LECTURE 10

Visualization, Part I

Visualizing Distributions and Relationships Between Quantitative Variables

Visualizations in the Real World, Goals

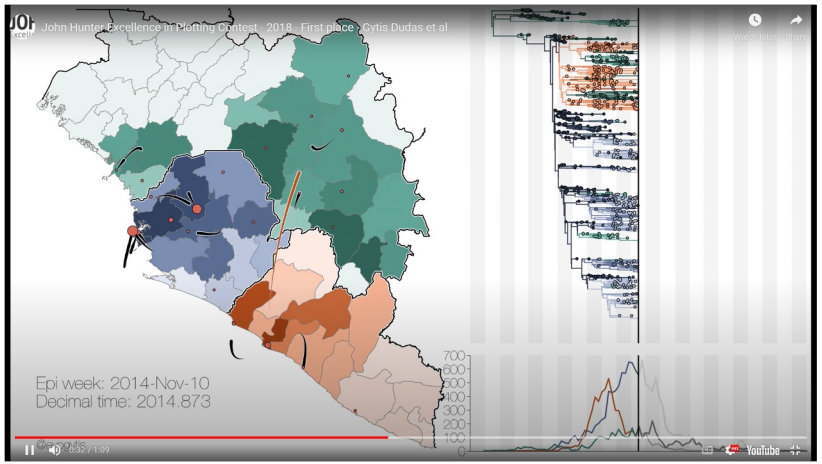
- **Visualizations In the Real World, Goals**
- Bar Plots/Histograms for Distributions
- Box Plots and Violin Plots
- Comparing Quantitative Distribution
- Relationships Between Quantitative Variables

Boutique Visualizations

Visualizations can also be much more complex.

Examples:

- https://projects.fivethirtyeight.com/
- https://www.nytimes.com/interactive/2020/04/14/science/coronavirus-transmission-cough-6-feet-ar-ul.html
- The John Hunter Excellence in Plotting Contest: https://jhepc.github.io/gallery.html



Link

Goals of Data Visualization

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

- Key part of exploratory data analysis.
- Useful throughout modeling as well.
- 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.
- Often time-consuming: bridges into design, even art.

A constant tool across the lifecycle of data science

Bar Plots/Histograms for Distributions

- Visualizations In the Real World, Goals
- **Bar Plots/Histograms for Distributions**
- Box Plots and Violin Plots
- Comparing Quantitative Distribution
- Relationships Between Quantitative Variables

What is a distribution?

As a case study, we'll devote a large fraction of our time today towards visualization of **distributions**.

A **distribution** describes the frequency at which values of a variable occur.

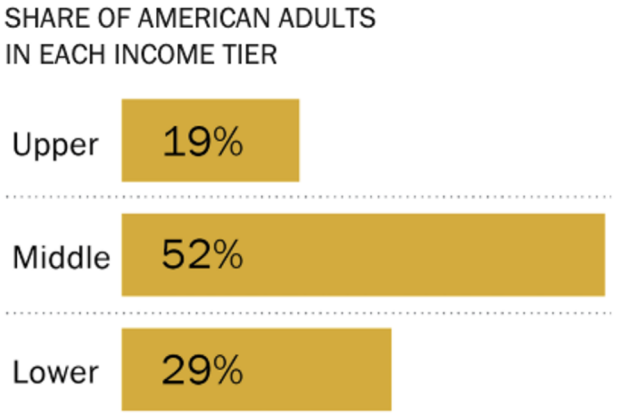
- All values must be accounted for **once, and only once**.
- The total frequencies must **add up to 100%**, or to the number of values that we're observing.

Let's look at some examples.

Bar Plots

Bar Plots are the most common way of displaying the **distribution** of a **qualitative (categorical)** variable.

- 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).



Example dataset

We will be using the baby weights dataset for most of our plots today. Here is what that looks like.

```
1 births = pd.read_csv('baby.csv')
```

```
1 births.head()
```

	Birth Weight	Gestational Days	Maternal Age	Maternal Height	Maternal Pregnancy Weight	Maternal Smoker
0	120	284	27	62	100	False
1	113	282	33	64	135	False
2	128	279	28	64	115	True
3	108	282	23	67	125	True
4	136	286	25	62	93	False

```
1 births.shape
```

```
(1174, 6)
```

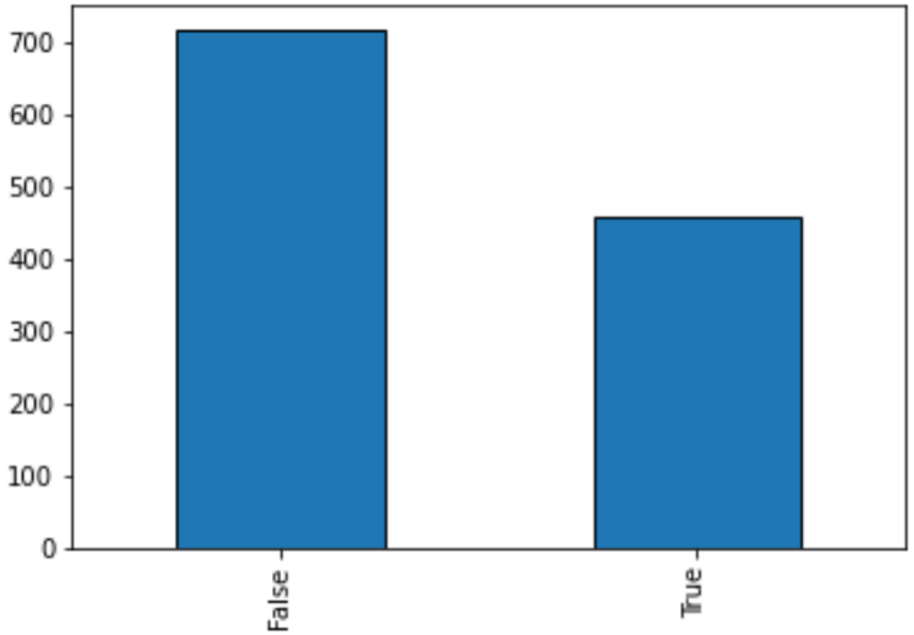
Generating Bar Plots in Python

there are many ways we could generate this bar code.

- Let’s see a few.

Generating Bar Plots in Python (Pandas Native)

Pandas dataframes and series have plotting methods, similar to the Table class.

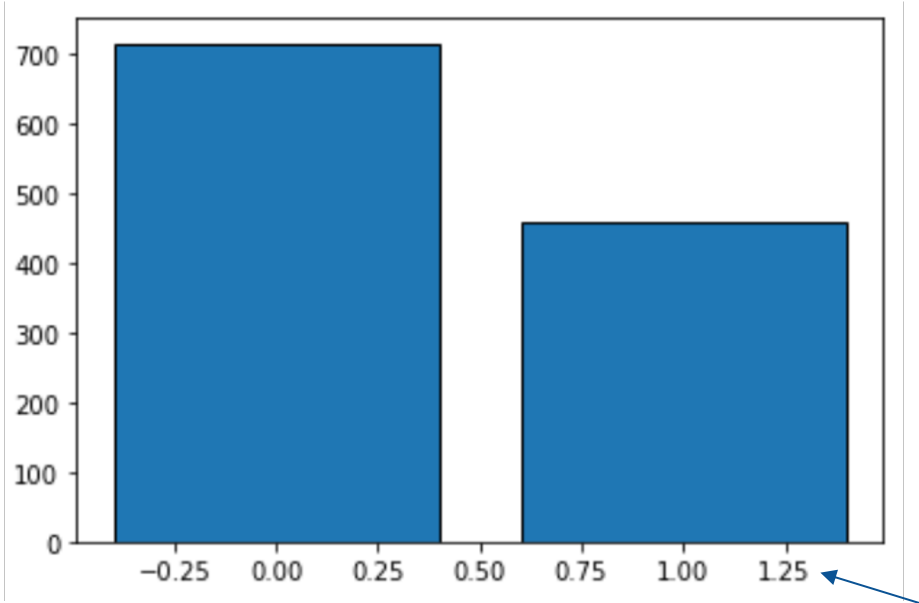



```
births['Maternal Smoker'].value_counts().plot(kind = 'bar')
```

Generating Bar Plots in Python (Matplotlib manual)

We could also directly invoke the matplotlib's `bar` function.

- We'll do this sometimes, but not often.



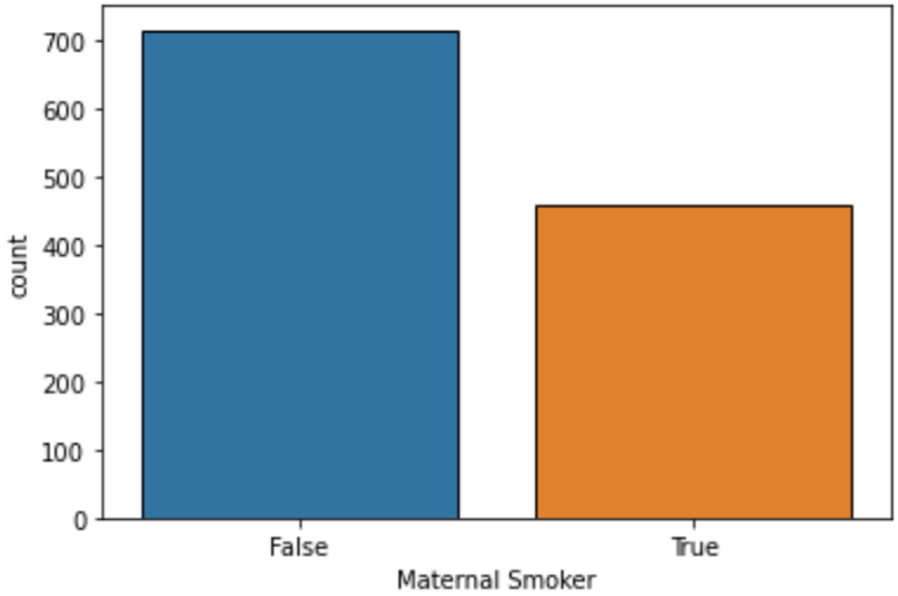
We could fix the x-axis labels, but I do not do so here.

```
ms = births['Maternal Smoker'].value_counts();
plt.bar(ms.index, ms);
```

Generating Bar Plots in Python (Seaborn)

We could use the Seaborn library's `countplot` function.

- Preferred approach.



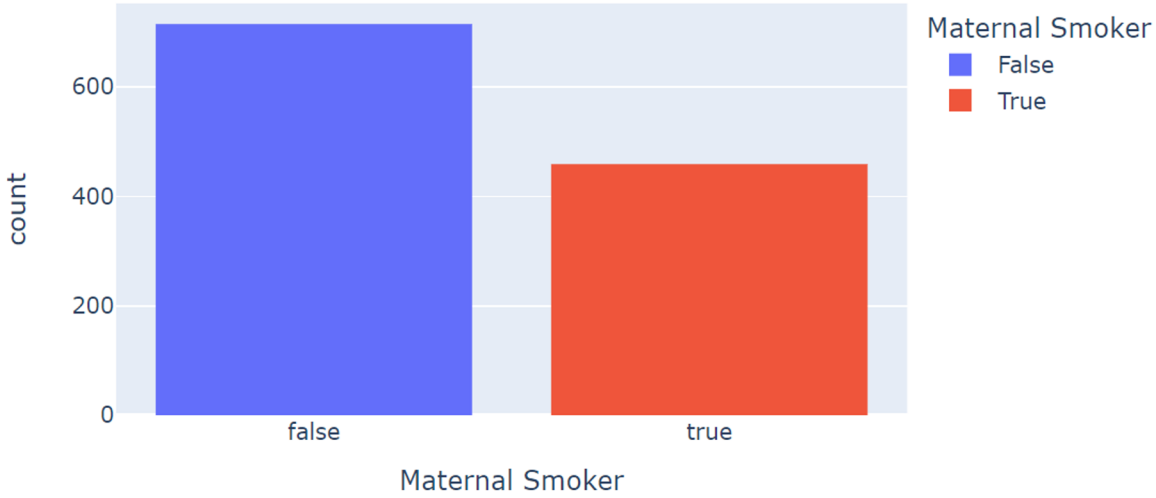
`countplot` operates at a higher level of abstraction!

You give it the entire dataframe and it does the counting for you.

```
import seaborn as sns
sns.countplot(data = births, x = 'Maternal Smoker');
```

Generating Bar Plots in Python (plotly)

We could use the plotly library's **`histogram`** function.



Note: **`plotly`** plots are interactive!

```
import plotly.express as px
px.histogram(births, x = 'Maternal Smoker', color = 'Maternal Smoker')
```

Generating Bar Plots in Python

We've seen four different ways to generate the same plot. The first three are matplotlib based:

- Pandas bar function (Series.bar).
- Matplotlib bar function (plt.bar).
- **Seaborn countplot function (`sns.countplot`).**
  - Uses matplotlib as its rendering engine.
  - Simpler to use. Superior aesthetics.

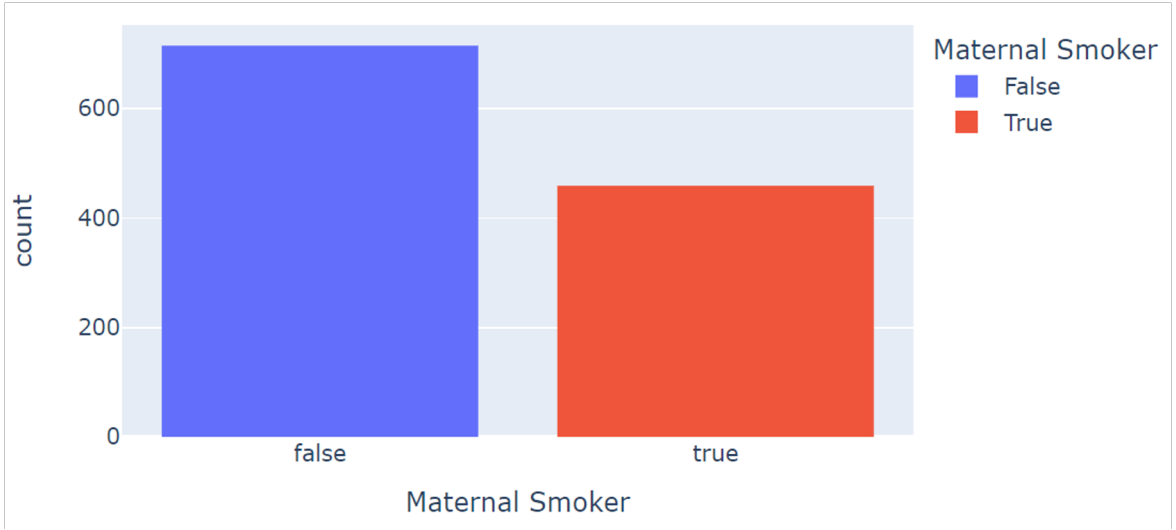
And the last is a non-matplotlib library:

- Plotly histogram function (px.histplot).
  - Aesthetics are often superior to seaborn.
  - Plots are interactive.

Questions About Our Bar Plot

Observations:

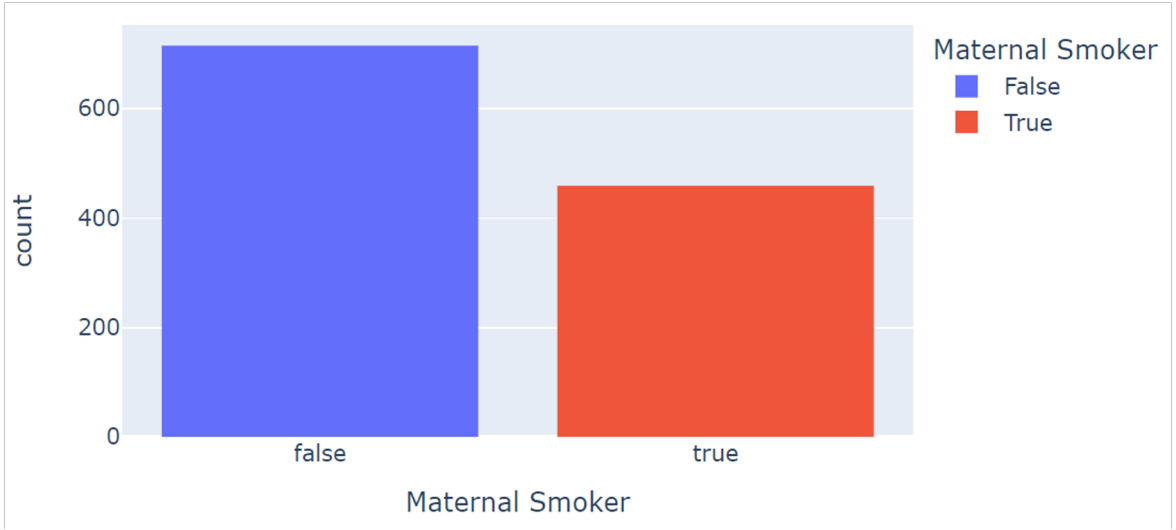
- The colors that were chosen (blue and red) are totally arbitrary and were not intended to convey any specific information, other than that there are two distinct categories.
- The widths of the bars encode no useful information and their arbitrarily chosen width is just to look nice.



Questions About Our Bar Plot

Questions:

- What colors should we use?
- How wide should the bars be?
- Should the legend exist?
- Should the bars and axes have dark borders?

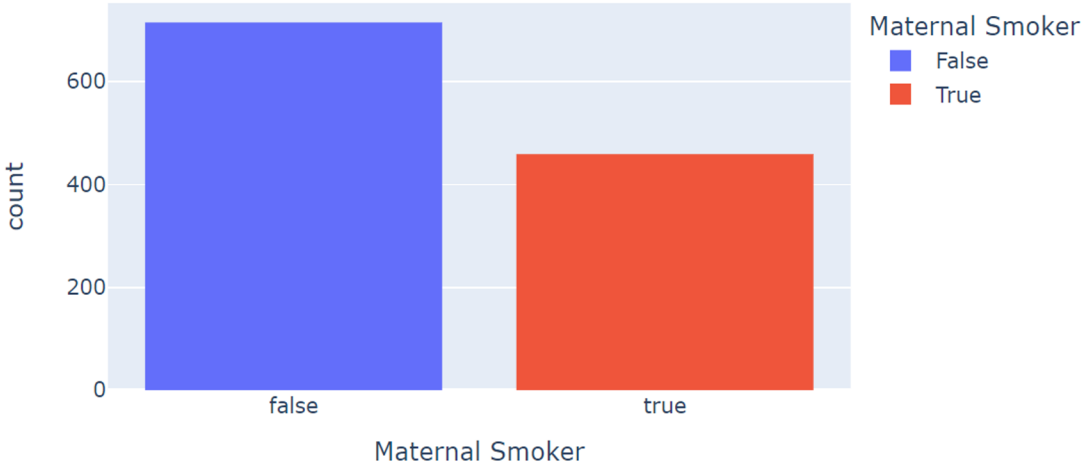


How do we decide? Are these purely aesthetic questions?

Questions About Our Bar Plot

Questions:

- What colors should we use?
- How wide should the bars be?
- Should the legend exist?
- Should the bars and axes have dark borders?



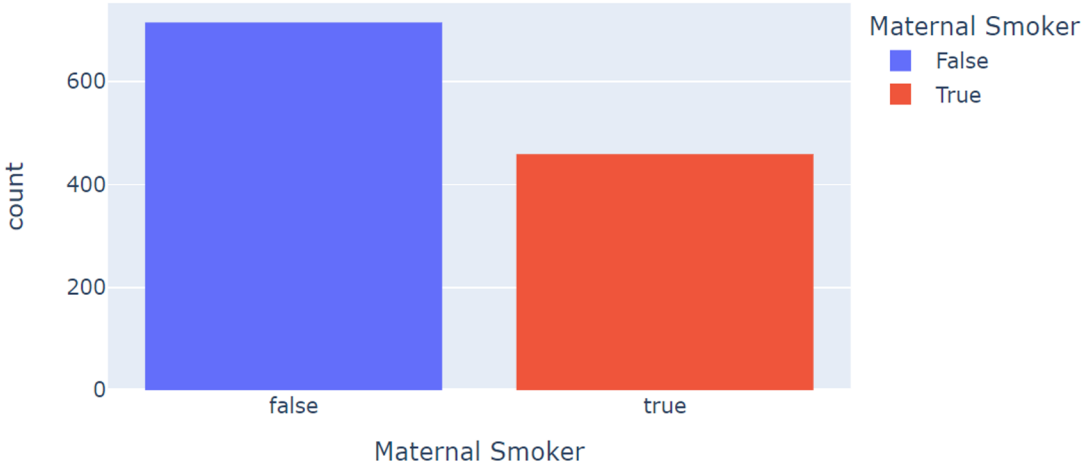
Thoughts:

- Could pick colors on the same gradient as a subtle cue that we're not trying to convey meaning.
  - If our goal was to convince people smoking is bad, maybe the current colors are good choice.
- Legend should exist, especially if it has numbers. Understand the info immediately. Better chance that a casual reader notices.
  - Legend feels messy and cluttered. It looks unprofessional
- Replace true and false with smoker and nonsmoker.
- Specify the exact count! Don't rely on the interactivity.

Questions About Our Bar Plot

Questions:

- What colors should we use?
- How wide should the bars be?
- Should the legend exist?
- Should the bars and axes have dark borders?



Thoughts:

- In many contexts, you want an annotation nearby saying where the data came from.

Questions About Other Features

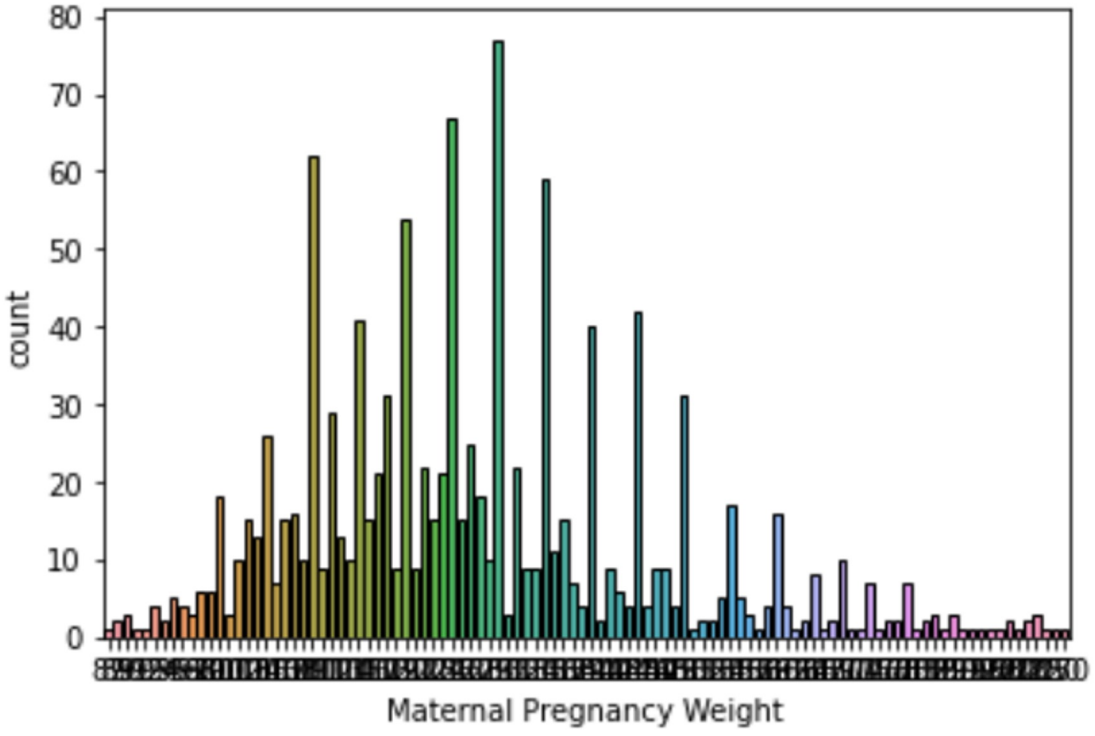
Our data set has many other features.

	Birth Weight	Gestational Days	Maternal Age	Maternal Height	Maternal Pregnancy Weight	Maternal Smoker
0	120	284	27	62	100	False
1	113	282	33	64	135	False
2	128	279	28	64	115	True
3	108	282	23	67	125	True
4	136	286	25	62	93	False
...	...	...	...	...	...	...

Suppose we want to plot the distribution of “Maternal Pregnancy Weight” as a bar plot.

- What are our “categories”? One natural choice: Each integer value, e.g. 100 is a category, 135 is a category, etc.

Maternal Pregnancy Weight



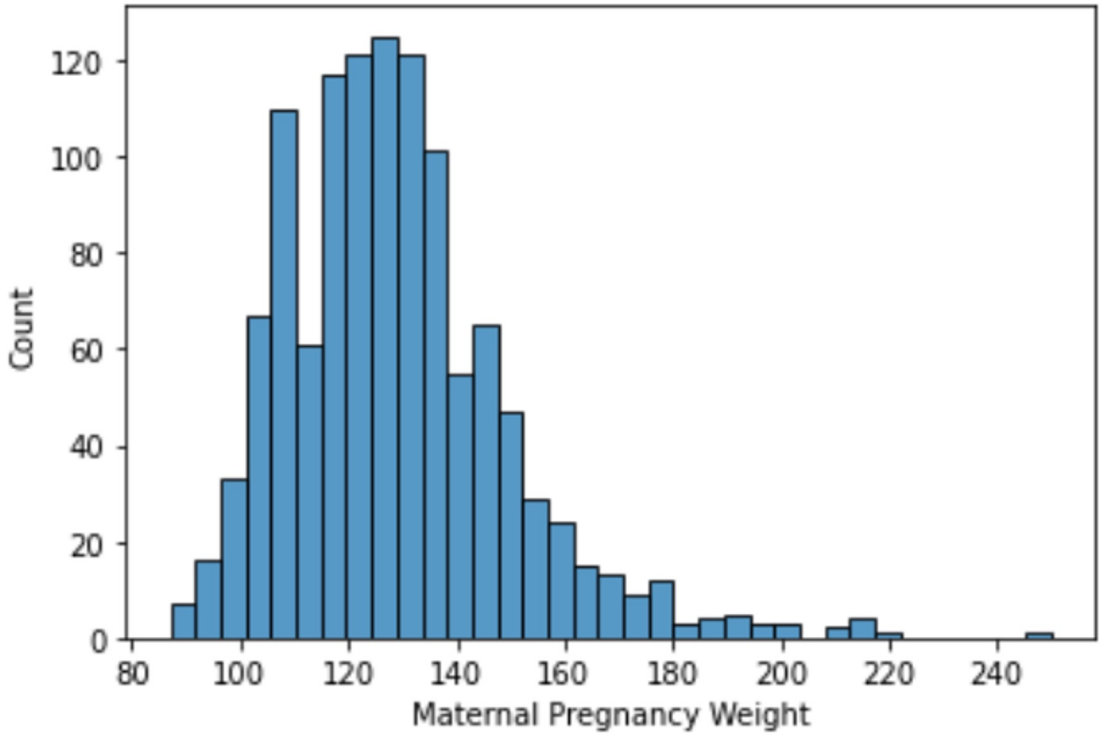
If we use each observed integer value as a category.

What's wrong here?

```
sns.countplot(data = births, x = 'Maternal Pregnancy Weight');
```

Maternal Pregnancy Weight

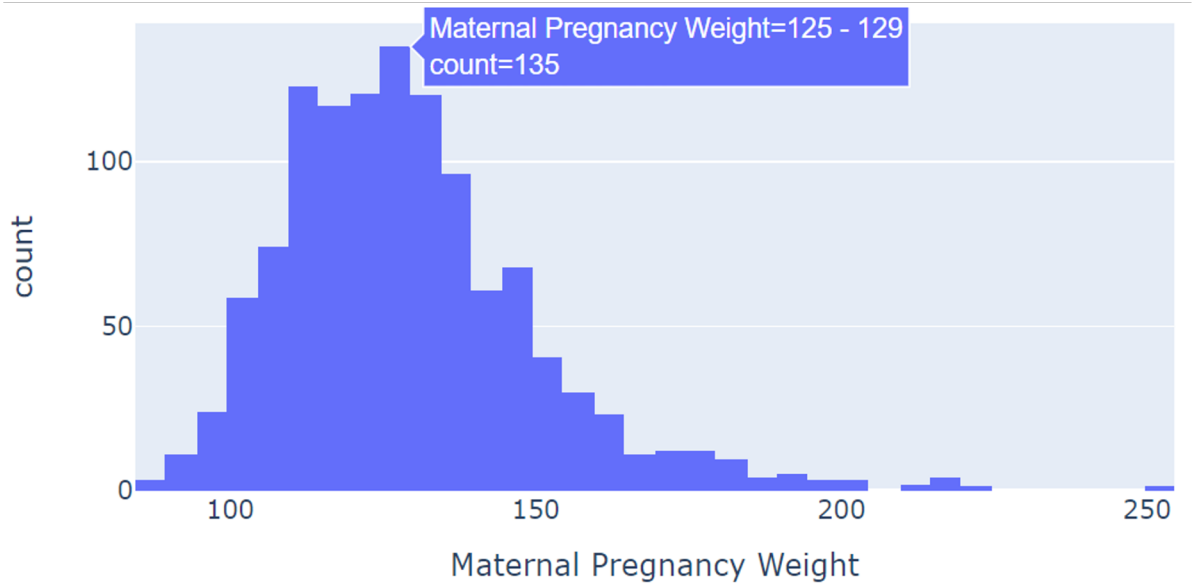
Quick note: These weights are **self-reported pre-pregnancy weights** from the 1960s!



If we use bins of integer values as a category.

```
sns.histplot(data = births, x = 'Maternal Pregnancy Weight');
```

Maternal Pregnancy Weight (in plotly)

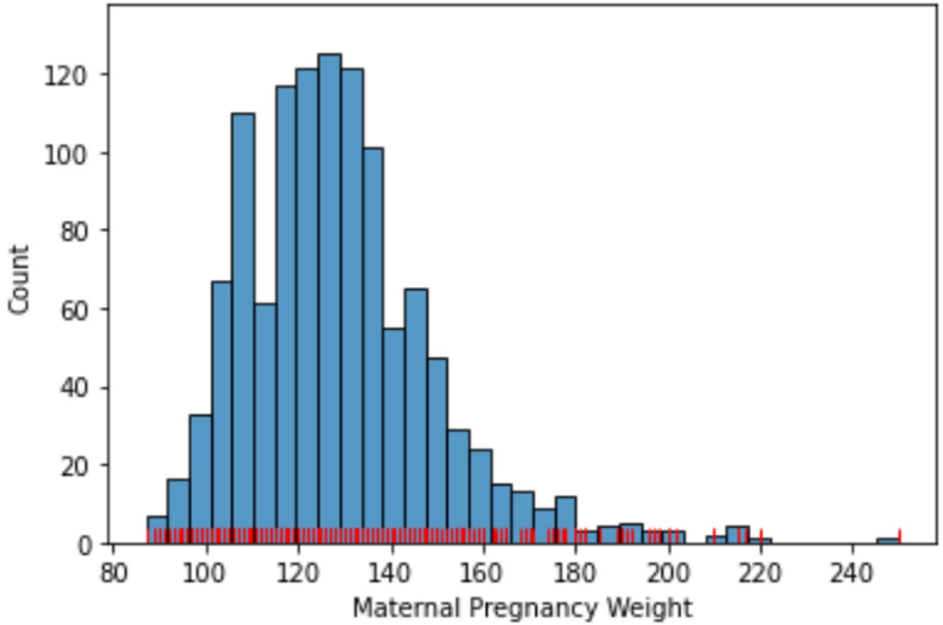



```
px.histogram(births, x = 'Maternal Pregnancy Weight')
```

Rugplots

We can add even more information to the bar plot.

- An overlaid “rug plot” lets us see the distribution of data points within each bin.



Not clear to me that the rug plot is useful in this context! But it’s an enhancement you might find useful sometimes.

```
sns.histplot(data = births, x = 'Maternal Pregnancy Weight');
sns.rugplot(data = births, x = 'Maternal Pregnancy Weight', color = "red");
```

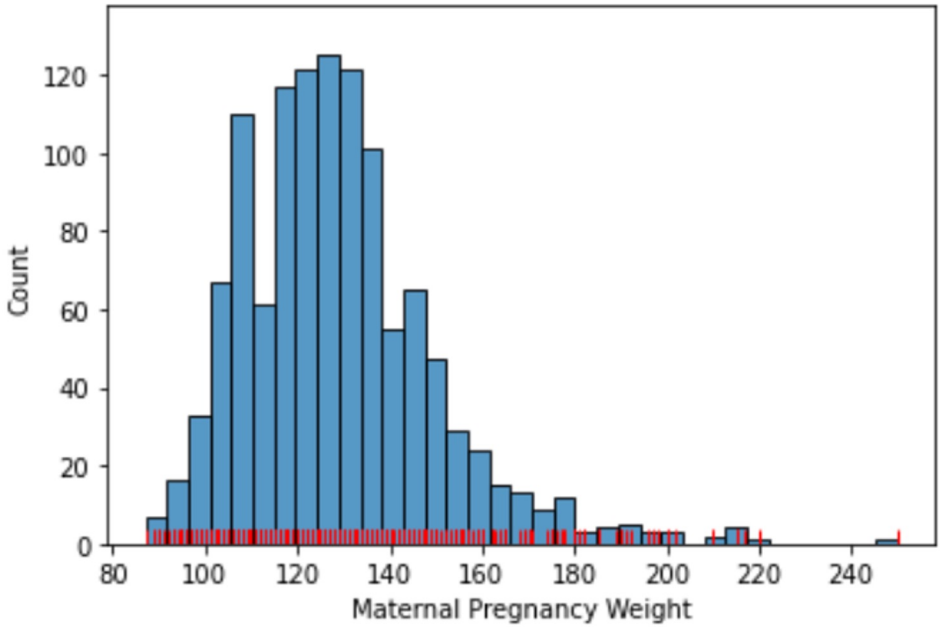
Describing distributions

Histograms allow us to assess a distribution by their shape.

Some of the terminology we use to describe distributions:

- **Skewness and Tails.**
  - Skewed left vs skewed right.
  - Left tail vs right tail.
- **Outliers.**
  - We'll define these arbitrarily.
- **Modes.**

Skewness and Tails

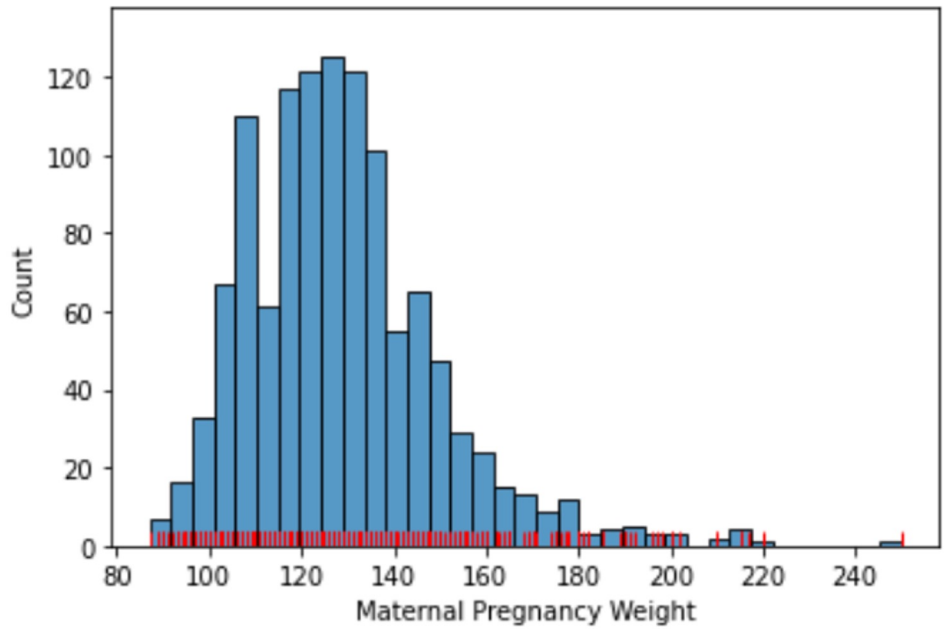


Median: 125, Mean: 128.5

If a distribution has a **long right tail**, we call it **skewed right.**

- Mean is typically to the right of the median.
  - Think of the mean as the "balancing point" of the density.
- If the **tail is on the left**, we say the data is **skewed left**.
- Our distribution can be also **symmetric**, when both tails are of equal size.

Outliers



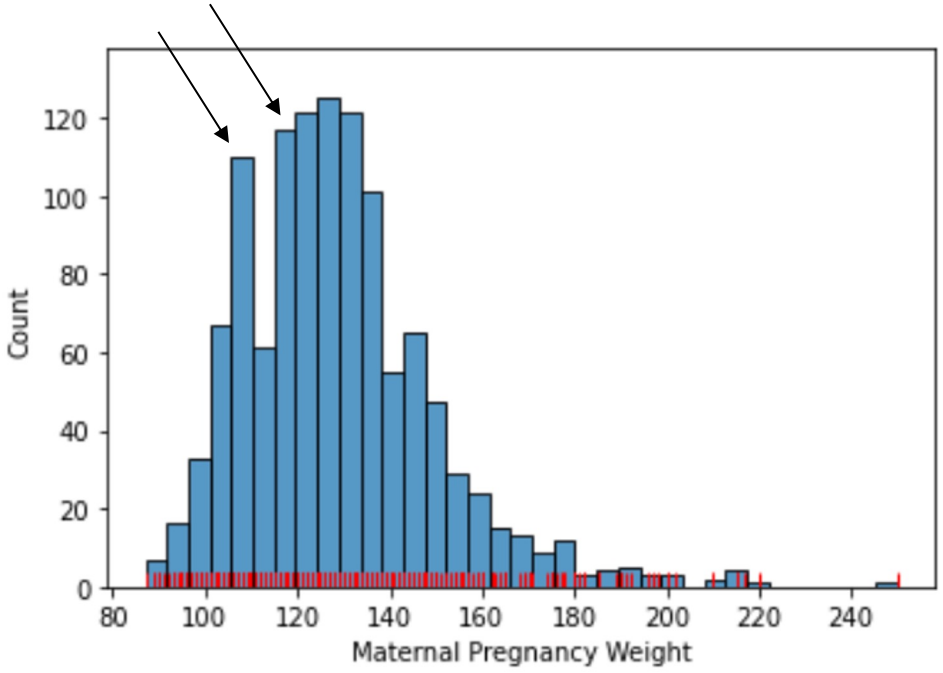
Median: 125, Mean: 128.5

This visualization lets us see **outlier(s)** in this sample on the far right.

- What constitutes an outlier is a choice we have to make.
  - Just the largest point?
  - The rightmost 4 bins?
- Will define outliers more carefully later when we talk about box plots.

Modes

Two distinct modes?



A **mode** of a distribution is a local or global maximum.

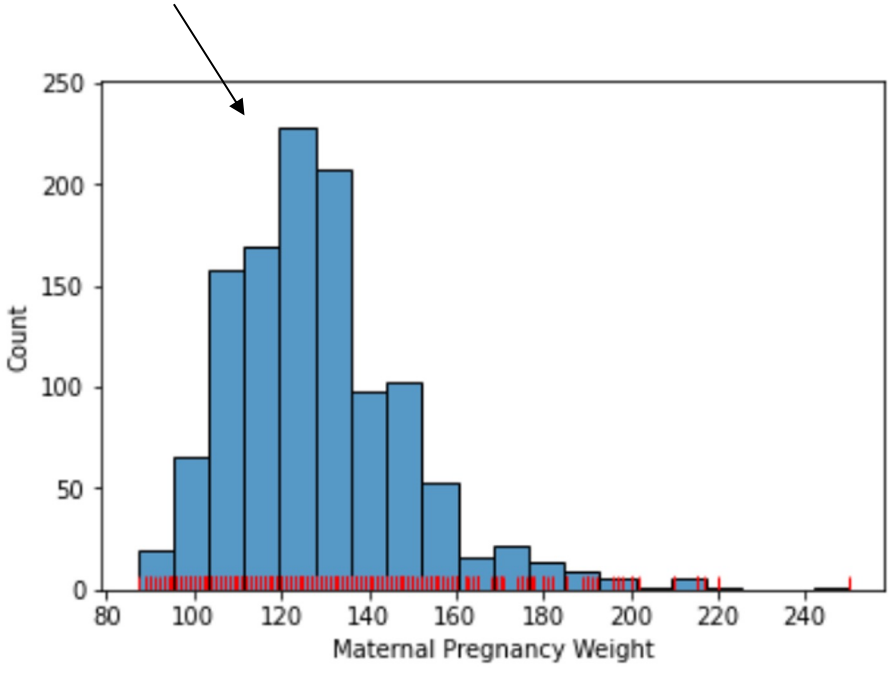
- A distribution with a single clear maximum is called unimodal.
- Distributions with two modes are called bimodal.
  - More than two: multimodal.
- Need to distinguish between **modes** and **random noise**.

Any ideas for how we can tell whether or not this is truly bimodal?

```
sns.histplot(data = births, x = 'Maternal Pregnancy Weight');
sns.rugplot(data = births, x = 'Maternal Pregnancy Weight', color = "red");
```

Modes

Unimodal?



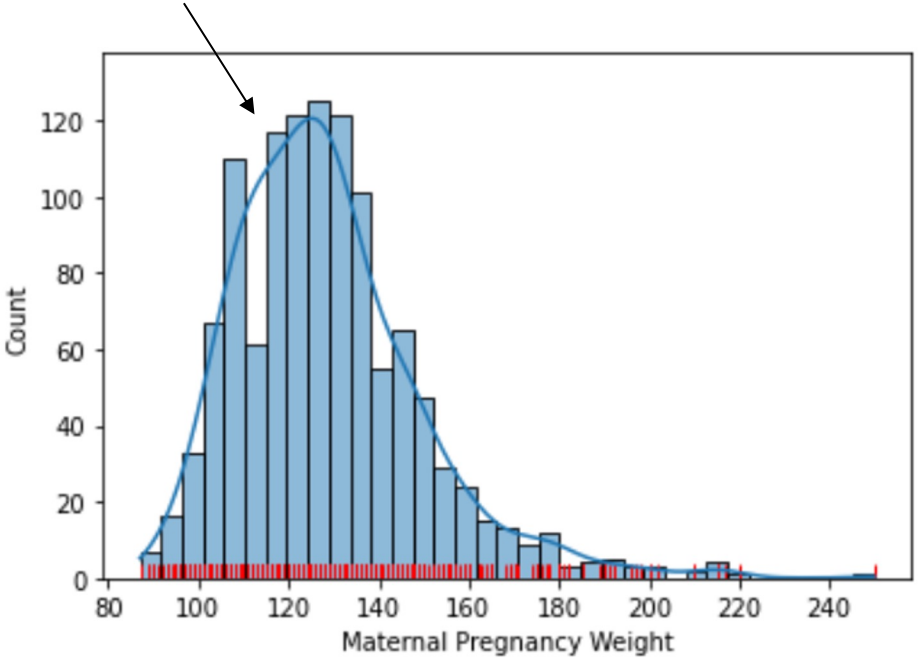
A **mode** of a distribution is a local or global maximum.

- A distribution with a single clear maximum is called unimodal.
- Distributions with two modes are called bimodal.
  - More than two: multimodal.
- Need to distinguish between **modes** and **random noise**.

```
sns.histplot(data = births, x = 'Maternal Pregnancy Weight', bins = 20);
sns.rugplot(data = births, x = 'Maternal Pregnancy Weight', color = "red");
```

Density curves

Unimodal



Instead of a discrete histogram, we can visualize what a continuous distribution corresponding to that same histogram could look like...

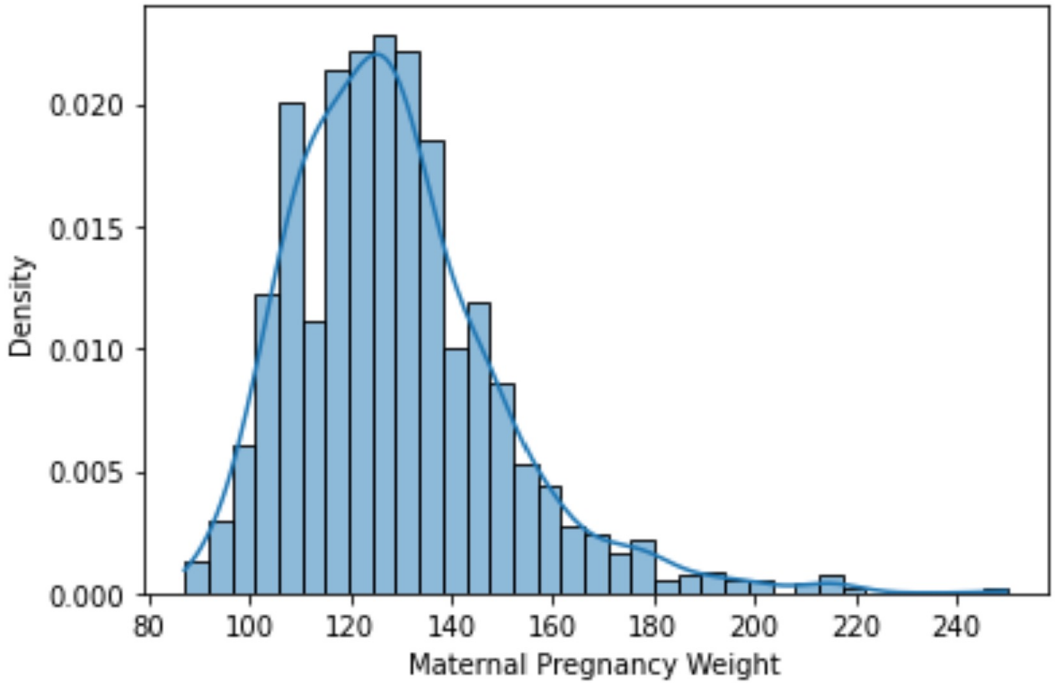
- The smooth curve drawn on top of the histogram here is called a **density curve**.

In later lectures , we will study how exactly to compute these density curves (using a technique is called Kernel Density Estimation).

```
sns.histplot(data = births, x = 'Maternal Pregnancy Weight', kde = True);
sns.rugplot(data = births, x = 'Maternal Pregnancy Weight', color = "red");
```

Reminder: Histograms and Density

Rather than labeling by counts, we can instead plot the density, as shown below:



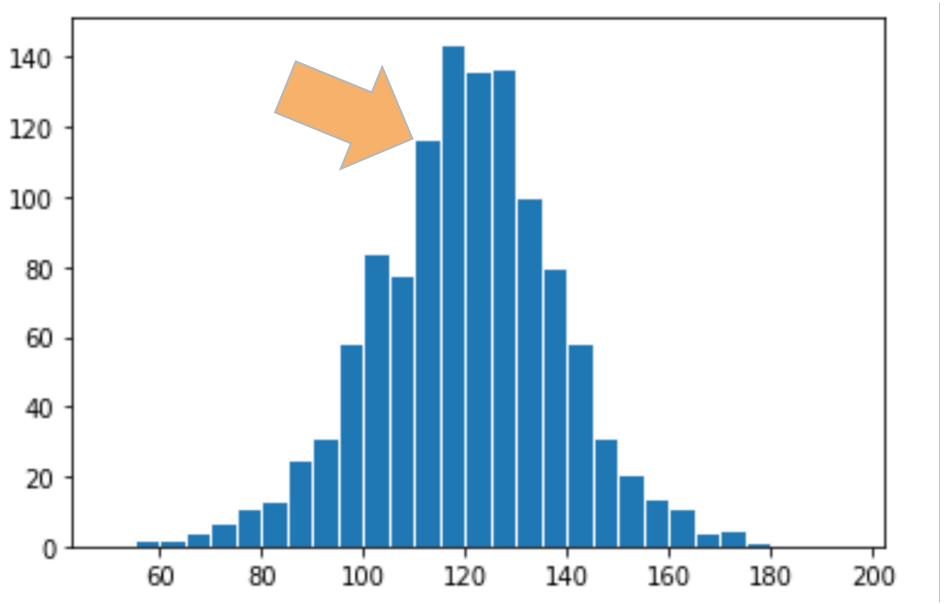

```
sns.histplot(data = births, x = 'Maternal Pregnancy Weight',
             kde = True, stat = "density");
```

Review Calculation: Computing Density from Counts and Bin Size

Approximately ~120 babies were born with a weight between 110 and 115.

There are 1174 observations total.

- Total area of this bin should be:
  - 120/1174 = ~10%
- Density of this bin is therefore:
  - 10% / (115 - 110) = 0.02

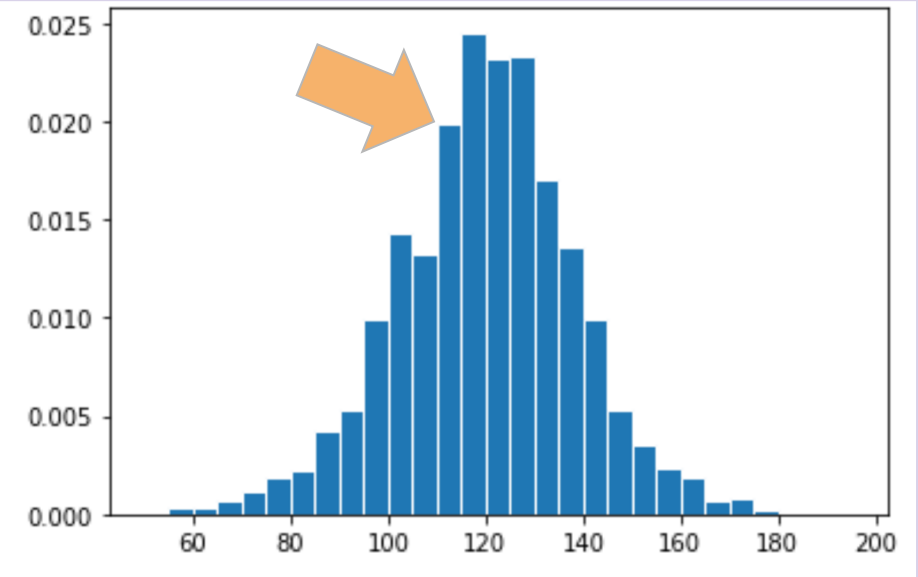


Review Calculation: Computing Count from Bin Size and Density

There are 1174 observations total.

- Width of bin [110, 115): 5
- Height of bar [110, 115): 0.02

How many observations are in this bin?

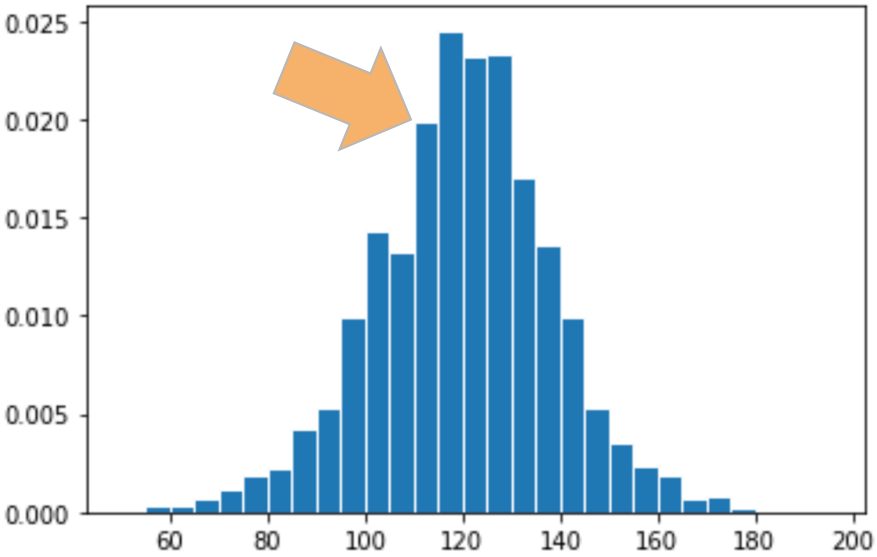


Review Calculation: Computing Count from Bin Size and Density

There are 1174 observations total.

- Width of bin [110, 115): 5
- Height of bar [110, 115): 0.02
- Proportion in bin = 5 * 0.02 = 0.1
- Number in bin = 0.1 * 1174 = **117.4**

This is roughly the number we got before (120)!



Box Plots and Violin Plots

- Visualizations In the Real World, Goals
- Bar Plots/Histograms
- for Distributions
- **Box Plots and Violin Plots**
- Comparing Quantitative Distributions
- Relationships Between Quantitative Variables

Quartiles

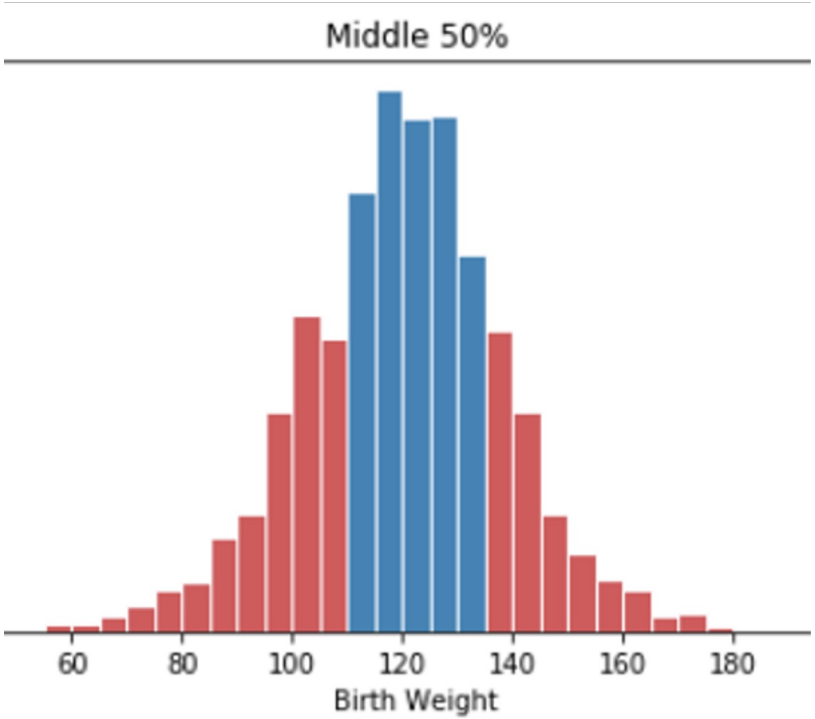
For a quantitative variable:

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

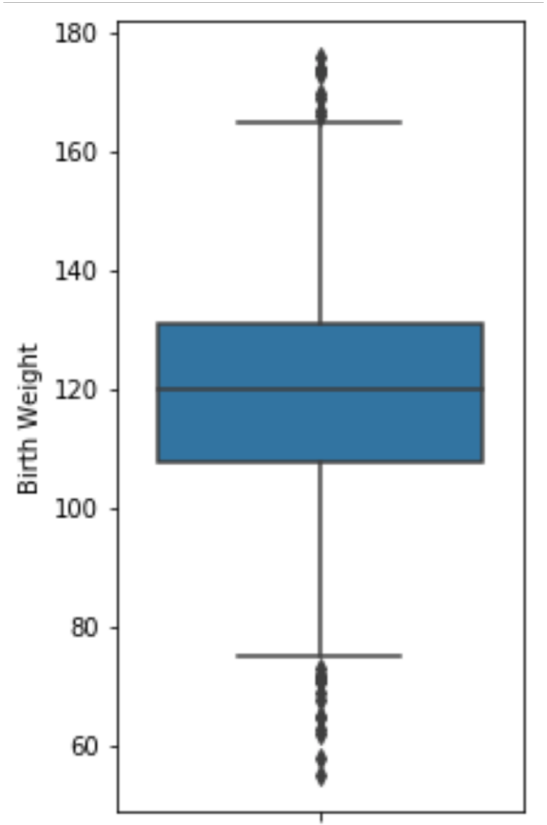
The interval [first quartile, third quartile] contains the “middle 50%" of the data.

Interquartile range (IQR) measures spread.

- IQR = third quartile – first quartile.



Box plots

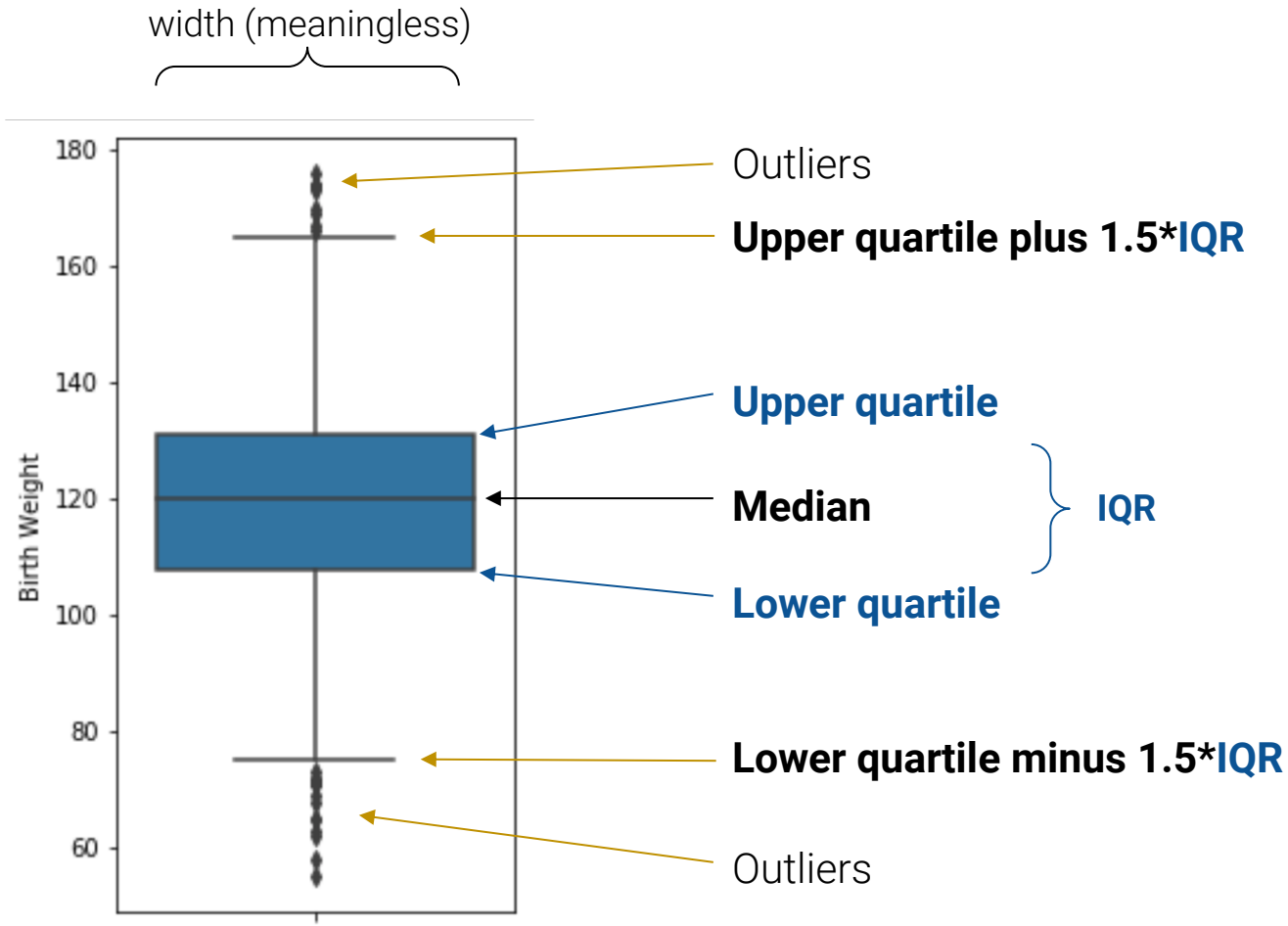


Box plots summarize several characteristics of a numerical distribution. They visualize:

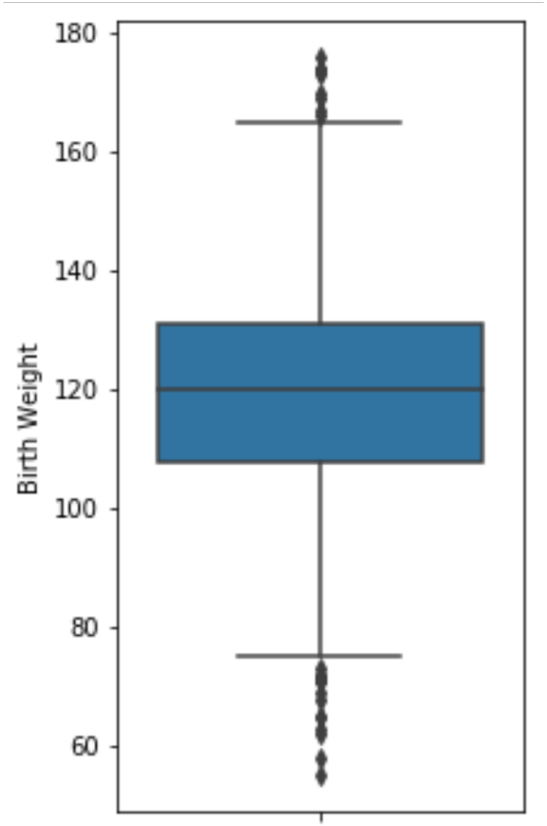
- **Lower quartile**.
- **Median**.
- **Upper quartile**.
- "**Whiskers**", placed at lower quartile minus 1.5*IQR and upper quartile plus 1.5*IQR.
- **Outliers**, which are defined as being further than 1.5*IQR from the extreme quartiles. Arbitrary definition!
- We lose a lot of information, too!

```
sns.boxplot(y = "Birth Weight", data = births)
```

Box plots


width (meaningless)
180
160
140
120
100
80
60
Birth Weight
Outliers
Upper quartile plus 1.5*IQR
Upper quartile
Median
IQR
Lower quartile
Lower quartile minus 1.5*IQR
Outliers

Box plots

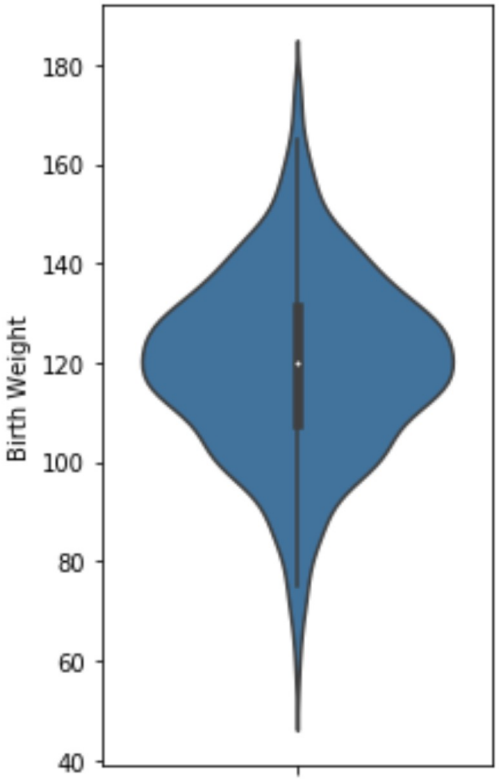



```
1  q1 = np.percentile(bweights, 25)
2  q2 = np.percentile(bweights, 50)
3  q3 = np.percentile(bweights, 75)
4  iqr = q3 - q1
5  whisk1 = q1 - 1.5*iqr
6  whisk2 = q3 + 1.5*iqr
7
8  whisk1, q1, q2, q3, whisk2
```

```
(73.5, 108.0, 120.0, 131.0, 165.5)
```

The five numbers above match what we see on the left.

Violin plots



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

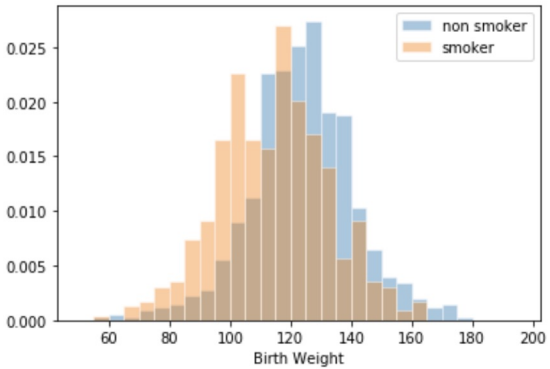
- The “width” of our “box” now has meaning!
- The three quartiles and “whiskers” are still present – look closely.

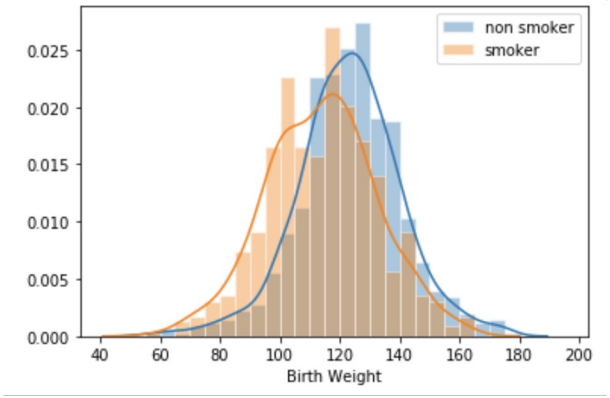
Next up: Box plots and violin plots are useful for comparing multiple distributions.

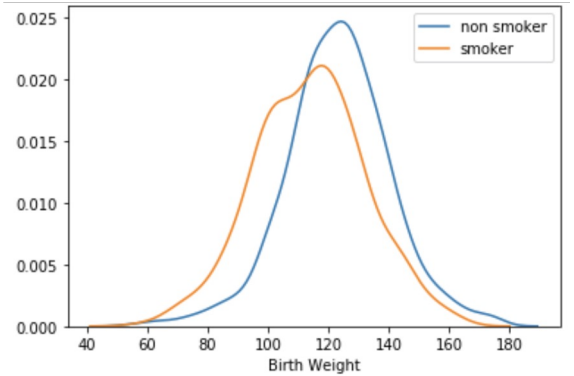
Comparing Quantitative Distributions

- Visualizations In the Real World, Goals
- Bar Plots/Histograms for Distributions
- Box Plots and Violin Plots
- **Comparing Quantitative Distributions**
- Relationships Between Quantitative Variables

Overlaid histograms and density curves



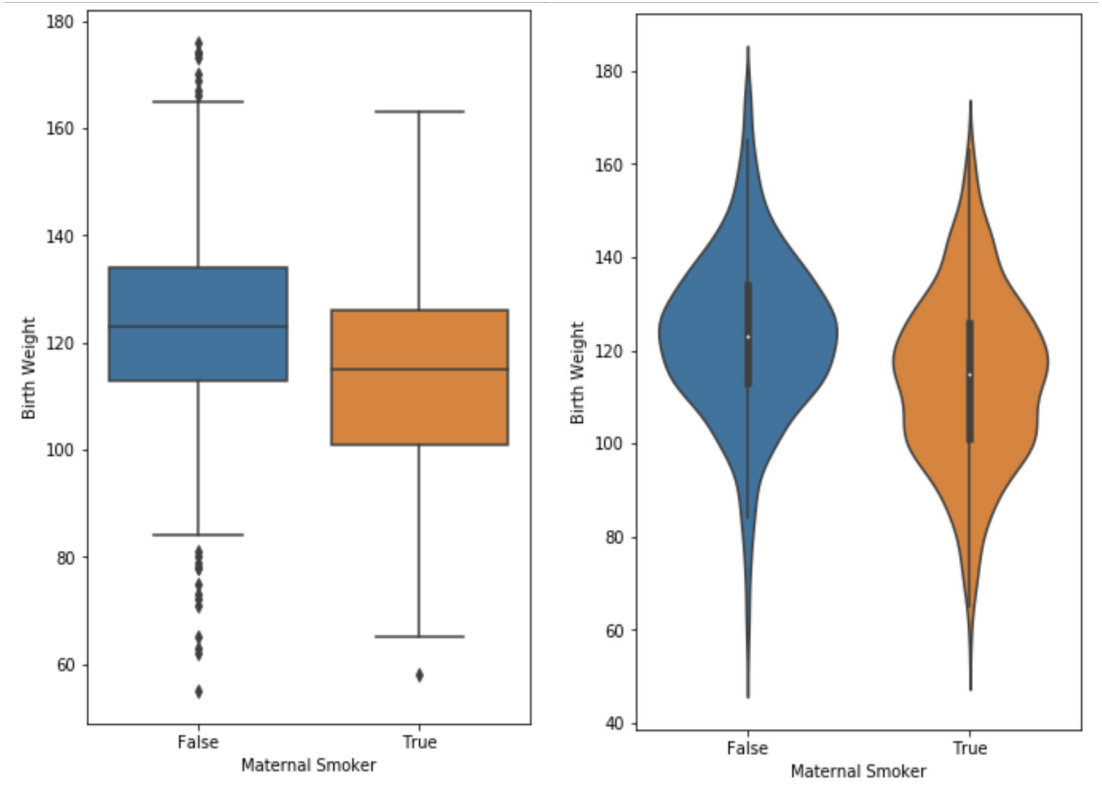




We can overlay multiple histograms and density curves on top of one another.

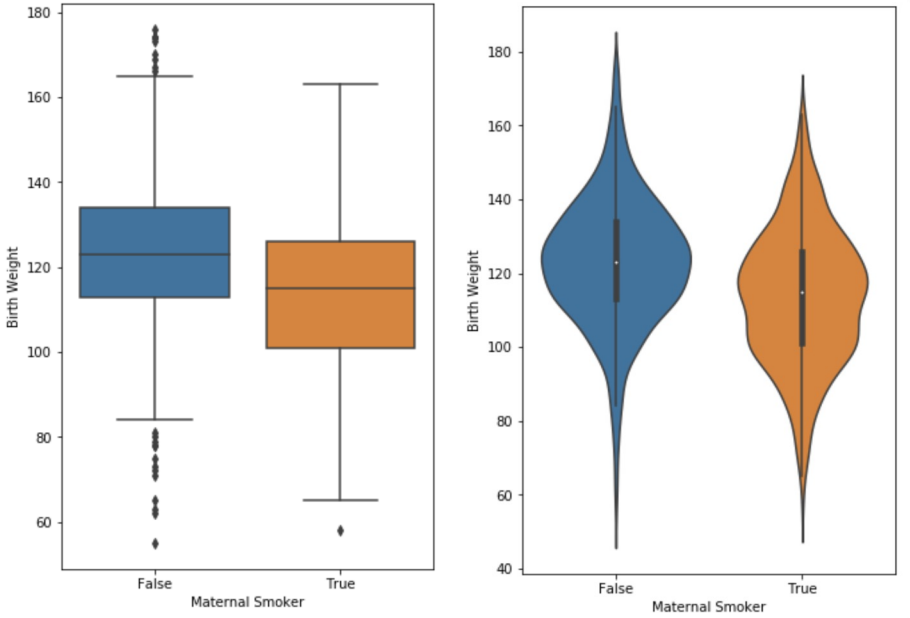
- First: Not terrible, but looks like three separate histograms.
- Second: Has the most information, but isn't very clear!
- Third: Rough estimate of both distributions, but is the most clear by far.
- Neither will generalize well to three or more categories.

Side by side box plots and violin plots




```
sns.boxplot(data=births, x='Maternal Smoker', y='Birth Weight')
sns.violinplot(data=births, x='Maternal Smoker', y='Birth Weight')
```

Side by side box plots and violin plots



Box plots and violin plots are concise, and thus are well suited to be stacked side by side to compare multiple distributions at once.

- At a glance, we can tell that the median birth weight is higher for babies whose mothers did not smoke while pregnant (“False”).
- The violin plot shows us the bimodal nature of the “True” category.

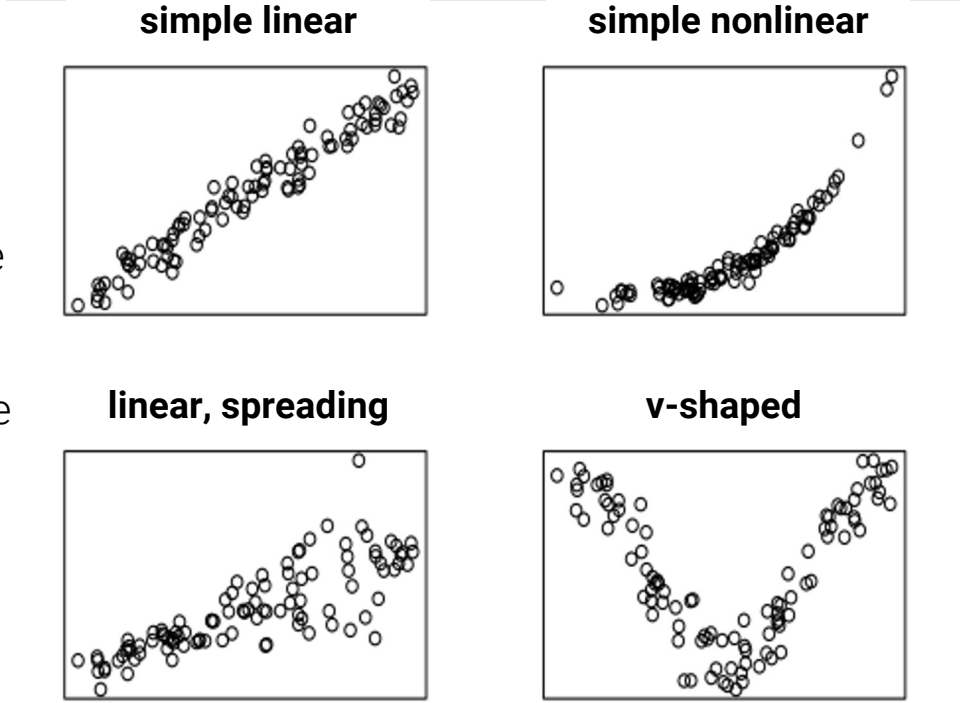
Relationships Between Quantitative Variables

- Visualizations In the Real World, Goals
- Bar Plots/ Histograms for Distributions
- Box Plots and Violin Plots
- Comparing Quantitative Distributions
- **Relationships Between Quantitative Variables**

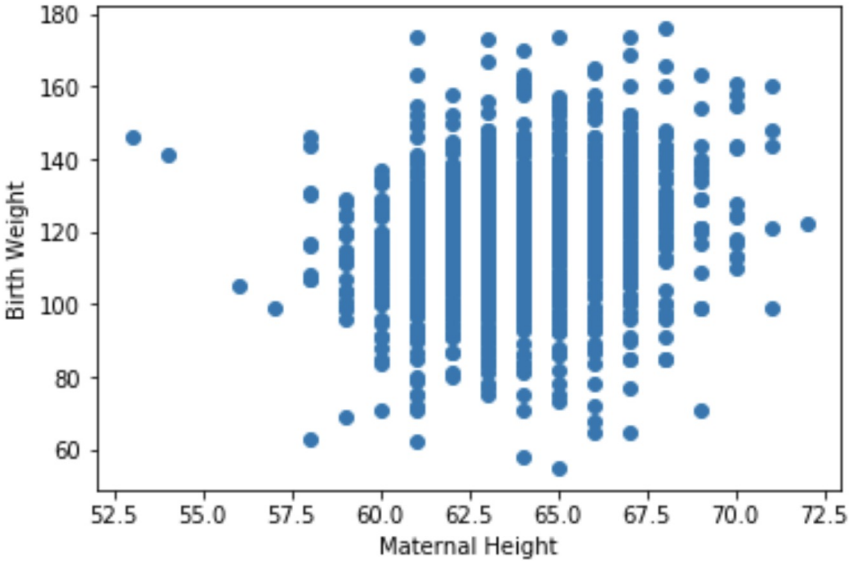
Scatter plots

Scatter plots are used to reveal relationships between pairs of numerical variables.

- The bottom left plot is labeled “linear, spreading” because the relationship appears linear, but with increasing spread as x gets larger.
- Visual assessment may help us decide what kind of model to build.
  - Example model: Simple Linear Regression example. Good for the left two, not so much for the right two.

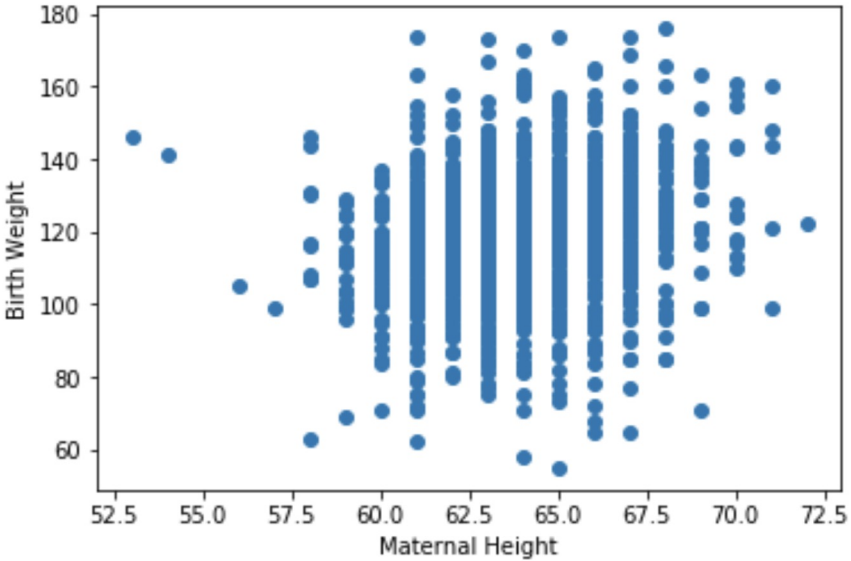


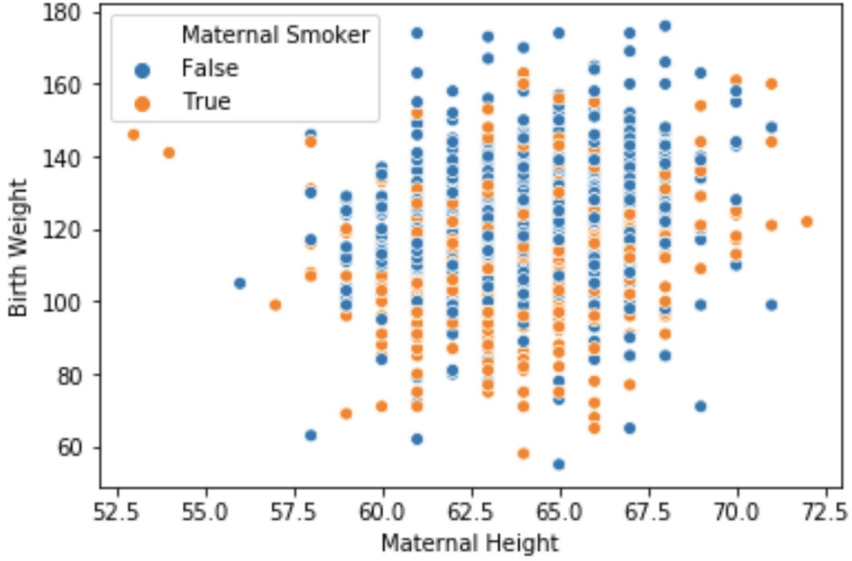
Scatter Plot on our Birth Data (Matplotlib Example)




```
plt.scatter(births['Maternal Height'], births['Birth Weight'])
plt.xlabel('Maternal Height')
plt.ylabel('Birth Weight')
```

Scatter Plot on our Birth Data (Seaborn Example)





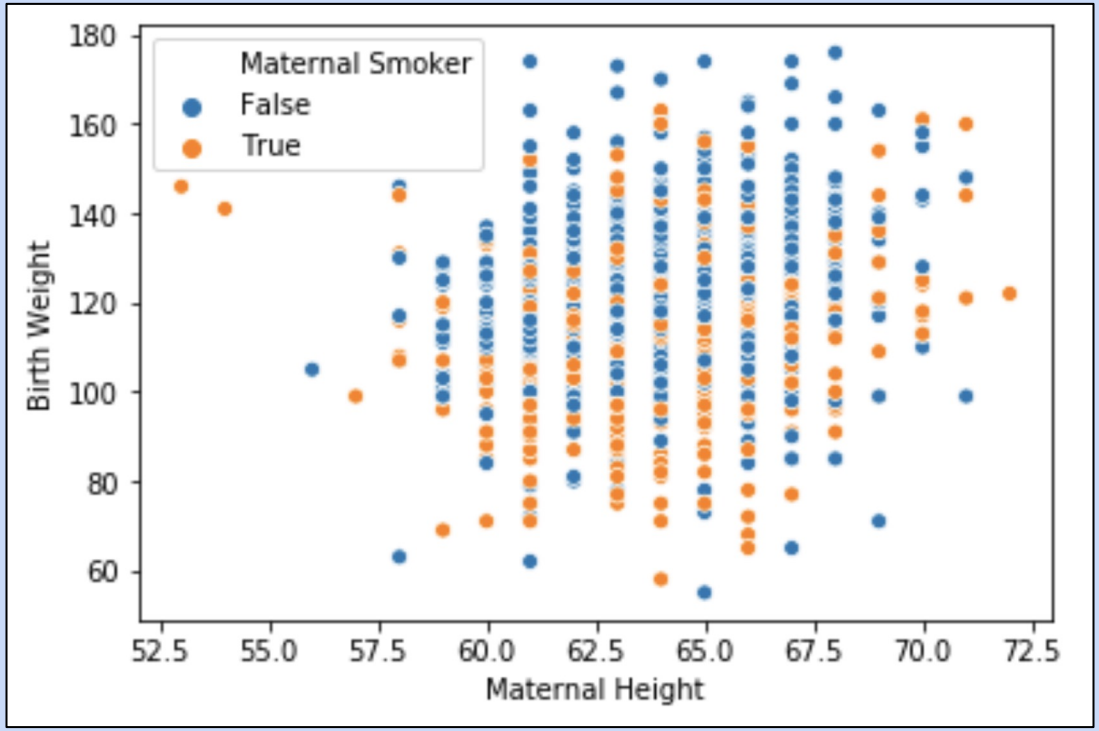

```
plt.scatter(births['Maternal Height'], births['Birth Weight'])
plt.xlabel('Maternal Height')
plt.ylabel('Birth Weight')
```

```
sns.scatterplot(data = births,
                x = 'Maternal Height',
                y = 'Birth Weight',
                hue = 'Maternal Smoker')
```

The Seaborn example on the right uses **color** to add a **third dimension** to our plot!

- Unlike earlier plots, the color represents information present nowhere else.
- Note that our box plot and violin plots were MUCH better assessments of these two distributions. Harder to see the lower weight for **True** babies.

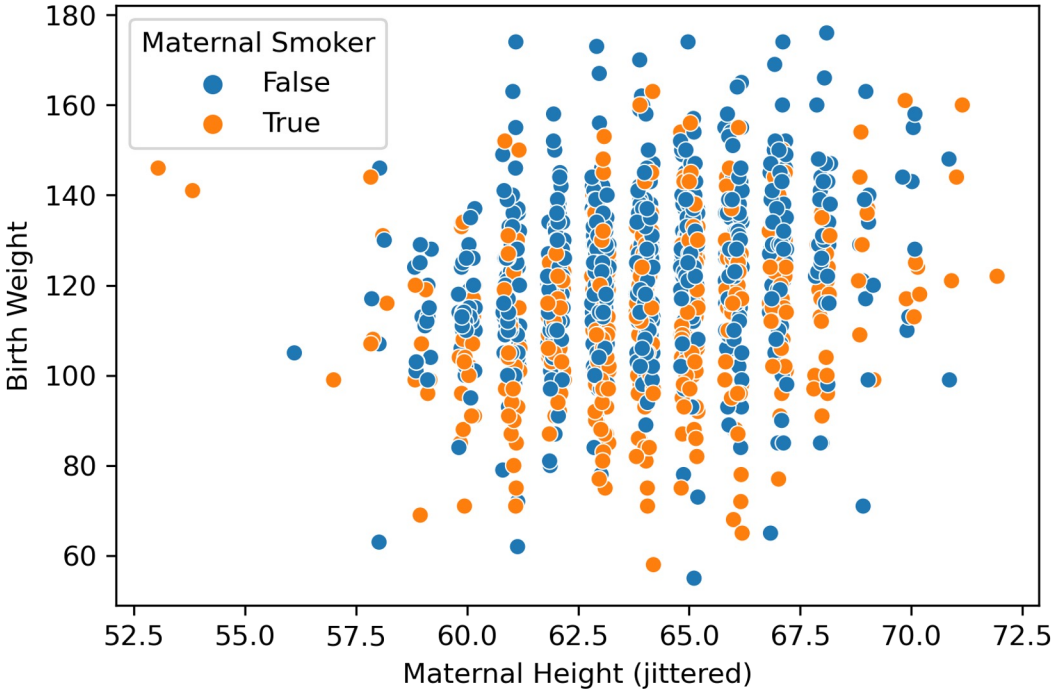
Scatter Plot on our Birth Data (Seaborn Example)



This plots suffers from overplotting – many of the points are on top of one another!

- Any suggestions for fixing this? (still has to be a scatter plot)

Scatter Plot on our Birth Data (Seaborn Example)

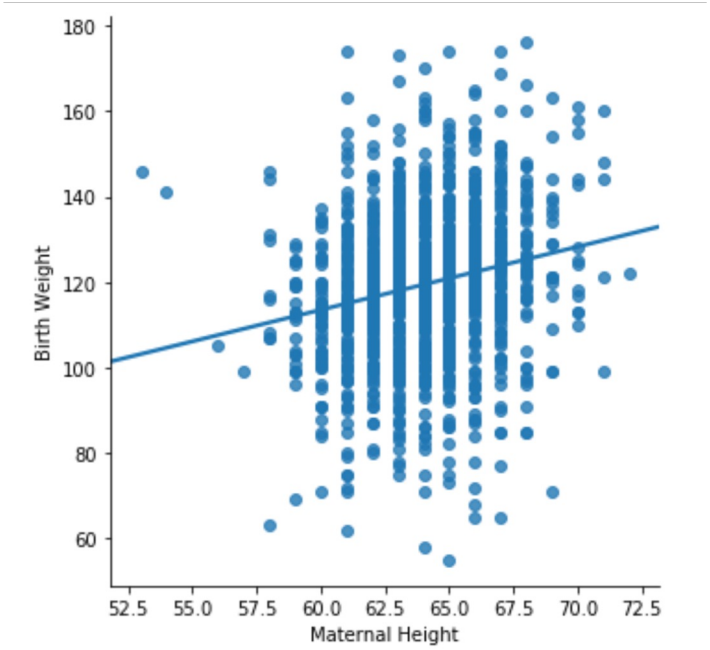


One solution: Add some random noise to the x variable.

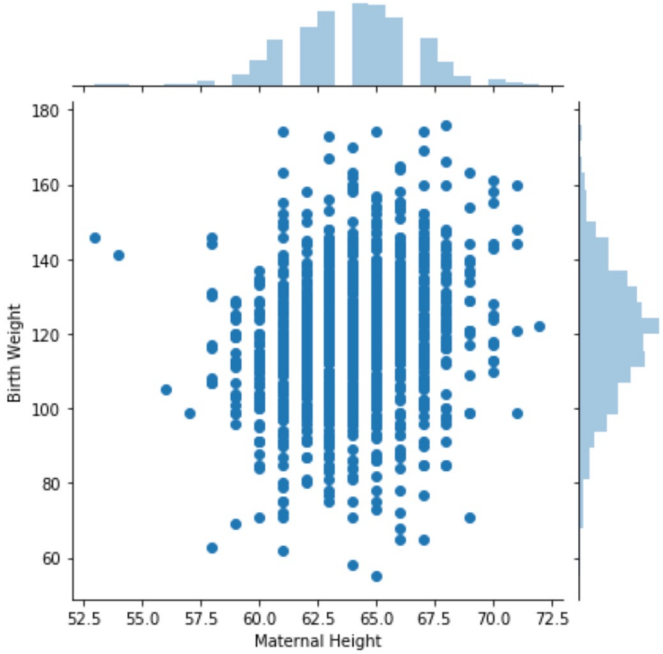
- Is this really reasonable???

```
births["Maternal Height (jittered)"] = births["Maternal Height"] + np.random.uniform(-0.2, 0.2, len(births))
```

Scatter plots




```
sns.lmplot(data=births, x='Maternal Height', y='Birth Weight', ci=False)
```




```
sns.jointplot(data=births, x='Maternal Height', y='Birth Weight')
```

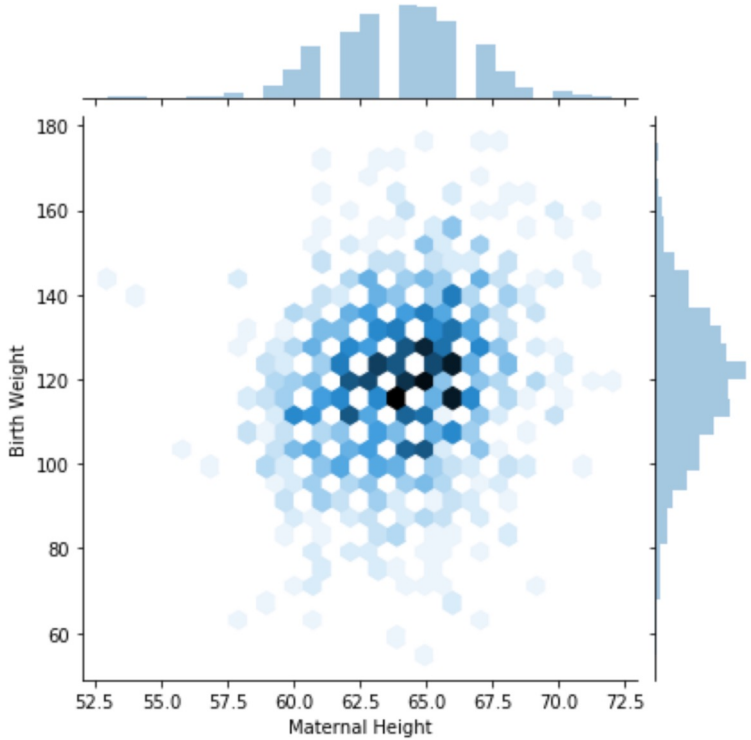
Hex plots

Can be thought of as a two dimensional histogram. Shows the joint distribution.

- The xy plane is binned into hexagons.
- More shaded hexagons typically indicate a greater density/frequency.

Why hexagons instead of squares?

- Easier to see linear relationships.
- More efficient for covering region.
- Visual bias of squares – drawn to see vertical and horizontal lines.




```
sns.jointplot(data=births, x='Maternal Height',
y='Birth Weight', kind='hex')
```

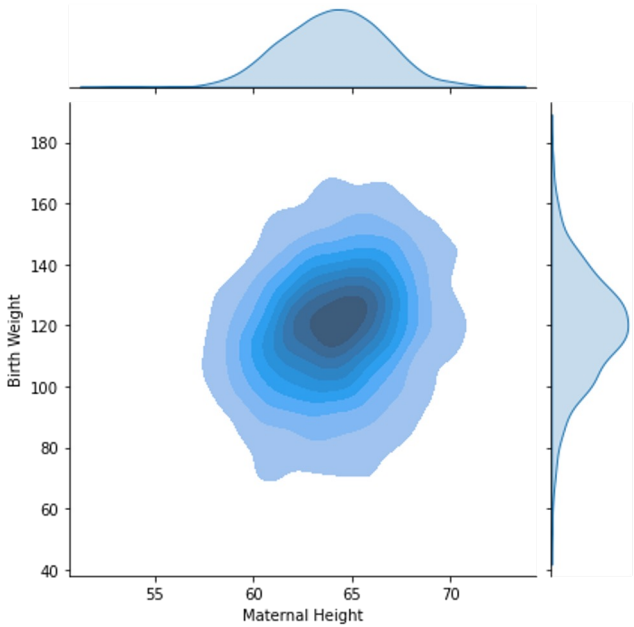
Contour plots

Contour plots are two dimensional versions of density curves.

- Will reappear when we study gradient descent!

Each of the last few plots has been created by **`sns.jointplot`**.

- By default, shows **marginal** distributions on the horizontal and vertical axes.
- These are the histograms/density curves of each variable independently.




```
sns.jointplot(data=births, x='Maternal Height',
y='Birth Weight', kind='kde', fill=True)
```

Kernel density estimation (KDE)

Kernel Density Estimation is used to estimate a **probability density function** (or **density curve**) from a set of data.

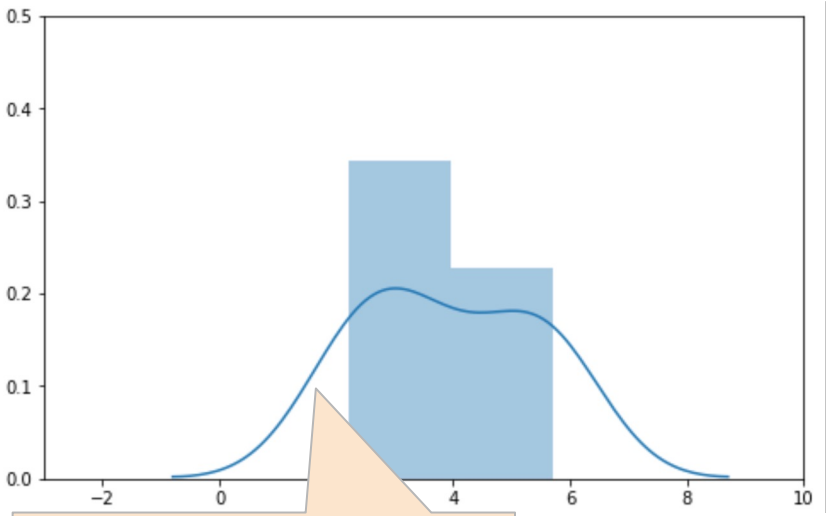
- Just like a histogram, a **density curve**'s total area must sum to 1.

To create a KDE:

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

To generate a curve we need to choose a **kernel** and **bandwidth.**

We will formally define "**probability density function**" later in the class.

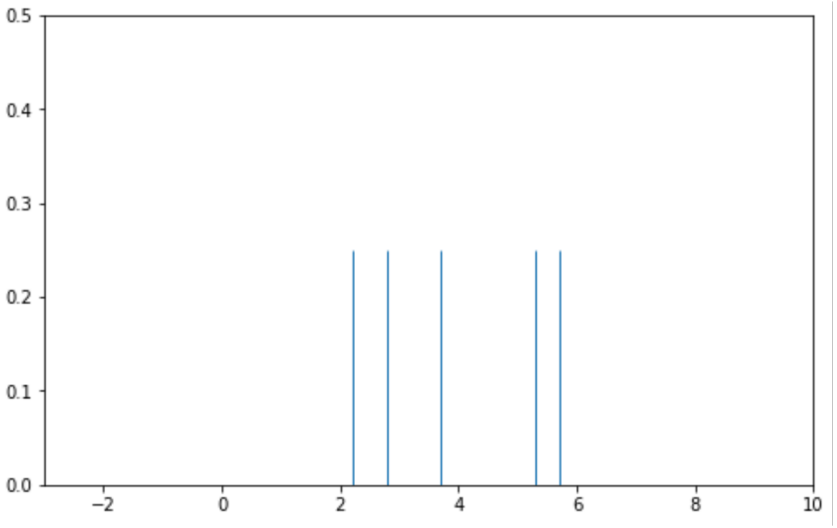


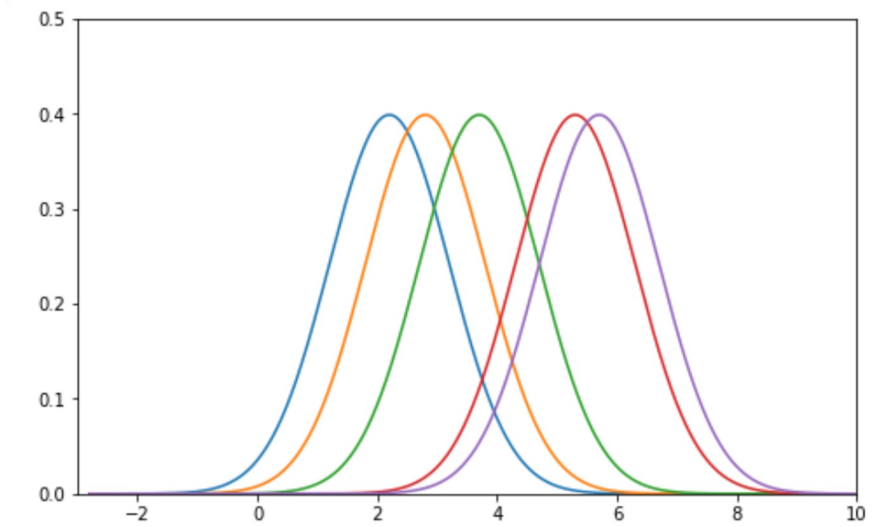
Our goal is to recreate this smooth curve ourselves.

Step 1 – place a kernel at each data point

At each of our 5 points (depicted in the rug plot on the left), we've placed a **Gaussian kernel** with **bandwidth** of **alpha = 1**. The idea is that there is a higher density near the points we've already seen.

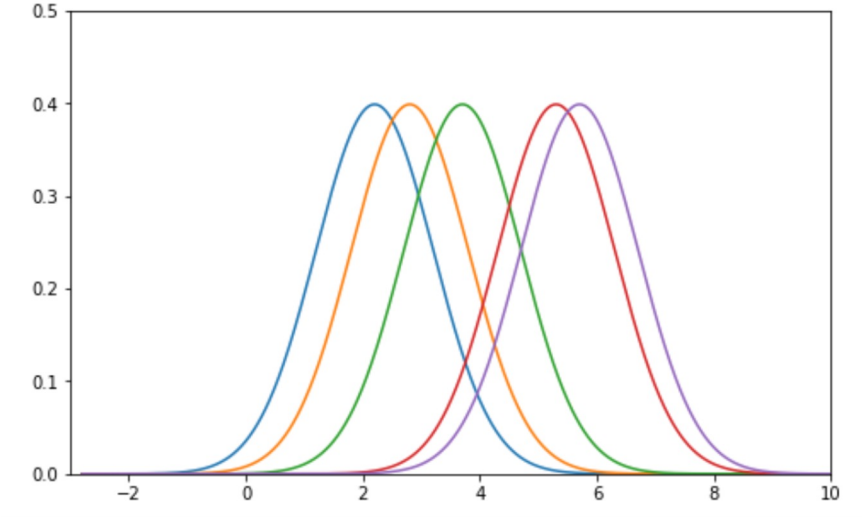
- We will precisely define a **Gaussian kernel** and **alpha** in a few slides.

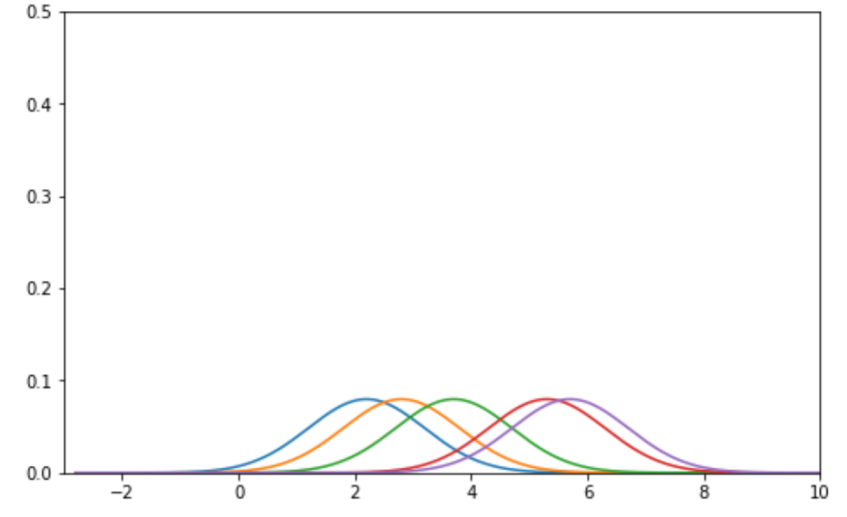




Step 2 – normalize kernels

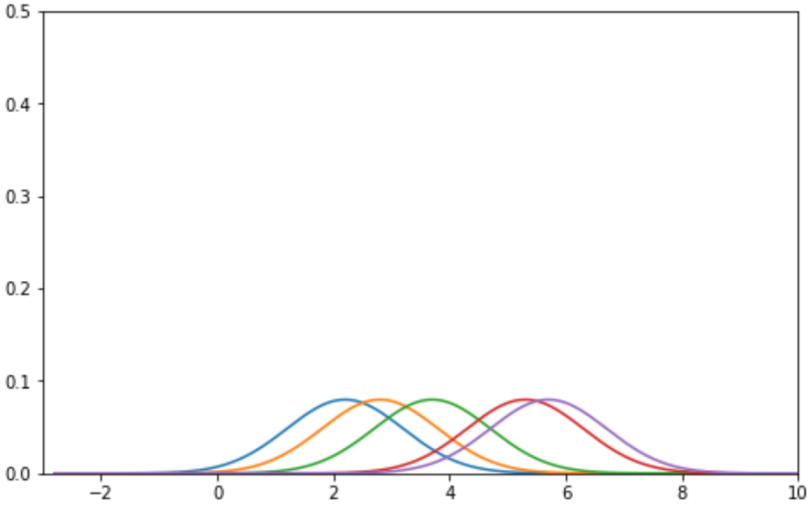
In Step 3, we will be summing each of these **kernels**. We want the result to be a valid density, that has area 1. Right now, we have 5 different **kernels**, each with an area 1. So, we **multiply each kernel by 1/5**.

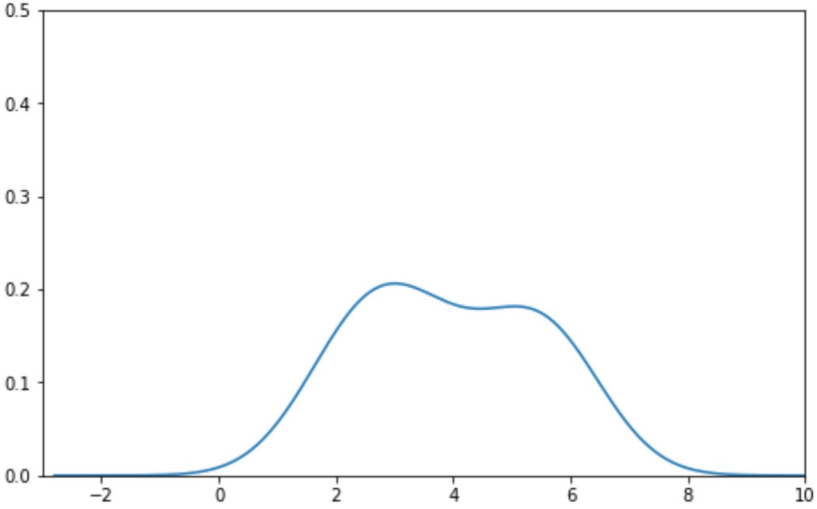




Step 3 – sum kernels

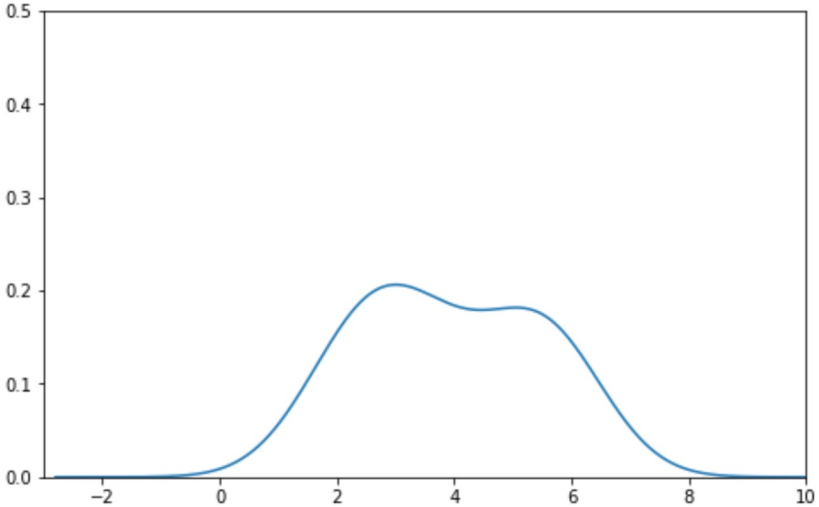
Our **kernel density estimate (KDE)** is the **sum of the normalized kernels at each point.** It is depicted below on the right.

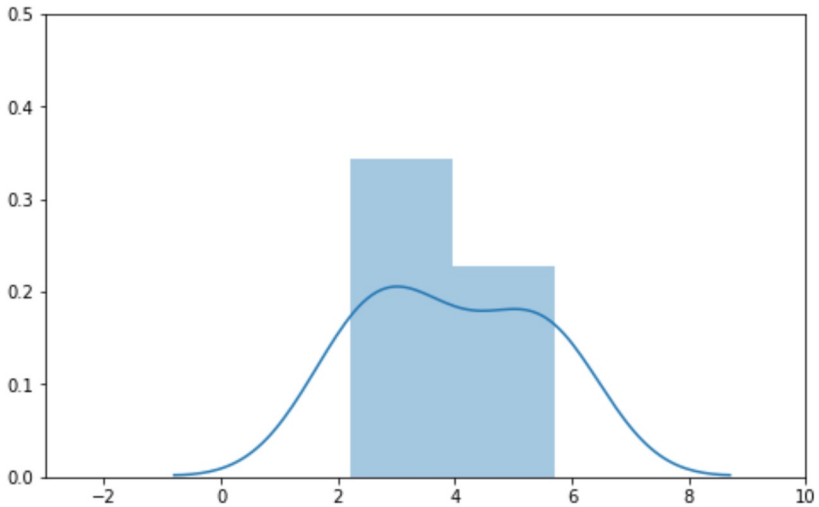




Kernel density estimates

The curve we manually created (left) exactly matches the one that `sns.distplot` creates for us (right)!





Kernels

A kernel (for our purposes) is a valid density function. That means it:

- Must be non-negative for all inputs.
- Must integrate to 1.

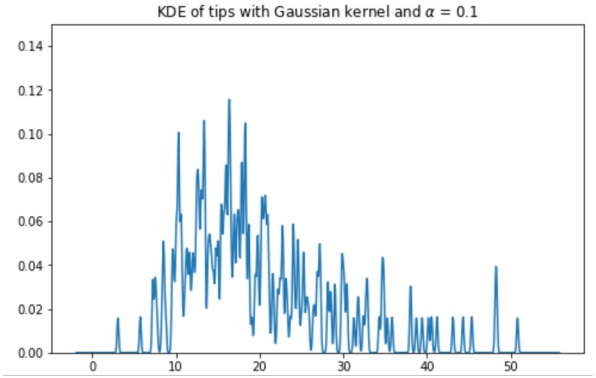
The most common **kernel** is the **Gaussian kernel**.

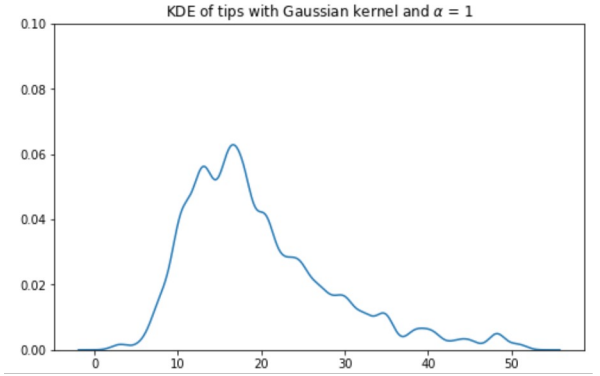
- Here, x represents any input, and x_i represents the ith observed value. The kernels are centered on our observed values (and so the mean of this distribution is x_i).
- **α** is the **bandwidth parameter**. It controls the smoothness of our KDE. Here, it is also the standard deviation of the Gaussian.

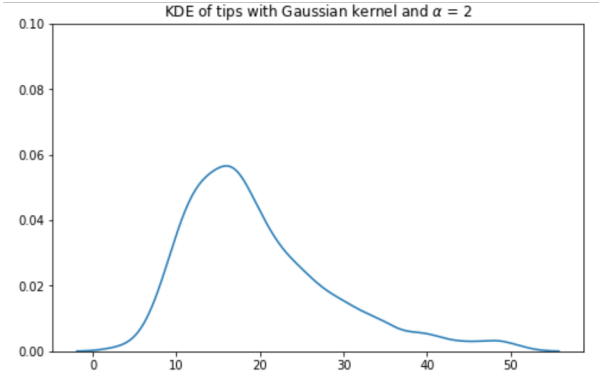
$$K_\alpha(x, x_i) = \frac{1}{\sqrt{2\pi\alpha^2}} e^{-\frac{(x-x_i)^2}{2\alpha^2}}$$

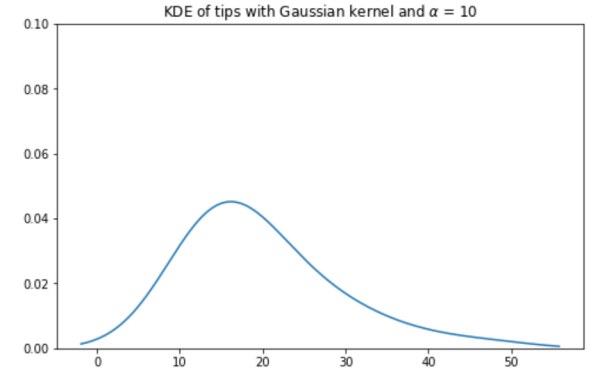
If you've taken a probability class, the mean of the kernel for the ith observed value is x_i, and the standard deviation of the kernel is **α.**

Effect of bandwidth on KDEs



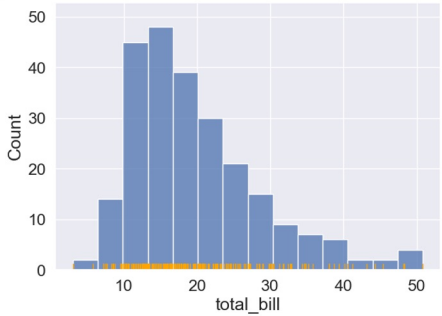






Bandwidth is analogous to the width of each bin in a histogram.

- 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).



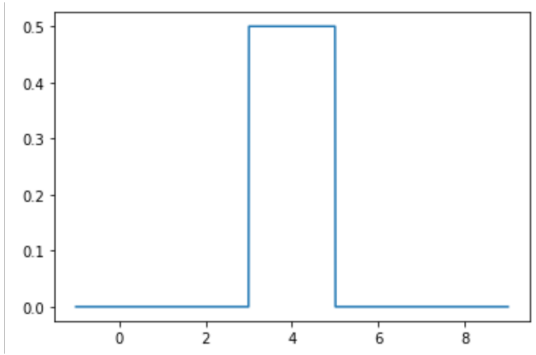
Kernels

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

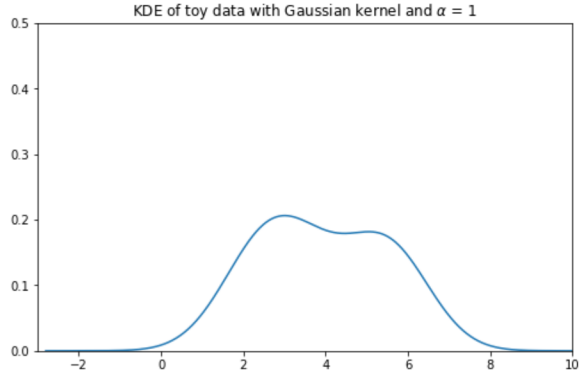
- It assigns uniform density to points within a "window" of the observation, and 0 elsewhere.
- Resembles a histogram… sort of.

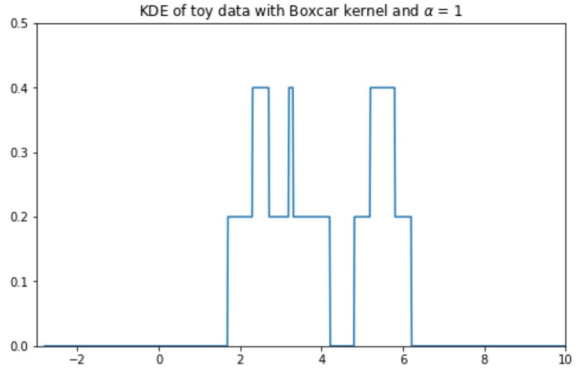
$$K_\alpha(x, x_i) = \begin{cases} \frac{1}{\alpha}, & |x - x_i| \leq \frac{\alpha}{2} \\ 0, & \text{else} \end{cases}$$

- For even more **kernels**, see https://en.wikipedia.org/wiki/Kernel_(statistics)



A **boxcar kernel** centered on $x_i = 4$ with $\boldsymbol{\alpha} = 2$.





Kernels in Practice

Despite a great deal of literature in statistics on ***kernel*** properties (e.g. the **Epanechnikov kernel** has some nice theoretical properties), the libraries we use in this class only support a **Gaussian kernel**.

seaborn.kdeplot ¶

`seaborn.kdeplot` (*x=None, *, y=None, shade=None, vertical=False, kernel=None, bw=None, gridsize=200, cut=3, clip=None, legend=True, cumulative=False, shade_lowest=None, cbar=False, cbar_ax=None, cbar_kws=None, ax=None, weights=None, hue=None, palette=None, hue_order=None, hue_norm=None, multiple='layer', common_norm=True, common_grid=False, levels=10, thresh=0.05, bw_method='scott', bw_adjust=1, log_scale=None, color=None, fill=None, data=None, data2=None, warn_singular=True, **kwargs*) ¶

Plot univariate or bivariate distributions using kernel density estimation.

kernel : *str*

Function that defines the kernel.

Deprecated since version 0.11.0: support for non-Gaussian kernels has been removed.

Summary of KDE

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

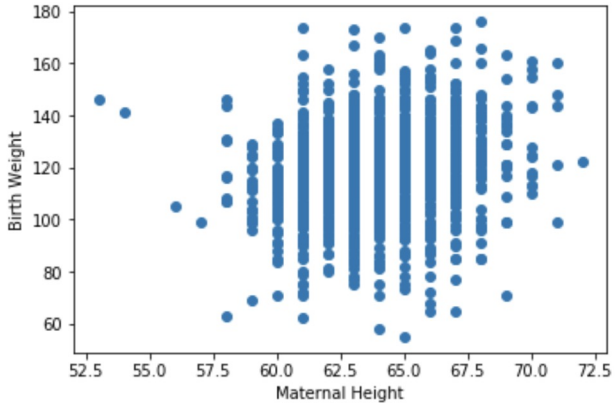
A general "KDE formula" function is given above. We will not cover this in lecture.

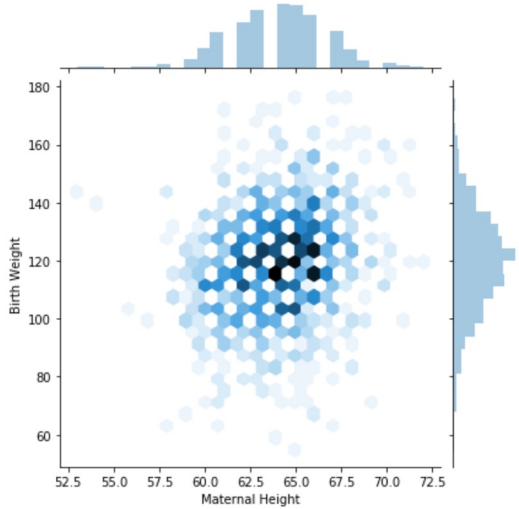
- $\mathbf{x}$ represents any number on the number line. It is the input to our function.
- $\mathbf{n}$ is the number of observed data points that we have.
- Each $\mathbf{x}_i$ ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$) represents an observed data point. These are what we use to create our KDE.
- $\boldsymbol{\alpha}$ is the bandwidth or smoothing parameter**.**
- $K_\alpha(x, x_i)$ is the kernel centered on the observation i.
    - Each kernel individually has area 1. We multiply by $\mathbf{1/n}$ so that the total area is still 1.

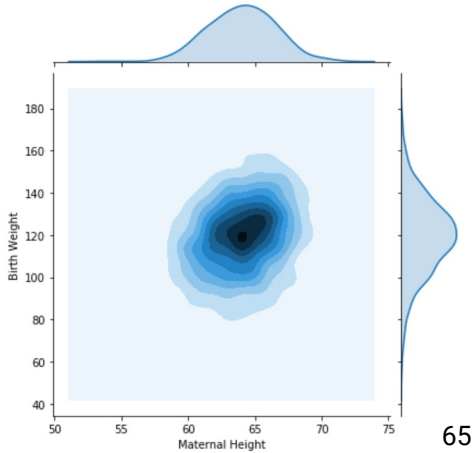
Generalizing to 2D (Contour Map)

We can also do the same thing analogously in 2D.

- Rug Plot :: Histogram :: KDE
- Scatter Plot :: Hex Plot :: Contour Map



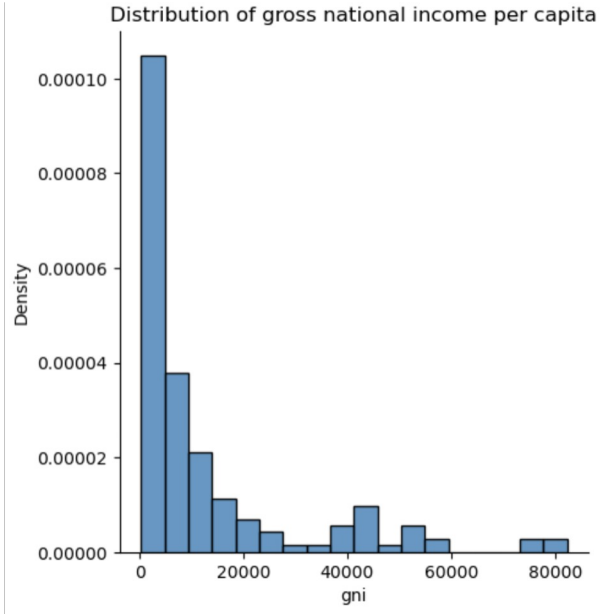




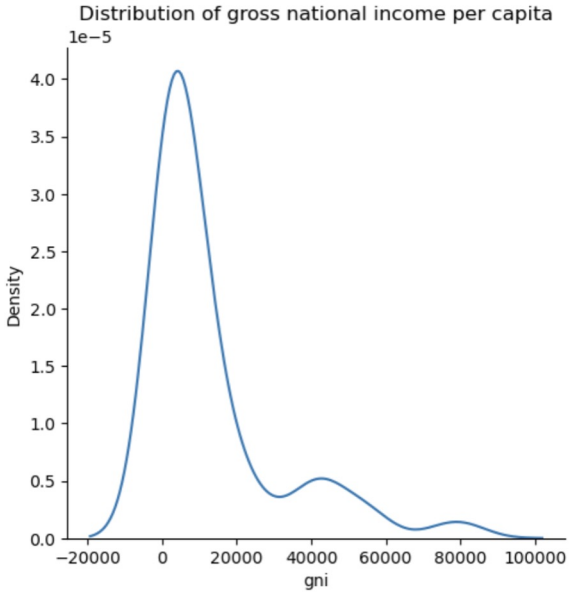
displot

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

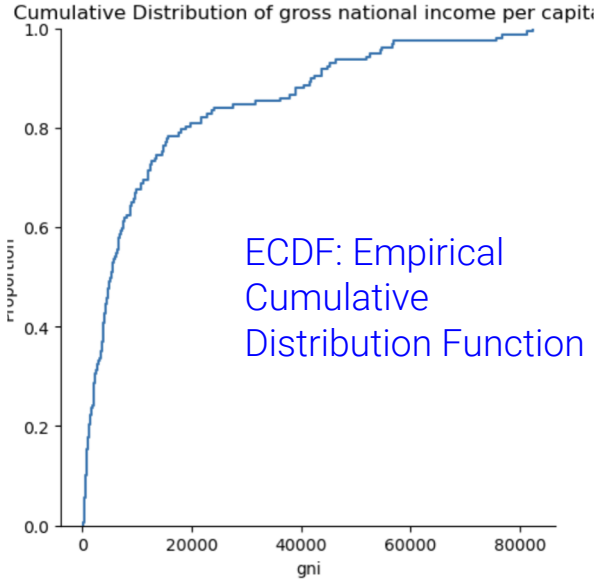
```
sns.displot(data=wb,
            x="gni",
            kind="hist",
            stat="density")
```




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




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



ECDF: Empirical Cumulative Distribution Function

Extra: interactive visualization on the web - D3

- https://d3js.org/
- D3 is a javascript library (D3.js) that is free for both noncommercial and commercial use
- Overall process
  - loads data into the browser's memory
  - binds data to elements within the document
  - sets visual properties of each elements
  - transition elements between states in response to user input

- Interactive Data Visualization (https://www.oreilly.com/library/view/interactive-data-visualization/9781449340223/)

- SVG works well with D3 too

Extra: interactive visualization on the web - ECharts

- An Open Source JavaScript Visualization Library
- It was developed by Baidu, one of the largest Chinese web services companies, and is now widely used worldwide. (Credit to Chatgpt!)
- A wide range of chart types, including line charts, bar charts, pie charts, scatter plots, radar charts, and more.
- Supports a variety of data formats, including JSON, CSV, and Excel, making it easy to integrate with your existing data sources.

Summary

- **Visualization requires a lot of thought!**
- Many tools for visualizing distributions.
  - Distribution of a single variable: rug plot, histogram, density plot, box, violin.
  - Joint distribution of two quantitative variables: scatter plot, hex plot, contour plot.
- This class primarily uses seaborn and matplotlib.
  - Pandas also has basic built-in plotting methods.
  - Many other visualization libraries exist. **`plotly`** is one of them.
    - It creates **interactive** plots very easily .
- For more interactions on the web, check out D3/ ECharts/ Etc…

In the next lecture we'll go deeper into the theory behind visualization.