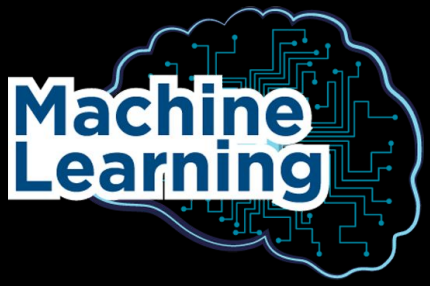
Visualization

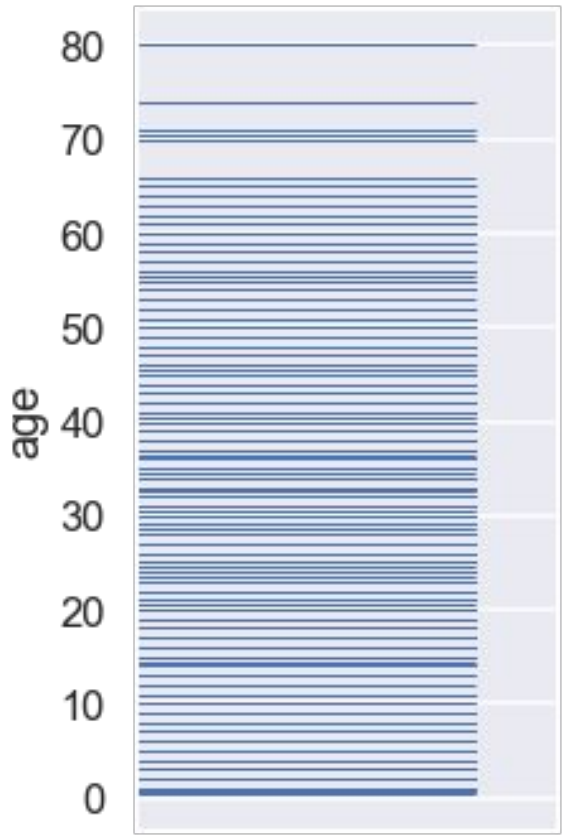


Seyed Abbas Hosseini
Sharif University of Technology

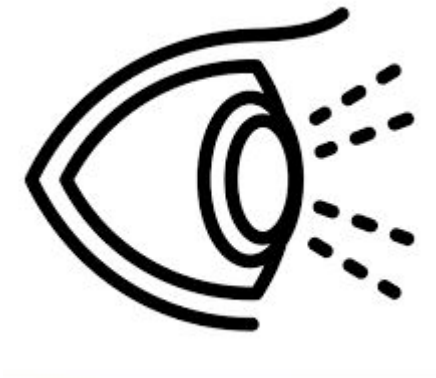
Adapted from Berkeley DS 100 Course in summer 2020 by Suraj Rampure

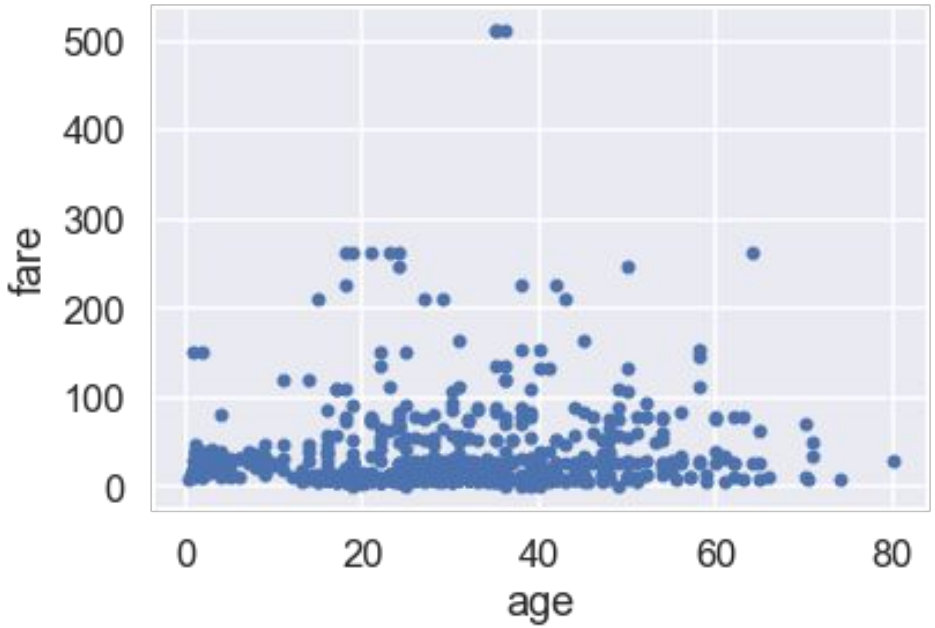
Computer readable vs. human readable

	age
0	22.0
1	38.0
2	26.0
...	...
888	NaN
889	26.0
890	32.0



Visualizations are for humans





“Looks like older people didn’t spend more than younger people.”

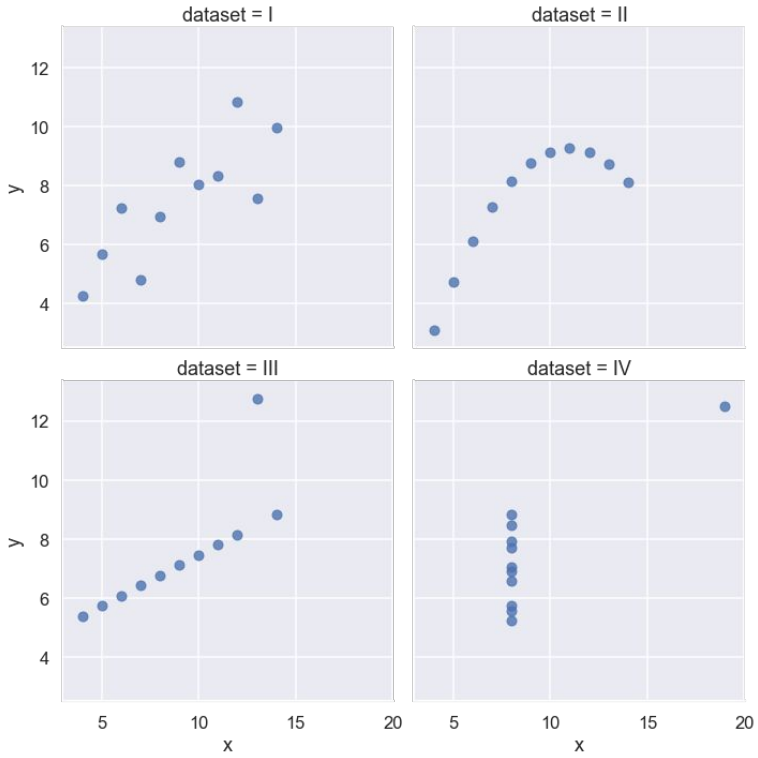
Visualize, then quantify!

x	y
10.0	8.04
8.0	6.95
13.0	7.58
9.0	8.81
11.0	8.33
14.0	9.96
6.0	7.24
4.0	4.26
12.0	10.84
7.0	4.82
5.0	5.68

x	y
10.0	9.14
8.0	8.14
13.0	8.74
9.0	8.77
11.0	9.26
14.0	8.10
6.0	6.13
4.0	3.10
12.0	9.13
7.0	7.26
5.0	4.74

x	y
10.0	7.46
8.0	6.77
13.0	12.74
9.0	7.11
11.0	7.81
14.0	8.84
6.0	6.08
4.0	5.39
12.0	8.15
7.0	6.42
5.0	5.73

x	y
8.0	6.58
8.0	5.76
8.0	7.71
8.0	8.84
8.0	8.47
8.0	7.04
8.0	5.25
19.0	12.50
8.0	5.56
8.0	7.91
8.0	6.89



Anscombe's Quartet

Each of these datasets has the same means, standard deviations, and correlation. As we will see in a few lectures, this means they have the same regression line.

Visualization complements statistics.

Why data visualization?

- One goal of data science is to inform human decisions.
  - Excellent plots **directly** address this goal.
  - Sometimes the most useful results from data analysis are the visualizations!
- Data visualization isn't as simple as calling `plot()`.
  - Many plots are possible, but only a few are useful!
  - Every visualization has tradeoffs.

Roadmap:

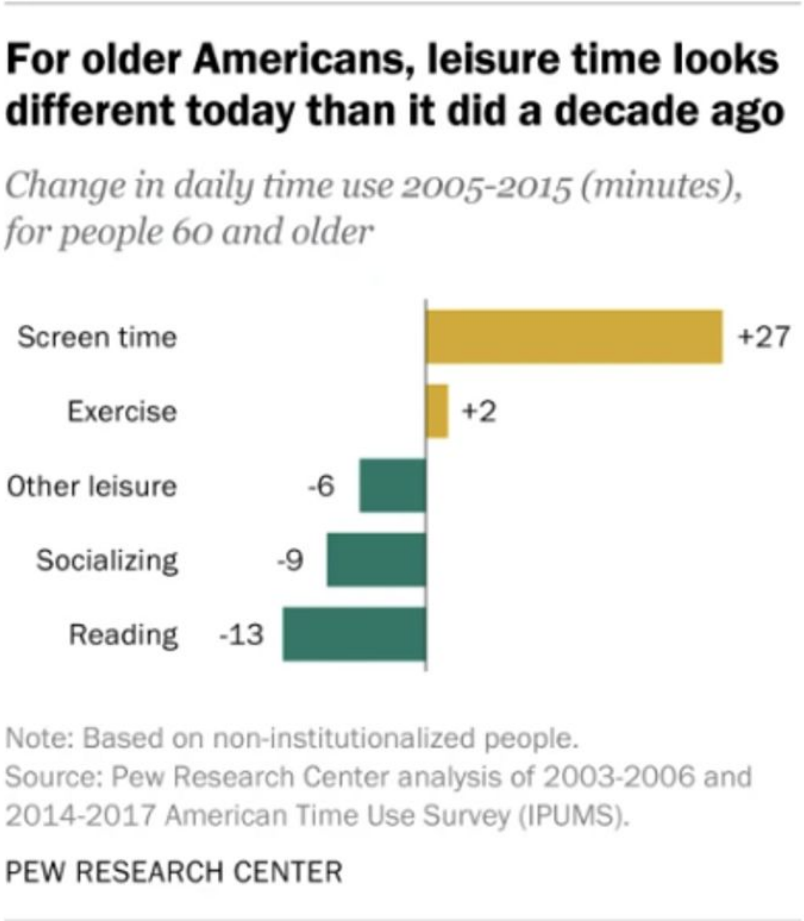
- Today: Establish when to use certain types of visualizations.
- Next lecture: Discuss various principles of visualization, along with kernel density estimation and transformation.

What is a distribution?

- 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.

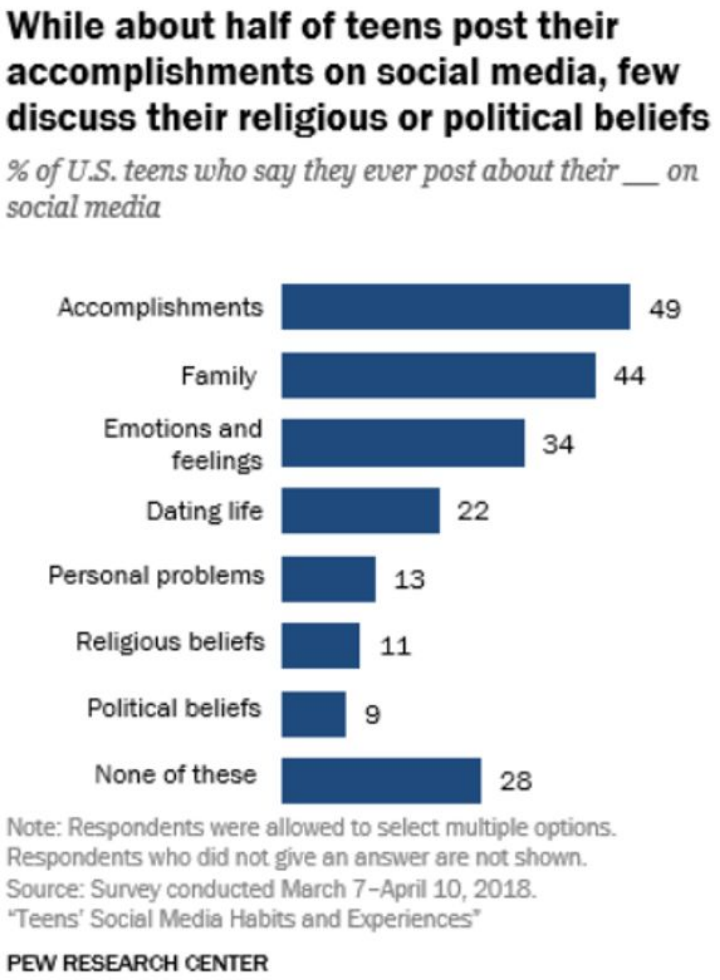
Does this chart show a distribution?



No.

- Individuals can be in more than one category.
- The numbers (and bar lengths) correspond to "time", not the proportion or number of individuals in the category.

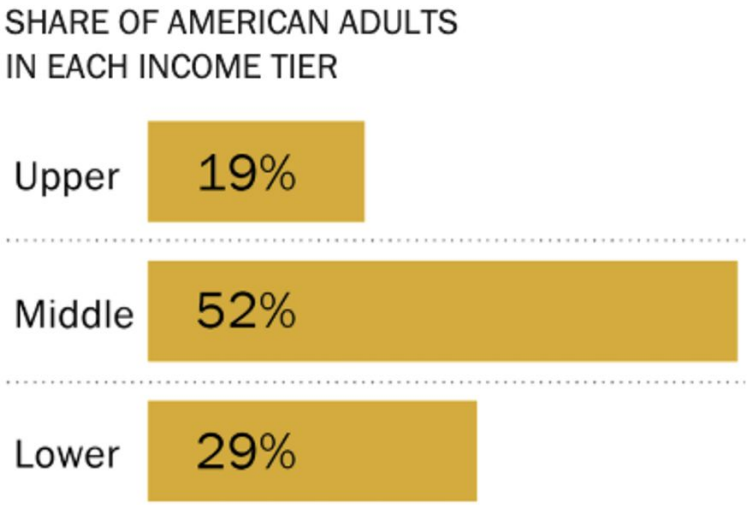
Does this chart show a distribution?



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).

Does this chart show a distribution?



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%.

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.
- They are also used to display a numerical variable that has been measured on individuals in different categories.
  - For example, the average GPAs of students at Berkeley in several majors.
  - Not a distribution! But bar plots still make sense.
- 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 from Data 8 for most of our plots today. Here is what that looks like.

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

```
births.head()
```

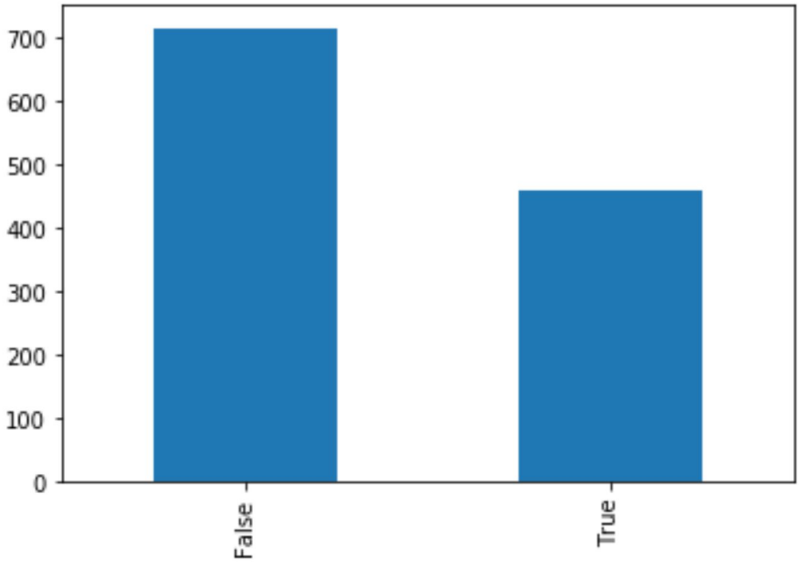
	Birth Weight	Gestational Days	Maternal Age	Maternal Height	Maternal Pregnancy Weight	Maternal Smoker
0	120	284	27	62	100	False
1	113	282	33	64	135	False
2	128	279	28	64	115	True
3	108	282	23	67	125	True
4	136	286	25	62	93	False

```
births.shape
```

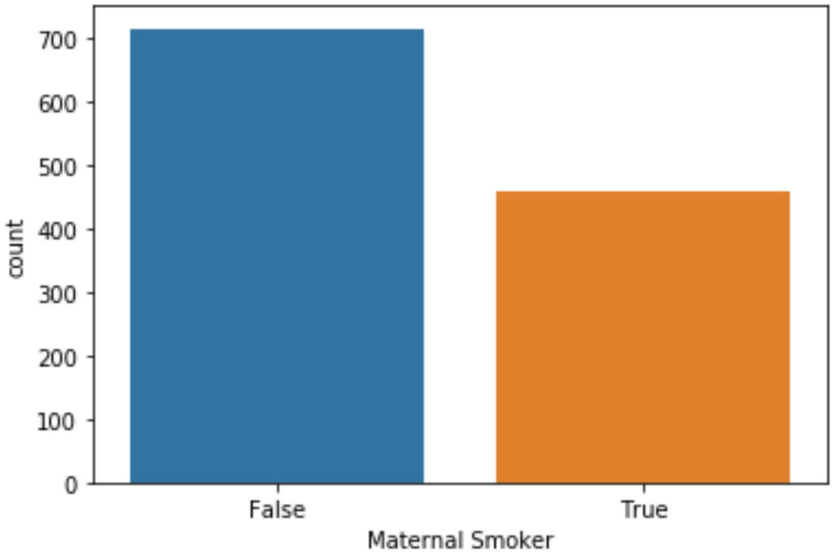
```
(1174, 6)
```

Bar plots

Suppose **births[‘Maternal Smoker’]** is a series containing True and False. Then:



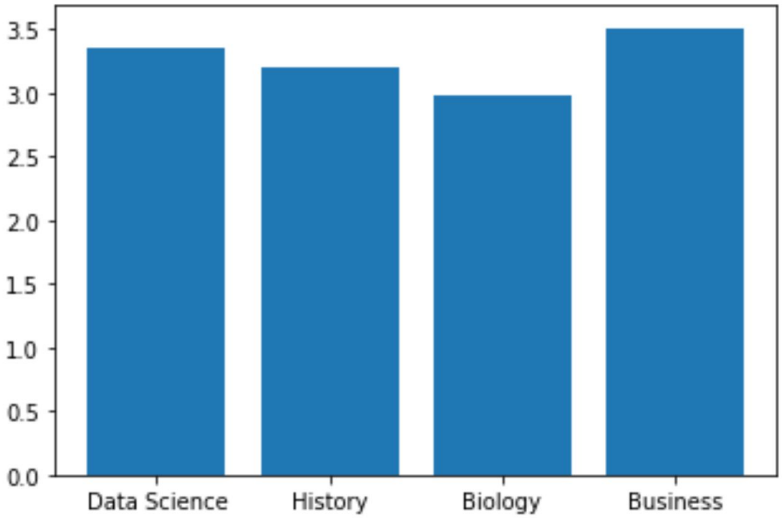

```
births['Maternal Smoker']
.value_counts().plot(kind =
'bar');
```




```
sns.countplot(births[‘Maternal
Smoker’])
```

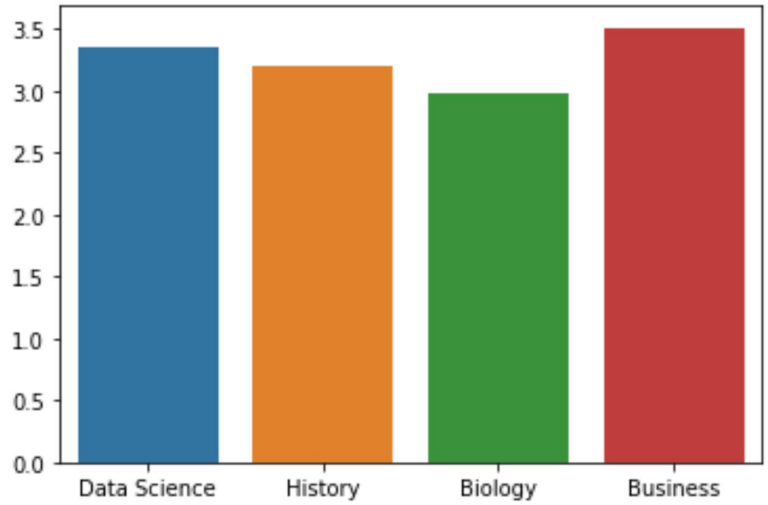
Bar plots

Suppose we have a list of majors and a list of gpas corresponding to those majors. Then:



`plt.bar(majors, gpas)`

To make horizontal:
`plt.hbar(majors, gpas)`



`sns.barplot(majors, gpas)`

Note: Here, color is meaningless.

Three ways to plot

- matplotlib (**`plt`**)
  - The underlying plotting library powering all three of these.
- pandas **`.plot()`**
  - Knows how to make some default plots for you!
- seaborn (**`sns`**)
  - Allows us to create sophisticated visualizations quickly.
  - Not just a colorful version of matplotlib!
- There are several other ways, but these are what we'll focus on.
- Moving forward, we won't necessarily show you all of the ways to plot something.
  - But we will give you the code for at least one way!
  - Play around with arguments in the supplemental notebook.

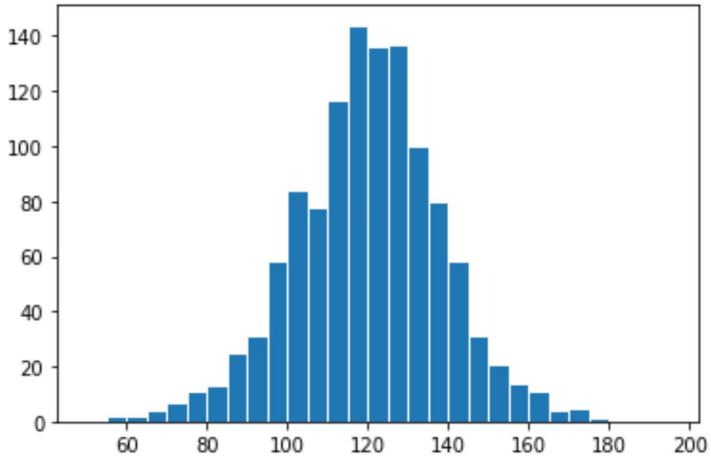
Histograms

- Histograms can be thought of as a smoothed version of a rug plot.
  - Lose granularity, but gain interpretability.
- Horizontal axis: the number line, divided into **bins**.
- **Areas represent proportions**!
  - Total area = 1 (or 100%).
- Units of height: proportion per unit on the x-axis.
  - Can be seen by dividing the above equation by “width of bin”.

$$\boxed{\text{proportion in bin} = \text{width of bin} \cdot \text{height of bar}}$$

Histograms

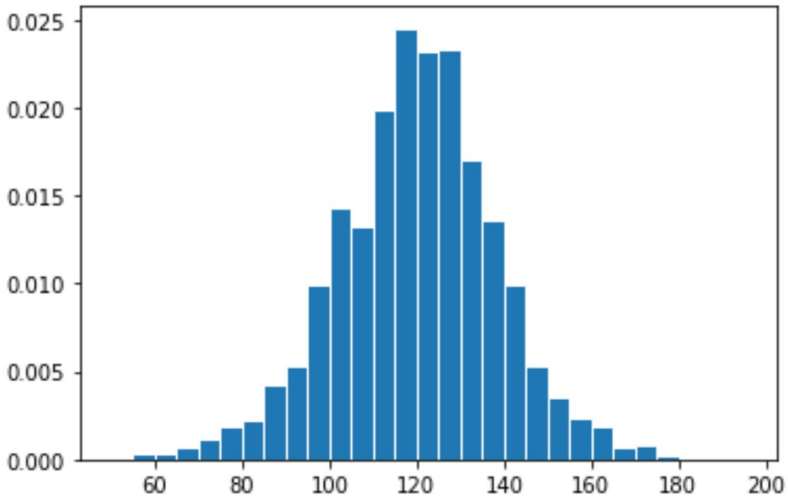
By default, **matplotlib** histograms show counts on the y-axis, not proportions per unit.




```
plt.hist(bweights,
bins=bw_bins, ec='w')
```

where `bw_bins = range(50, 200, 5)`

We use the optional **density** parameter to fix the y-axis. After doing this, the total area sums to 1.

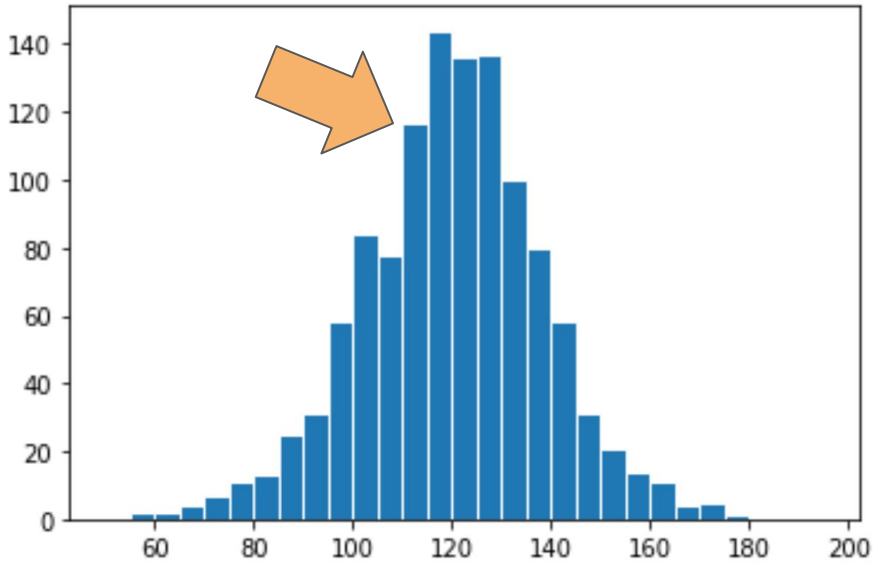



```
plt.hist(bweights, density=True,
bins=bw_bins, ec='w')
```

Example calculation

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

There are 1174 observations total.



Example calculation

There are 1174 observations total.

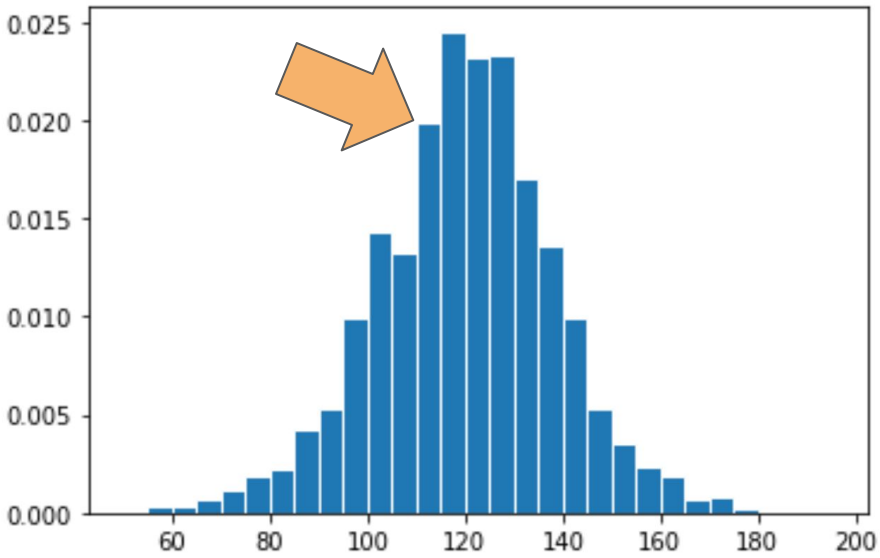
Width of bin [110, 115): 5

Height of bar [110, 115): 0.02

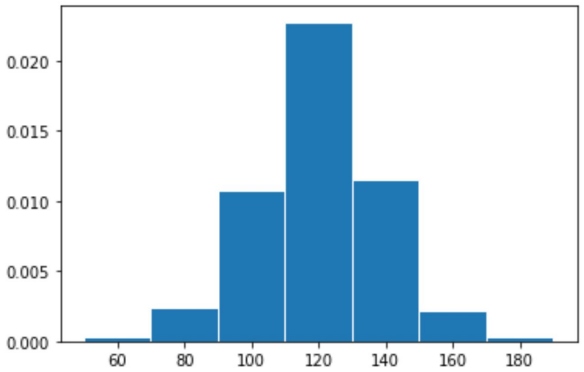
Proportion in bin = 5 * 0.02 = 0.1

Number in bin = 0.1 * 1174 = **117.4**

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

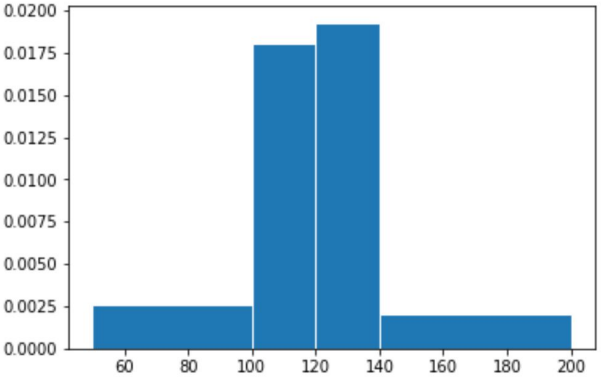


Histograms



If we decrease the number of bins (or increase bin width), we lose some granularity. This can be fine, depending on the purpose of the graph. **There's no "right number" of bins.**

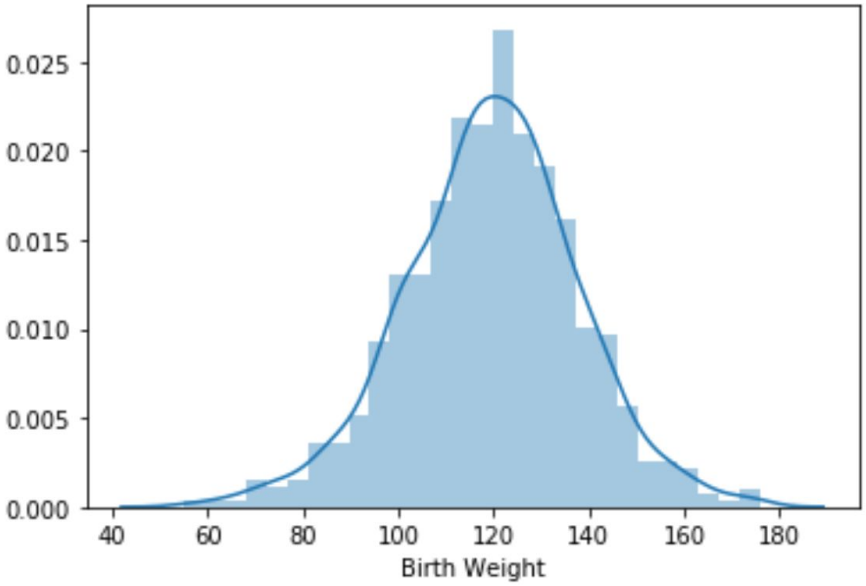
```
plt.hist(bweights, bins = np.arange(50, 200, 20), density=True, ec='w')
```



Bins don't need to have the same width! When they don't, it's especially crucial to think of proportions as areas.

```
plt.hist(bweights, bins = [50, 100, 120, 140, 200], density=True, ec='w');
```

Density curves

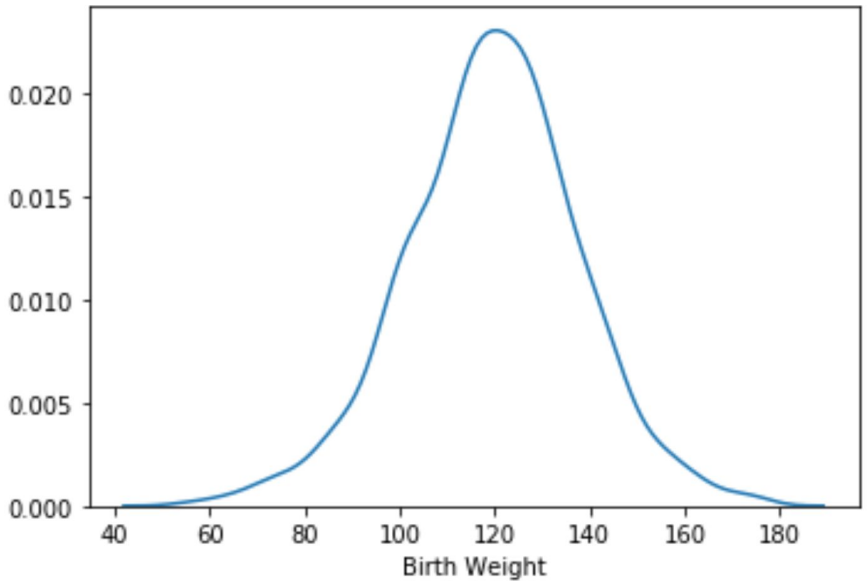



```
sns.distplot(bweights)
```

Sometimes, we don't need all of the detail that a histogram provides us with, and instead want a general idea of what our distribution looks like.

The smooth curve drawn on top of the histogram here is called a **density curve**, and it is just that!

Density curves




```
sns.distplot(bweights, hist=False)
```

We can also plot a density curve by itself, by appropriately setting the parameters of sns.distplot.

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

With the appropriate parameter, we can also add a rug plot to our density curve.

Describing distributions

One of the benefits of a histogram or density curve is that they show us the "bigger picture" of our distribution (something we don't get with a rug plot).

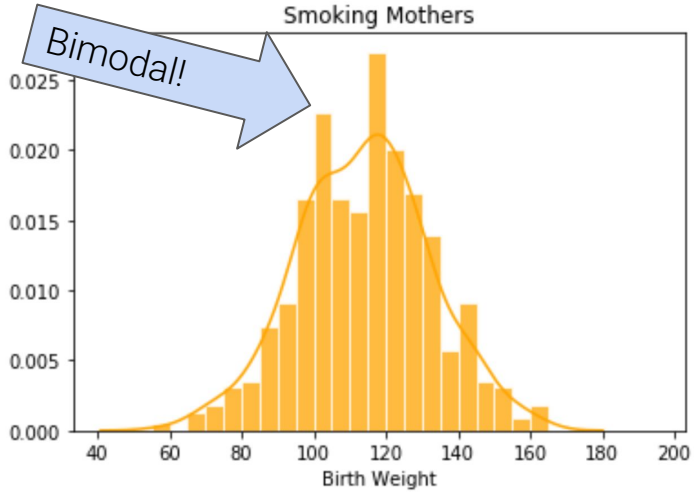
Some of the terminology we use to describe distributions:

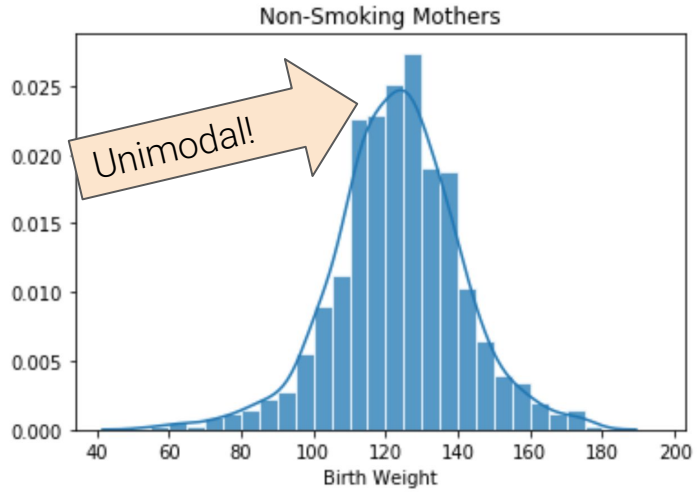
- **Modes.**
- **Skewness.**
  - Skewed left vs skewed right.
- **Tails.**
  - Left tail vs right tail.
- **Outliers.**
  - Define these arbitrarily.
  - Will see one definition in the next section.

Modes

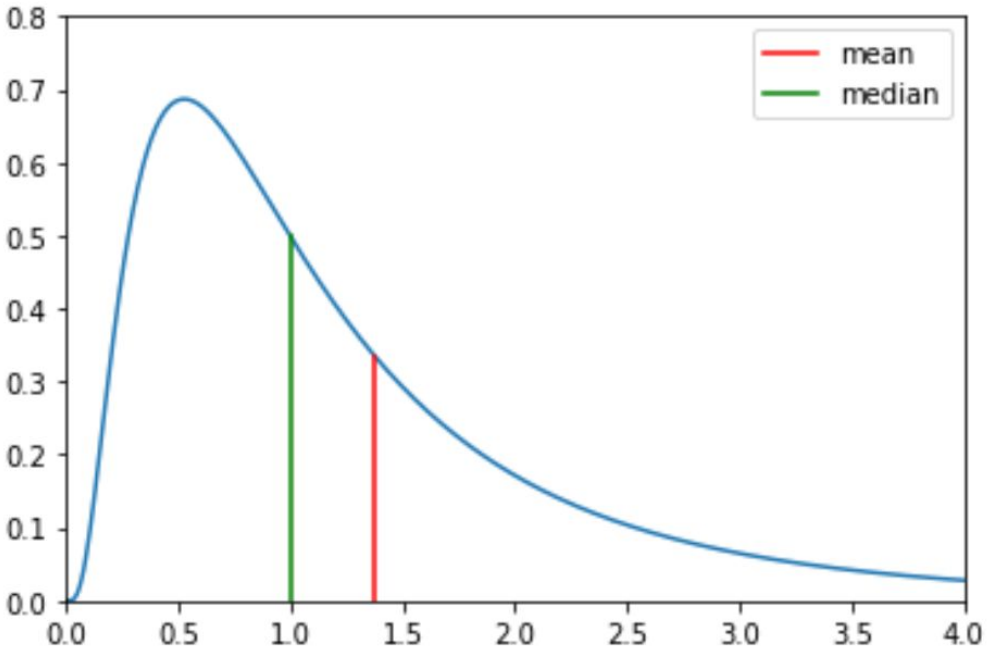
A **mode** of a distribution is a local or global maximum.

- 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.





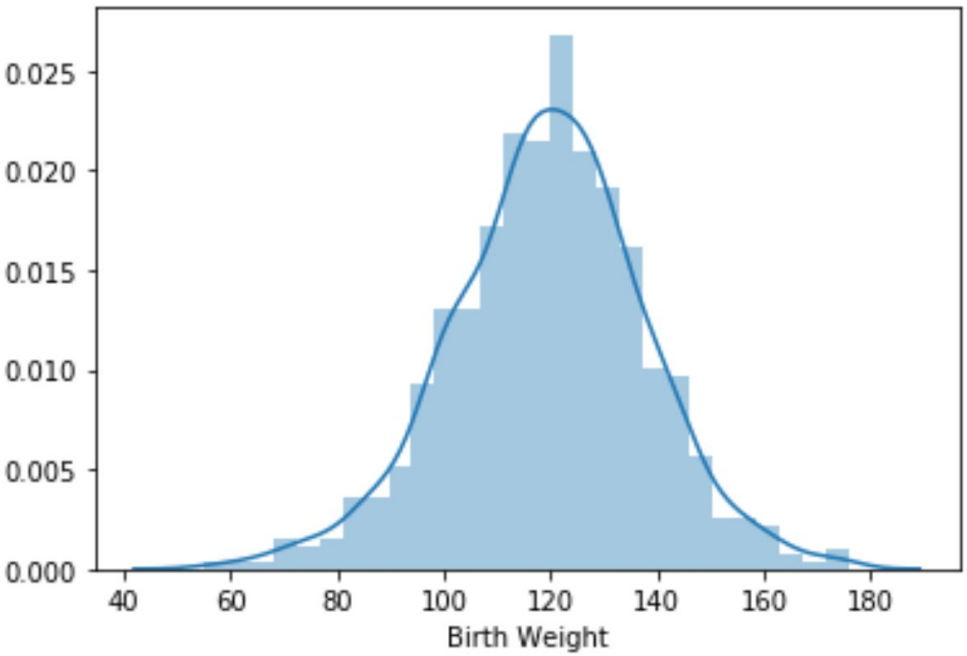
Skew and tails



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

- Such an example is on the left.
- In such cases, the mean is typically to the right of the median.
  - Think of the mean as the "balancing point" of the density.
- In the event that the tail is on the left, we say the data is skewed left.
- Our distribution can be symmetric, when both tails are of equal size.

Example



Consider the distribution of birth weights shown to the left. We might describe this as being:

- Unimodal. There is a single clear peak.
- Symmetric. It doesn't appear to be skewed in any direction.
  - Mean is very close to the median.
- Roughly normal.

Quartiles

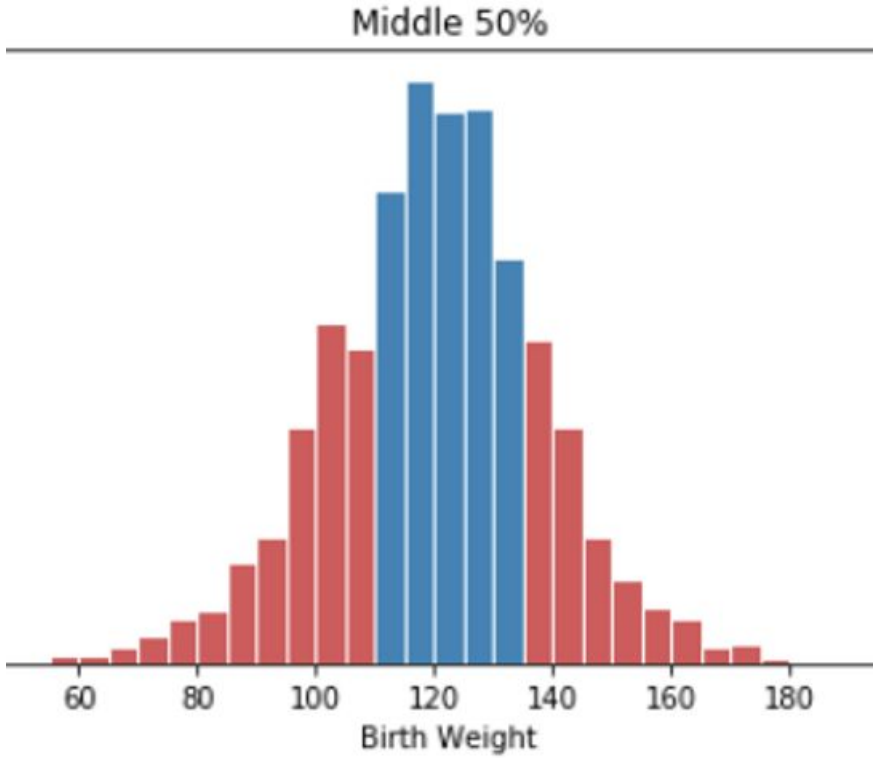
For a quantitative variable:

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

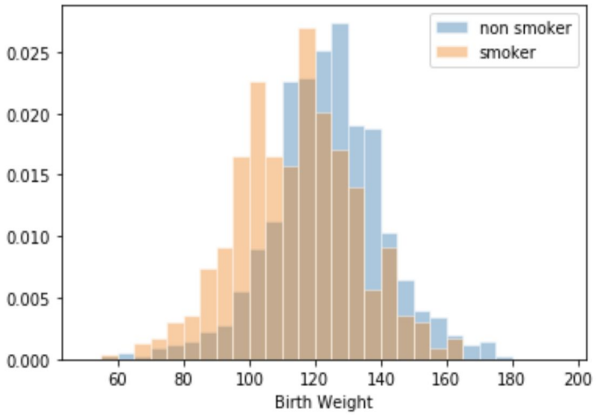
The interval [first quartile, third quartile] contains the “middle 50% of the data.

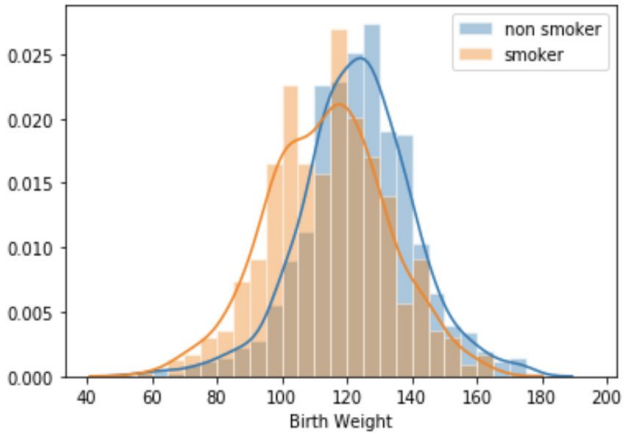
Interquartile range (IQR) measures spread.

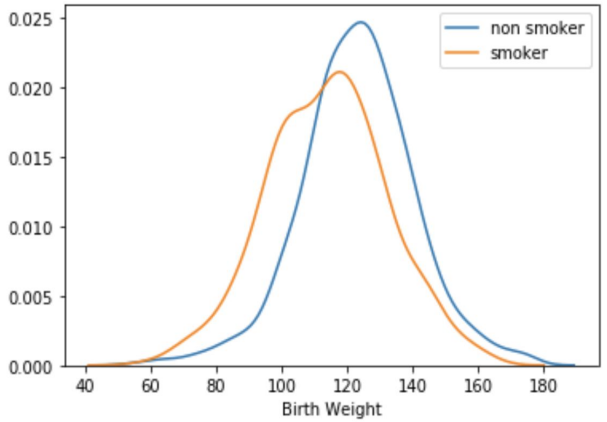
- IQR = third quartile – first quartile.



Overlaid histograms and density curves





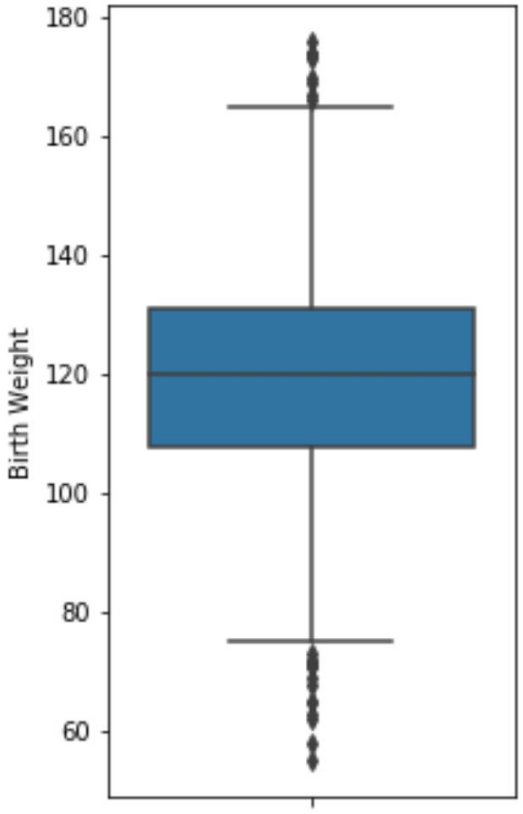


We can overlay multiple histograms and density curves on top of one another.

- 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.

Code is in the jupyter notebook.

Box plots

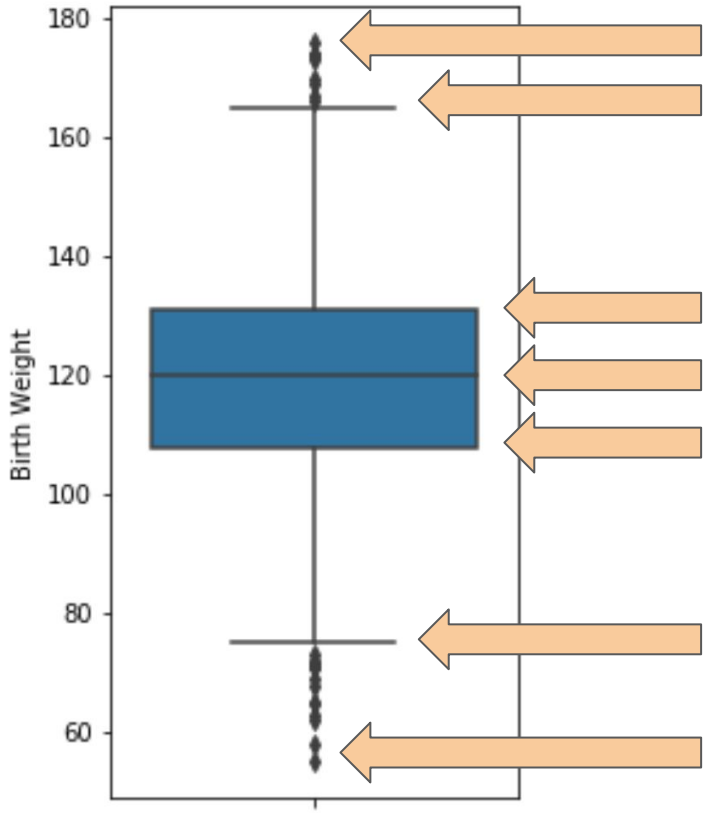



```
sns.boxplot(bweights)
```

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!

Box plots



Outliers

Upper quartile plus 1.5*IQR

Upper quartile

Median

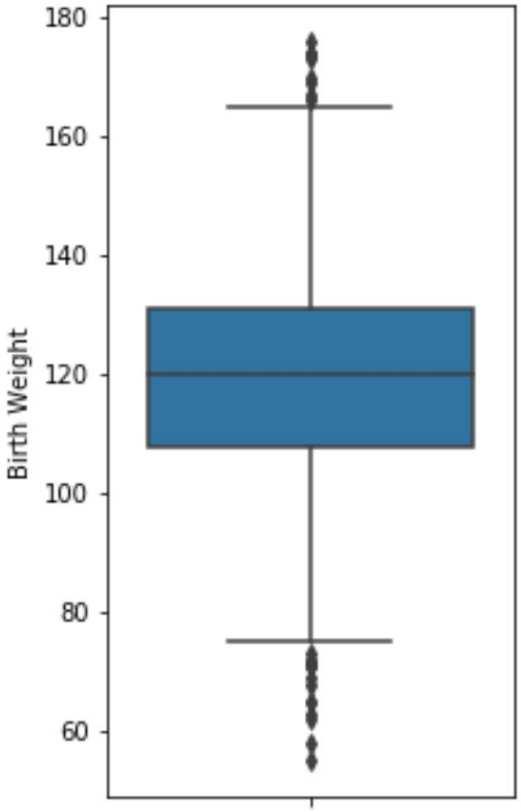
Lower quartile

Lower quartile minus 1.5*IQR

Outliers

Note: The box width is meaningless.

Box plots

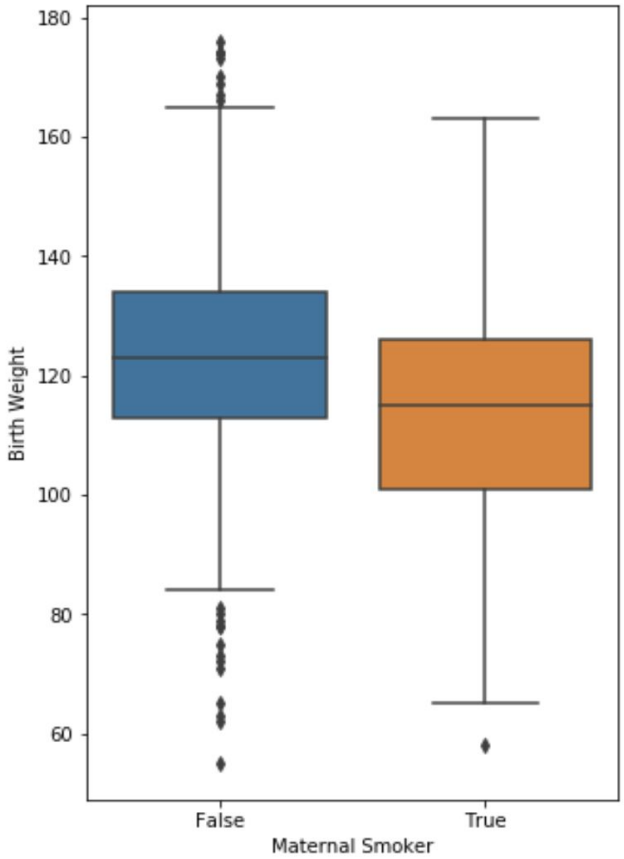


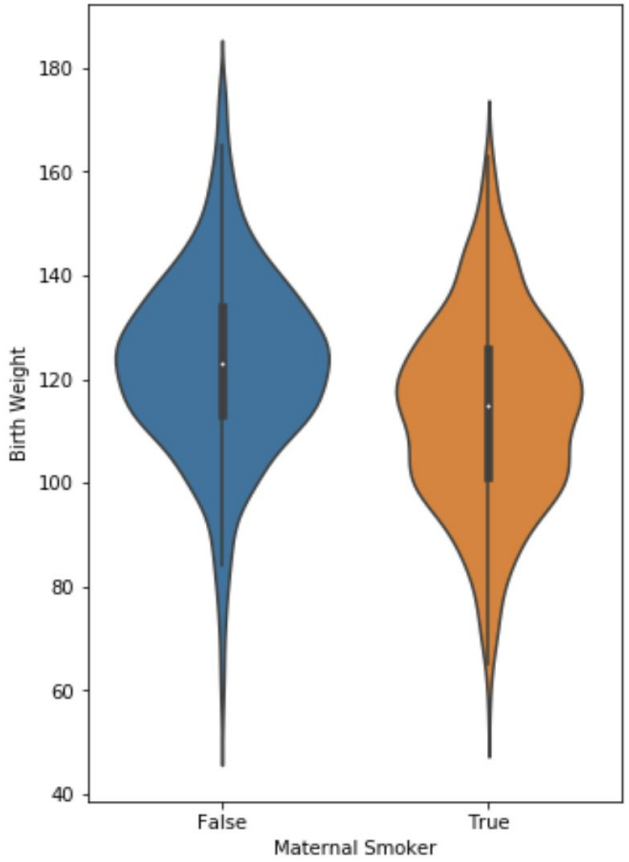

```
1  q1 = np.percentile(bweights, 25)
2  q2 = np.percentile(bweights, 50)
3  q3 = np.percentile(bweights, 75)
4  iqr = q3 - q1
5  whisk1 = q1 - 1.5*iqr
6  whisk2 = q3 + 1.5*iqr
7
8  whisk1, q1, q2, q3, whisk2
```

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

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

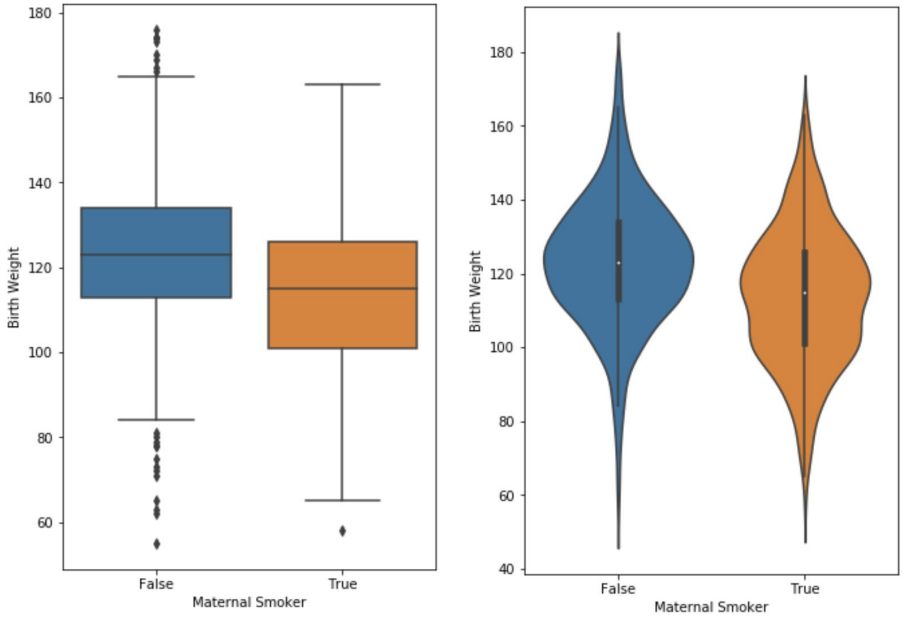
Side by side box plots and violin plots





Code is in the jupyter notebook.

Side by side box plots and violin plots

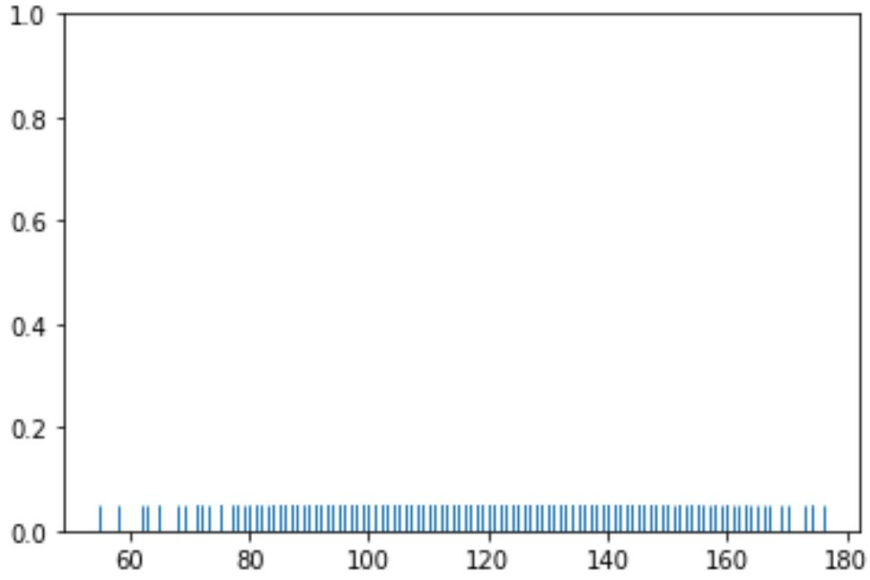


Box plots and violin plots are concise, and thus are well suited to be stacked side by side to compare multiple distributions at once.

- 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.

Rug plot

- Rug plots are used to show the distribution of a single quantitative (**numerical**) variable.
- They show us each and every value!
- Issues with rug plots:
  - Too much detail.
  - Hard to see the bigger picture.
  - **Overplotting**.
    - How many birth weights were at 120?
    - Can't tell – they're all on top of each other.

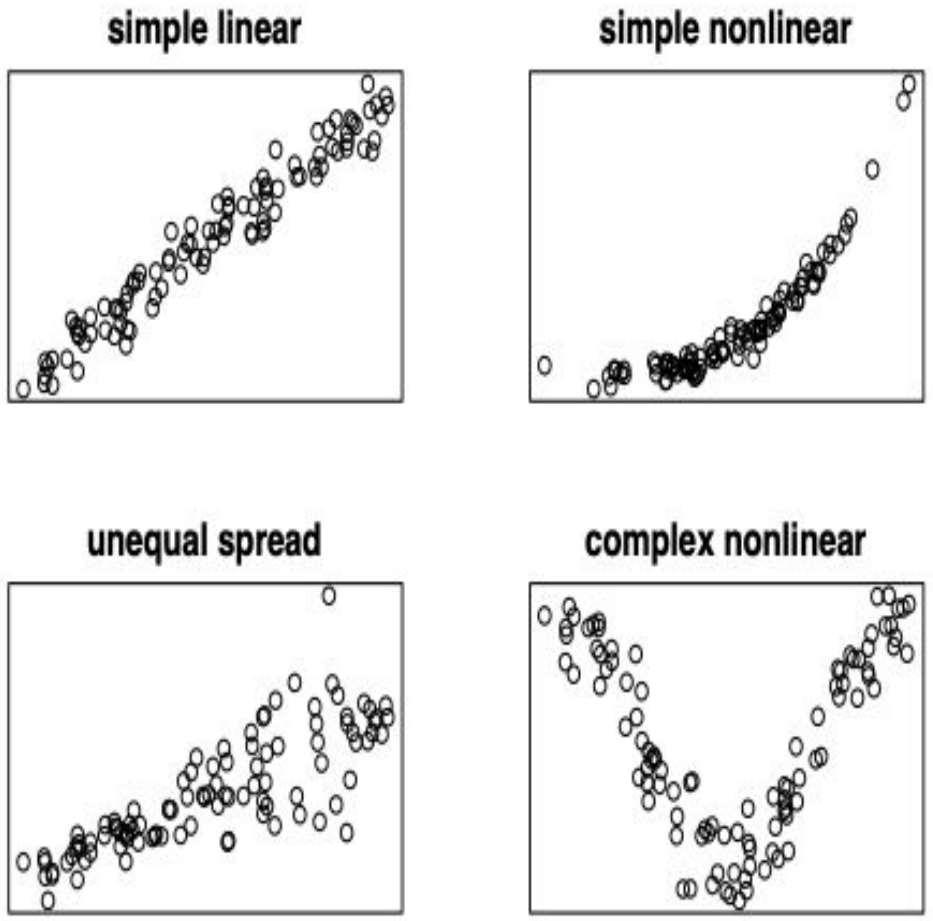



```
sns.rugplot(bweights)
```

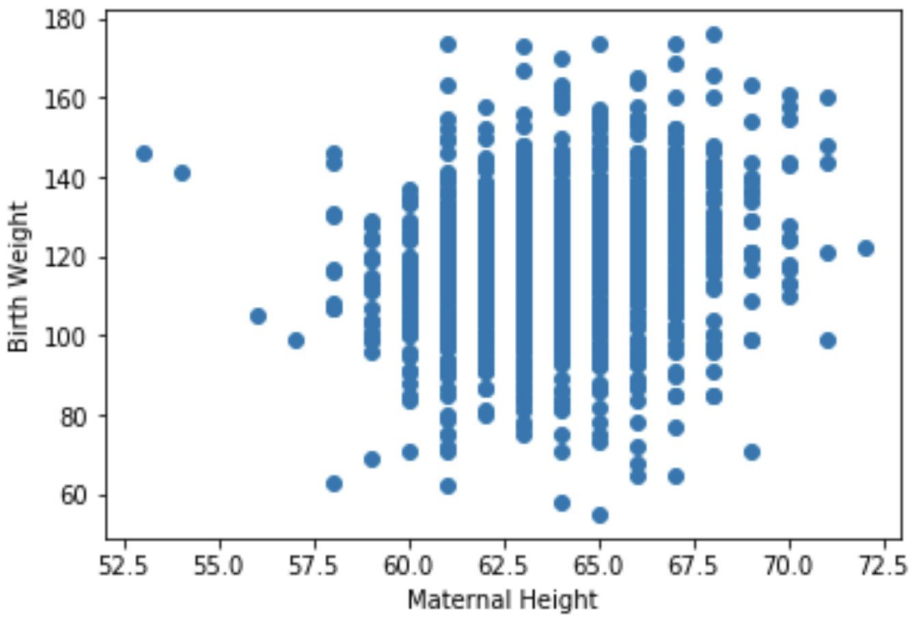
Scatter plots

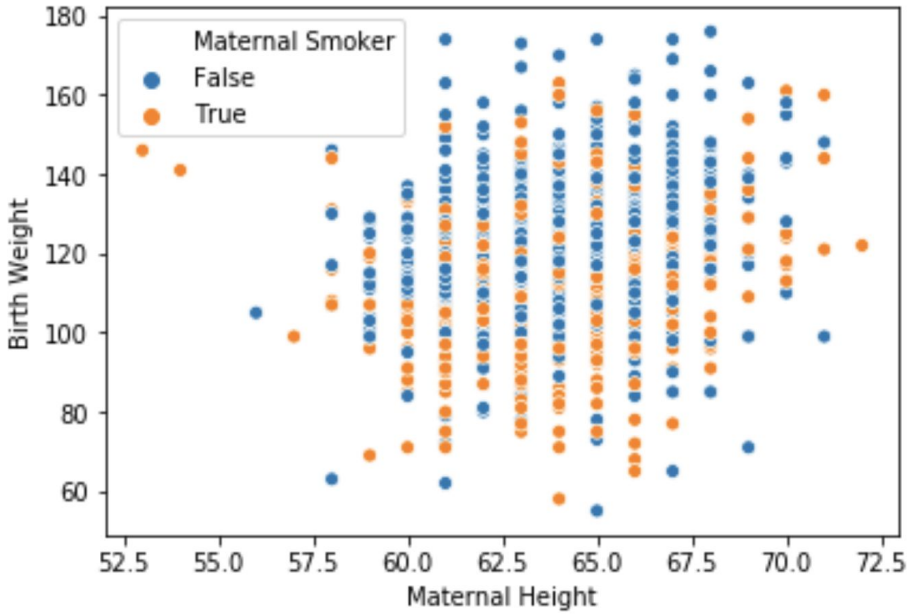
Scatter plots are used to reveal relationships between pairs of numerical variables.

- We often use scatter plots to help inform modeling choices.
- For instance, the simple linear model requires the trend in our data to be roughly linear, and for spread to be roughly equal.



Scatter plots

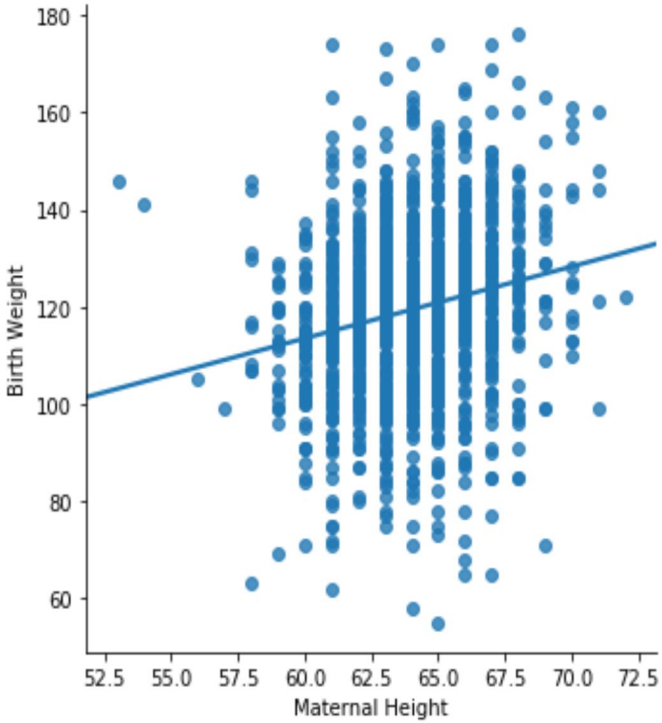




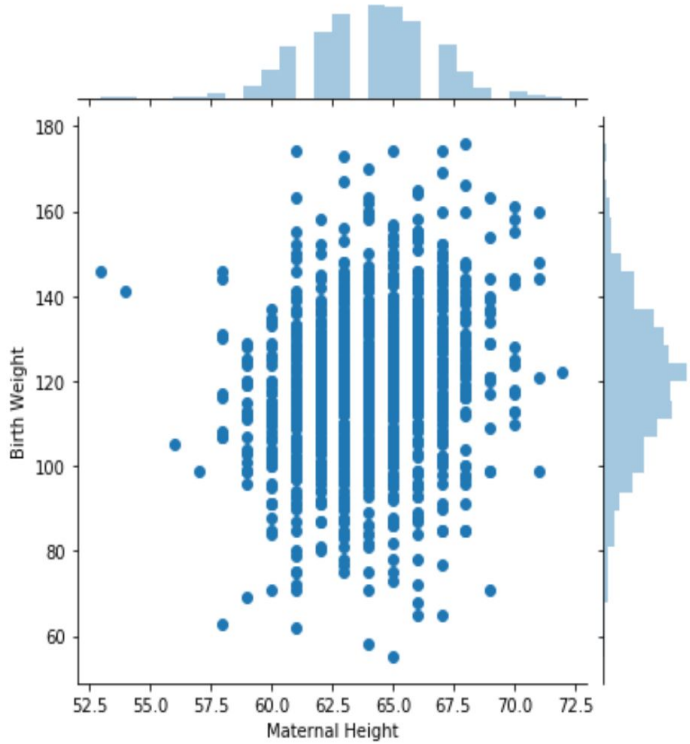
- We can also use color to encode categorical variables.
- These plots suffer from overplotting – many of the points are on top of one another!
    - One solution: add a small amount random noise in both the x and y directions.

Code is in the jupyter notebook.

Scatter plots




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




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

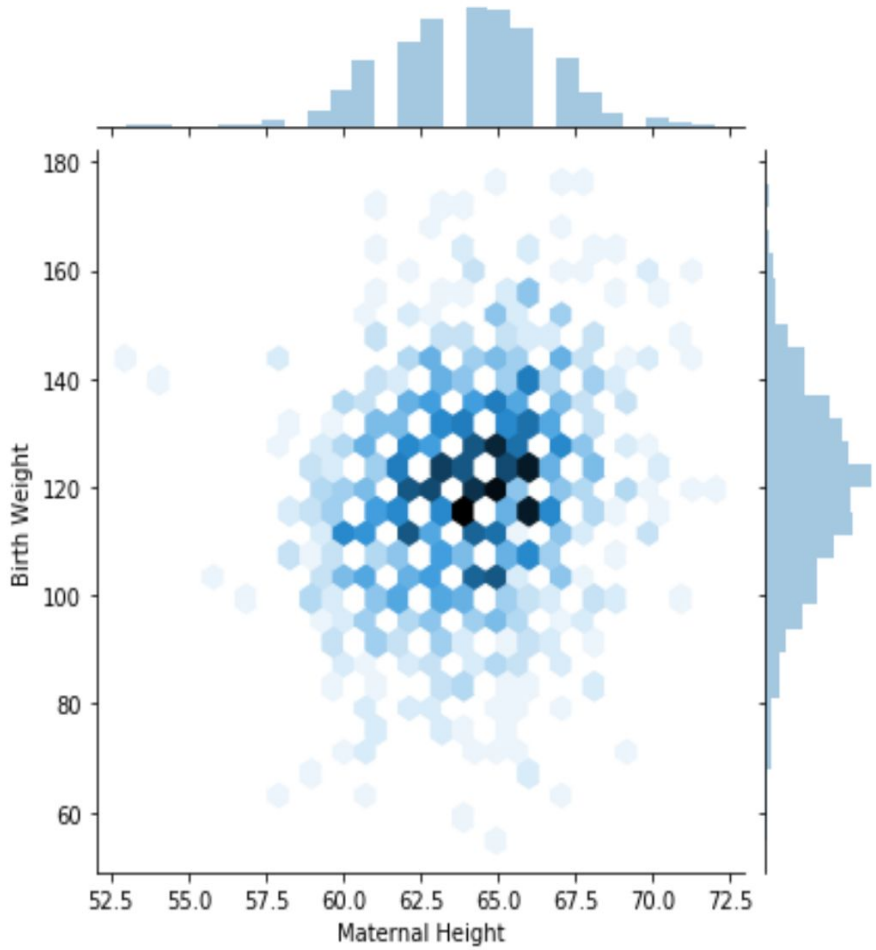
Hex plots

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?

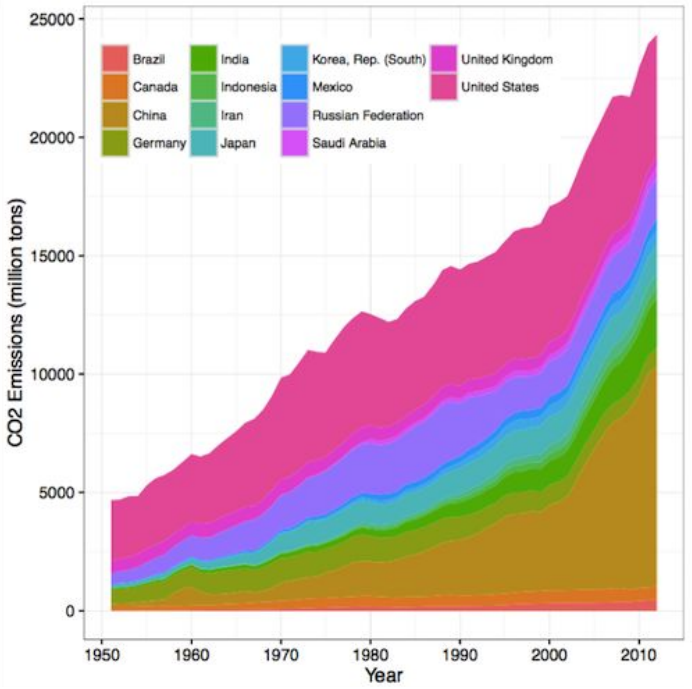
- Easier to see linear relationships.
- More efficient for covering region.
- Visual bias of squares – drawn to see vertical and horizontal lines.

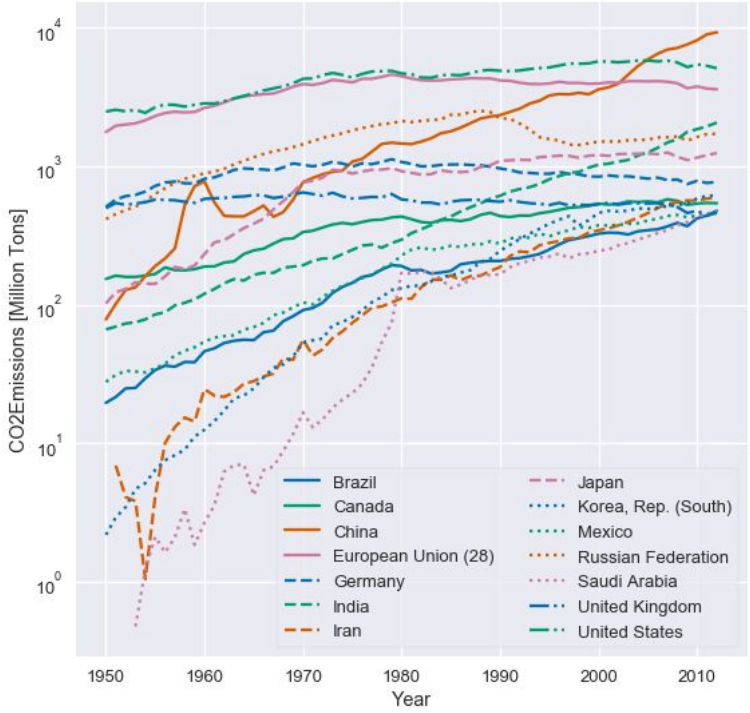



```
sns.jointplot(data=births, x='Maternal Height', y='Birth Weight', kind='hex')
```

Line plots

Avoid jiggling the baseline





Here, by switching to a line plot, comparisons are made much easier.

Code for a similar line plot is in the jupyter notebook.

Add context directly to plot

A publication-ready plot needs:

- Informative title (takeaway, not description).
  - “Older passengers spend more on plane tickets” instead of “Scatter plot of price vs. age”.
- Axis labels.
- Reference lines, markers, and labels for important values.
- Legends, if appropriate.
- Captions that describe the data.

The plots you create in this class always need titles and axes labels, too.

Captions

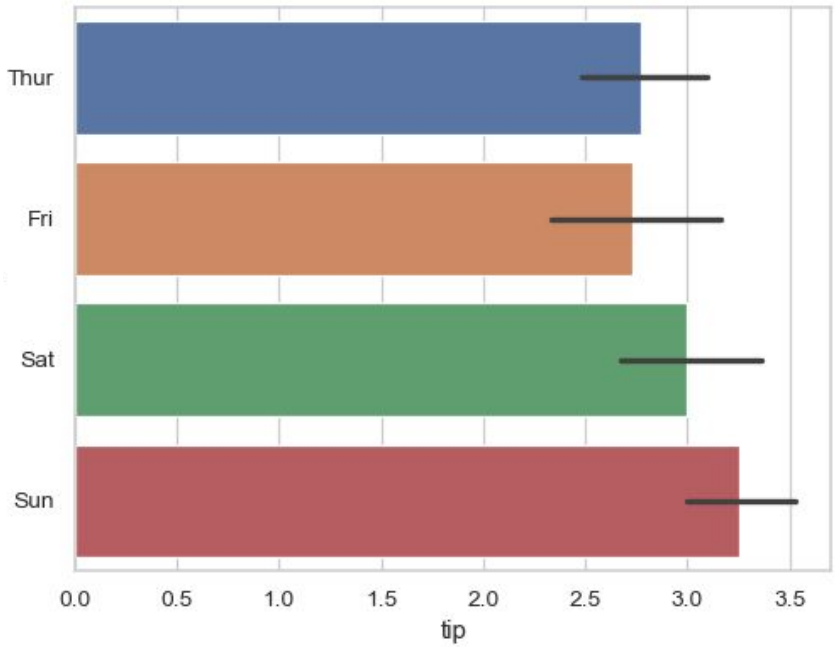
A picture is worth a thousand words, but not all thousand words you want to tell may be in the picture. In many cases, we need captions to help tell the story.

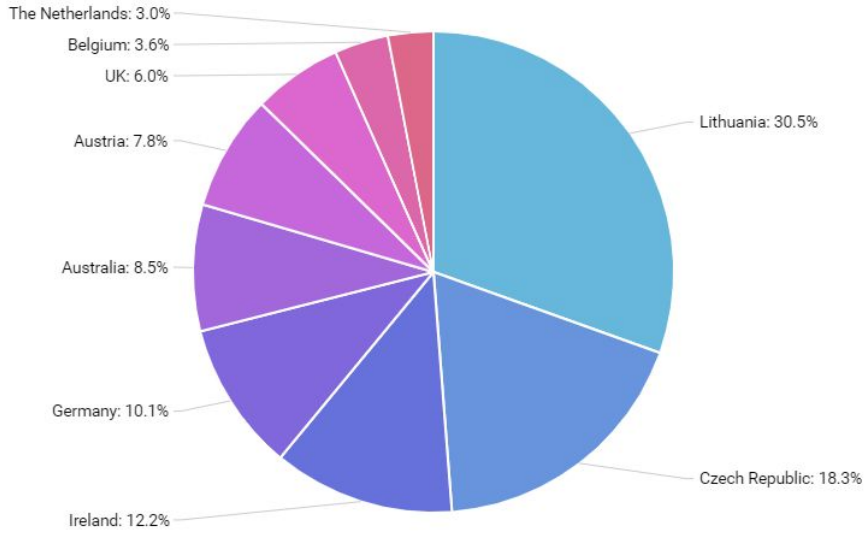
Captions should be:

- Comprehensive and self-contained.
- Describe what has been graphed.
- Draw attention to important features.
- Describe conclusions drawn from graph.

Pie charts

Lengths are easy to distinguish; angles are hard



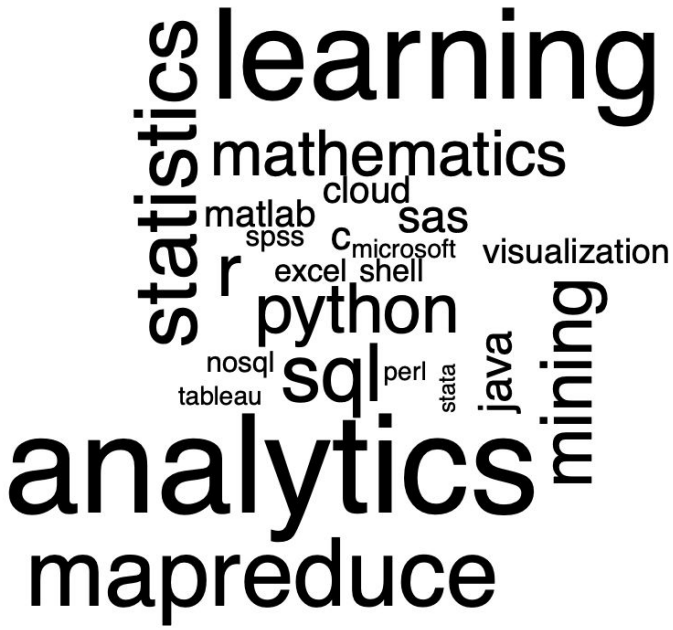


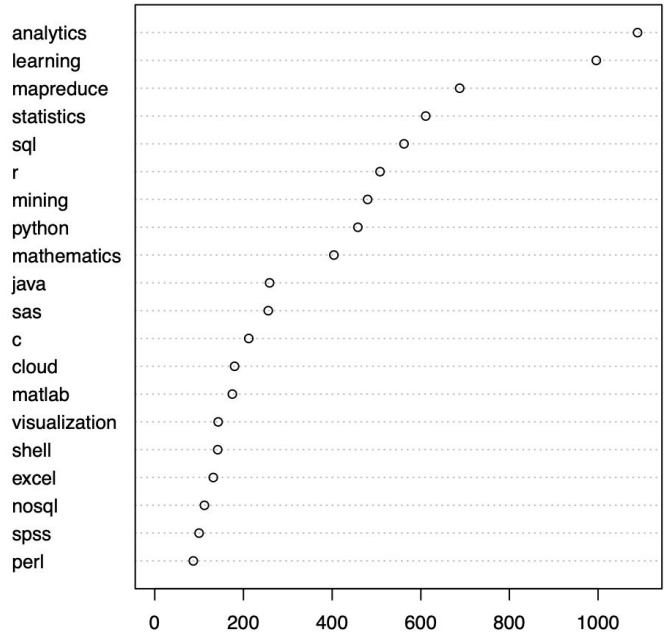
Don’t use pie charts! Angle judgements are inaccurate.

Code for a similar pie chart is in the jupyter notebook.

Areas are hard to distinguish

Areas are hard to distinguish



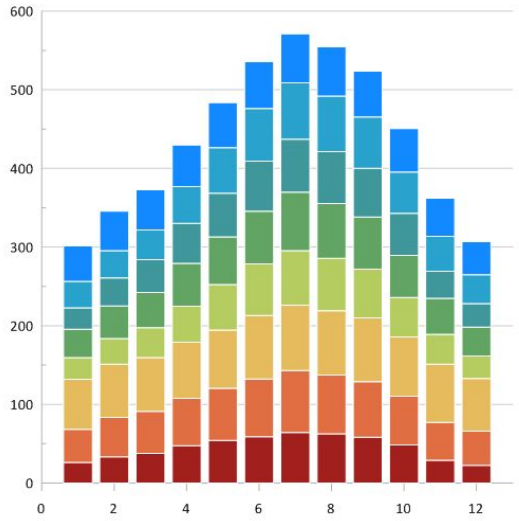


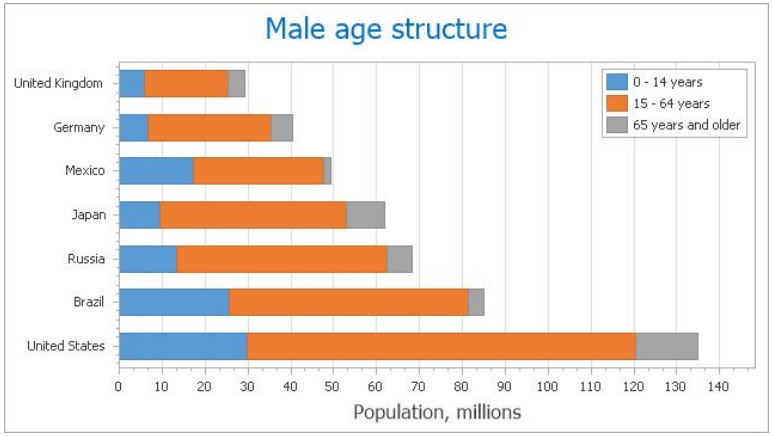
Avoid word clouds too! It's hard to tell the area taken up by a word.

Avoid jiggling the baseline

Stacked bar charts, histograms, and area charts are hard to read because the baseline moves.

- In the first plot, the top blue bars are all roughly of the same length. But that's not immediately obvious!
- In the second plot, comparing the number of 15-64 year old males in Germany and Mexico is difficult.

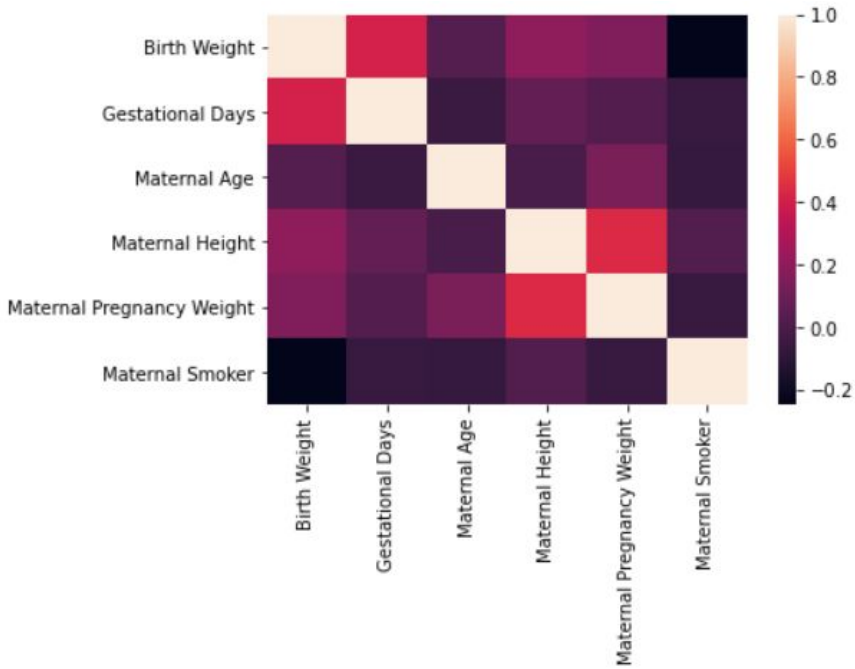




Heatmaps

Use heatmaps for correlation matrices

	Birth Weight	Gestational Days	Maternal Age	Maternal Height	Maternal Pregnancy Weight	Maternal Smoker
Birth Weight	1.000000	0.407543	0.026983	0.203704	0.155923	-0.246800
Gestational Days	0.407543	1.000000	-0.053425	0.070470	0.023655	-0.060267
Maternal Age	0.026983	-0.053425	1.000000	-0.006453	0.147322	-0.067772
Maternal Height	0.203704	0.070470	-0.006453	1.000000	0.435287	0.017507
Maternal Pregnancy Weight	0.155923	0.023655	0.147322	0.435287	1.000000	-0.060281
Maternal Smoker	-0.246800	-0.060267	-0.067772	0.017507	-0.060281	1.000000

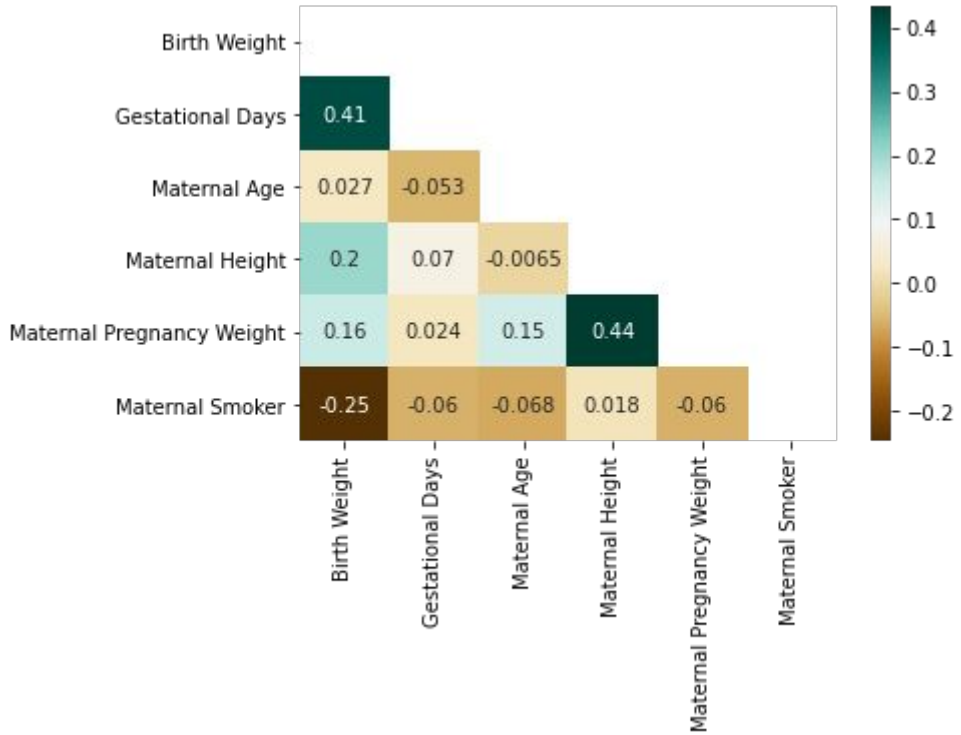



```
sns.heatmap(births.corr(), annot=False);
```

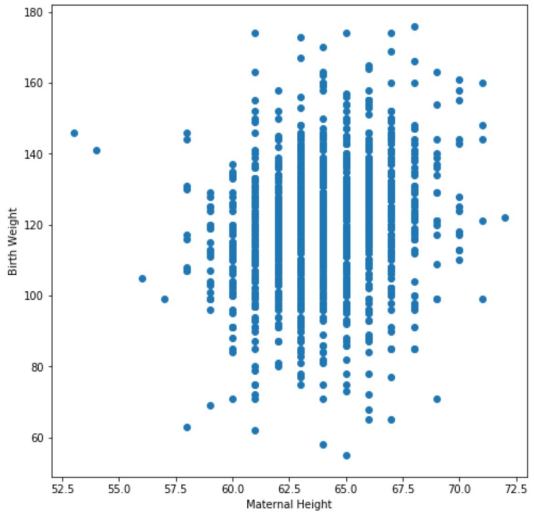
Heatmaps

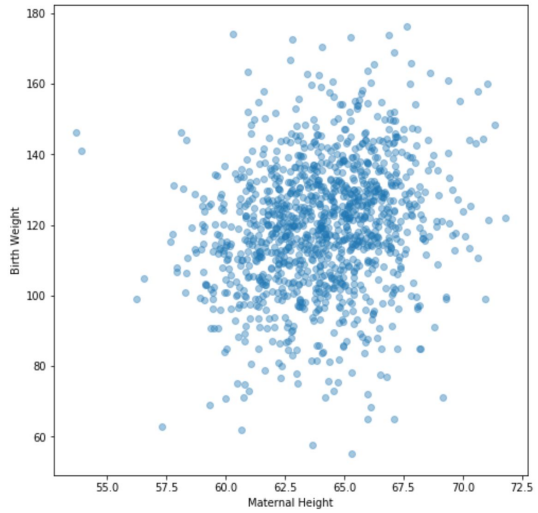
Correlation matrix is symmetric. If you cut away half of it along the diagonal line marked by 1-s, you would not lose any information.

Code is in the jupyter notebook.



Overplotting





In the plot on the left, it's hard to tell exactly how many points are being visualized.

- Many on top of one another.
- Observations only on lattice points.

Some solutions:

- Add small random noise to both x and y ("jittering").
- Make points smaller (wouldn't help here though).

Kernel density estimation (KDE)

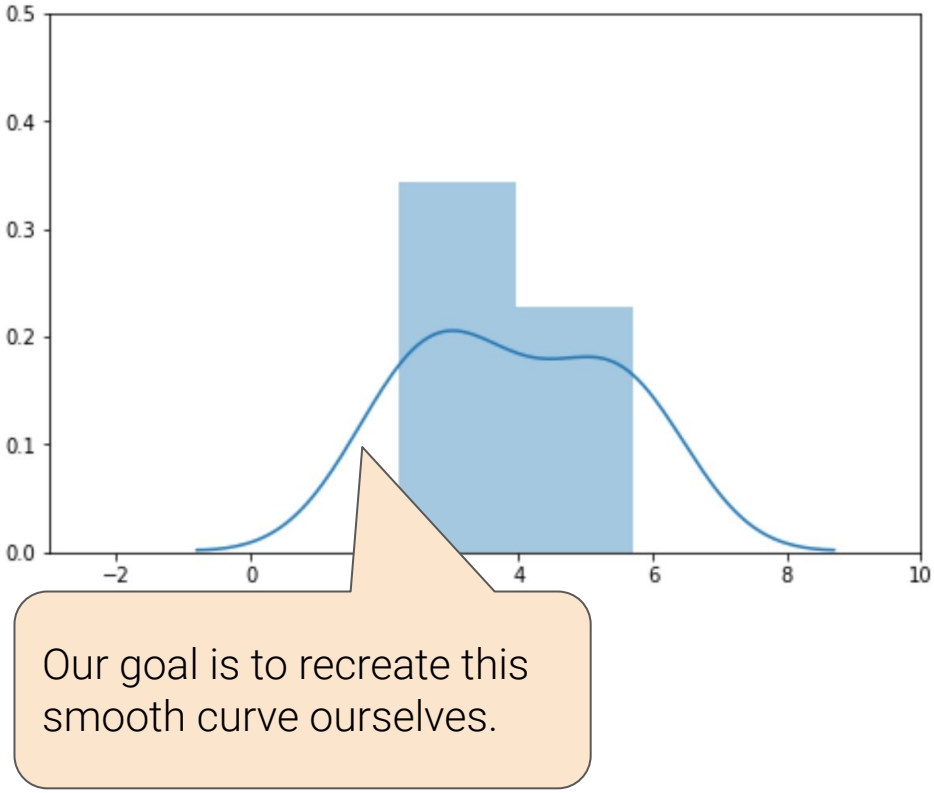
Kernel Density Estimation is used to estimate a **probability density function** (or density curve) from a set of data.

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

To create a KDE:

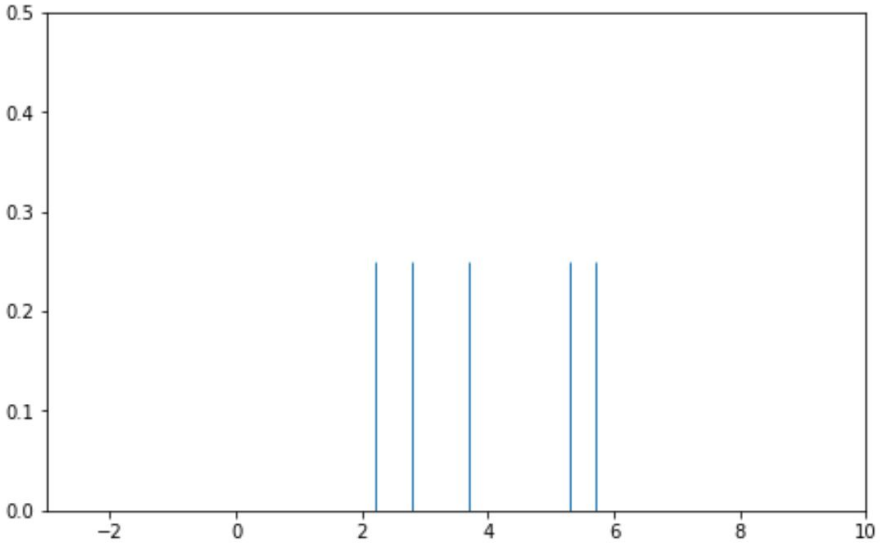
- Place a **kernel** at each data point.
- Normalize kernels so that total area = 1.
- Sum all kernels together.

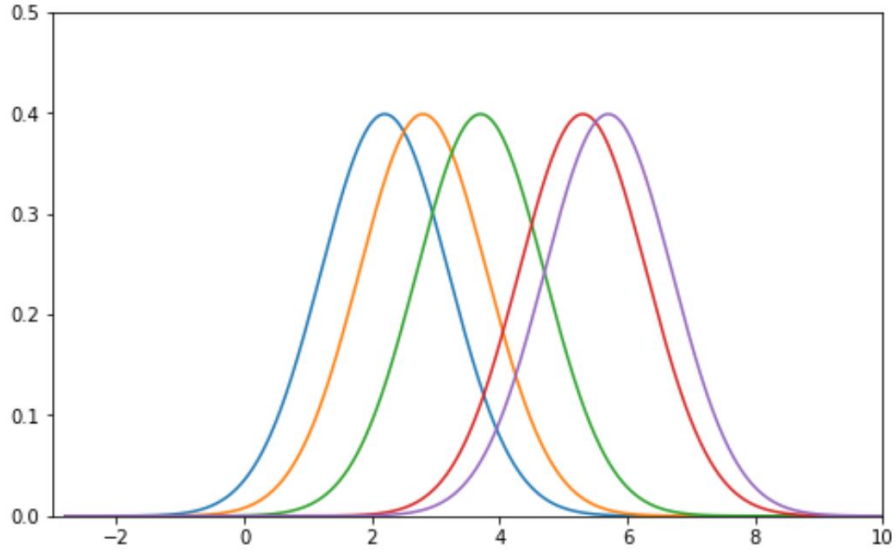
We also need to choose a kernel and **bandwidth.**



Step 1: Place a kernel at each data point

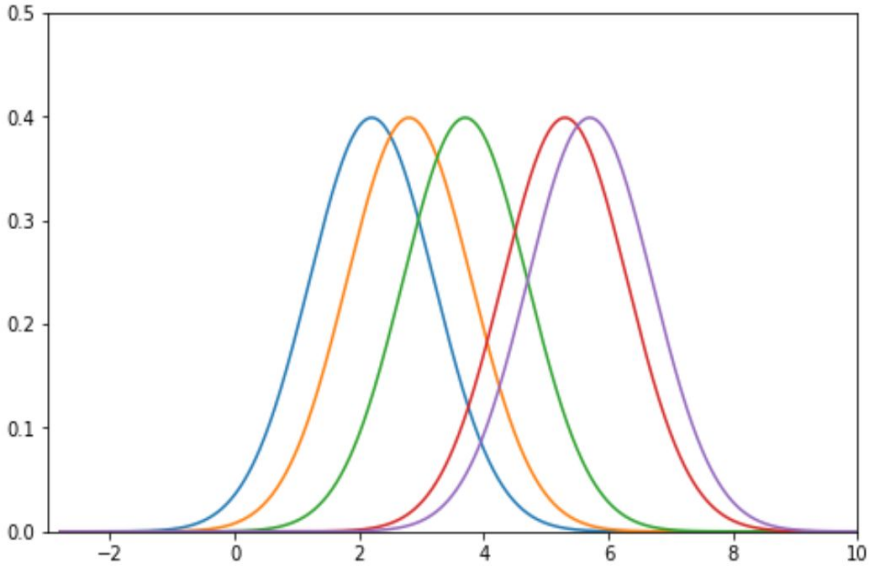
At each of our 5 points (depicted in the rug plot on the left), we've placed a **Gaussian** kernel with **alpha = 1**. The idea is that there is a higher density near the points we've already seen.

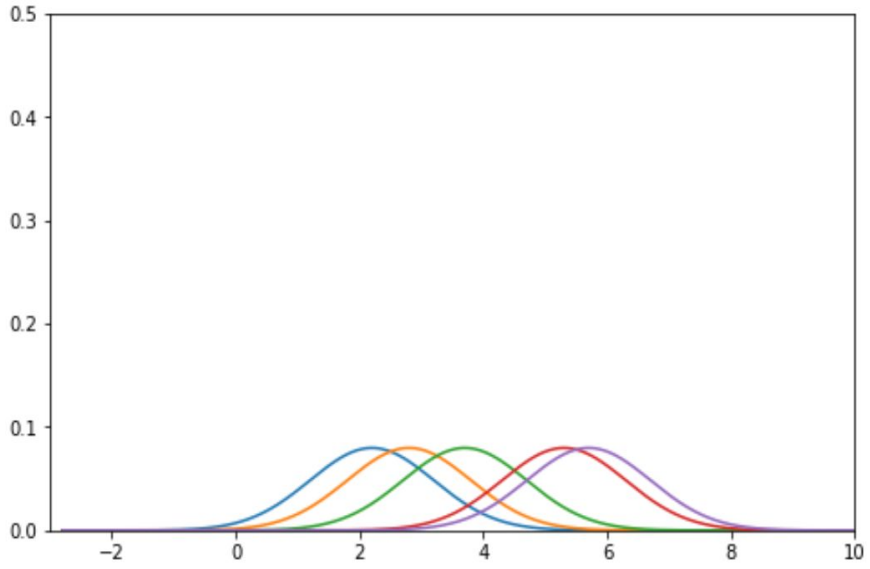




Step 2: Normalize kernels

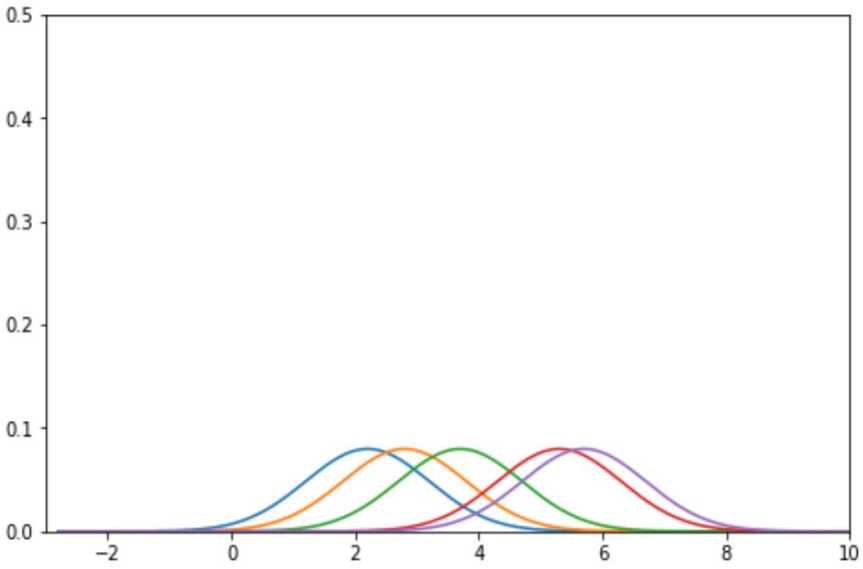
In Step 3, we will be summing each of these kernels. We want the result to be a valid density, that has area 1. Right now, we have 5 different kernels, each with an area 1. So, we **multiply each by 1/5**.

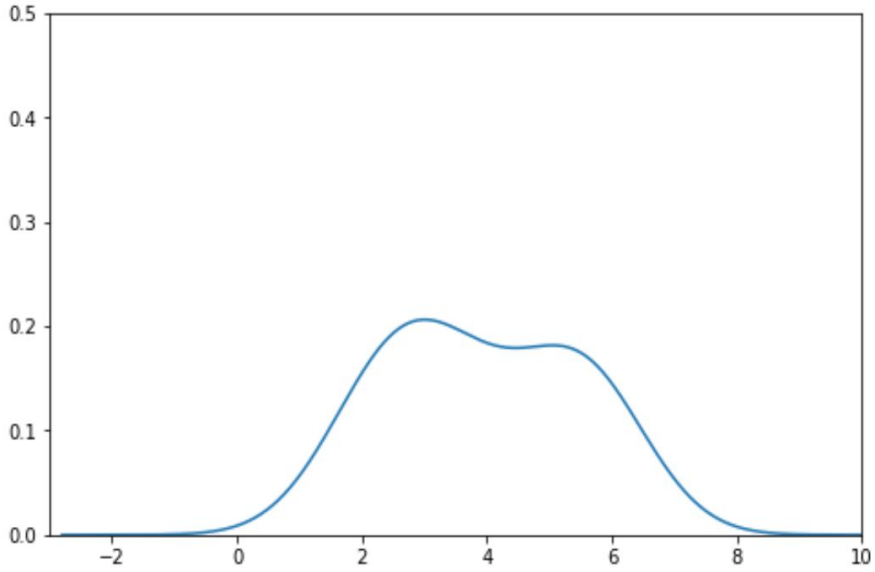




Step 3: Sum kernels

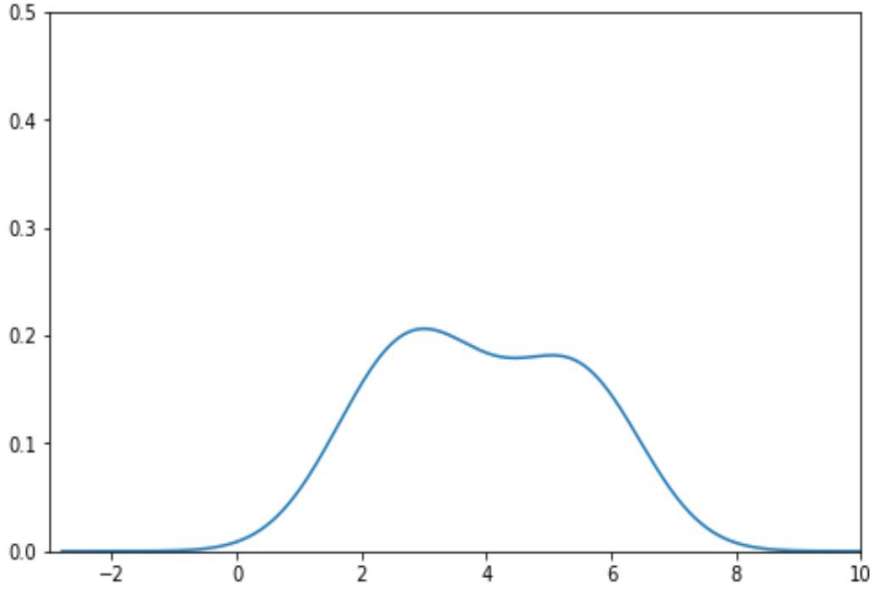
Our **kernel density estimate** is the **sum of the normalized kernels at each point.** It is depicted below on the right.

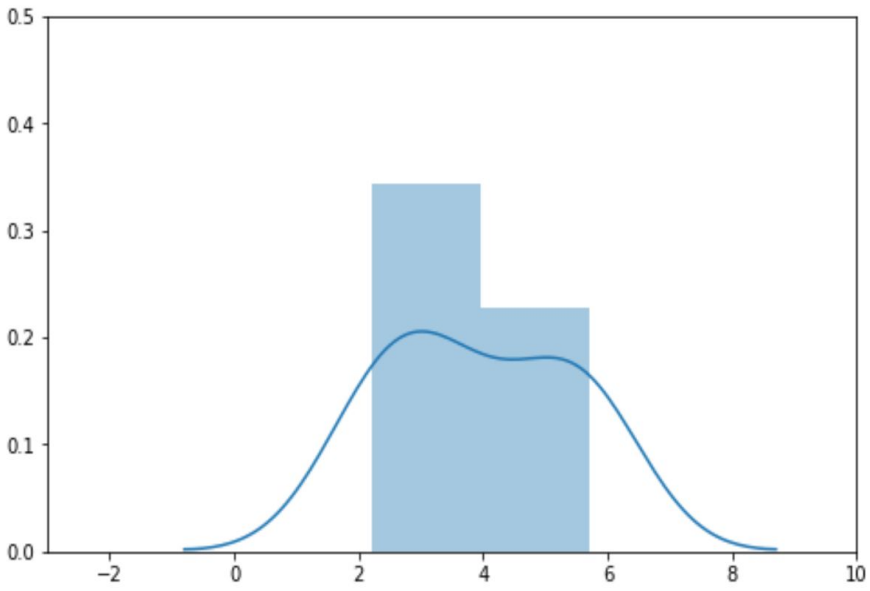




Kernel density estimates

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





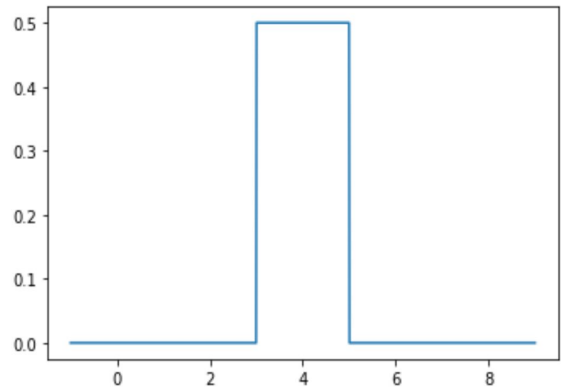
Kernels

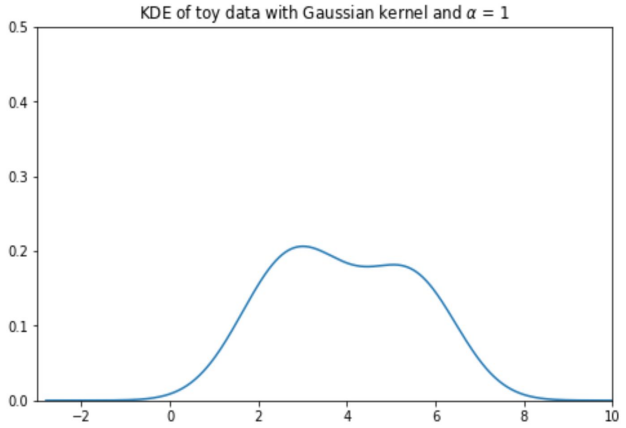
- A kernel (for our purposes) is a valid density function. That means it:
  - Must be non-negative for all inputs.
  - Must integrate to 1.
- The most common kernel is the **Gaussian** kernel.
  - Here, x represents any input, and xi represents the ith observed value. The kernels are centered on our observed values (and so the mean of this distribution is xi).
  - **α** is the **bandwidth parameter**. It controls the smoothness of our KDE. Here, it is also the standard deviation of the Gaussian.

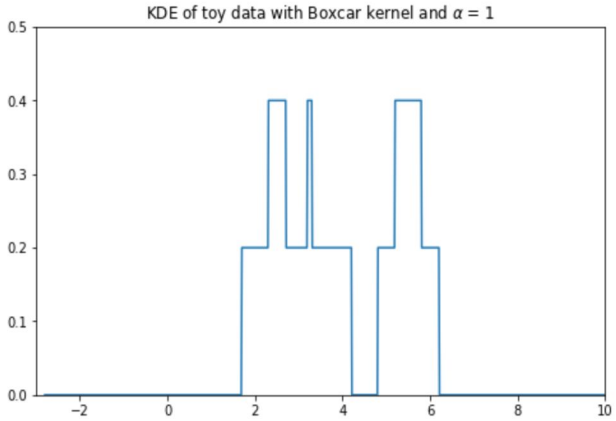
$$K_\alpha(x, x_i) = \frac{1}{\sqrt{2\pi\alpha^2}} e^{-\frac{(x-x_i)^2}{2\alpha^2}}$$

Kernels

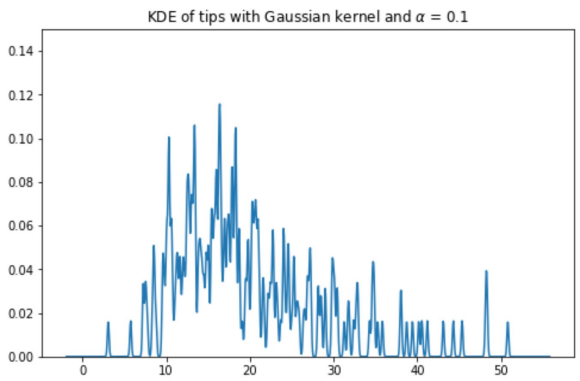
- Another common kernel is the **boxcar** kernel.
  - It assigns uniform density to points within a “window” of the observation, and 0 elsewhere.
  - Resembles a histogram… sort of.

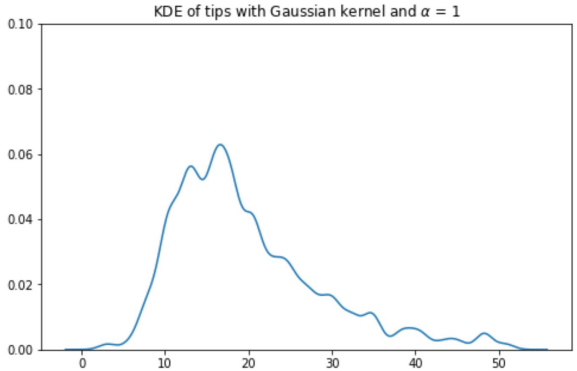


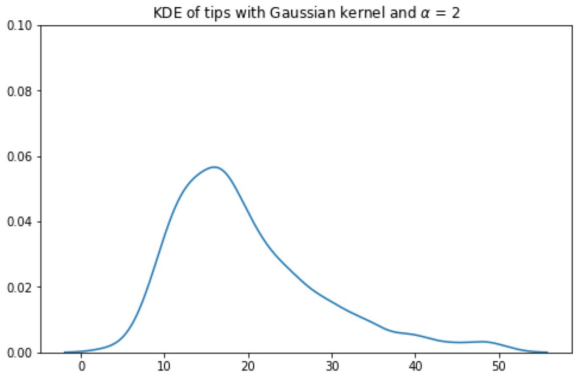


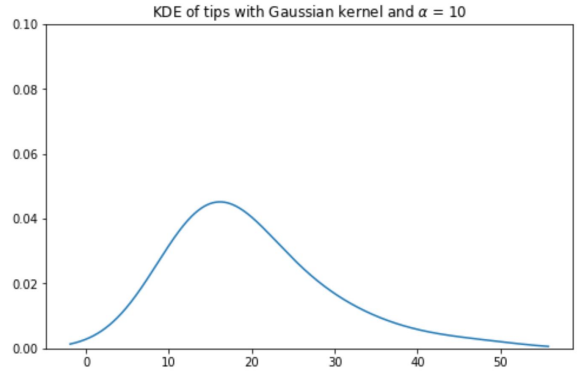


Effect of bandwidth on KDEs









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

- As **α** increases, the KDE becomes more smooth.
- This makes it simpler to understand, but also gets rid of potentially important distributional information.
- We call **α** a **hyperparameter**. Be familiar with this term!

Summary of KDE

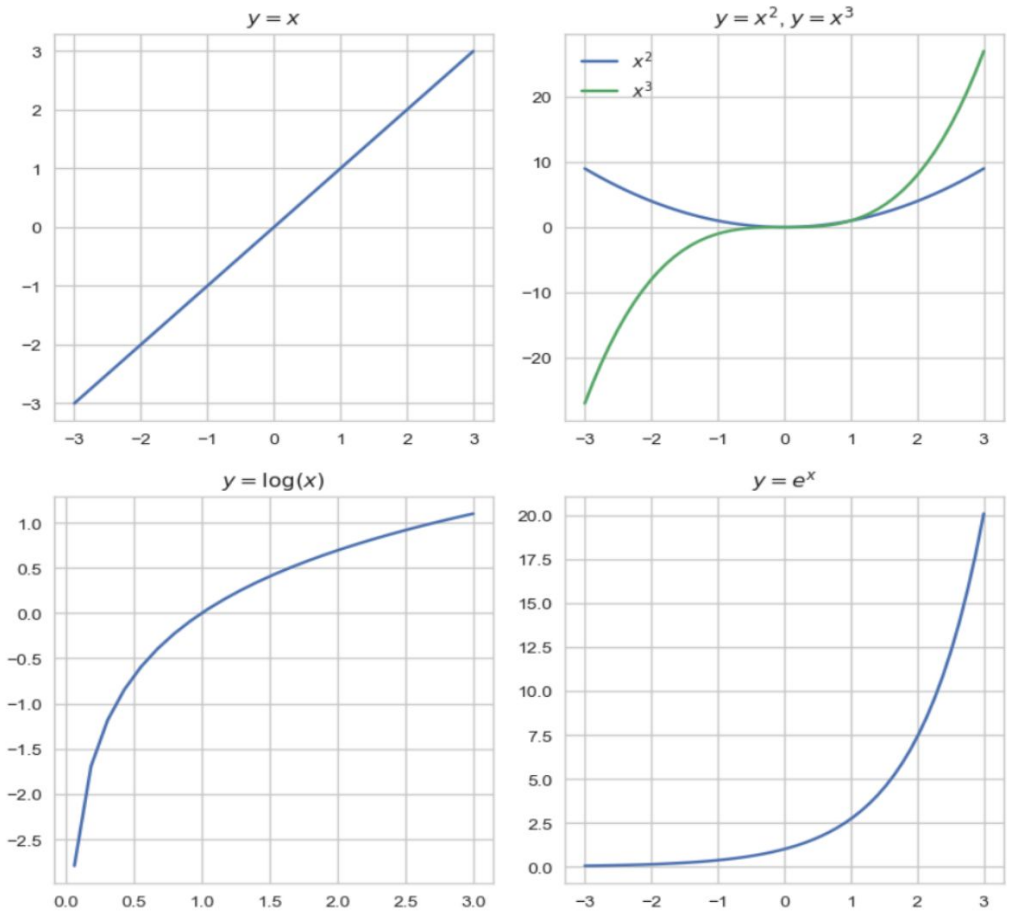
$$f_\alpha(x) = \sum_{i=1}^{n} \frac{1}{n} \cdot K_\alpha(x, x_i) = \frac{1}{n} \sum_{i=1}^{n} K_\alpha(x, x_i)$$

The “KDE formula” is above.

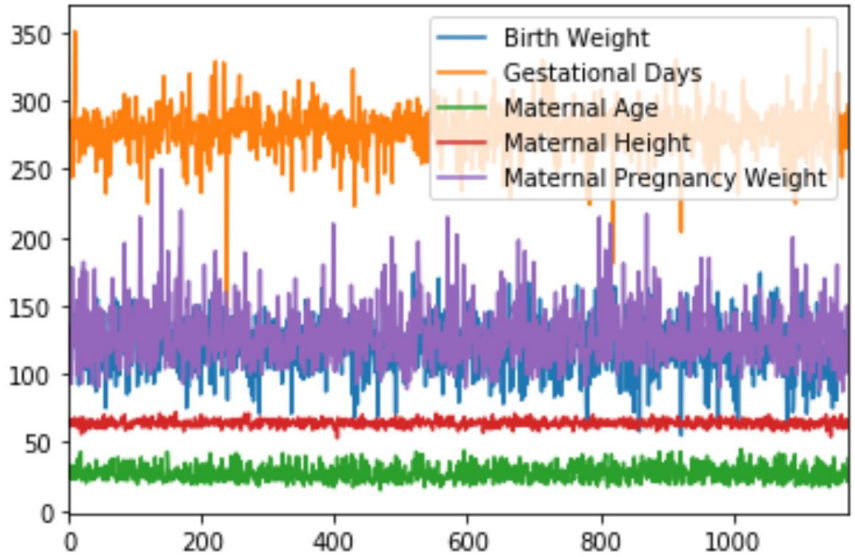
- $\mathbf{x}$ represents any number on the number line. It is the input to our function.
- $\mathbf{n}$ is the number of observed data points that we have.
- Each $\mathbf{xi}$ ($\mathbf{x1}$, $\mathbf{x2}$, …, $\mathbf{xn}$) represents an observed data point. These are what we use to create our KDE.
- $\boldsymbol{\alpha}$ is the bandwidth or smoothing parameter.
- $K\boldsymbol{\alpha}(x, xi)$ 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.

Basic functional relations

Knowing the general shapes of polynomial, exponential, and logarithmic curves (regardless of base) will go a long way.



Fun fact



`births.plot()`

This is the result of calling **`births.plot()`**. If you don't provide any specifications, pandas just guesses what you want visualized. It often makes no sense!

Summary

- **Visualization requires a lot of thought!**
- Types of variables constrain the charts that you can make.
  - Single quantitative: rug plot, histogram, density plot.
  - Two quantitative: scatter plot, hex plot, contour plot.
  - Combination: bar plot, overlaid histograms/density plots, SBS box/violin plots.
- 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 very easily creates **interactive** plots.
    - It will appear in lecture code and assignments!

Summary

- Choose appropriate scales.
- Condition in order to make comparisons more natural.
- Choose colors and markings that are easy to interpret correctly.
- Add context and captions that help tell the story.
- Smoothed estimates of distributions help with big-picture interpretation.
  - Kernel Density Estimates are a method of smoothing data.
- Transforming our data can linearize relationships.
  - Helpful when we start linear modeling next lecture.
- **More generally – reveal the data!**
  - Eliminate anything unrelated to the data itself – “chart junk.”
  - It’s fine to plot the same thing multiple ways, if it helps fit the narrative better.

Any Questions?