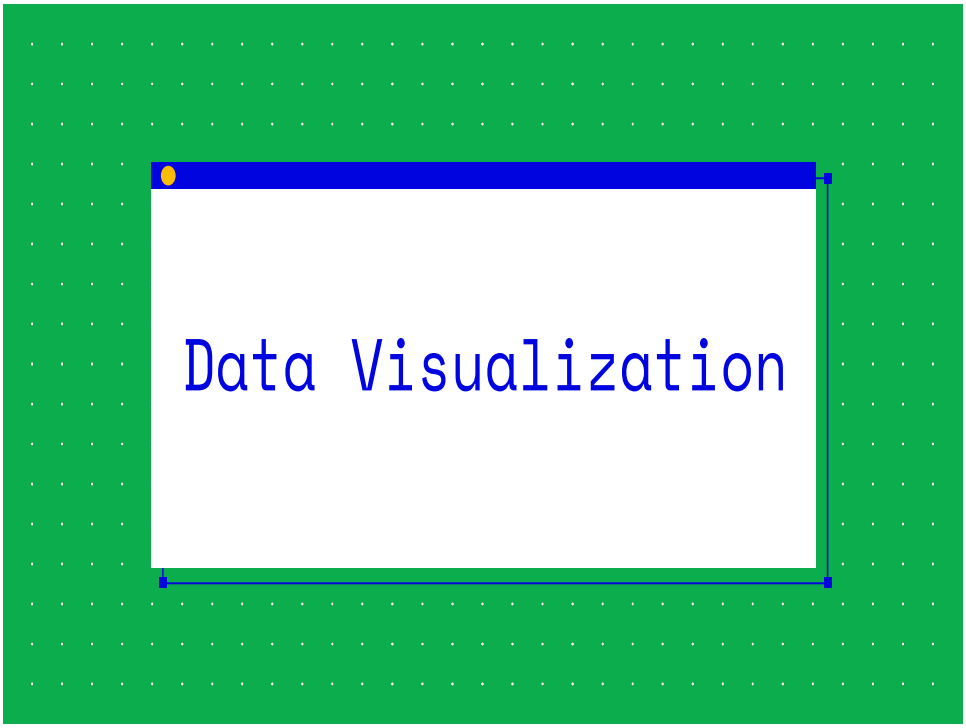

Data Visualization

Agenda:

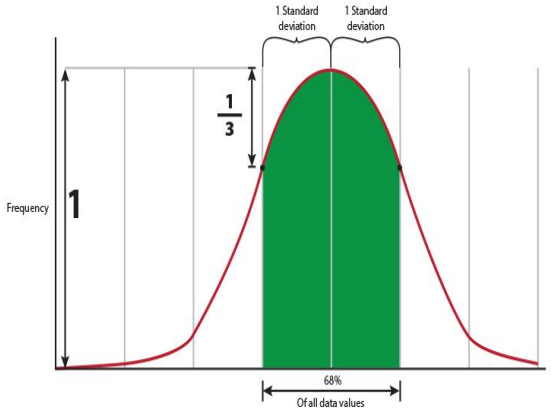
1. Normal Distributions Intro
2. Histogram
3. Matplotlib Artist Layer Examples
4. Questions

Here is a quick overview of what we'll cover today.

Normal Distributions Intro

Normal Distributions & 1 Standard Deviation

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

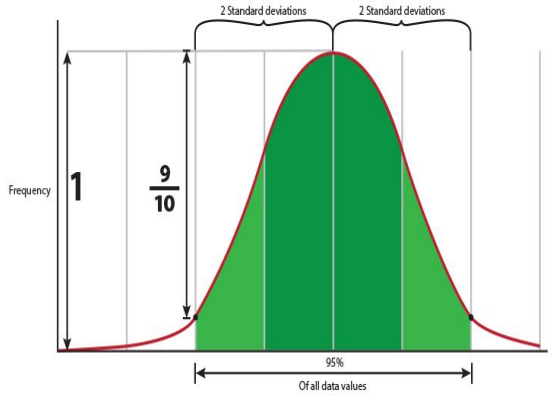


The **Normal Distribution** received its name because many early statisticians observed that numerous natural phenomena (such as human heights) tended to cluster around an average value and decrease symmetrically away from it.

- This pattern was considered the "**normal**" or **usual** distribution of data.
- While originally referred to as the "**Gaussian distribution**" (after Carl Friedrich Gauss) or the "**Laplace-Gaussian curve**,"
    - Sir Francis Galton popularized the term "**normal curve**" in the late 19th century, and
    - Karl Pearson later encouraged the use of "**normal distribution**" to avoid debates over who deserved priority (Laplace or Gauss) and to emphasize its frequent occurrence in data.

Normal Distributions & 2 Standard Deviation

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

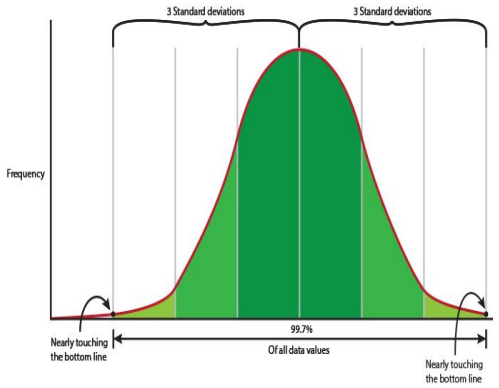


The term "**Standard Deviation**" was introduced by mathematician Karl Pearson in his 1893 paper.

- He coined it to replace older, more cumbersome names like "mean error" (used by Gauss).
- It effectively represents the **typical** or **average** distance a data point falls from the mean.
- The name is descriptive of
    - **Deviation**: It measures the spread or distance of data points from the **mean** of the dataset.
    - **Standard**: It is a **standardized** way to express this deviation, as it takes the square root of the variance to bring the measurement back into the original units of the data.
- **Example**: The Standard Deviation is 3. Therefore, on average, the data points in this set deviate from the mean by 3 units.

Normal Distributions & 3 Standard Deviation

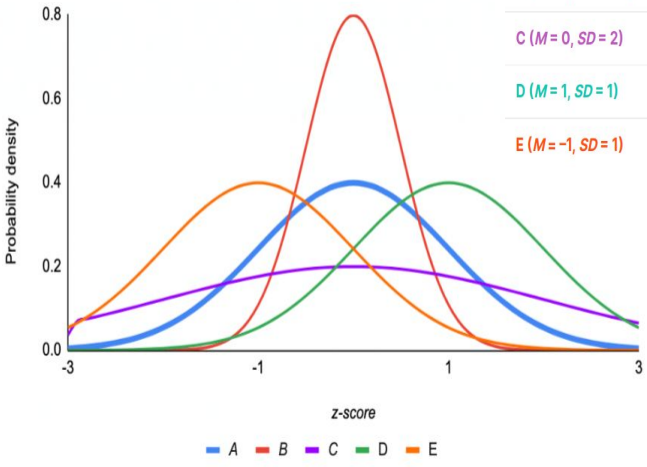
99.7% of data is **< 3** standard deviations away from the mean



The square of the deviation is calculated for two main reasons:

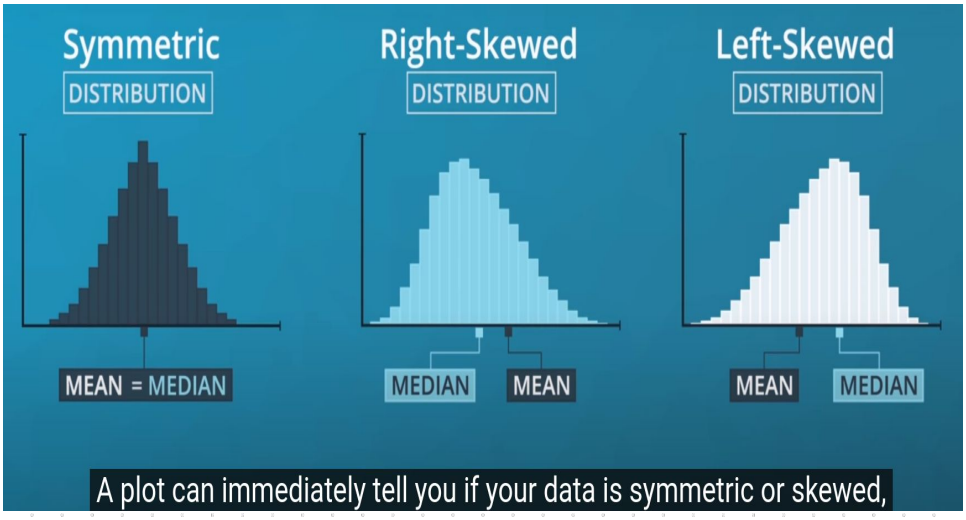
1. to eliminate negative values and, more importantly,
2. to give greater weight to larger deviations (emphasizing outliers and larger spread).
    - This feature makes the Standard Deviation highly sensitive to outliers or data points far from the mean.
    - It reflects a core principle: large errors or high variability are a greater concern than many small errors.
3. Squaring also results in a measure, called **variance**, that has better mathematical properties for advanced statistical analysis.

Standard Normal Distribution



A ($M = 0$, $SD = 1$)	Standard normal distribution
B ($M = 0$, $SD = 0.5$)	Squeezed, because $SD < 1$
C ($M = 0$, $SD = 2$)	Stretched, because $SD > 1$
D ($M = 1$, $SD = 1$)	Shifted right, because $M > 0$
E ($M = -1$, $SD = 1$)	Shifted left, because $M < 0$

Mean & Median



The **Median** is the middle value in a dataset when the values are arranged in ascending or descending order.

- If the dataset has an odd number of values, the median is the single value exactly in the middle.
- If the dataset has an even number of values, the median is the average (mean) of the two middle values.

Mode

The **Mode** of a data set is the value that appears most frequently in the set. A data set can have:

- One Mode (unimodal)
- Two Modes (bimodal)
- More than two Modes (multimodal)
- No Mode (if all values appear with the same frequency)

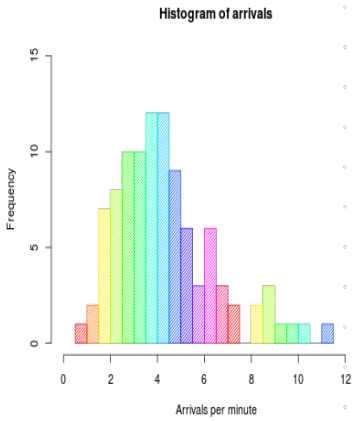
Histogram

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

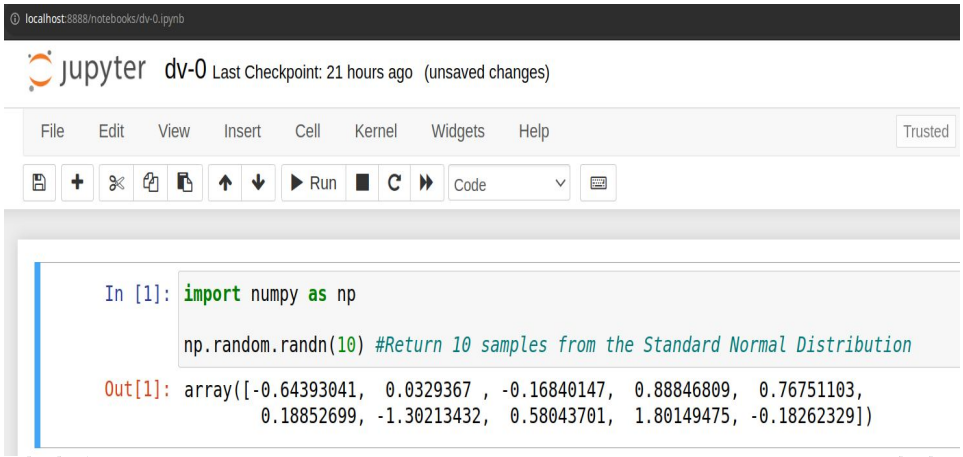
- `Histogram`
  - used for **continuous data**, where the bins represent ranges of data

- `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

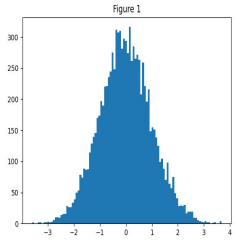


NumPy: Numerical Python library for **Scientific Computing**

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
  - https://github.dev/matplotlib/matplotlib/blob/main/lib/matplotlib/backends/backend_agg.py

- 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**

A **package** is a directory that logically groups related modules (Python files ending in .py).

- A **package** can has sub-packages.
- When a **package** is large, provides a rich set of functionalities, and is published for others to use (e.g., NumPy, Pandas), it is typically referred to as a "**library**."

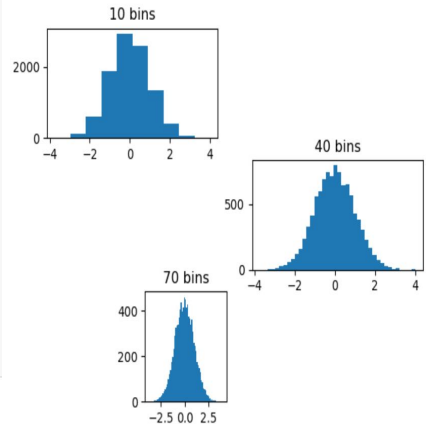
Backends

- Matplotlib is a plotting library. It relies on some backend to actually render the plots.
  - https://github.dev/matplotlib/matplotlib/blob/main/lib/matplotlib/backend_bases.py
- Matplotlib supports a variety of backends, including
  - interactive backends (e.g., TkAgg, GTKAgg, Qt5Agg, WxAgg) and

    - non-interactive backends for generating image files (e.g., Agg, SVG, PDF, PS)
- The default backend is the agg backend.

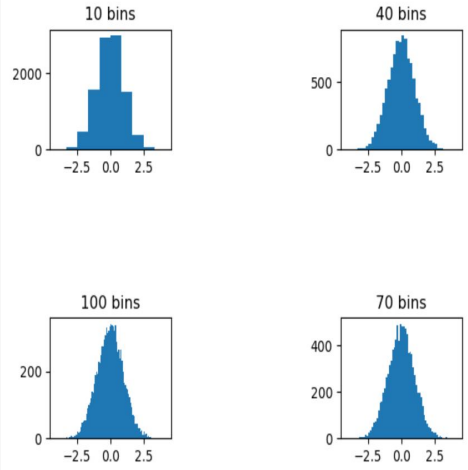
Artist Layer Example 2

```
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')
```



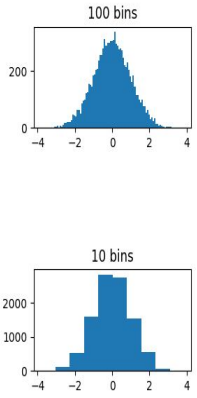
Artist Layer (Using GridSpec) Example 1

```
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')
```



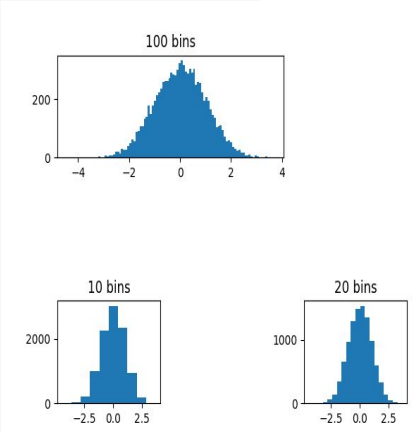
Artist Layer (Using GridSpec) Example 2

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

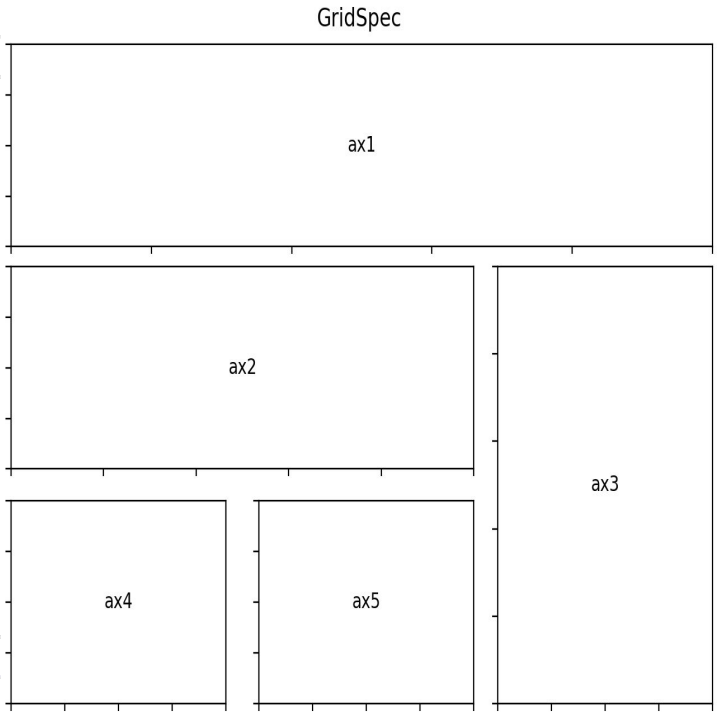


Artist Layer (Using GridSpec) Example 3

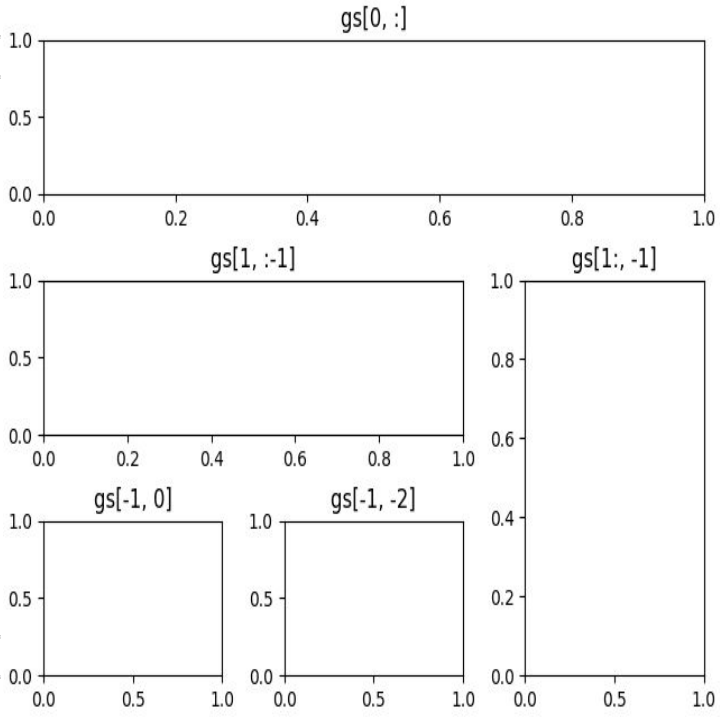
```
import numpy as np
x = np.random.randn(10000)

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')
```

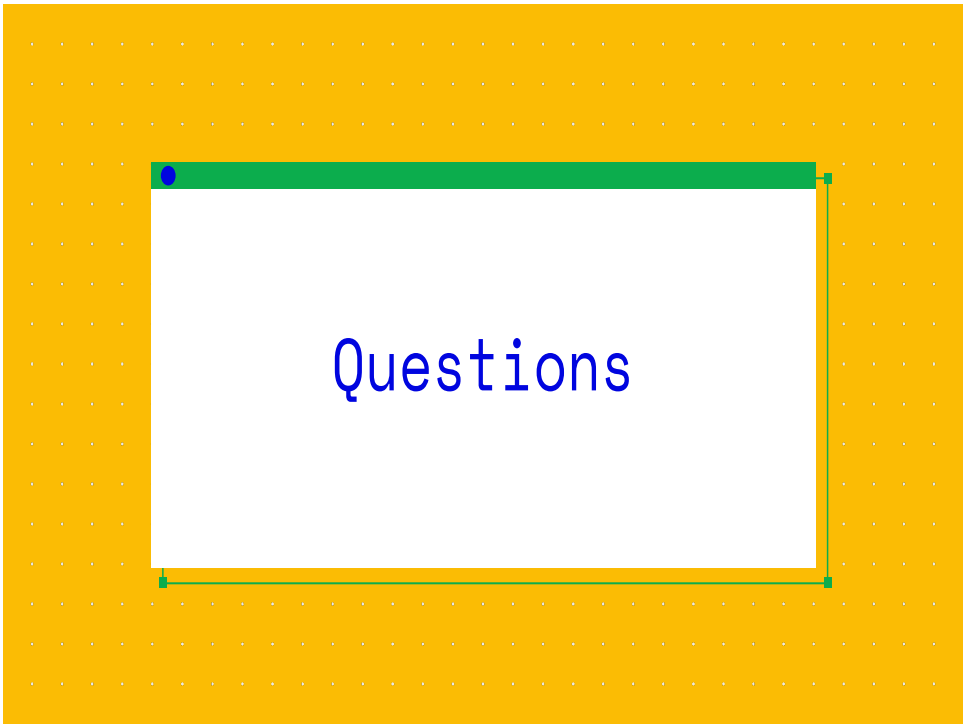


GridSpec Question



GridSpec Solution




Questions

Links

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

References

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