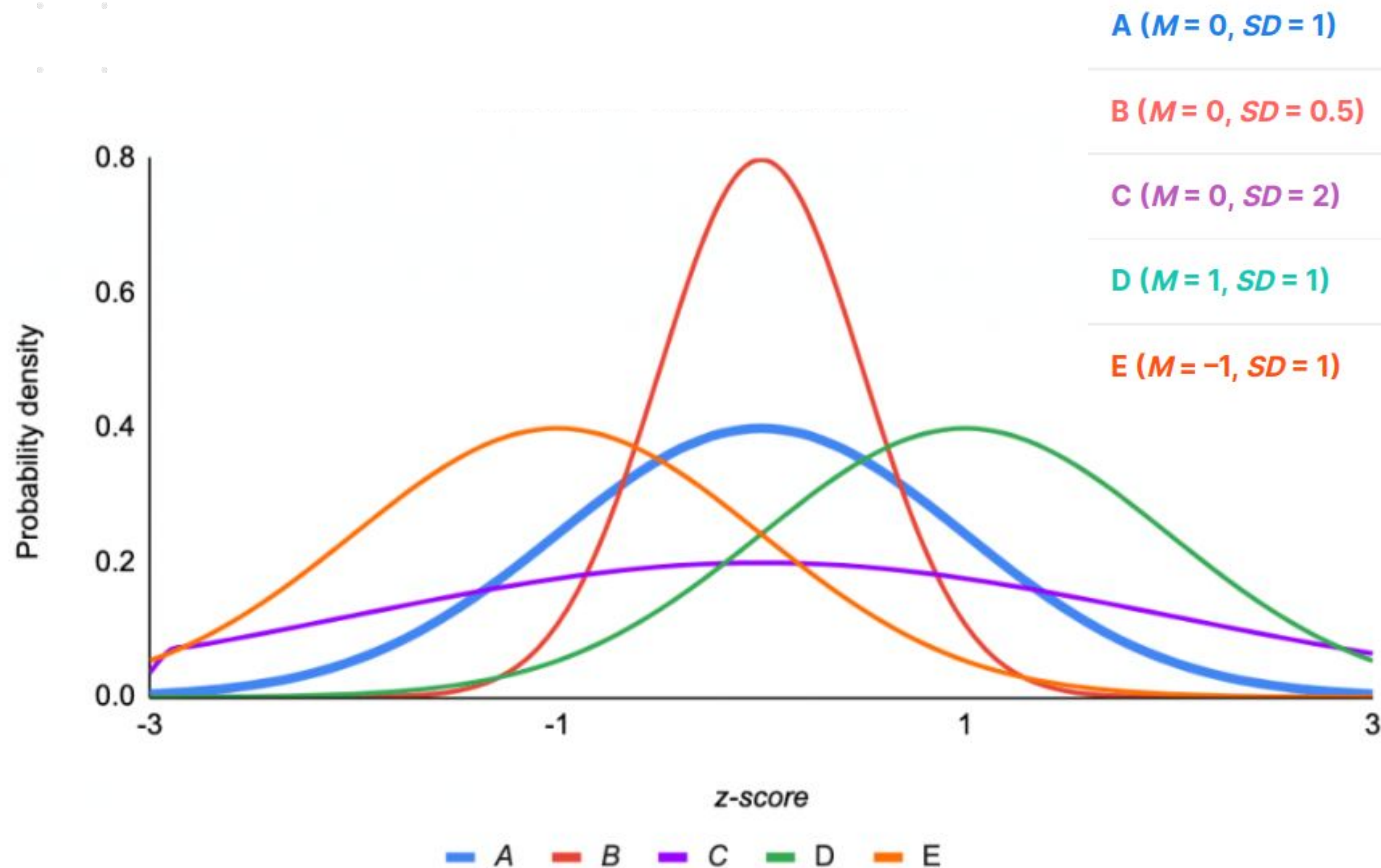


Normal Distributions (3 Standard Deviation)

99.7% of data is ≤ 3 standard deviations away from the mean



Standard Normal Distribution





Histogram

Histogram

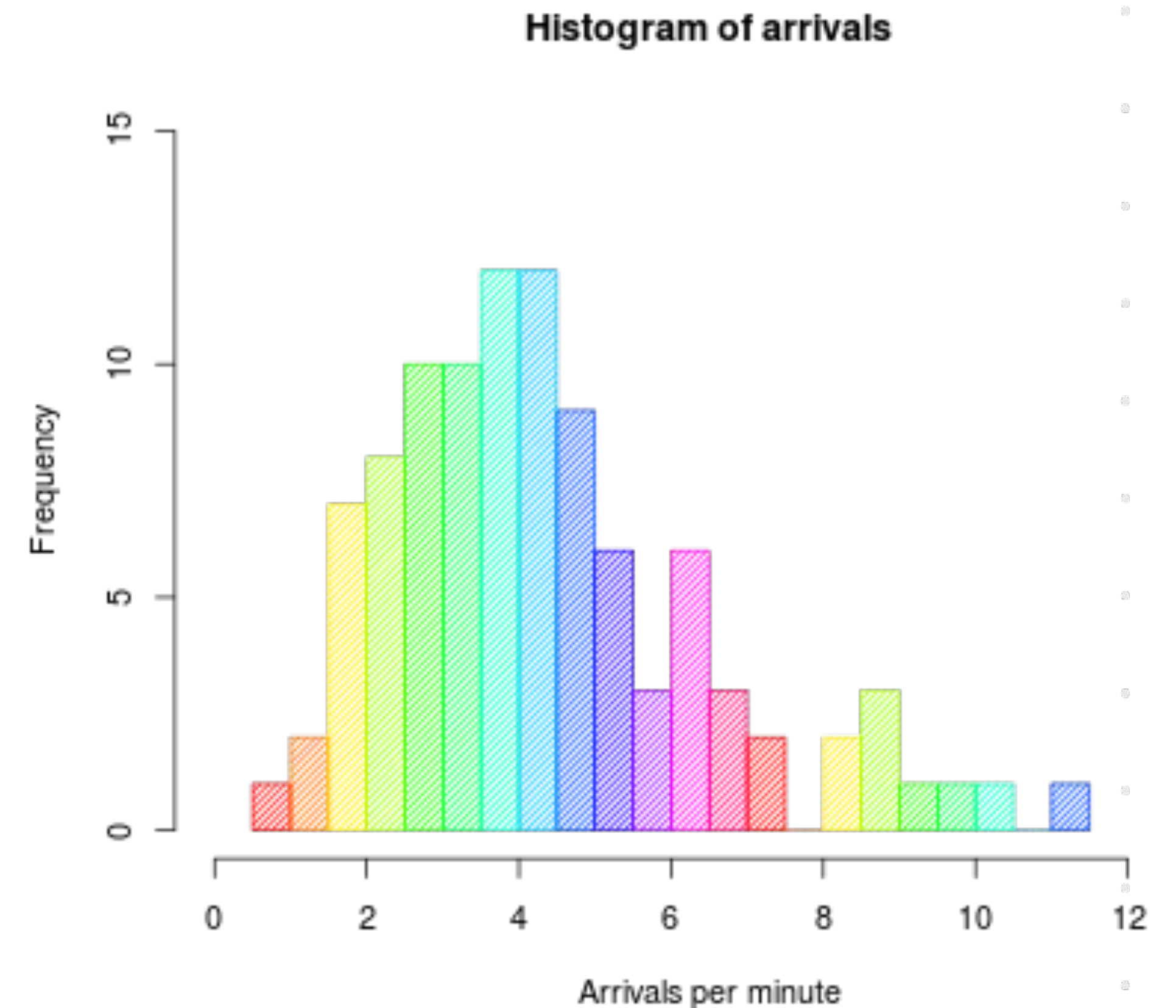
- approximate representation of numerical data distribution

- **Construct Histogram**

- bin (bucket) the range of values
- count how many values fall into each interval

- The **Bins** are usually specified as

- consecutive
- non-overlapping intervals



Histogram vs Column Chart

- **Histogram**

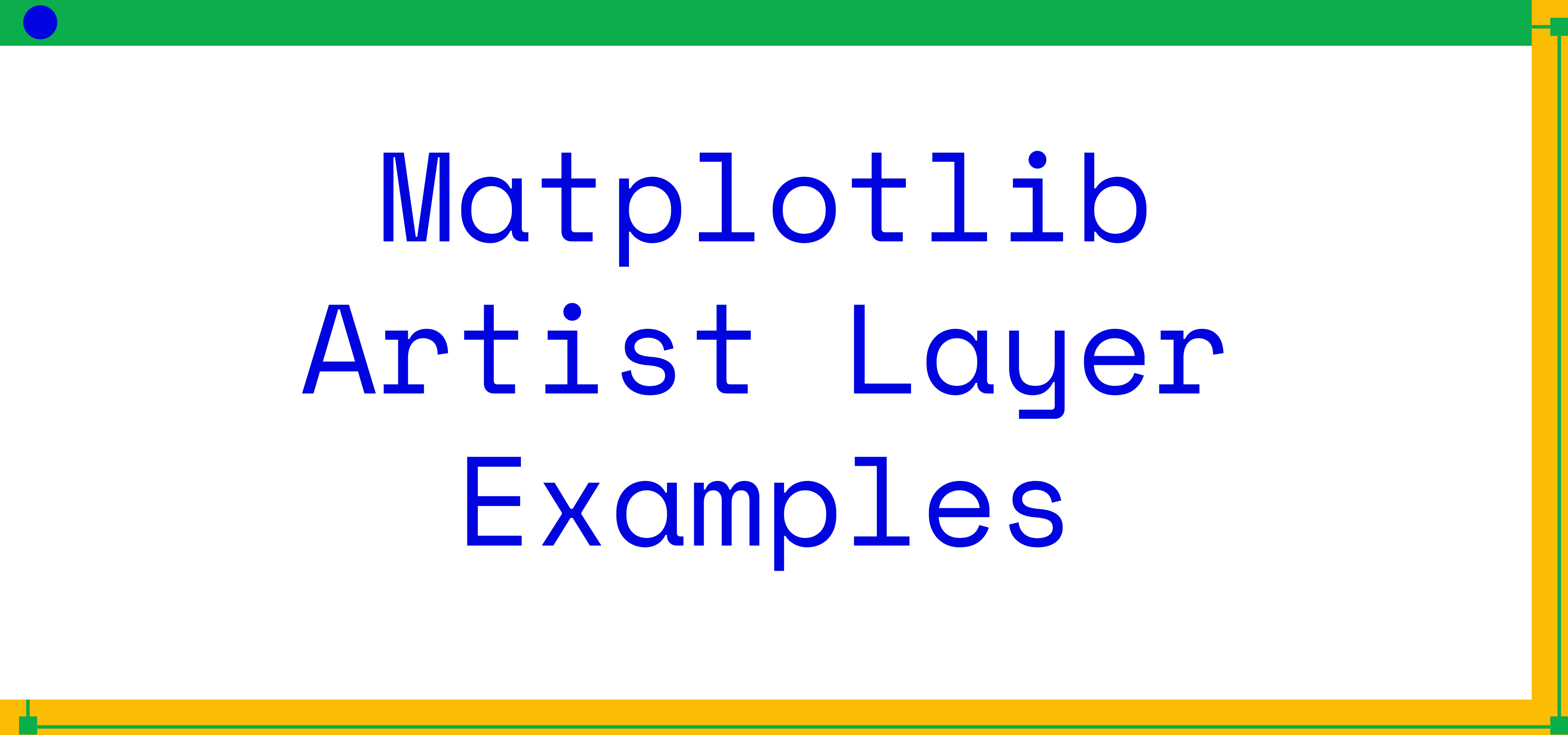
- used for **continuous data**, where the bins represent **ranges of data**

- **Column Chart**

- plot of **categorical variables**

- **Recommendation**

- **Histogram** rectangles touch each other to indicate: original variable is **continuous**
- **Column Chart** has gaps between the rectangles to clarify the distinction



Matplotlib Artist Layer Examples

NumPy Example

localhost:8888/notebooks/dv-0.ipynb

 jupyter dv-0 Last Checkpoint: 21 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted



Run



Code



```
In [1]: import numpy as np
```

```
np.random.randn(10) #Return 10 samples from the Standard Normal Distribution
```

```
Out[1]: array([-0.64393041,  0.0329367 , -0.16840147,  0.88846809,  0.76751103,  
               0.18852699, -1.30213432,  0.58043701,  1.80149475, -0.18262329])
```

NumPy: Numerical Python library for Scientific Computing

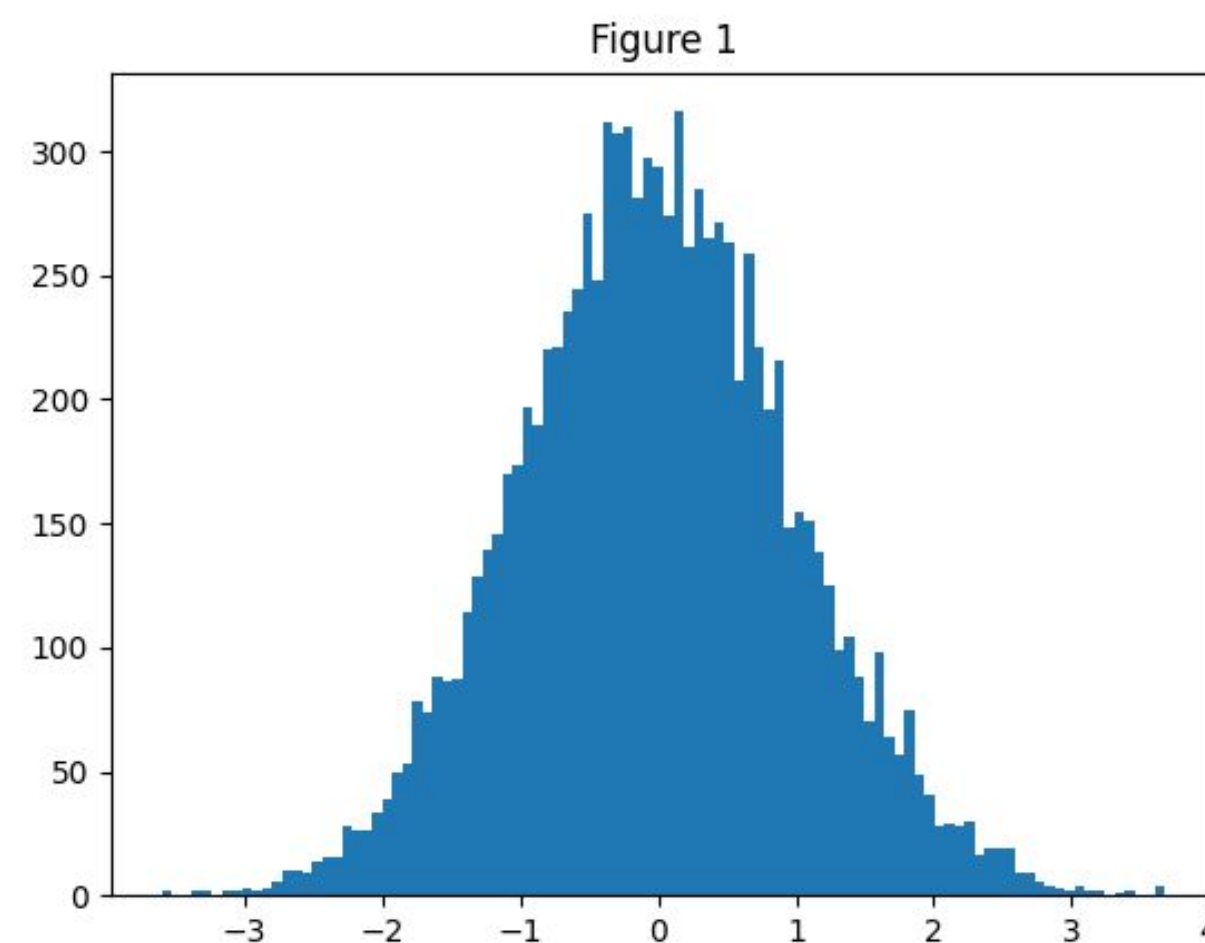
https://github.dev/numpy/numpy/blob/main/numpy/random/_init_.py

Artist Layer Example 1

```
In [2]: from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
import numpy as np

x = np.random.randn(10000)

fig = Figure() #Create an Artist (the Figure Artist)
canvas = FigureCanvas(fig) #Create a Canvas and attach the Artist to it
ax = fig.add_subplot(111) #Create a second Artist (the Axes Artist)
ax.hist(x, 100) #Call hist method to generate the histogram with 100 bins
ax.set_title('Figure 1') #We should write an appropriate title
fig.savefig('hist1.png') #Save the figure
```



Artist Layer Example 1 - Notes

- Use **Artist Layer** to generate histogram of 10000 random numbers
- **Anti Grain Geometry (AGG)**
 - a high-performance library that produces attractive images
- use **111** (from MATLAB convention)
 - creates a grid with 1 row and 1 column
 - uses the first cell in that grid for the location of the new **Axes Artist**
- **hist** method
 - creates a sequence of **Rectangle Artists**

Artist Layer Example 2

```
import numpy as np
x = np.random.randn(10000)

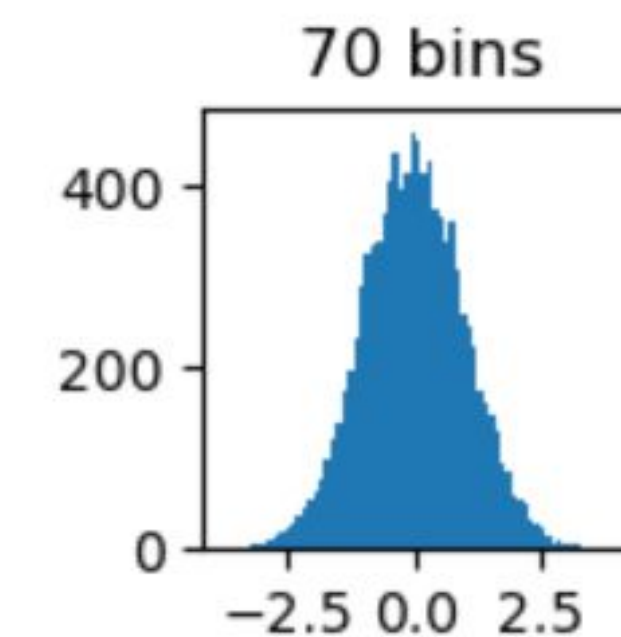
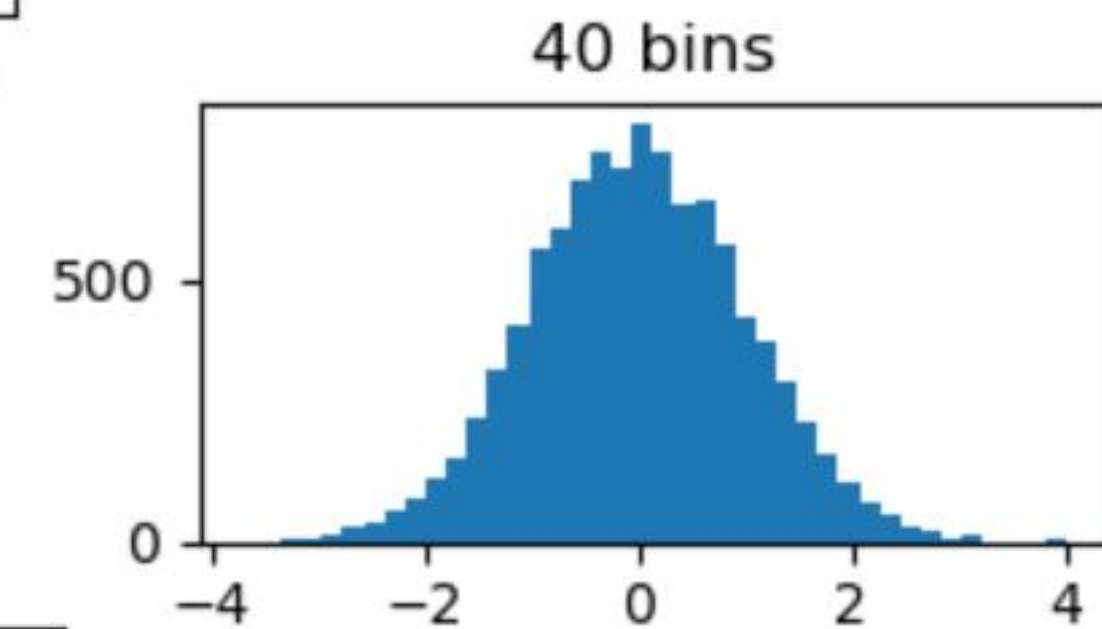
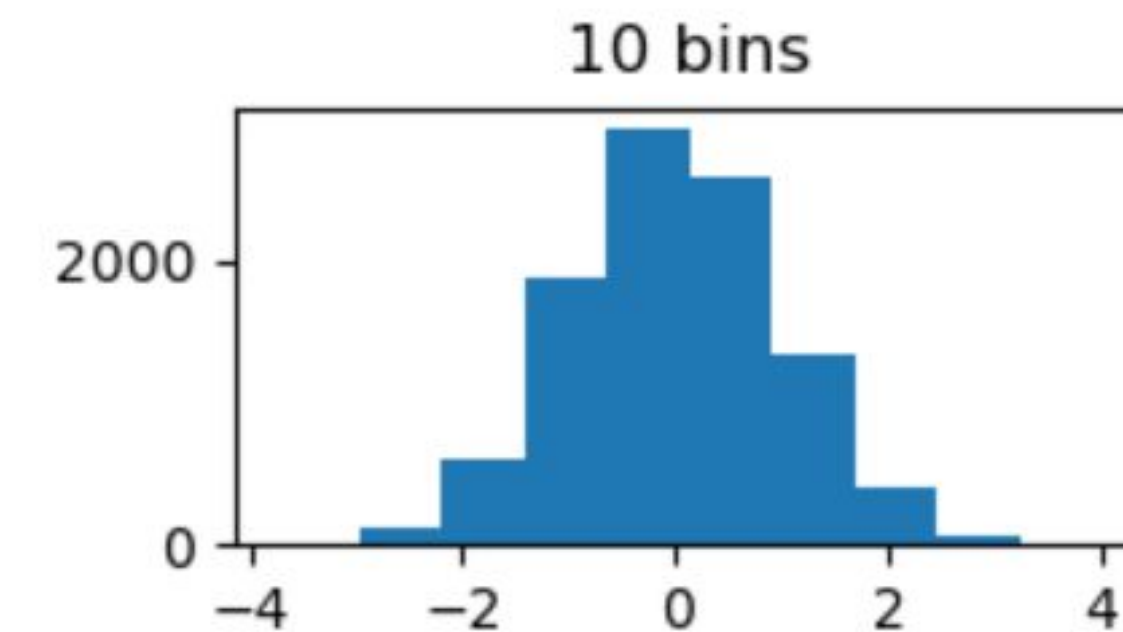
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
fig = Figure()
canvas = FigureCanvas(fig)

ax1 = fig.add_subplot(321)
ax1.hist(x, 10)
ax1.set_title('10 bins')

ax2 = fig.add_subplot(324)
ax2.hist(x, 40)
ax2.set_title('40 bins')

ax3 = fig.add_subplot(3,4,10)
ax3.hist(x, 70)
ax3.set_title('70 bins')

fig.savefig('fig4.png')
```



GridSpec

GridSpec

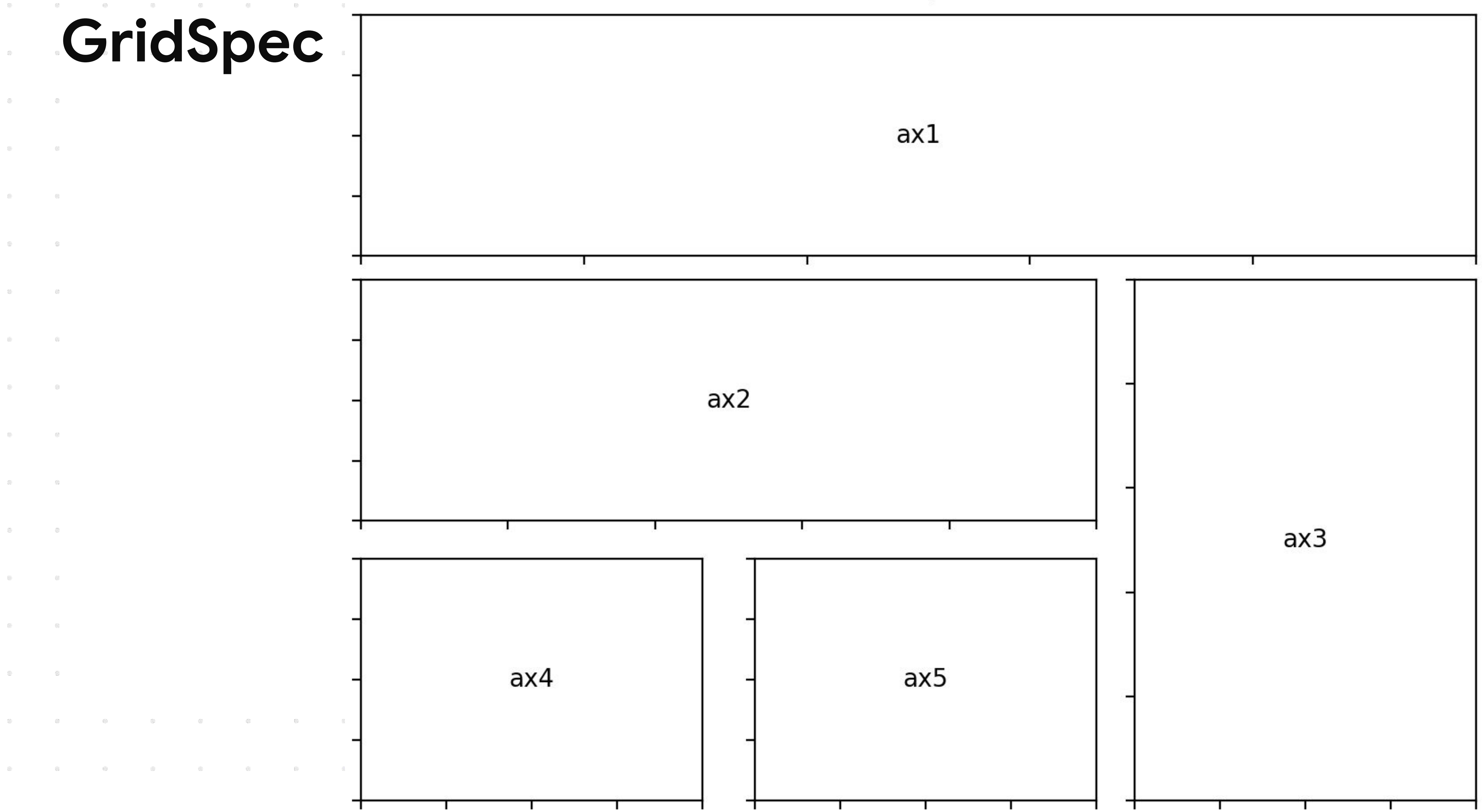
ax1

ax2

ax3

ax4

ax5



Artist Layer (Using GridSpec) Example 1

```
import numpy as np
x = np.random.randn(10000)

from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
fig = Figure()
canvas = FigureCanvas(fig)
gs = fig.add_gridspec(3, 3)

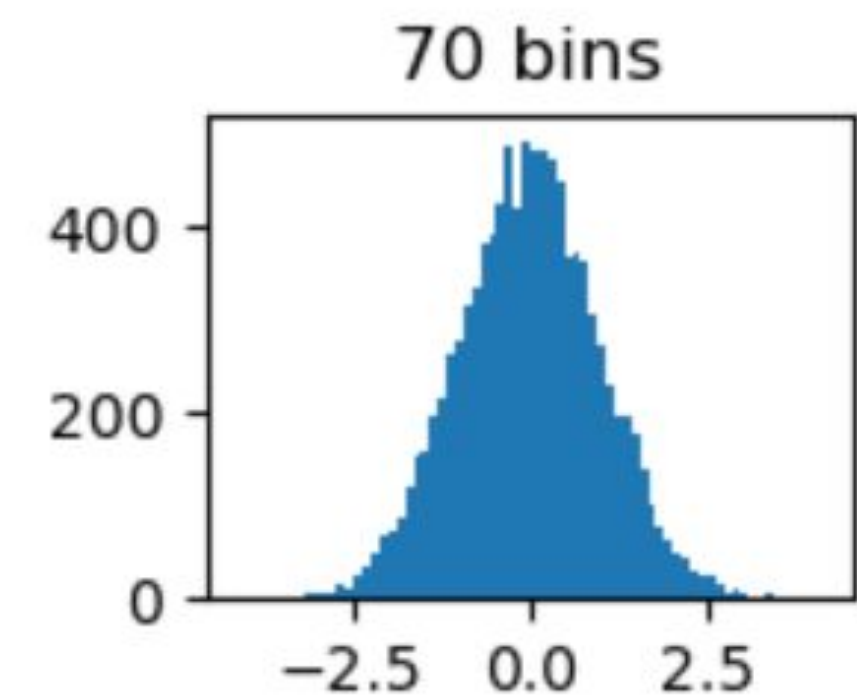
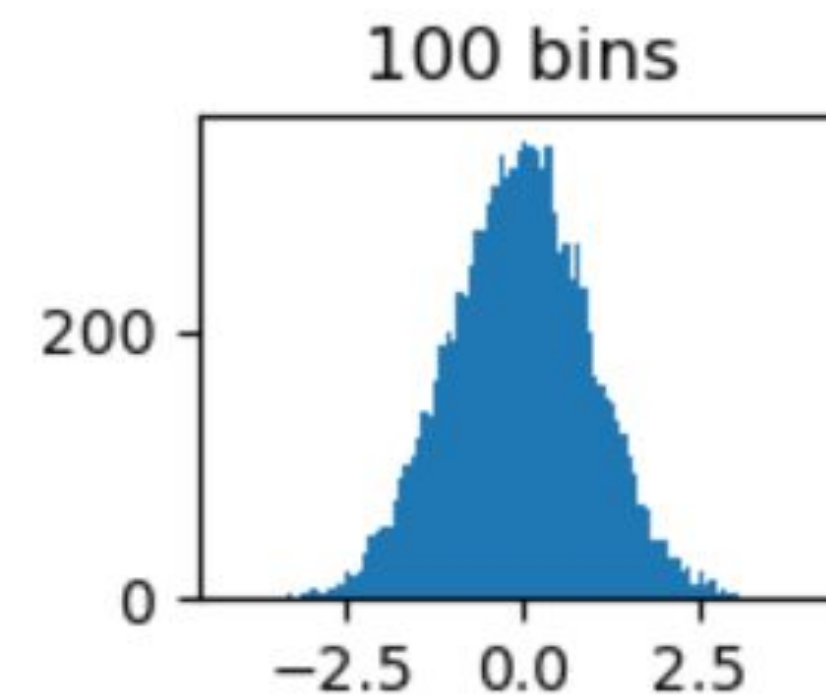
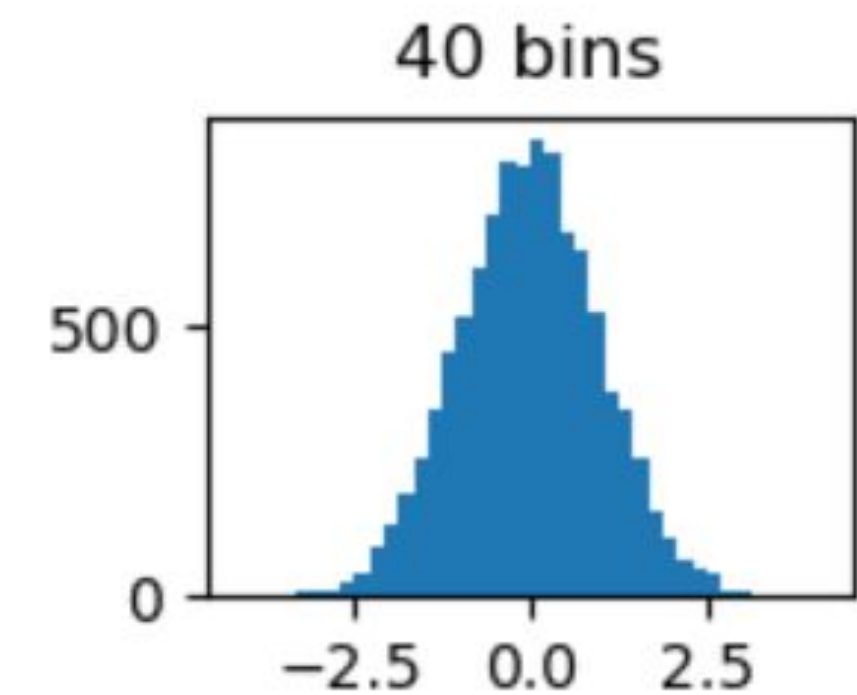
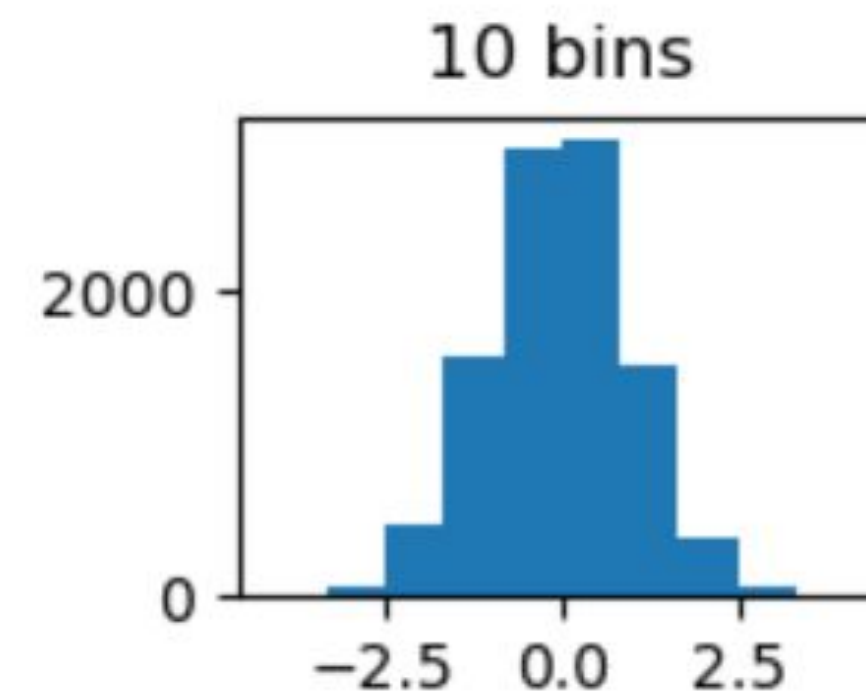
ax1 = fig.add_subplot(gs[0, 0])
ax1.hist(x, 10)
ax1.set_title('10 bins')

ax2 = fig.add_subplot(gs[0, 2])
ax2.hist(x, 40)
ax2.set_title('40 bins')

ax3 = fig.add_subplot(gs[2, 2])
ax3.hist(x, 70)
ax3.set_title('70 bins')

ax4 = fig.add_subplot(gs[2, 0])
ax4.hist(x, 100)
ax4.set_title('100 bins')

fig.savefig('fig5.png')
```



Artist Layer (Using GridSpec) Example 2

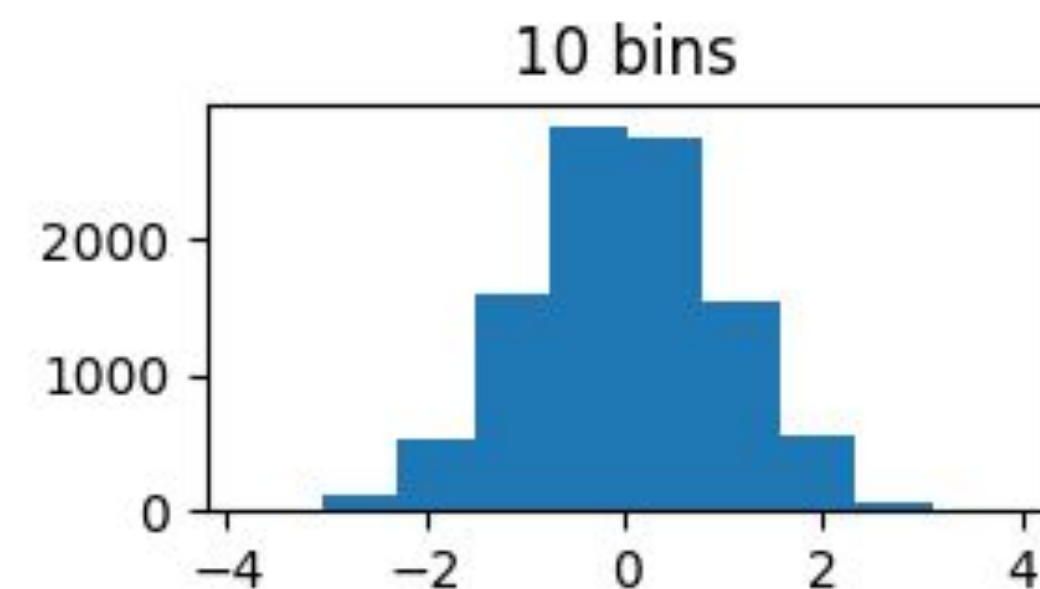
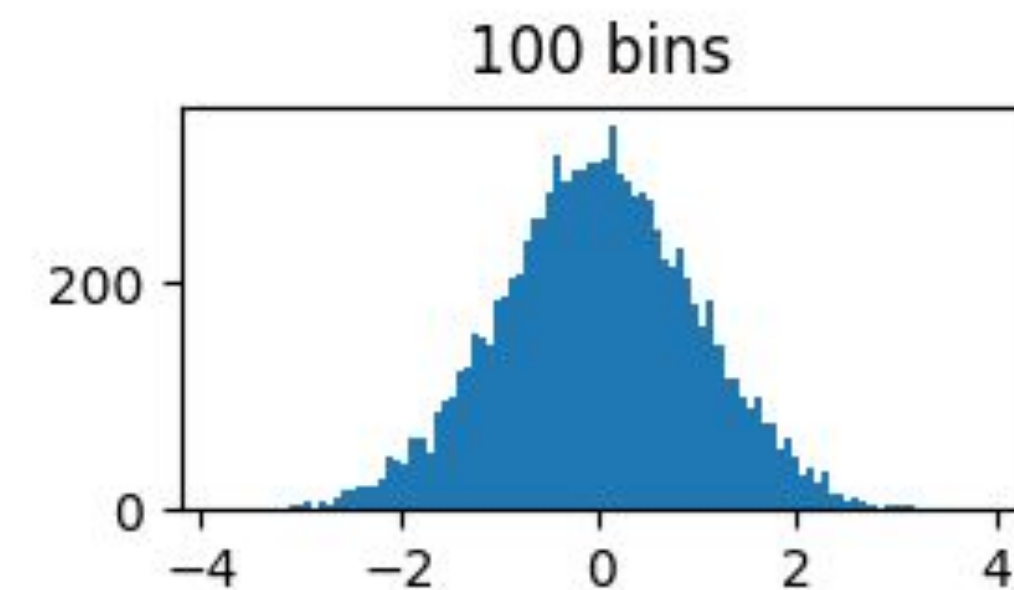
```
import numpy as np
x = np.random.randn(100000)

from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
fig = Figure()
canvas = FigureCanvas(fig)
gs = fig.add_gridspec(3, 2)

ax1 = fig.add_subplot(gs[0, 0])
ax1.hist(x, 100)
ax1.set_title('100 bins')

ax2 = fig.add_subplot(gs[2, 0])
ax2.hist(x, 10)
ax2.set_title('10 bins')

fig.savefig('2-axes.png')
```



Artist Layer (Using GridSpec) Example 3

```
import numpy as np
x = np.random.randn(10000)

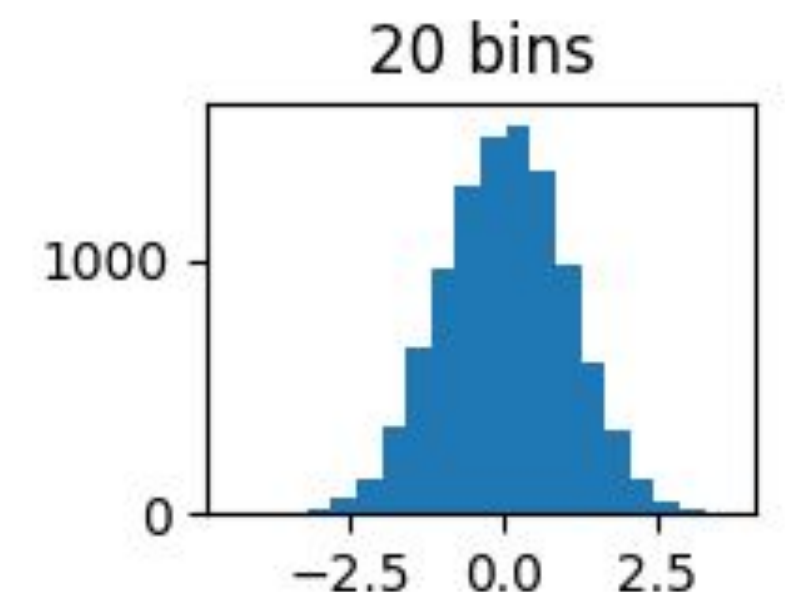
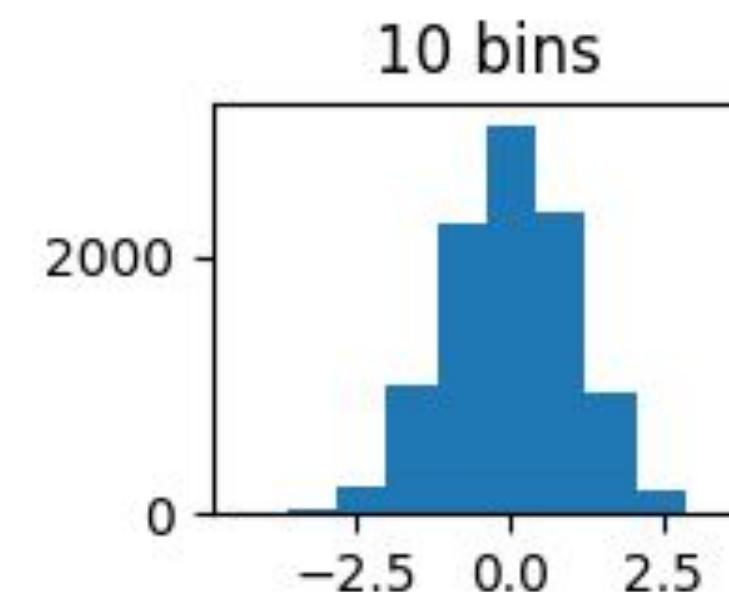
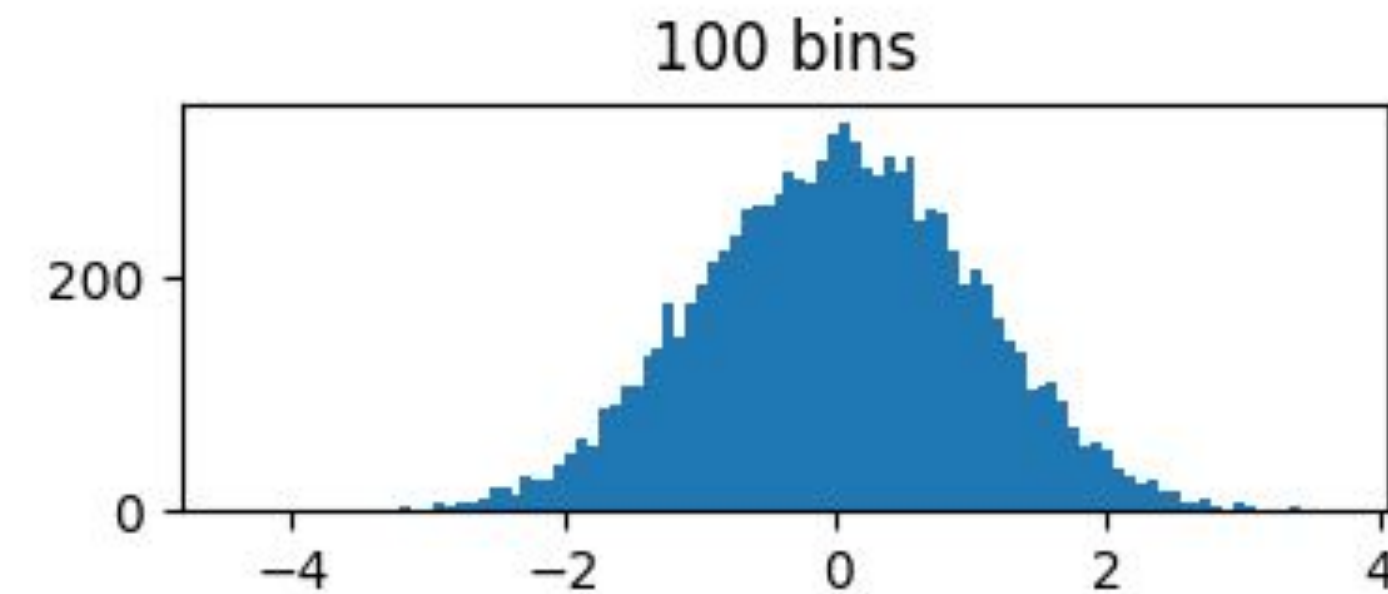
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
fig = Figure()
canvas = FigureCanvas(fig)
gs = fig.add_gridspec(3, 3)

ax1 = fig.add_subplot(gs[0, 0:2])
ax1.hist(x, 100)
ax1.set_title('100 bins')

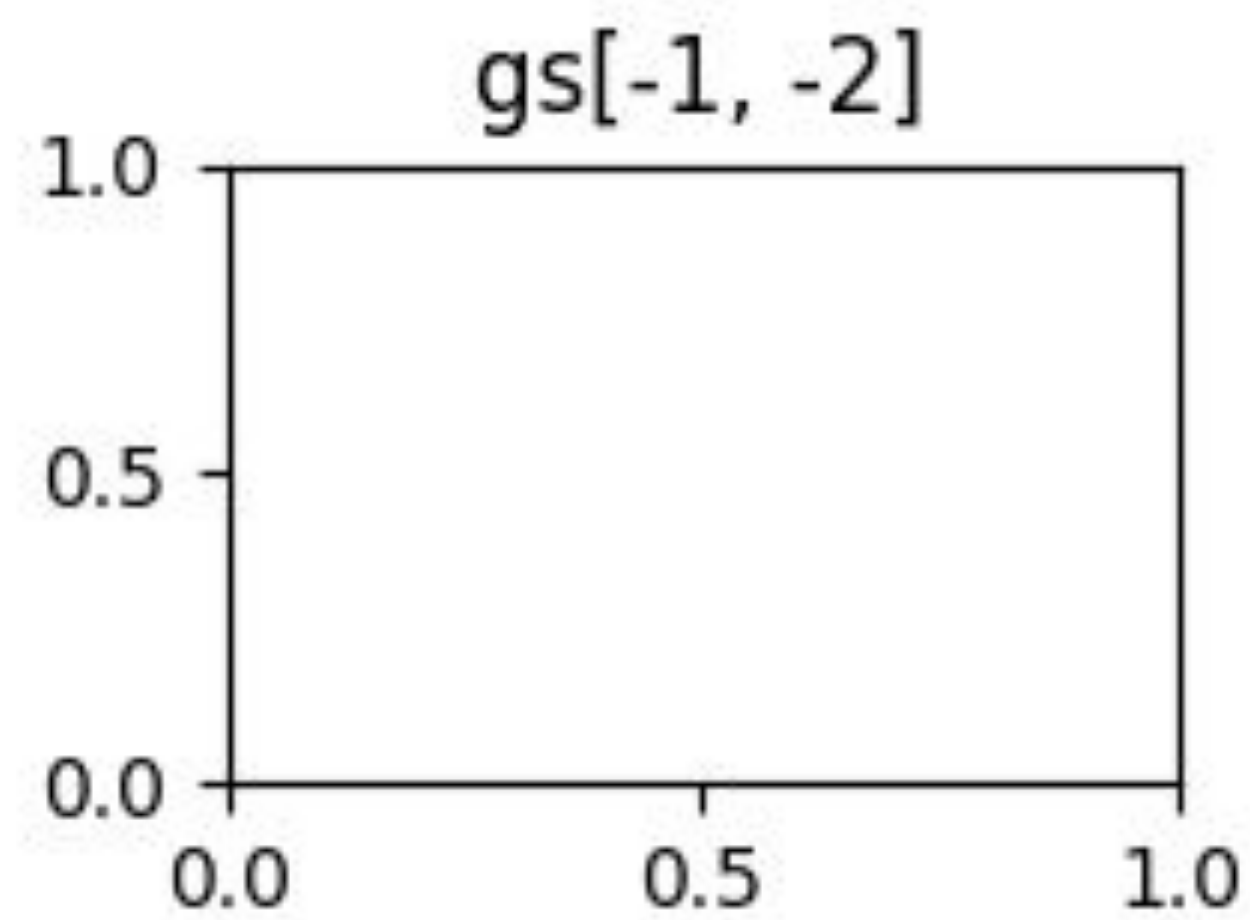
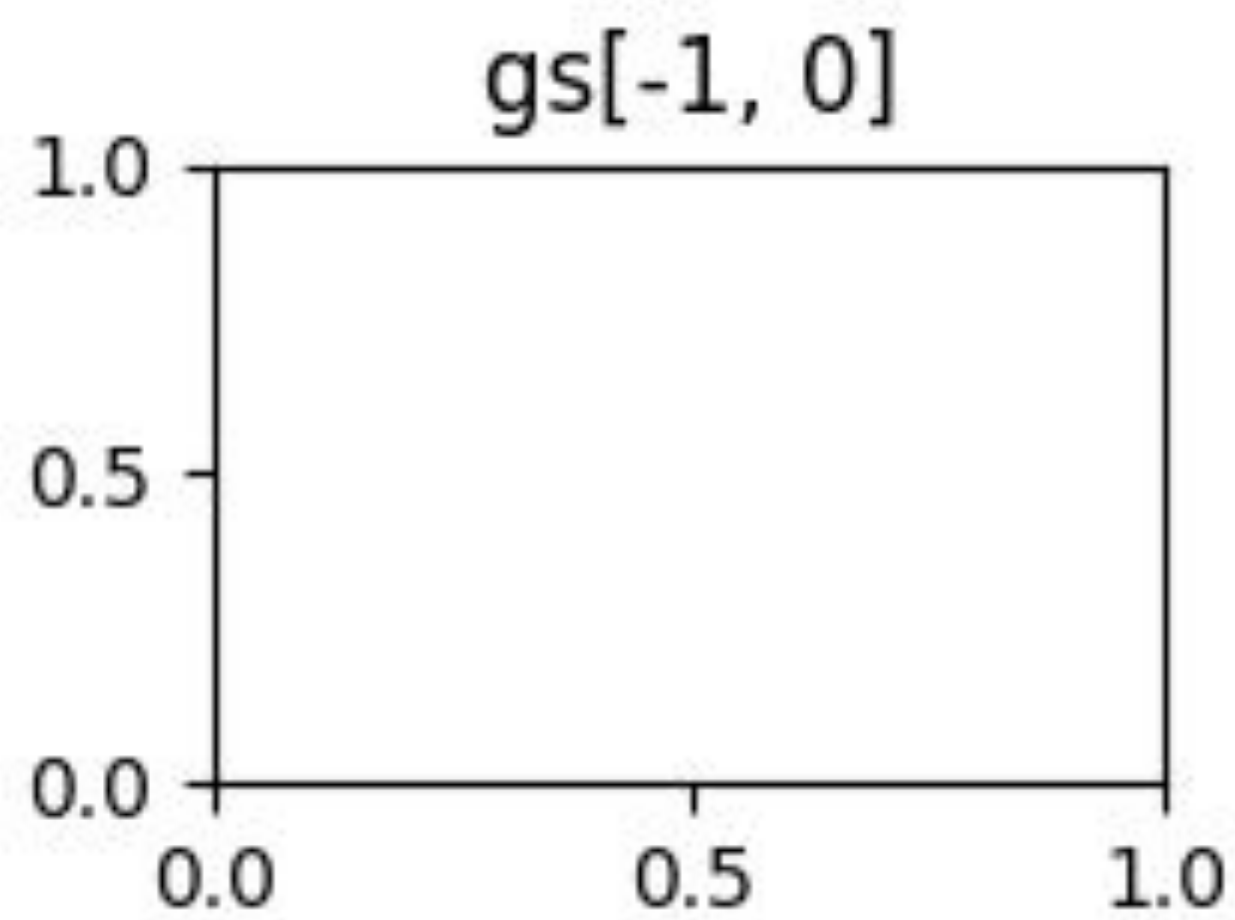
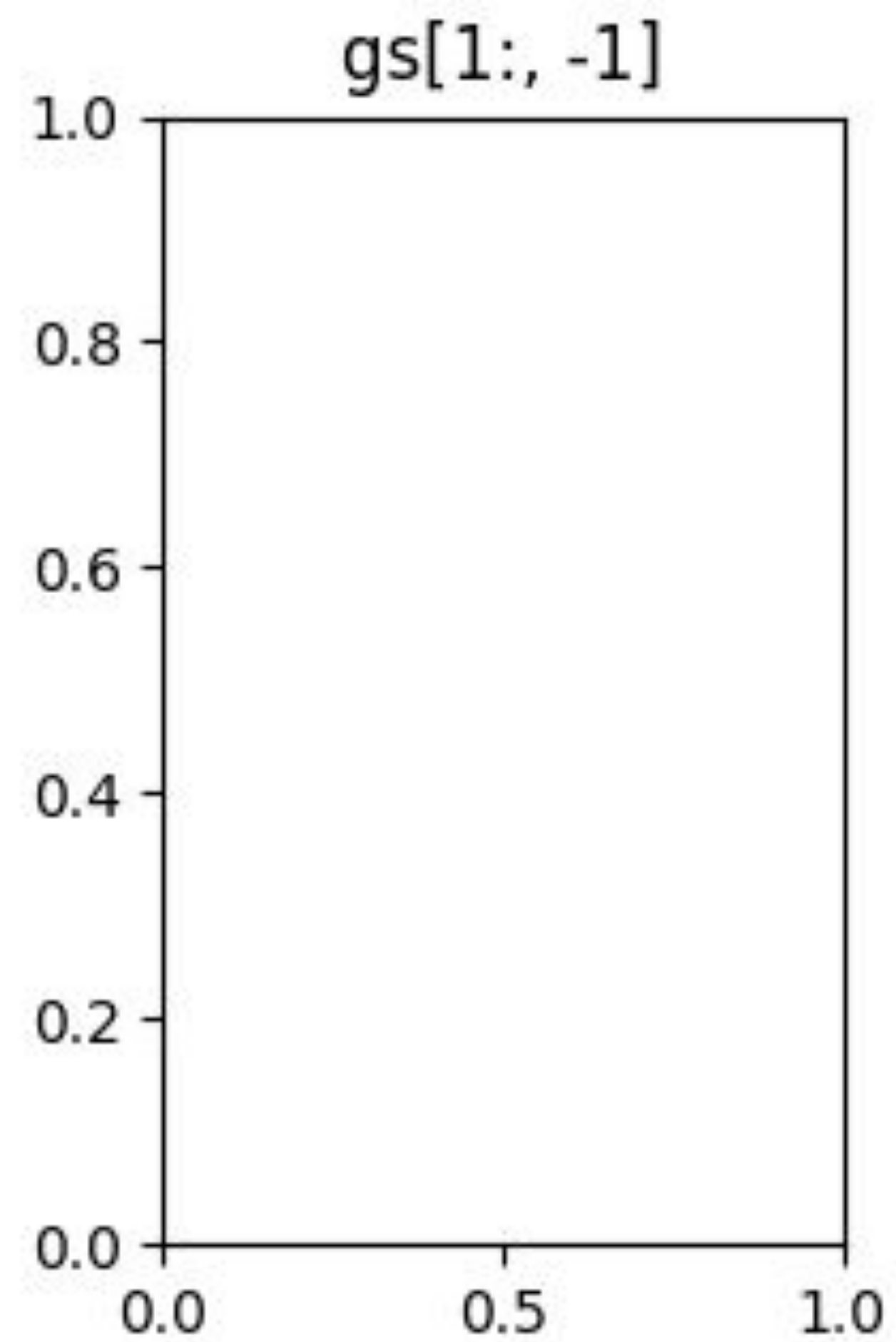
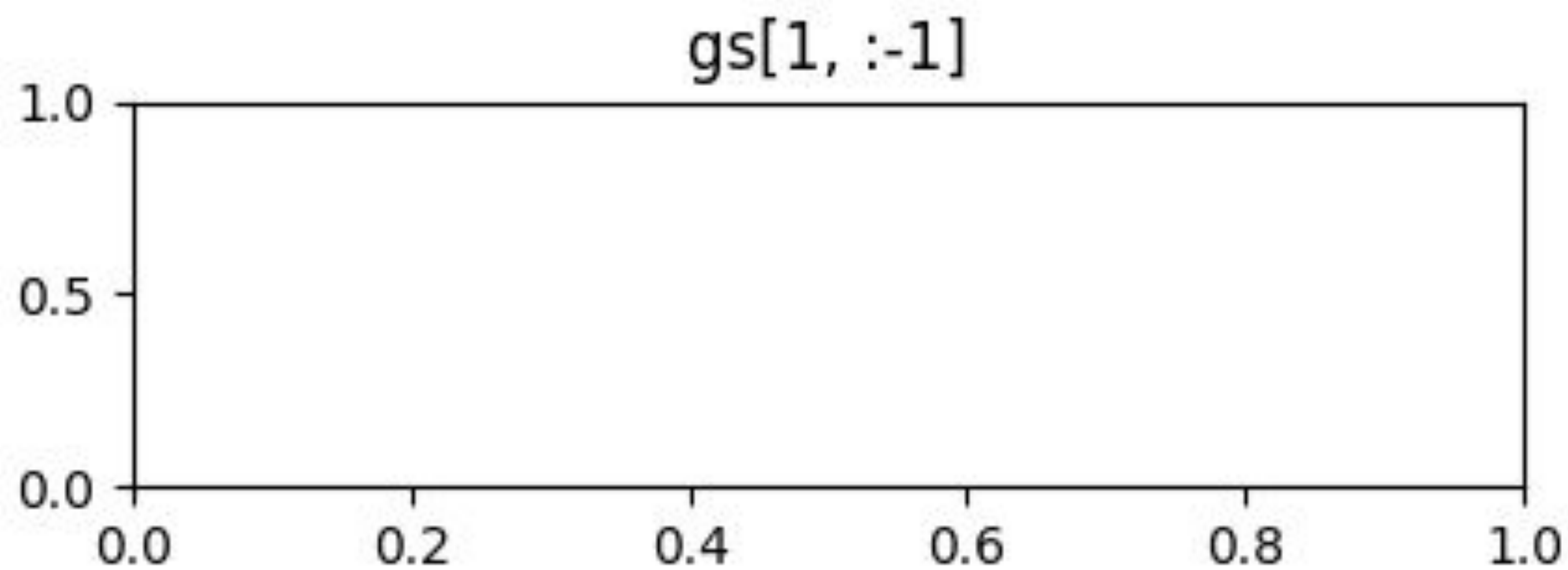
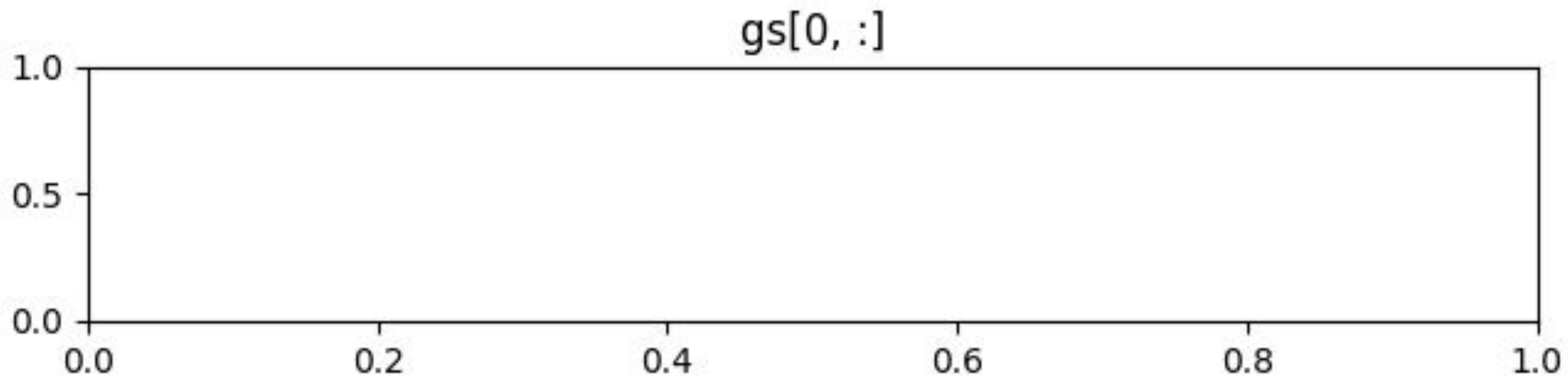
ax2 = fig.add_subplot(gs[2, 0])
ax2.hist(x, 10)
ax2.set_title('10 bins')


ax2 = fig.add_subplot(gs[2, -1])
ax2.hist(x, 20)
ax2.set_title('20 bins')

fig.savefig('span.png')
```



GridSpec





Questions

Links

<https://github.com/fcai-b/dv>

References

1. <https://www.coursera.org/learn/python-for-data-visualization>