1823766

slido

Join at slido.com
#1823766

Click **Present with Slido** or install our Chrome extension to display joining instructions for participants while presenting.

LECTURE 7

Visualization I

Visualizing distributions and KDEs

Data 100/Data 200, Spring 2025 @ UC Berkeley

Narges Norouzi and Josh Grossman

Content credit: [Acknowledgments]

Goals for this Lecture

Lecture 7, Data 100 Spring 2025

1823766

Understand the theories behind effective visualizations and start to generate plots of our own

- The necessary "pre-thinking" before creating a plot
- Python libraries for visualizing data

Agenda

Lecture 7, Data 100 Spring 2025

- Visualization
  - Goals of visualization
  - Visualizing distributions
  - Kernel density estimation

1823766

Goals of Visualization

Lecture 7, Data 100 Spring 2025

- **Visualization**
  - **Goals of visualization**
  - Visualizing distributions
  - Kernel density estimation

Where Are We?

1823766

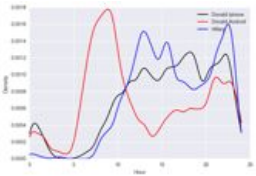
?

Question & Problem Formulation

Data Acquisition

Prediction and Inference

Exploratory Data Analysis



Reports, Decisions, and Solutions

(today)

Data Wrangling Intro to EDA	Working with Text Data Regular Expressions	Plots and variables Seaborn KDE	Viz principles Transformations

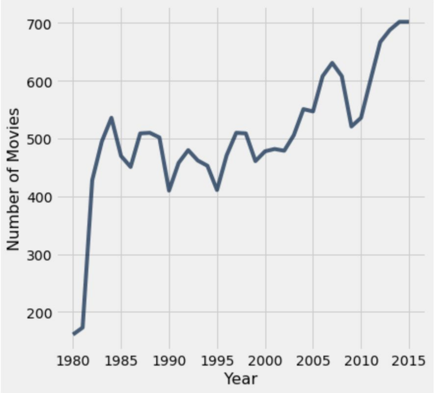
(Part I: Processing Data) (Part II: Visualizing and Reporting Data)

Visualizations in Data 8 (and Data 100, so far)

1823766

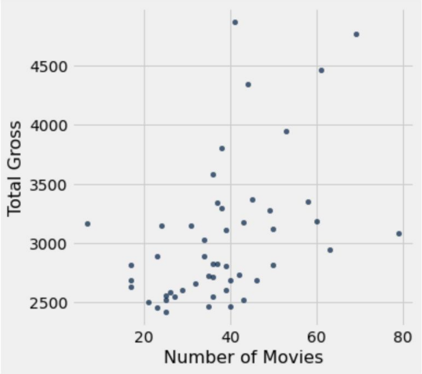
You worked with many types of visualizations throughout Data 8.

```python
# The call is
# t.plot(x_label, y_label)

movies.plot('Year', 'Number of Movies')
```

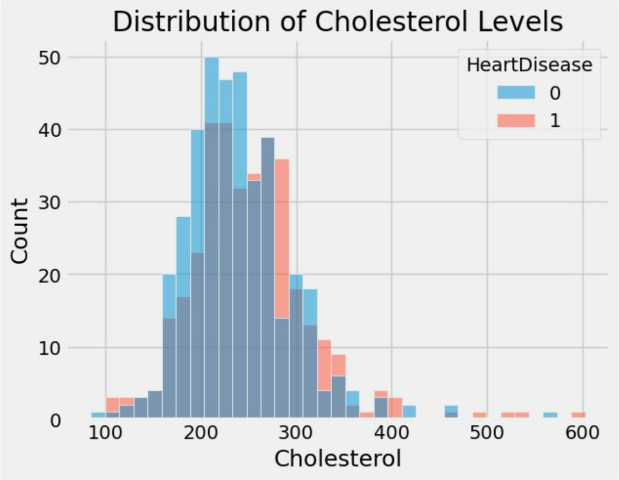


Line plot

```python
# The call is
# t.scatter(x_label, y_label)

actors.scatter('Number of Movies', 'Total Gross')
```



Scatter plot



Histogram from Homework #1

What did these achieve?

- Provide a high-level overview of a complex dataset.
- Communicated trends to viewers.

Goals of Data Visualization

1823766

Goal 1: To **help your own understanding** of your data/results.

- Key part of exploratory data analysis.
- Summarize trends visually before in-depth analysis.
- Lightweight, iterative and flexible.

Goal 2: To **communicate results/conclusions to others**.

- Highly editorial and selective.
- Be thoughtful and careful!
- Fine-tuned to achieve a communications goal.
- Considerations: clarity, accessibility, and necessary context.

What do these goals imply?

Visualizations aren't a matter of making "pretty" pictures.

We need to do a lot of thinking about what stylistic choices communicate ideas most effectively.

1823766

What do these goals imply?

Visualizations aren't a matter of making "pretty" pictures.

We need to do a lot of thinking about what stylistic choices communicate ideas most effectively.

First half of visualization topics in Data 100:
Choosing the "right" plot for

- Introducing plots for different variable types
- Generating these plots through code

Second half of visualization topics in Data 100:
Stylizing plots appropriately

- Smoothing and transforming visual data
- Providing context through labeling and color

Visualizing Distributions

Lecture 7, Data 100 Spring 2025

1823766

- **Visualization**
  - Goals of visualization
  - **Visualizing distributions**
  - Kernel density estimation

Distributions

1823766

A distribution describes...

- The set of values that a variable can possibly take.
- The frequency with which each value occurs.

...for a **single** variable

Example: Distribution of faculty to different departments at Cal.

- The list of departments at Cal.
- The number of faculty in each department.

In other words: How is the variable distributed across all of its possible values?

This means that percentages **should sum to 100%** (if using proportions) and counts should **sum to the total number of datapoints** (if using raw counts).

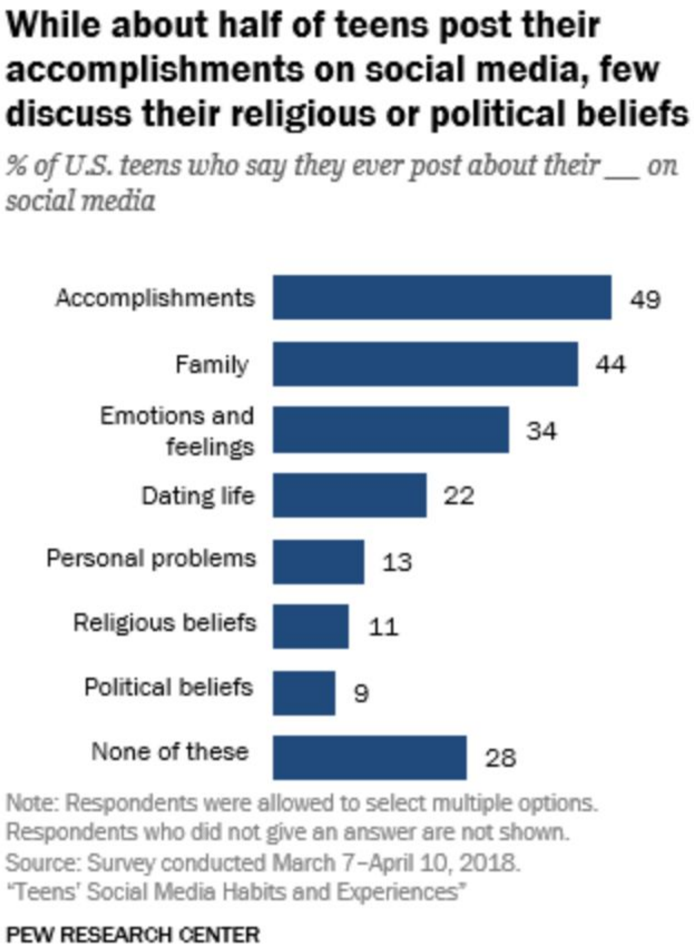
Let's see some examples.

slido

1823766

Does this chart show a distribution?

Click **Present with Slido** or install our [Chrome extension] to activate this poll while presenting.



1823766

Does this chart show a distribution?

No.

- The chart does show percents of individuals in different categories!
- But, this is not a distribution because individuals can be in more than one category (see the fine print).

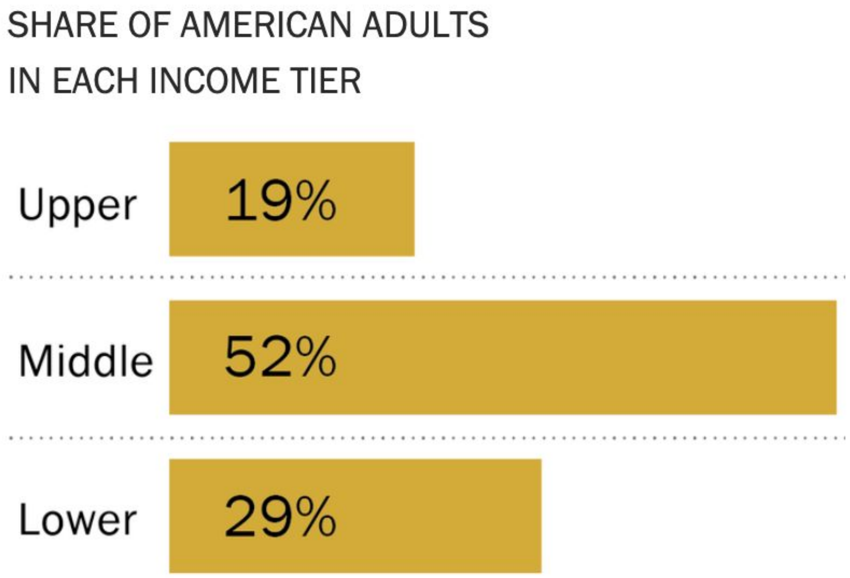
1823766

slido

Does this chart show a distribution?

Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

Does this chart show a distribution?



1823766

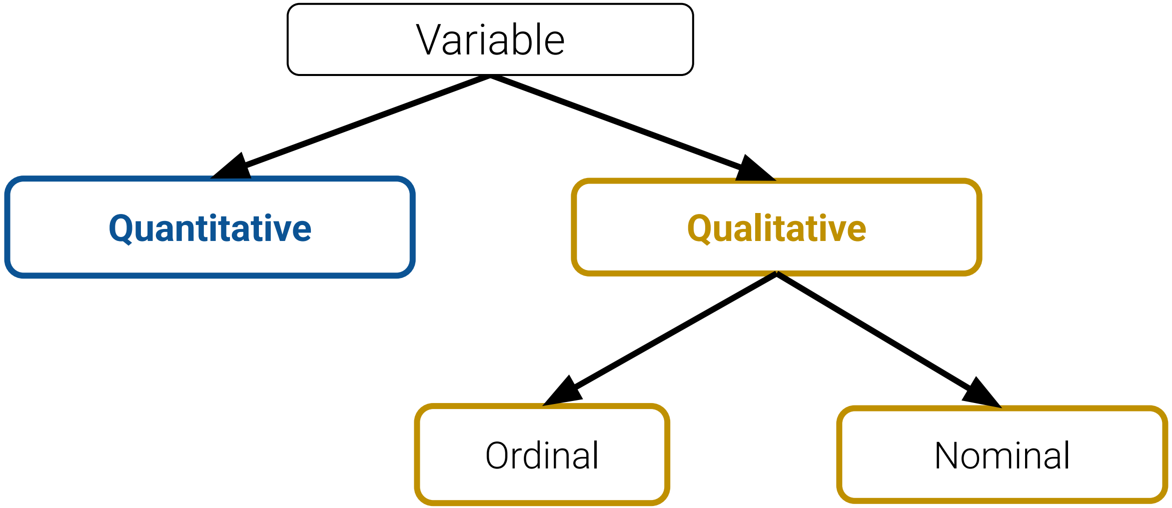
Yes!

- This chart shows the distribution of the qualitative ordinal variable "income tier."
- Each individual is in exactly one category.
- The values we see are the proportions of individuals in that category.
- Everyone is represented, as the total percentage is 100%.

Variable Types Should Inform Plot Choice

1823766

Different plots are more or less suited for displaying particular types of variables.



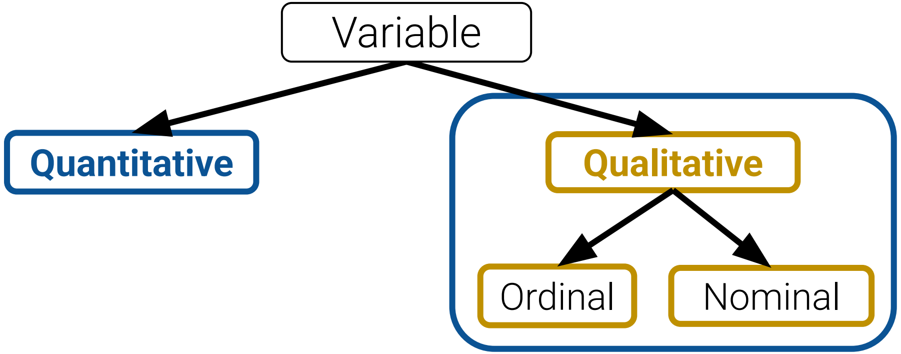
First step of visualization: Identify the variables being visualized. Then, select a plot type accordingly.

Bar Plots: Distributions of Qualitative Variables

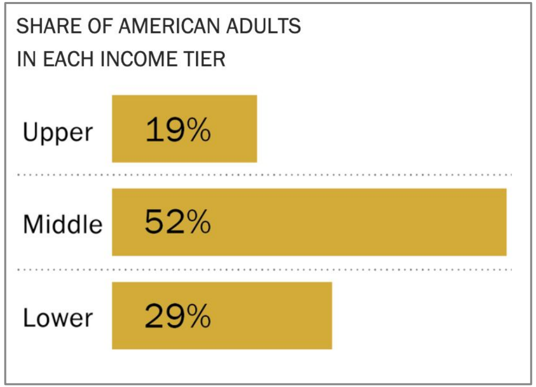
1823766

Bar plots are the most common way of displaying the **distribution** of a **qualitative** variable.

*Sometimes quantitative discrete data too, if there are few unique values.



- For example, the proportion of adults in the upper, middle, and lower classes.
- Lengths encode values.
  - *Widths* encode *nothing*!
  - *Color* could indicate a sub-category (but not necessarily).



World Bank Dataset

1823766

We will be using the `wb` dataset about world countries for most of our work today.

	Continent	Country	Primary completion rate: Male: % of relevant age group: 2015	Primary completion rate: Female: % of relevant age group: 2015	Lower secondary completion rate: Male: % of relevant age group: 2015	Lower secondary completion rate: Female: % of relevant age group: 2015	Youth literacy rate: Male: % of ages 15-24: 2005-14	Youth literacy rate: Female: % of ages 15-24: 2005-14	Adult literacy rate: Male: % ages 15 and older: 2005-14	Adult literacy rate: Female: % ages 15 and older: 2005-14
0	Africa	Algeria	106.0	105.0	68.0	85.0	96.0	92.0	83.0	68.0
1	Africa	Angola	NaN	NaN	NaN	NaN	79.0	67.0	82.0	60.0
2	Africa	Benin	83.0	73.0	50.0	37.0	55.0	31.0	41.0	18.0
3	Africa	Botswana	98.0	101.0	86.0	87.0	96.0	99.0	87.0	89.0
5	Africa	Burundi	58.0	66.0	35.0	30.0	90.0	88.0	89.0	85.0

Generating Bar Plots: Matplotlib

1823766

In Data 100, we will mainly use two libraries for generating plots: [Matplotlib]() and [Seaborn]().

Most `matplotlib` plotting functions follow the same structure: We pass in a sequence (`list`, `array`, or `series`) of values to be plotted on the x-axis, and a second sequence of values to be plotted on the y-axis.

```
import matplotlib.pyplot as plt
plt.plotting_function(x_values, y_values)
```

`matplotlib` is typically given the alias `plt`

To add labels and a title:

```
plt.xlabel("x axis label")
plt.ylabel("y axis label")
plt.title("Title of the plot");
```

Generating Bar Plots: `matplotlib`

1823766

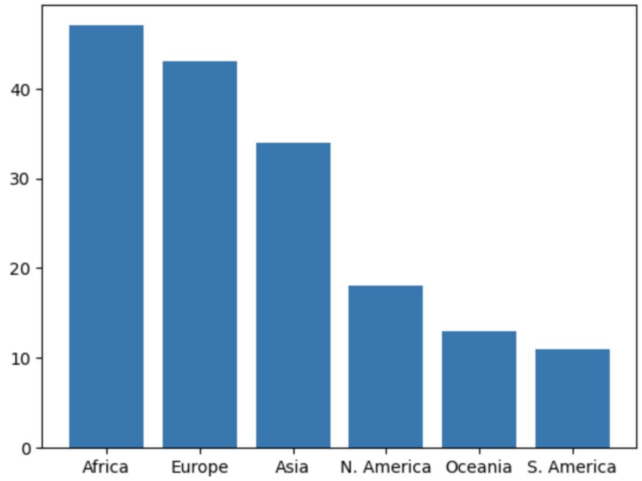
To create a bar plot in **matplotlib**: `plt.bar( )`
[Documentation]

```
continents = wb["Continent"].value_counts()
```

```
Africa        47
Europe        43
Asia          34
N. America    18
Oceania       13
S. America    11
Name: Continent, dtype: int64
```

```
plt.bar(continents.index, continents.values);
```

$\underbrace{\texttt{continents.index}}_{\text{x values}}$, $\underbrace{\texttt{continents.values}}_{\text{y values}}$



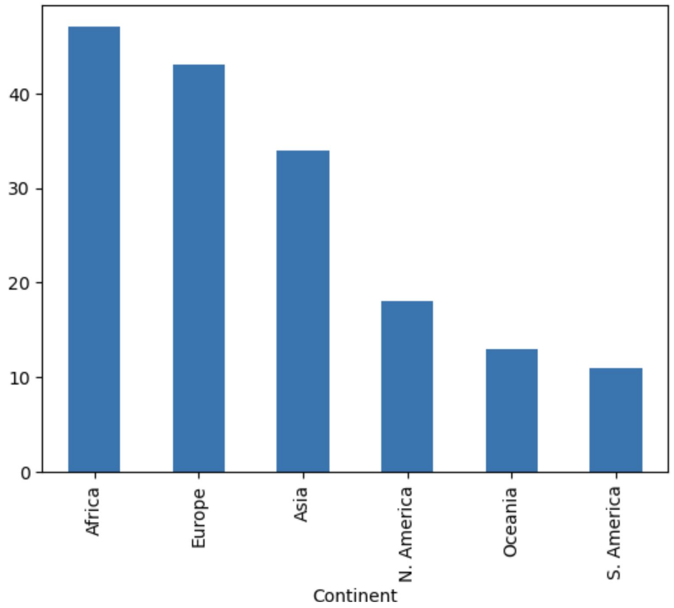
Generating Bar Plots: `pandas` Native Plotting

1823766

To create a bar plot in native **pandas**: `.plot(kind='bar')`

```
Africa        47
Europe        43
Asia          34
N. America    18
Oceania       13
S. America    11
Name: Continent, dtype: int64
```

```
wb["Continent"].value_counts().plot(kind='bar')
```



Generating Bar Plots: `seaborn`

1823766

Seaborn plotting functions use a different structure: Pass in an entire **`DataFrame`**, then specify what column(s) to plot.

Seaborn is typically given the alias **`sns`**

```
import seaborn as sns
sns.plotting_function(data=df, x="x_col", y="y_col")
```

To add labels and a title, use the same syntax as before:

```
plt.xlabel("x axis label")
plt.ylabel("y axis label")
plt.title("Title of the plot");
```

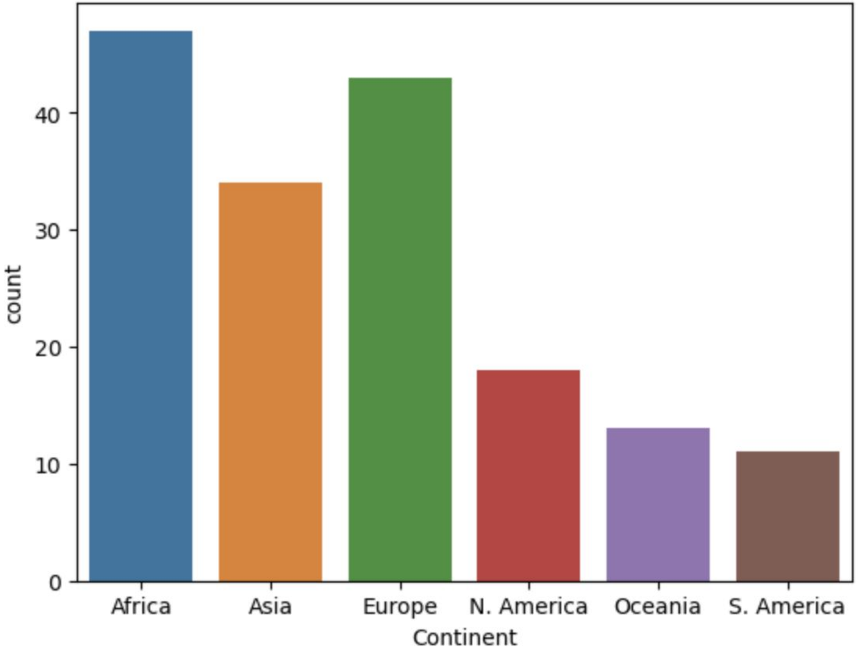
`seaborn` *is built on* `matplotlib`*! When using* `seaborn`*, you're actually using* `matplotlib` *under the hood, but with an easier-to-use interface for working with* ***`DataFrame`****s and creating certain types of plots.*

Generating Bar Plots: `seaborn`

1823766

To create a bar plot in `seaborn`: `sns.countplot( )`

[Documentation]



`countplot` operates at a higher level of abstraction!

You give it the entire `DataFrame` and it does the counting for you.

```python
import seaborn as sns

sns.countplot(data=wb, x="Continent");
```

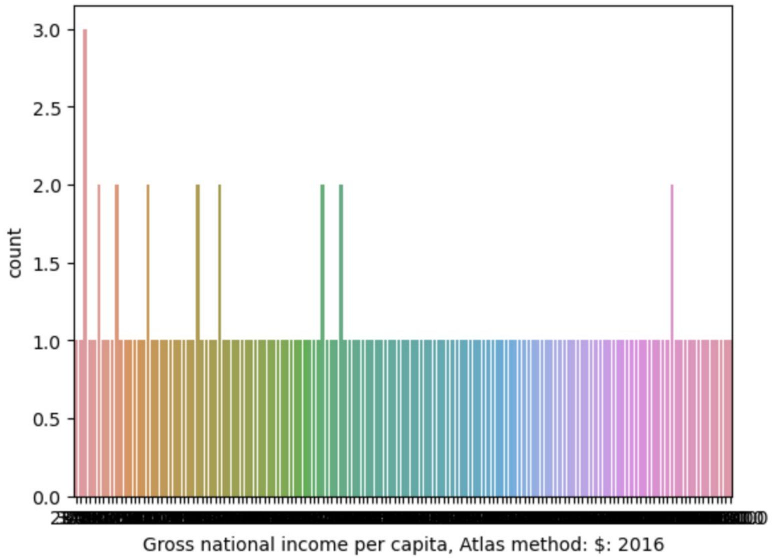
Distributions of Quantitative Variables

1823766

Earlier, we said that bar plots are appropriate for distributions of qualitative variables.

Why only qualitative? Why not quantitative as well?

- For example: The distribution of gross national income per capita.

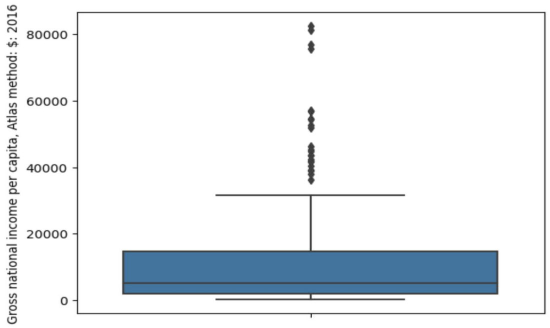


A bar plot will create a separate bar for each unique value. This leads to too many bars for continuous data!

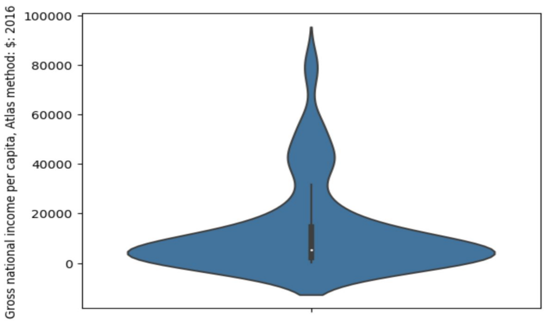
Distributions of Quantitative Variables

1823766

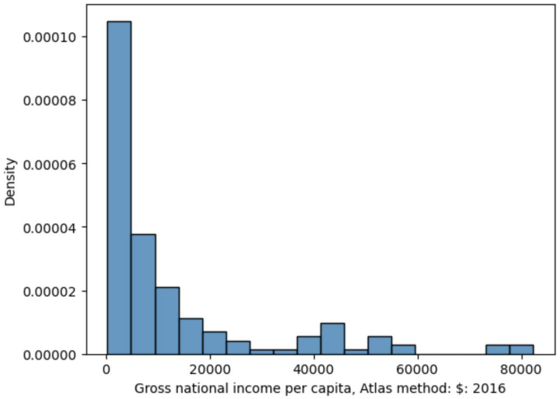
To visualize the distribution of a continuous quantitative variable:



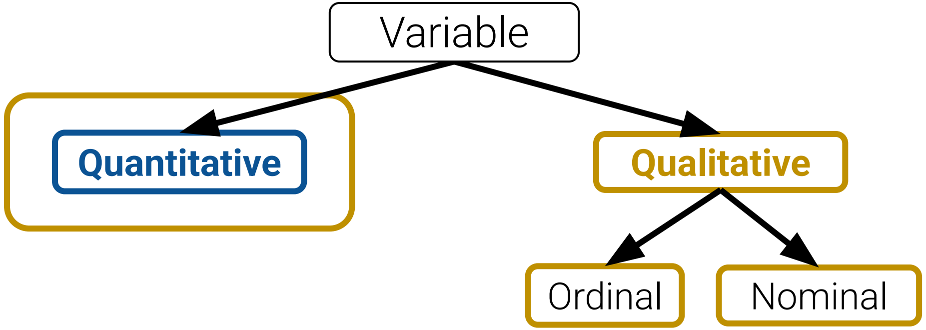
Box plot



Violin plot



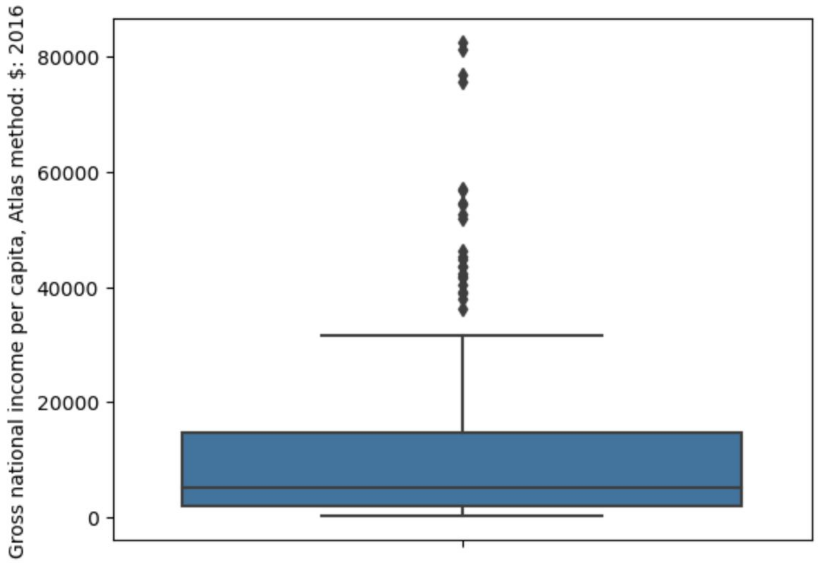
Histogram



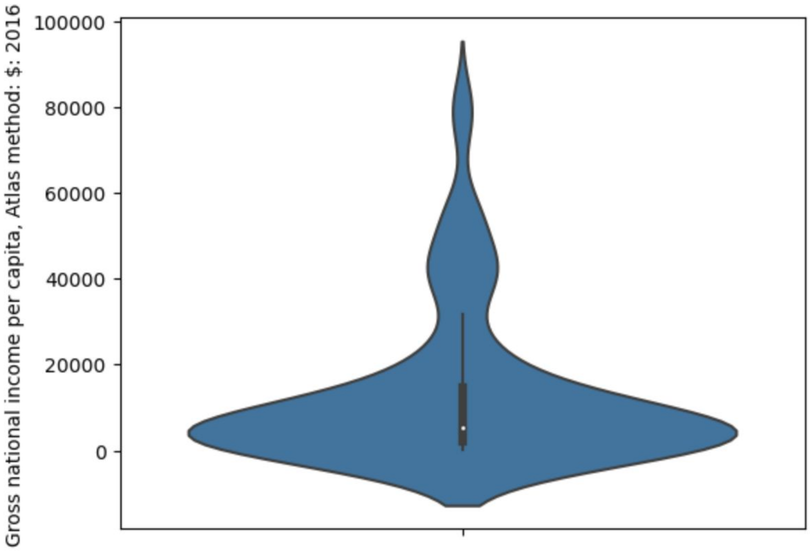
Box plots and Violin Plots

Box plots and violin plots display distributions using information about **quartiles**.

- In a box plot, the width of the box encodes no meaning.
- In a violin plot, the width of the "violin" indicates the density of datapoints at each value.



`sns.boxplot(data=df, y="y_variable");`
[Documentation]



`sns.violinplot(data=df, y ="y_variable");`
[Documentation]

Quartiles

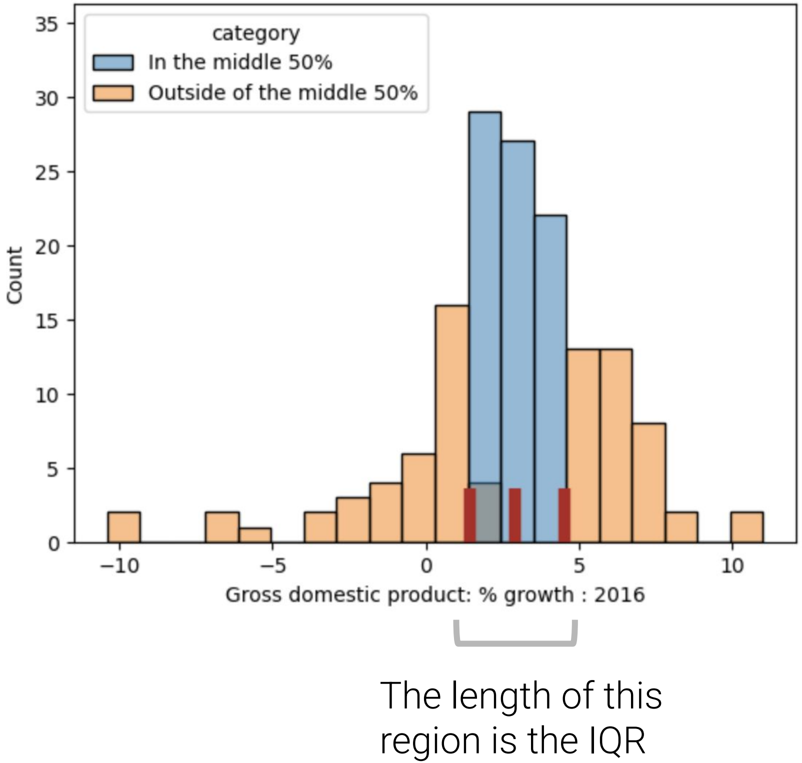
For a quantitative variable:

- First or lower quartile: 25th percentile.
- Second quartile: 50th percentile (median).
- Third or upper quartile: 75th percentile.

The interval [first quartile, third quartile] contains the "middle 50%" of the data.

Interquartile range (IQR) measures spread.

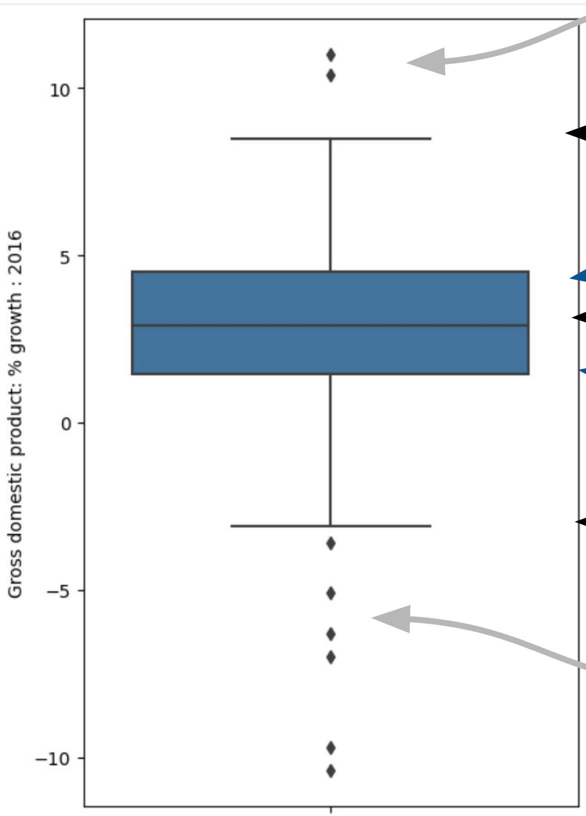
- IQR = third quartile – first quartile.



Box Plots

1823766

Why an outlier? [link]



Outliers

Whisker: upper quartile + 1.5*IQR

Third quartile (75th percentile)

Second quartile (median)

First quartile (25th percentile)

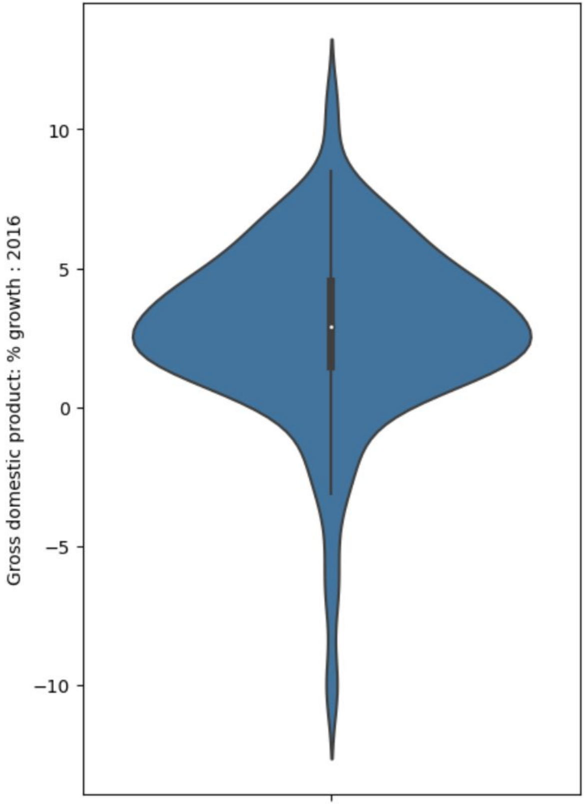
Whisker: lower quartile - 1.5*IQR

Outliers

```
sns.boxplot(data=wb, y="Gross domestic product: % growth : 2016")
```

Violin Plots

1823766



Violin plots are similar to box plots, but also show smoothed density curves.

- The "width" of our "box" now has meaning!
- The three quartiles and "whiskers" are still present – look closely.

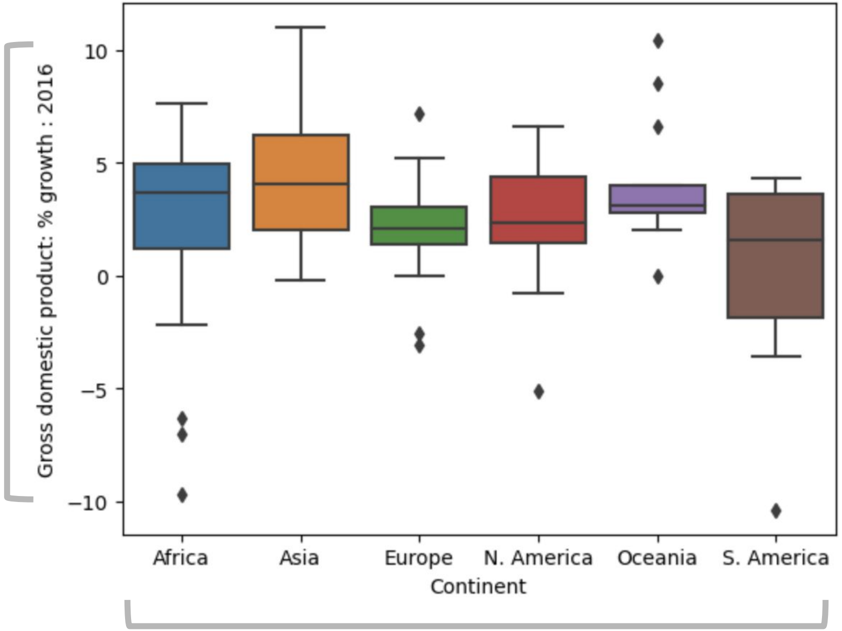
```
sns.violinplot(data=wb, y="Gross domestic product: % growth : 2016")
```

Side-by-side Box and Violin Plots

What if we wanted to incorporate a *qualitative* variable as well? For example, compare the distribution of a quantitative continuous variable *across* different qualitative categories.

```
sns.boxplot(data=wb, x="Continent", y="Gross domestic product: % growth : 2016");
```

GDP growth:
quantitative continuous

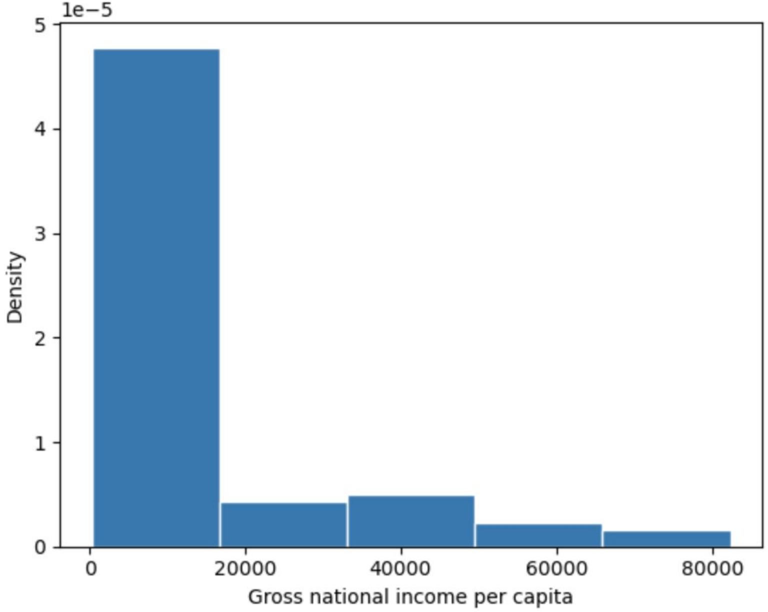


Continent: qualitative nominal

Histograms

A histogram:

- Collects datapoints with similar values into a shared "bin".
- Scales the bins such that the **area** of each bin is equal to the **percentage** of datapoints it contains (as in [Data 8]).



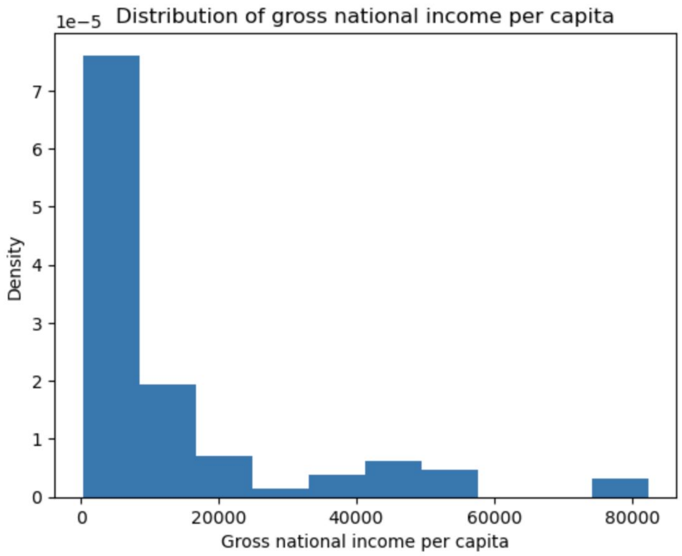
The first bin has a width of \$16410
height of 4.77×10^{-5}

This means that it contains $16410 \times (4.77 \times 10^{-5}) = 78.3\%$ of all datapoints in the dataset.

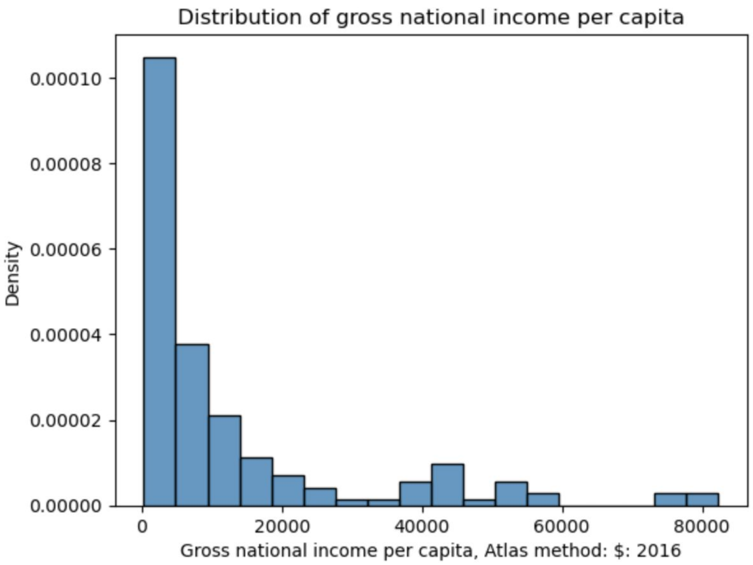
Histograms in Code

In Matplotlib [Documentation]: `plt.hist(x_values, density=True)`

In Seaborn [Documentation]: `sns.histplot(data=df, x="x_column", stat="density")`



Matplotlib

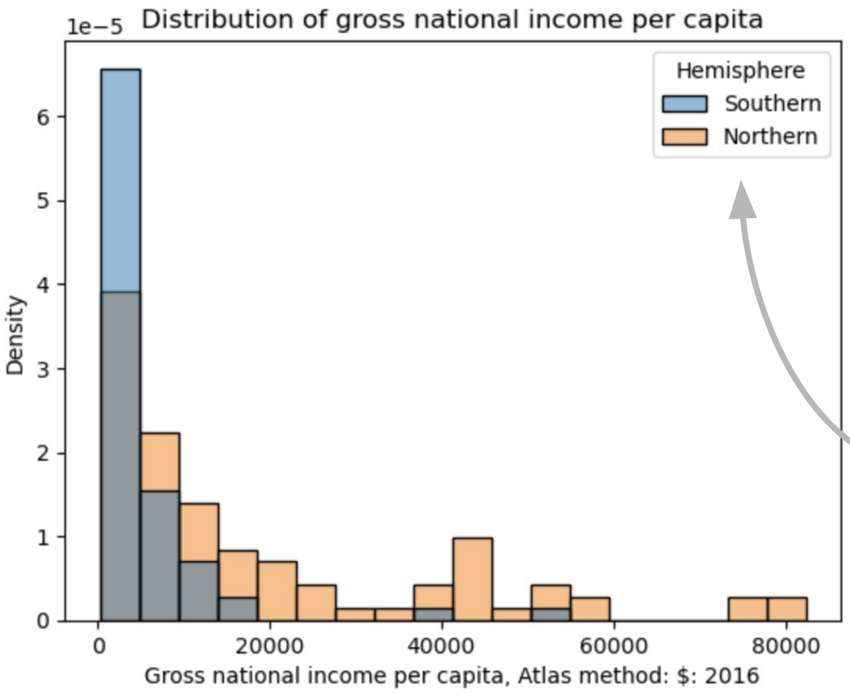


Seaborn

Overlaid Histograms

1823766

To compare a quantitative variable's distribution across qualitative categories, overlay histograms on top of one another.



The **hue** parameter of Seaborn plotting functions sets the column that should be used to determine color.

```
sns.histplot(data=wb, hue="Hemisphere",
x="Gross national income...")
```

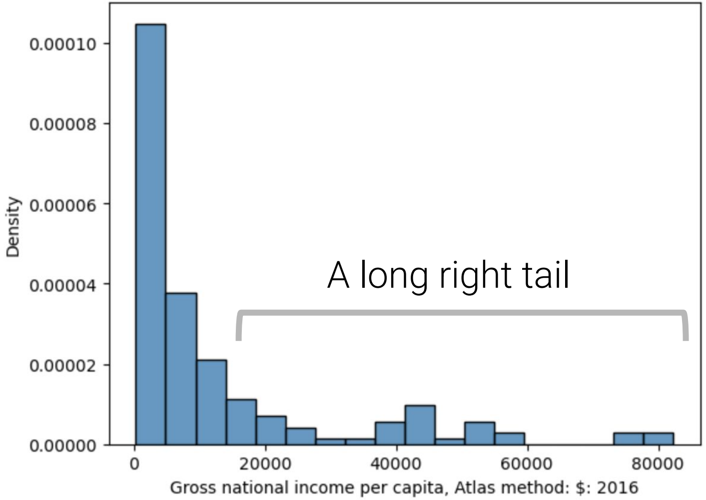
Always include a legend when color is used to encode information!

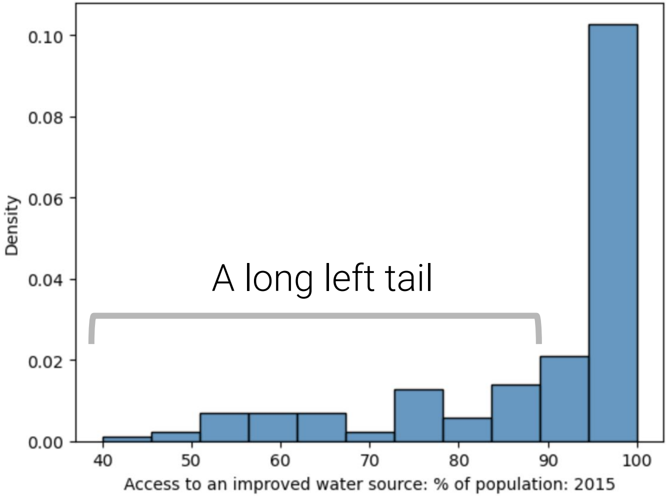
Interpreting Histograms

The **skew** of a histogram describes the direction in which its "tail" extends.

- A distribution with a long right tail is skewed right.
- A distribution with a long left tail is skewed left.

A histogram with no clear skew is called symmetric.



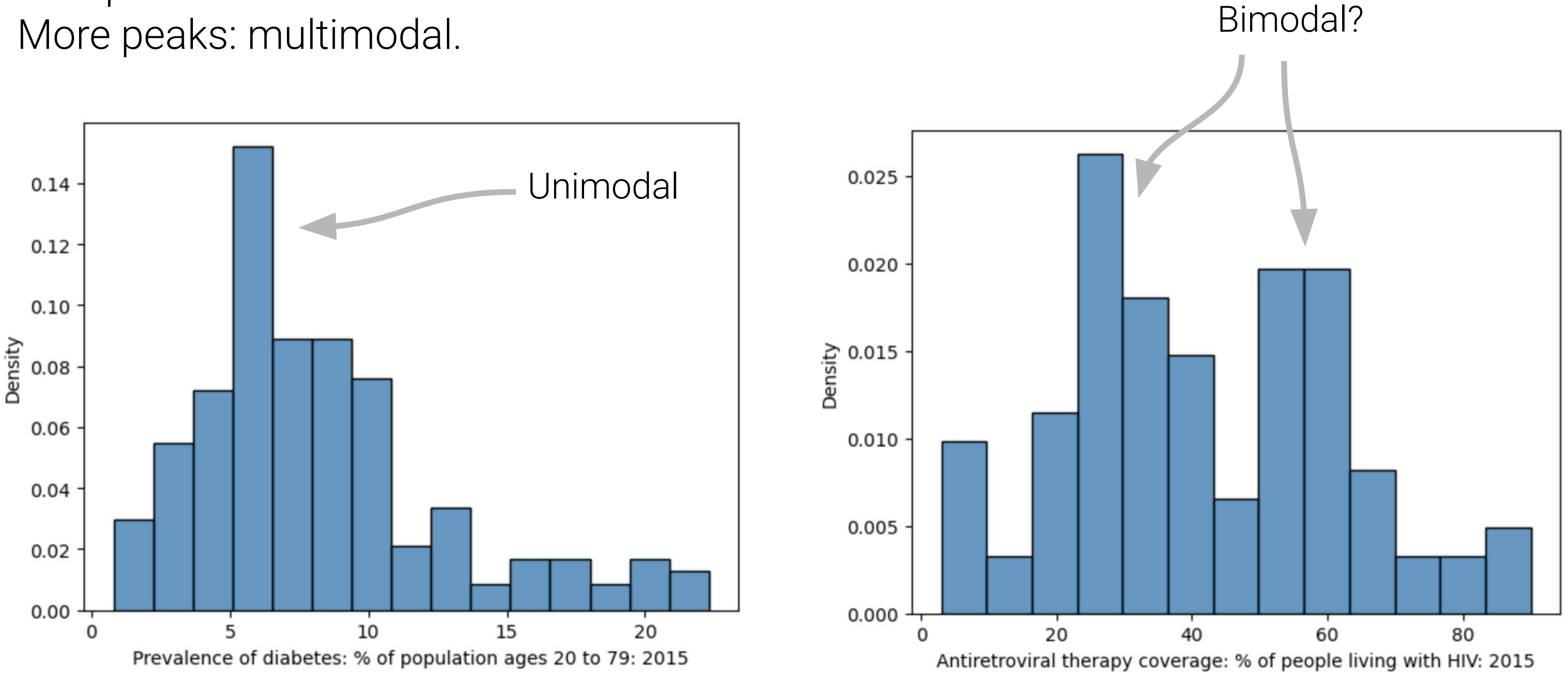


Interpreting Histograms

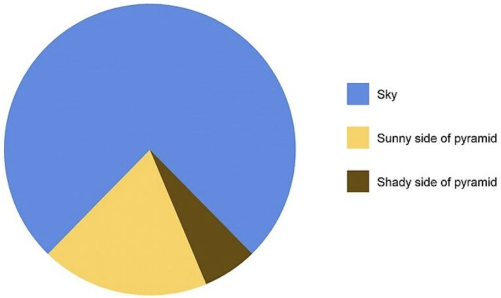
1823766

The **mode(s)** of a histogram are the peak values in the distribution.

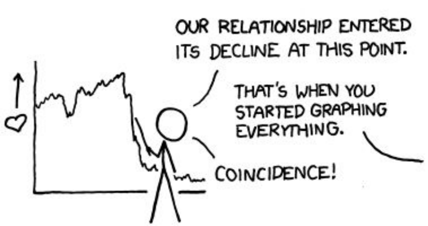
- A distribution with one clear peak is called unimodal.
- Two peaks: bimodal.
- More peaks: multimodal.

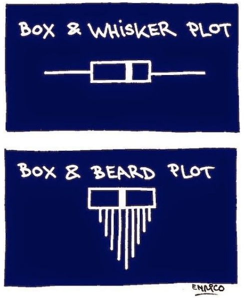


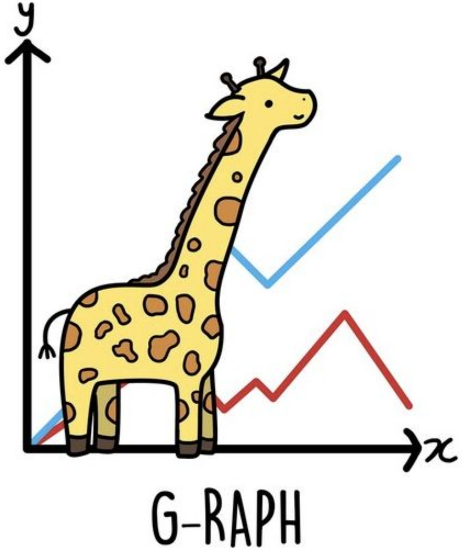
Interlude


Sky
Sunny side of pyramid
Shady side of pyramid


"Bar graph of how much door I've painted."


OUR RELATIONSHIP ENTERED ITS DECLINE AT THIS POINT.
THAT'S WHEN YOU STARTED GRAPHING EVERYTHING.
COINCIDENCE!


BOX & WHISKER PLOT
BOX & BEARD PLOT


y
x
G-RAPH

Kernel Density Estimation

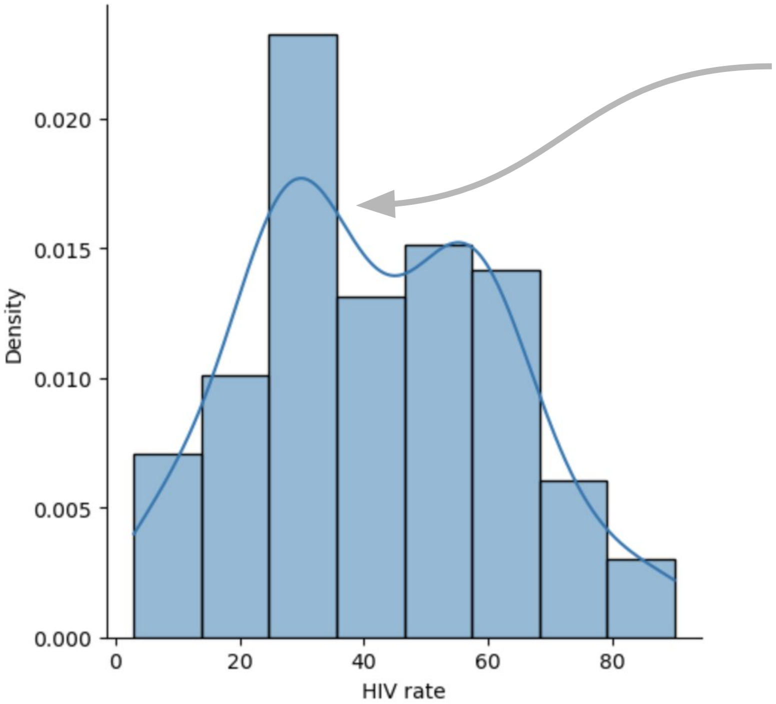
Lecture 7, Data 100 Spring 2025

- **Visualization**
  - Goals of visualization
  - Visualizing distributions
  - **Kernel density estimation**

Kernel Density Estimation: Intuition

1823766

Often, we want to identify *general* trends across a distribution, rather than focus on detail. Smoothing a distribution helps generalize the structure of the data and eliminate noise.



A KDE curve

Idea: approximate the probability distribution that generated the data.

- Assign an “error range” to each data point in the dataset – if we were to sample the data again, we might get a different value.
- Sum up the error ranges of all data points.
- Scale the resulting distribution to integrate to 1.

Kernel Density Estimation: Process

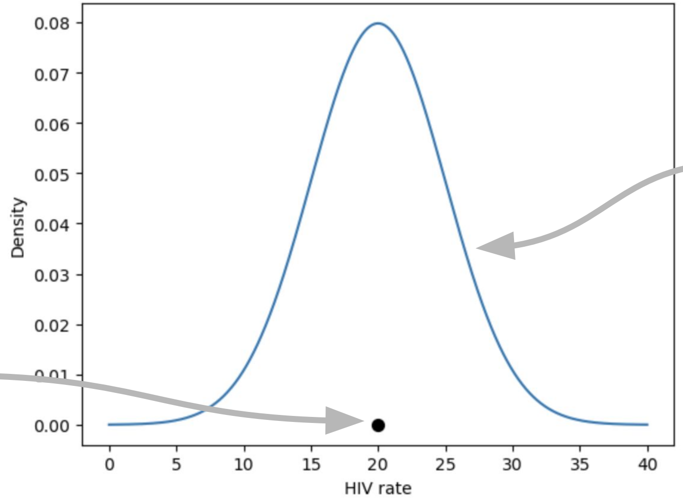
1823766

Idea: Approximate the probability distribution that generated the data.

- Place a kernel at each data point.
- Normalize kernels so that total area = 1.
- Sum all kernels together.

A **kernel** is a function that tries to capture the randomness of our sampled data.

A **datapoint** in our dataset



The **kernel** models the probability of us sampling that datapoint.

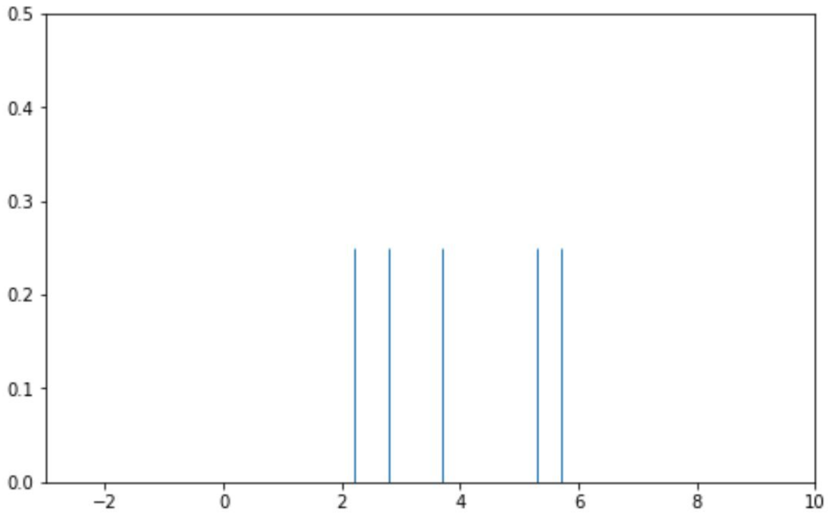
Area below integrates to 1

Step 1 – Place a Kernel at Each Data Point

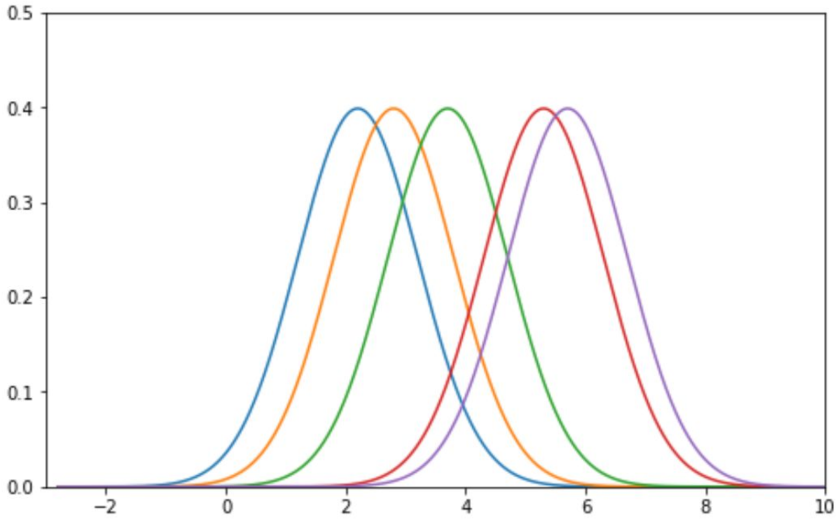
1823766

Consider a fake dataset with just five collected datapoints.

- Place a **Gaussian kernel** with **bandwidth** of **alpha = 1**.
- We will precisely define both the **Gaussian kernel** and **bandwidth** in a few slides.



Each line represents a datapoint in the dataset (e.g. one country's HIV rate).



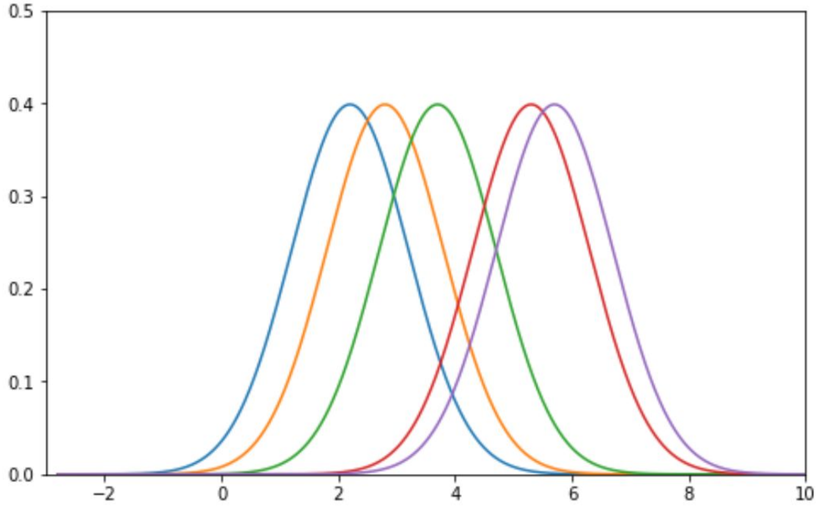
Place a kernel on top of each datapoint.

Step 2 – Normalize Kernels

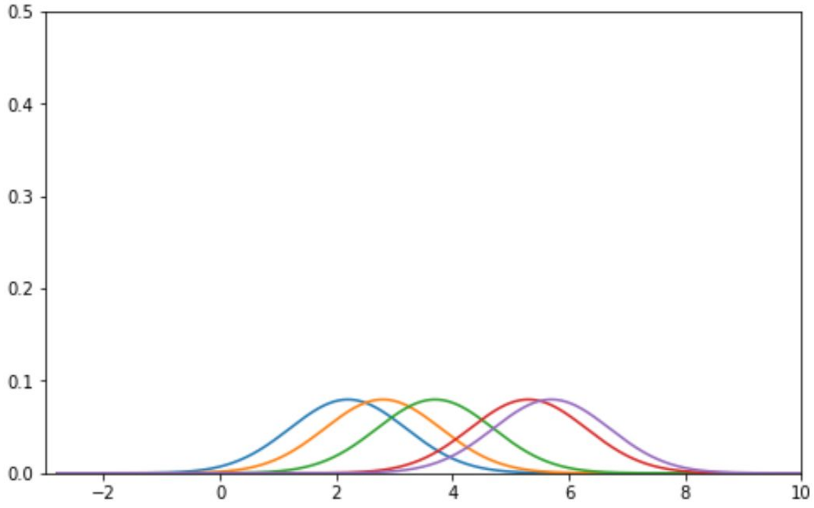
1823766

In Step 3, We will be summing each of these kernels to produce a probability distribution.

- We want the result to be a valid probability distribution that has area 1.
- We have 5 different kernels, each with an area 1.
- So, we normalize by multiplying each kernel by ⅕.



Each kernel has area 1.

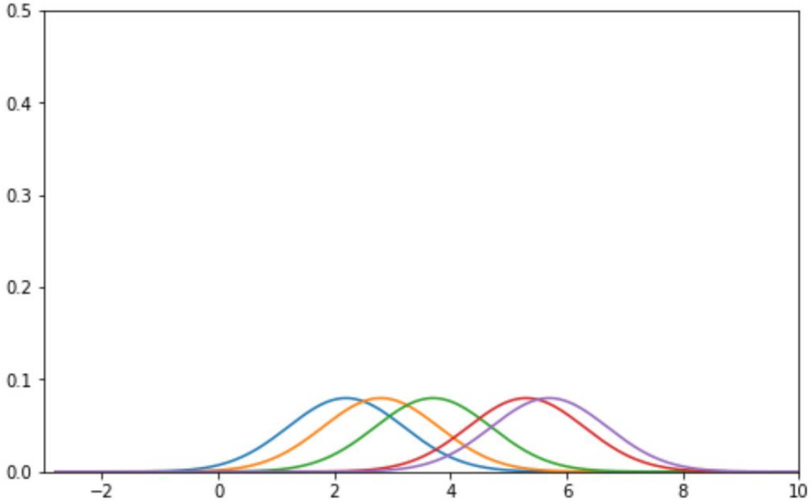


Each normalized kernel has density ⅕.

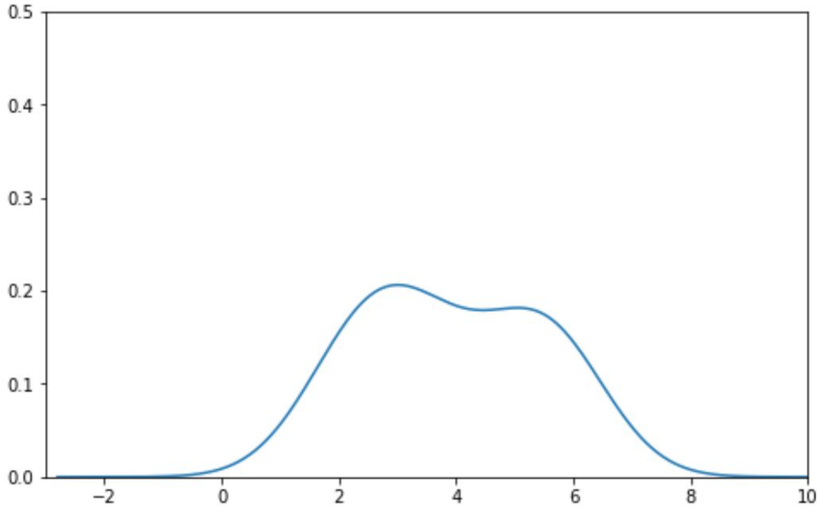
Step 3 – Sum the Normalized Kernels

1823766

At each point in the distribution, add up the values of all kernels. This gives us a smooth curve with area 1 – an approximation of a probability distribution!



Sum these five normalized curves together.



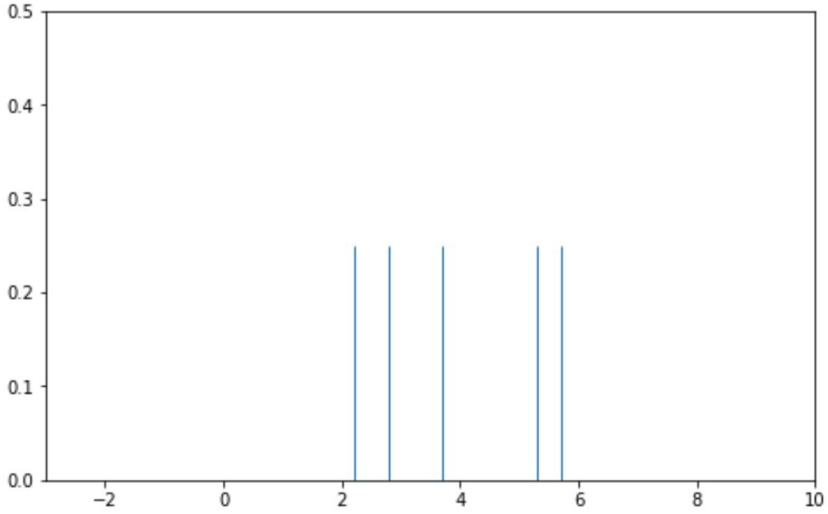
The final KDE curve.

Result

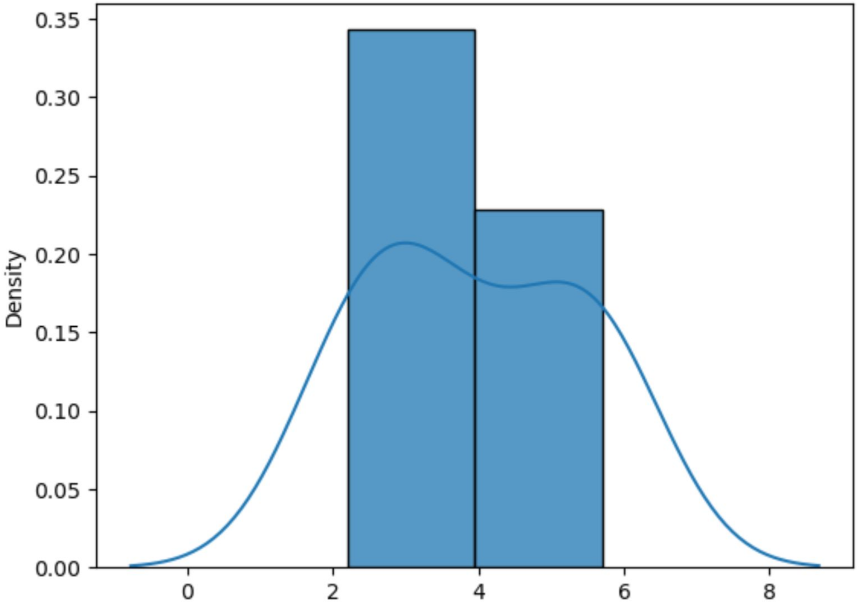
1823766

- A summary of the distribution using KDE.

```
sns.kdeplot(points, bw_method=0.65)
sns.histplot(points,
             stat="density",
             bins=2);
```



Each line represents a datapoint in the dataset (e.g. one country's HIV rate).

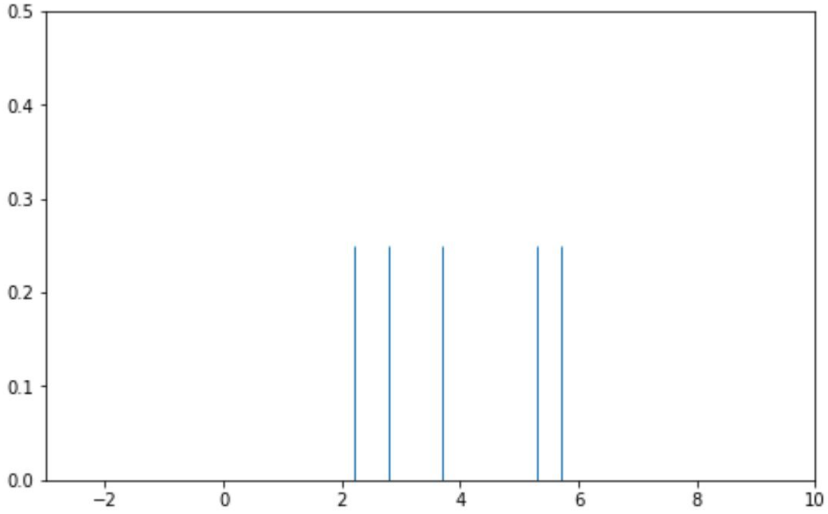


The density at each point corresponds to the KDE calculated based on kernels placed on all data points

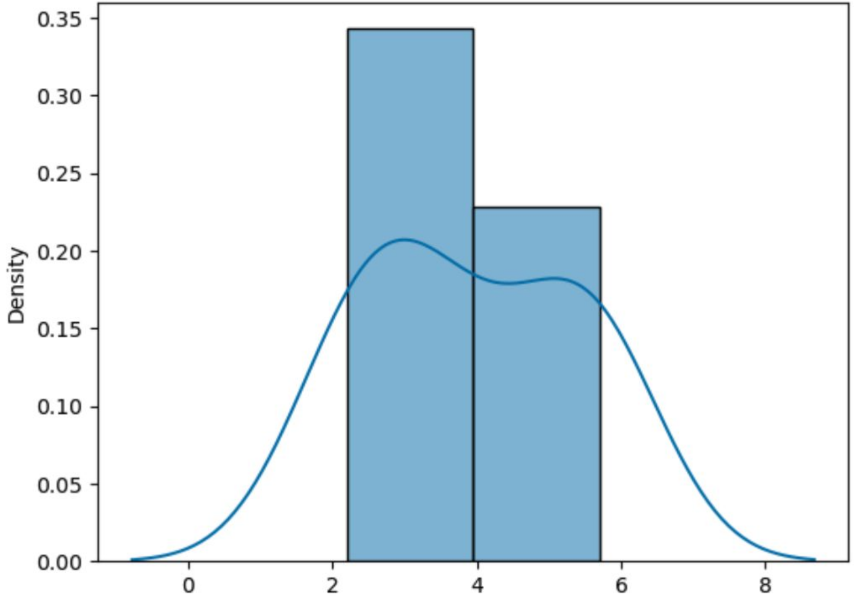
Result - Alternate Method

- A summary of the distribution using KDE.

```python
sns.histplot(points, bins=2, kde=True,
             stat="density",
             kde_kws=dict(cut=3,
                          bw_method=0.65))
```



Each line represents a datapoint in the dataset (e.g. one country's HIV rate).



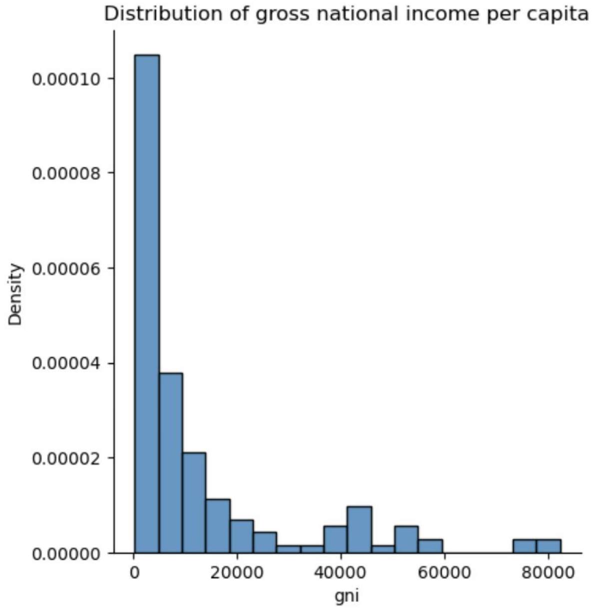
The density at each point corresponds to the KDE calculated based on kernels placed on all data points

displot

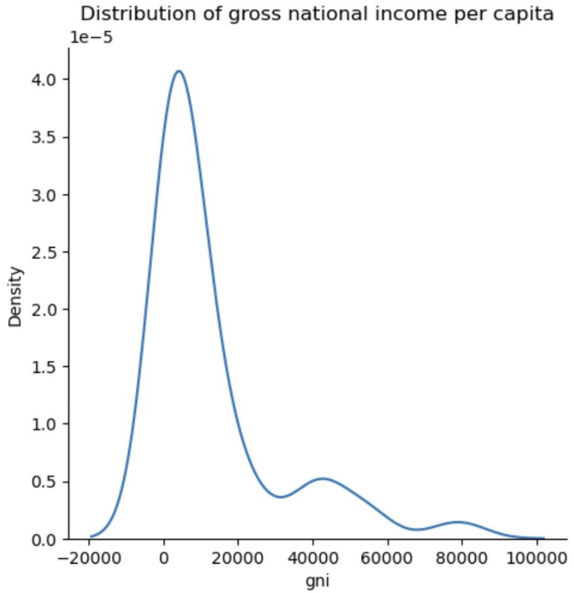
1823766

`displot` is a wrapper for **`histplot`**, **`kdeplot`**, and **`ecdfplot`** to plot distributions.

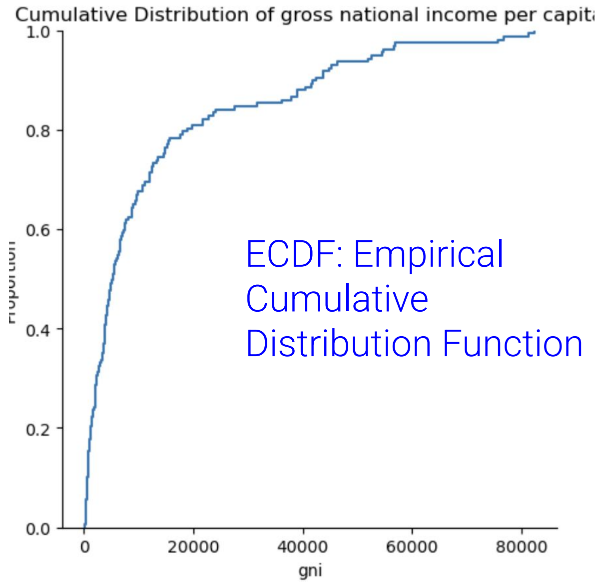
```python
sns.displot(data=wb,
            x="gni",
            kind="hist",
            stat="density")
```




```python
sns.displot(data=wb,
            x="gni",
            kind="kde")
```



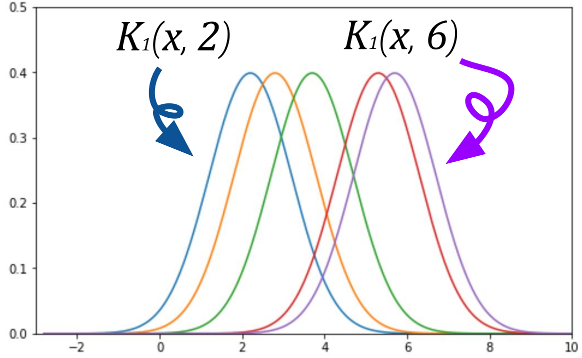

```python
sns.displot(data=wb,
            x="gni",
            kind="ecdf")
```



ECDF: Empirical Cumulative Distribution Function

Summary of KDE

$$f_\alpha(x) = \overset{\boxed{2}}{\frac{1}{n}} \overset{\boxed{3}}{\sum_{i=1}^{n}} \overset{\boxed{1}}{K_\alpha(x, x_i)}$$



A general “KDE formula” function is given above.

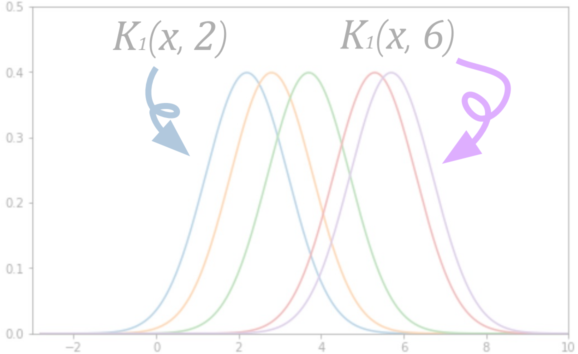
$K_\alpha(x, x_i)$ is the **kernel** function centered on the observation i.

- Each kernel individually has area 1.
- K represents our kernel function of choice. We’ll talk about the math of these functions soon.

Summary of KDE

1823766

$$f_\alpha(x) = \overset{\boxed{2}}{\frac{1}{n}} \overset{\boxed{3}}{\sum_{i=1}^{n}} \overset{\boxed{1}}{K_\alpha(x, x_i)}$$



A general "KDE formula" function is given above.

1. $K_\alpha(x, x_i)$ is the **kernel** centered on the observation i.
   - Each kernel individually has area 1.
   - x represents any number on the number line. It is the input to our function.
2. n is the number of observed data points that we have.
   - We multiply by $1/n$ to normalize the kernels so that the total area of the KDE is still 1.
3. Each x_i ($x_1, x_2, \ldots, x_n$) represents an observed data point. We sum the kernels for each datapoint to create the final KDE curve.

α is the **bandwidth** or **smoothing parameter**.

Lecture 7 ended here!

We will cover the rest in lecture 8

Kernels

1823766

A **kernel** (for our purposes) is a valid density function, meaning:

- It must be non-negative for all inputs.
- It must integrate to 1(area under curve = 1).

The most common kernel is the **Gaussian kernel**.

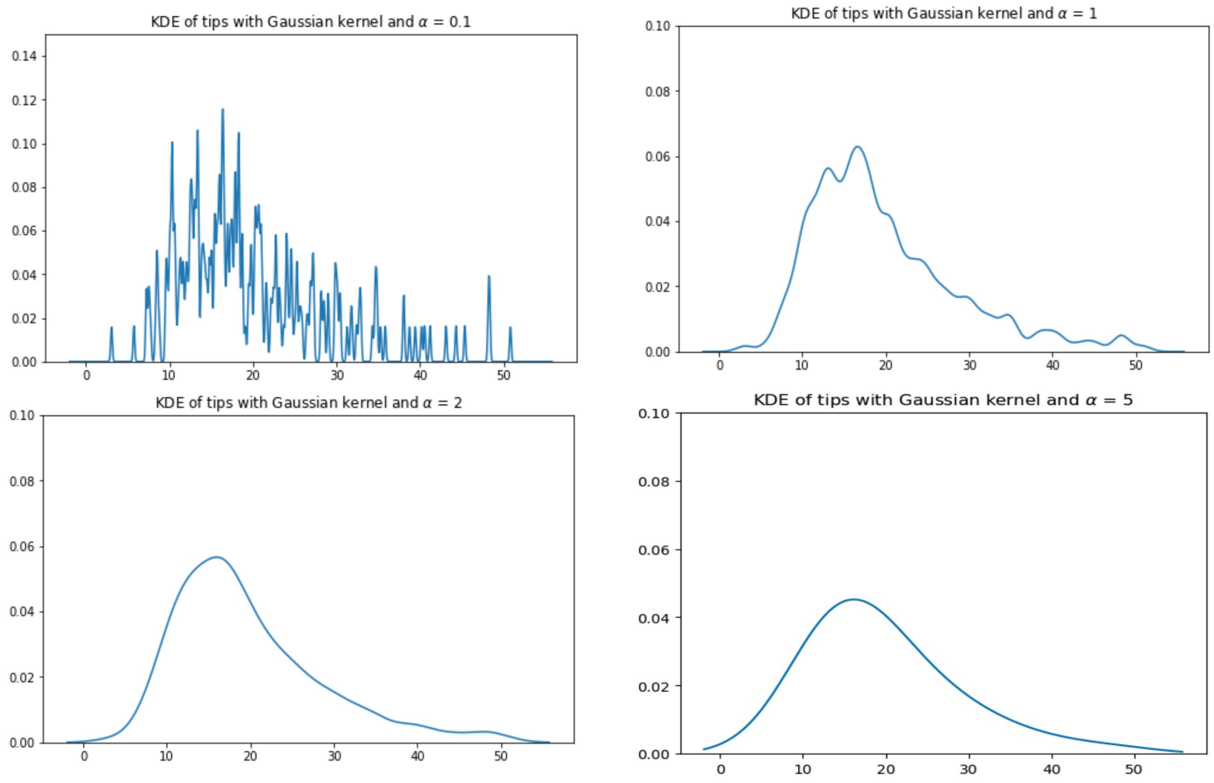
- Gaussian = Normal distribution = bell curve.
- Here, x represents any input, and x_i represents the ith observed value (datapoint).
- Each kernel is **centered** on our observed values (and so its distribution mean is x_i).
- **α is the bandwidth parameter**. It controls the smoothness of our KDE. Here, it is also the standard deviation of the Gaussian.

$$K_\alpha(x, x_i) = \frac{1}{\sqrt{2\pi\alpha^2}} e^{-\frac{(x-x_i)^2}{2\alpha^2}}$$

Memorizing this formula is less important than knowing the shape and how the bandwidth parameter **α** smoothes the KDE.

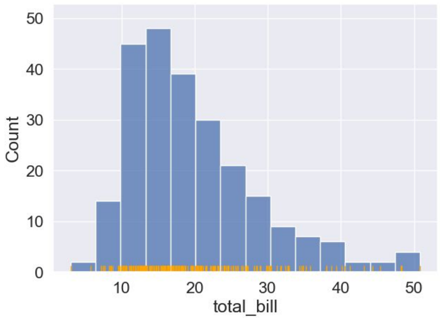
Effect of Bandwidth on KDEs

1823766



Bandwidth is analogous to the width of each bin in a histogram.

- As **α** increases, the KDE becomes more smooth.
- Large **α** KDE is simpler to understand, but gets rid of potentially important distributional information (e.g. multimodality).



Other Kernels: Boxcar

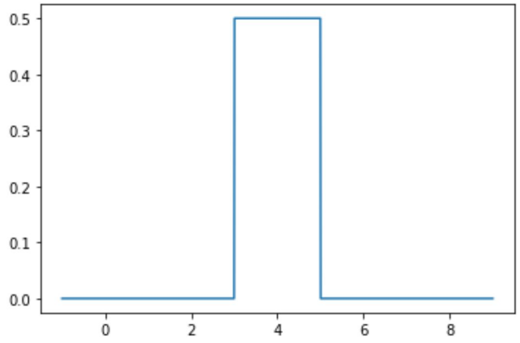
1823766

As an example of another kernel, consider the **boxcar kernel**.

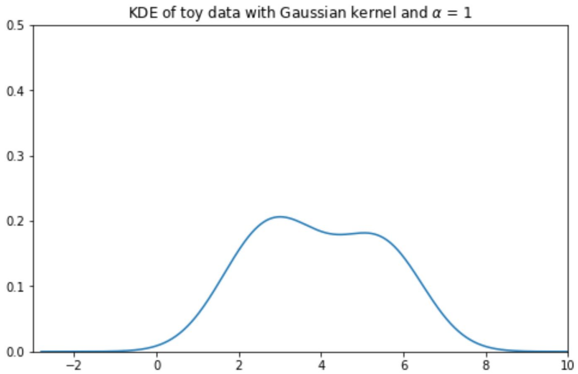
- It assigns uniform density to points within a “window” of the observation, and 0 elsewhere.
- Resembles a histogram… *sort of*.

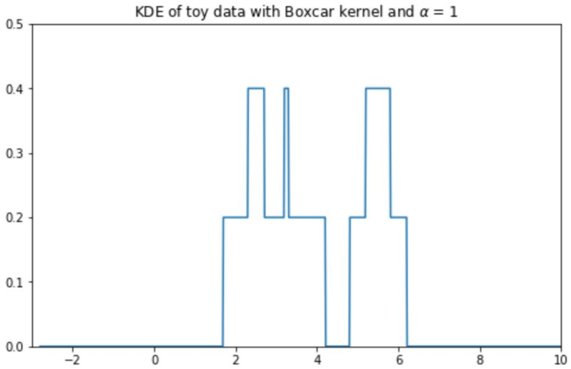
$$K_\alpha(x, x_i) = \begin{cases} \frac{1}{\alpha}, & |x - x_i| \le \frac{\alpha}{2} \\ 0, & \text{else} \end{cases}$$

- Not of any practical use in Data 100! Presented as a simple theoretical alternative.



A boxcar kernel centered on $x_i = 4$ with $\alpha = 2$.





Have a Normal Day!

1823766

1823766

LECTURE 7

Visualization I

Content credit: Acknowledgments