In [2]:
```
import numpy as np
import pandas as pd
import altair as alt
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
alt.data_transformers.disable_max_rows()

air = pd.read_csv('data/air.csv').rename(
    columns = {'PM2.5': 'PM25_98pct',
               'PM2.5.1': 'PM25_mean',
               'NO2': 'NO2_98pct',
               'NO2.1': 'NO2_mean'}
)
```

# Data Science Concepts and Analysis

## Week 4: Smoothing

### (Exploratory data analysis I)

- What is EDA?
- Smoothing techniques
- Making connections with probability

# This week: smoothing

**Objective**: introduce exploratory analysis and a tool for exploring variation in one variable.

- **What is exploratory data analysis (EDA)?**
  - The role of data: information or evidence?
  - Exploratory vs. confirmatory analysis
  - Essential exploratory questions: variation and co-variation

- **Smoothing**
  - Why smooth?
  - Kernel density estimation (KDE)

- **Connections with probability**
  - What's a probability distribution again?
  - EDA and conditional distributions

# What is exploratory data analysis (EDA)?

- The role of data: information or evidence?
- Exploratory vs. confirmatory analysis
- Essential exploratory questions: variation and co-variation

# Origins

The term and spirit of EDA is attributed to John Tukey, whose philosophically-leaning work in statistics in the 1960's and 1970's elaborated on the need for certain shifts in the standard disciplinary paradigms of statistics.

> For a long time I have thought I was a statistician, interested in inferences from the particular to the general. (1962)

The classical statistical paradigm: sample $\longrightarrow$ estimate population properties $\longrightarrow$ test hypotheses. The starting point is often a set of assumptions.

# Origins

The term and spirit of EDA is attributed to John Tukey, whose philosophically-leaning work in statistics in the 1960's and 1970's elaborated on the need for certain shifts in the standard disciplinary paradigms of statistics.

> All in all, I have come to feel that my central interest is in **data analysis** [which] is a larger and more varied field than inference. (1962)

Non-inferential techniques (like graphics and tabular summaries) can help to identify the particularities of a dataset, which may not conform to common statistical assumptions.

# Origins

The term and spirit of EDA is attributed to John Tukey, whose philosophically-leaning work in statistics in the 1960's and 1970's elaborated on the need for certain shifts in the standard disciplinary paradigms of statistics.

> 'Data analysis' ... allows us to use probability where it is ineeded and avoid it when we should. (1972)

An intitial 'model-free' stage of investigation pays off, leading to more realistic models or alternative analyses when modeling is not an appropriate choice.

# The role of data: evidence

When inferences are the central goal of an analysis, data often serve the role of **evidence** for or against prespecified hypotheses.

For example, in 2020's COVID vaccine trials, the objective was to determine whether vaccines were effective.

- Trial participants were vaccinated or given a placebo
- After an initial time period, reported cases among participants were examined by group

If the vaccine had little or no effect, observed cases would be distributed about 50-50 between the groups.

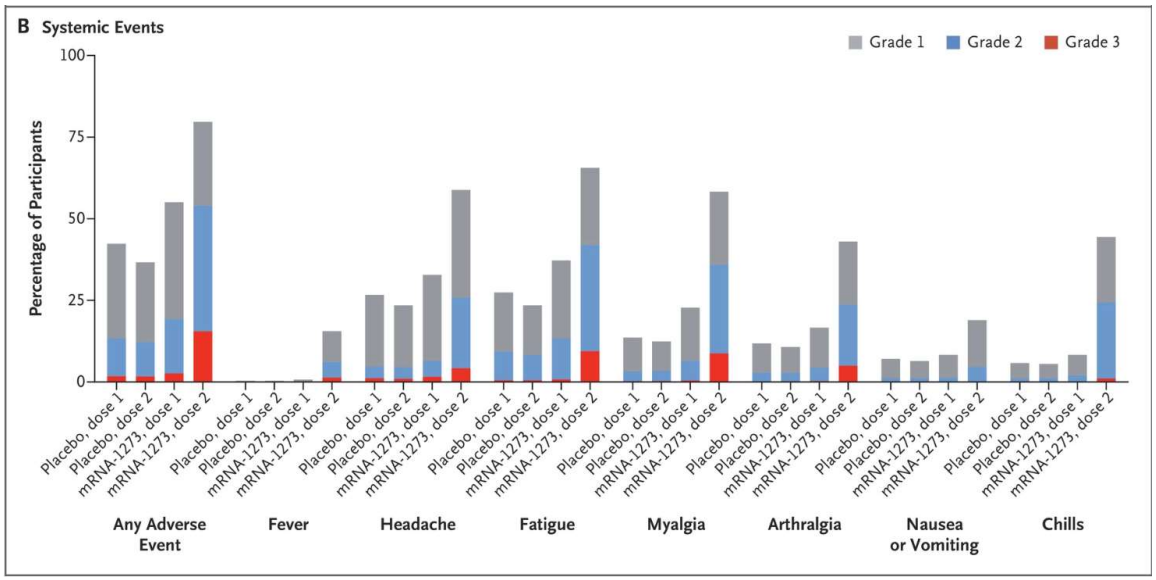
| Vaccine group | Placebo group |
|---|---|
| 11 cases | 185 cases |

The estimated probability that a case was in the vaccine group is 0.056 $\implies$ evidence of effect.

# The role of data: information

In many of the applications we've seen so far, data instead serve the role of **information** about some phenomenon.

To stick with the same example, the *very same* vaccine trial data was also used to assess safety by mointoring side effects. For this trial target, there was no specific hypothesis (*e.g.*, the vaccine causes headaches).



# Exploratory and confirmatory analysis

Tukey's ideas were a driving force in establishing a tradition of **exploratory data analysis** (EDA):

- the role of data in EDA is entirely informational;
- the goal of EDA is to identify patterns;
- undertaken with few or vague preconceptions;
- requires no assumptions.

EDA stands in contrast to **confirmatory data analysis** (CDA):

- the role of data in CDA is evidential;
- the goal of CDA is to draw conclusions;
- undertaken with specific hypotheses;
- requires statistical assumptions.

# EDA first

The picture that most practitioners have of modern data science is that EDA precedes CDA.

In this view, EDA is used to establish a background and a plan by:

- getting a grasp on what's going on;
- generating hypotheses;
- refining research questions;
- developing or ruling out possible models.

Then, CDA consists in:

- estimation and model fitting;
- hypothesis testing;
- resolving research questions.

**Aside:** Historically, statistics has focused on CDA -- and therefore a lot of your PSTAT coursework does, too.

# Essential exploratory questions

Wickham encourages us to do EDA by asking questions:

> When you ask a question, the question focuses your attention on a specific part of your dataset and helps you decide which graphs, models, or transformations to make. (R for Data Science)

There are two basic kinds of questions that are always useful:

1. What type of variation occurs within variables?
2. What type of *co*variation occurs between variables?

## Smoothing

The primary tools of EDA are graphics and tables, and arguably moreso the former (graphics).

You've *already* seen a lot of examples in HW and lab of useful graphics and tables.

We will focus on question 1 this week -- variation within variables -- and introduce **smoothing** as a visual EDA technique.

# Smoothing

- Why smooth?
- Kernel density estimation (KDE)

## Variation in one variable

**Variation** in data is the tendency of values to change from measurement to measurement.

The following are four measurements of particulate matter 2.5 microns or smaller in Aberdeen, South Dakota. They're around 8 $\frac{\mu g}{m^3}$, but each one is different.

In [3]:
```
pmdata = air.loc[:, ['PM25_mean', 'City', 'State', 'Year']].dropna()
```

In [4]:
```
pmdata.head(4)
```

Out[4]:

| | PM25_mean | City | State | Year |
|---|---|---|---|---|
| **0** | 8.6 | Aberdeen | SD | 2000 |
| **1** | 8.6 | Aberdeen | SD | 2001 |
| **2** | 7.9 | Aberdeen | SD | 2002 |
| **3** | 8.4 | Aberdeen | SD | 2003 |

So what does it mean to ask what 'type' of variation there is in a variable?

## Questions about variation

Well, there isn't a methodical classification system for types of variation.

So often the best one can do is seek to answer some useful questions about the values.

- (**Common**) Which values are most common? Why?
- (**Rare**) Which values are rare? Why?
- (**Spread**) How spread out are the values and how are they spread out?
- (**Shape**) Are values spread out evenly or irregularly? Why?

These questions often lead the way to more focused ones.

# Example: particulate matter

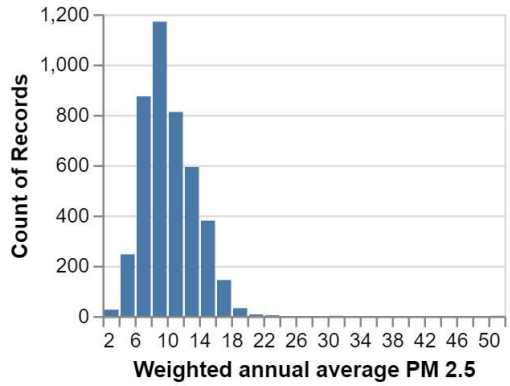
There are a lot of ways to describe variation quantitatively, but perhaps the simplest approach is to visualize the distribution of its values.

The following histogram shows the distribution of PM 2.5 concentrations from several thousand measurements taken in about 200 U.S. cities over a 20-year period.

In [5]:
```
hist1a = alt.Chart(pmdata).mark_bar().encode(
    x = alt.X('PM25_mean',
              bin = alt.Bin(maxbins = 30),
             title = 'Weighted annual average PM 2.5'),
    y = 'count()'
).properties(height = 150, width = 200)
```

In [6]:
```
hist1a
```

Out[6]:




- The **common** values have the highest bars -- values between 8 and 10.

- Values under 4 and over 18 are **rare**.
- Values are **concentrated** between 4 and 18, but are **spread** from 2 to 52.
- The **shape** is pretty even but a little more spread out to the right.

The national standard is 12 micrograms per cubic meter. Over 1,000 measurements exceeded this. Was it just a few cities, or more widespread?

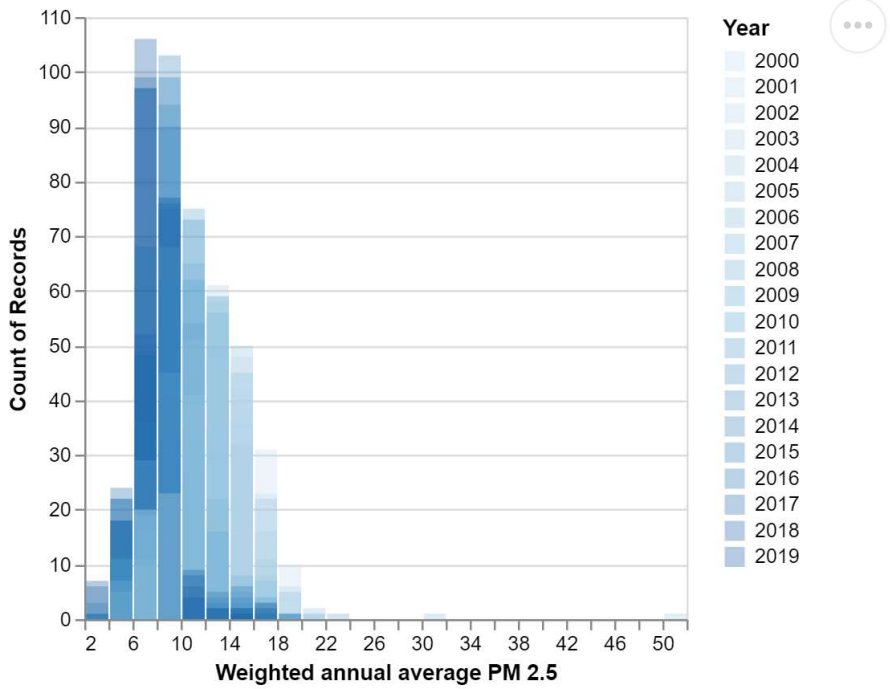
# Example: particulate matter

Maybe it would help to see years separately, so that we can see the distribution across cities each year to get a sense of how many *cities* exceed the benchmark in a typical year.

In [9]:
```
hist_annual = alt.Chart(pmdata).mark_bar(opacity = 0.3).encode(
    x = alt.X('PM25_mean',
              bin = alt.Bin(maxbins = 30),
             title = 'Weighted annual average PM 2.5'),
    y = alt.Y('count()', stack = None),
    color = 'Year:O'
).properties(height = 300, width = 300)
```

In [10]:
```
hist_annual
```

Out[10]:




But there are a lot of years, and it's hard to understand the overlapping bars.

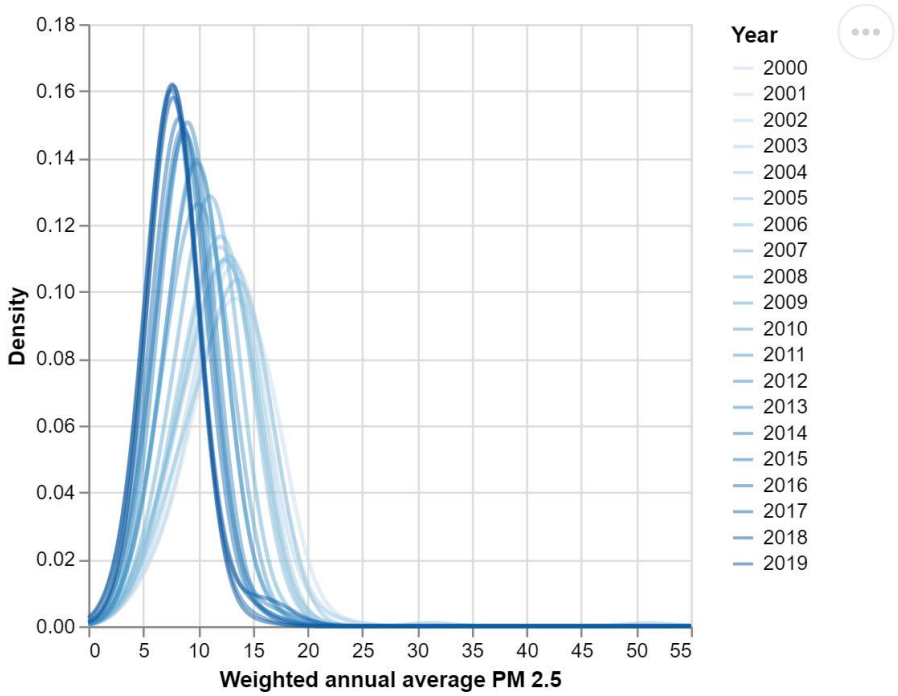
# Example: particulate matter

Visually, it's a lot easier to distinguish overlapping lines than overlapping bars. Smoothing out the histogram produces this:

In [15]:
```
density = alt.Chart(pmdata.dropna()).transform_density(
    density = 'PM25_mean',
    as_ = ['Weighted annual average PM 2.5', 'Density'],
    groupby = ['Year'],
    bandwidth = 2,
    extent = [0, 55],
    steps = 500
).mark_line(opacity = 0.5).encode(
    x = alt.X('Weighted annual average PM 2.5:Q'),
    y = alt.Y('Density:Q'),
    color = 'Year:O'
).properties(height = 300, width = 300)
```

In [16]:
```
density
```

Out[16]:




This shows the variation in PM 2.5 is diminishing over time, and the typical values are getting smaller. Good news!

These curves are known as *kernel density estimates*, and they are often useful for answering questions about variation in EDA.

- Easier to see the shape, spread, and typical values quickly.
- Easier to compare multiple distributions.

To explain how they're constructed, we'll first need to talk about rescaling histograms. Then we can get to smoothing.

# Histograms and probability densities

You might recall from 120A that a probability density/mass function has two properties:

1. Nonnegative: $f(x) \geq 0$ for every $x \in \mathbb{R}$.
2. Sums/integrates to one: $\int_{\mathbb{R}} f(x)dx = 1$ or $\sum_{x \in \mathbb{R}} f(x) = 1$

Histograms are *almost* proper density functions: they satisfy (1) but not (2).

A simple rescaling can fix this. You may appreciate why this is desirable in more depth later on.

For now, a practical view is that this ensures that any two histograms are on comparable scales.

# Count scale histograms

Typically, the bar height is simply a count of the number of observations in each bin. This is a **count scale histogram**.

If the values are $x_1, \ldots, x_n$, then the height of the bar for the $j$th bin $B_j = (a_j, b_j]$ is:

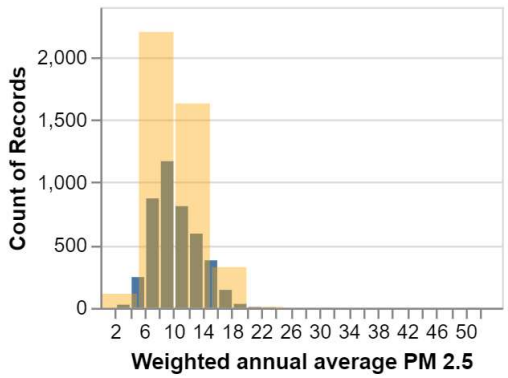
$$height_j = \sum_{i=1}^{n} \mathbf{1}\{x_i \in B_j\} = \#\{x_i \in B_j\}$$

While intuitive, this makes the bar heights incomparable in scale whenever the bin width is changed (or the sample size).

In [21]:
```
hist2a = alt.Chart(pmdata).mark_bar(opacity = 0.4, color = 'orange').encode(
    x = alt.X('PM25_mean',
              bin = alt.Bin(maxbins = 15),
             title = 'Weighted annual average PM 2.5'),
    y = 'count()'
).properties(height = 150, width = 200)
```

In [22]:
```
hist1a + hist2a
```

Out[22]:




# Density scale histograms

A fix that ensures comparability of scale for any two histograms is to normalize heights by bin width and sample size.

$$height_j = \frac{1}{nb}\sum_{i=1}^{n} \mathbf{1}\{x_i \in B_j\} = \frac{1}{nb}\#\{x_i \in B_j\} \quad \text{where} \quad b = |B_j|$$

Now the area under the histogram is $\sum_j |B_j| \times height_j = 1$.

So we call this a **density scale** histogram, because it is a valid probability density.

# Density scale histograms

Here are two density scale histograms with different choices of bin width. The bar heights are now on comparable scales.

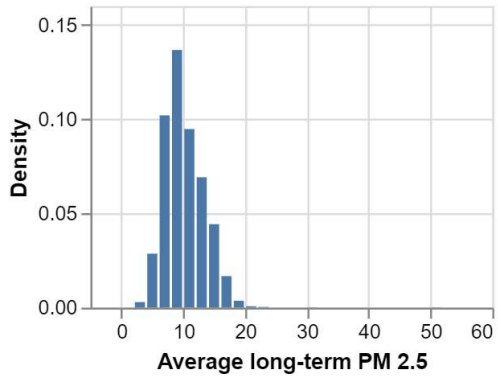
In [23]:
```
# filter, bin, count, convert scale, and plot
hist1b =  alt.Chart(pmdata).transform_bin(
    as_ = 'bin',
    field = 'PM25_mean',
    bin = alt.Bin(step = 2)
).transform_aggregate(
    Count = 'count()',
    groupby = ['bin']
).transform_calculate(
    Density = 'datum.Count/(2*4280)',
    binshift = 'datum.bin + 1'
).mark_bar(size = 5).encode(
    x = alt.X('binshift:Q', title = 'Average long-term PM 2.5', scale = alt.Scale(d
    y = alt.Y('Density:Q', scale = alt.Scale(domain = (0, 0.15)))
).properties(width = 200, height = 150)

# filter, bin, count, convert scale, and plot
hist2b =  alt.Chart(pmdata).transform_bin(
    as_ = 'bin',
    field = 'PM25_mean',
    bin = alt.Bin(step = 4)
```

```
).transform_aggregate(
    Count = 'count()',
    groupby = ['bin']
).transform_calculate(
    Density = 'datum.Count/(4*4280)',
    binshift = 'datum.bin + 2'
).mark_bar(size = 12, opacity = 0.5, color = 'orange').encode(
    x = alt.X('binshift:Q', title = 'Average long-term PM 2.5', scale = alt.Scale(d
    y = alt.Y('Density:Q', scale = alt.Scale(domain = (0, 0.15)))
).properties(width = 200, height = 150)
```

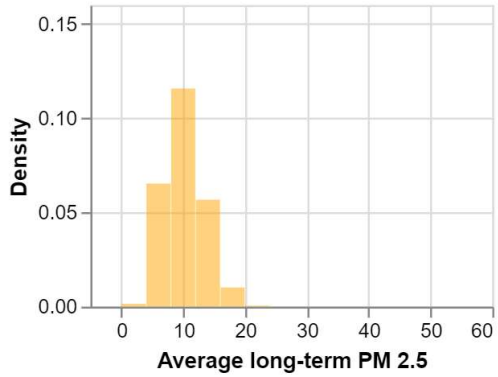
In [24]:
```
hist1b
```

Out[24]:




In [25]:
```
hist2b
```

Out[25]:




# Smoothing

**Kernel density estimates** are ***local smoothings of the density scale histogram***.

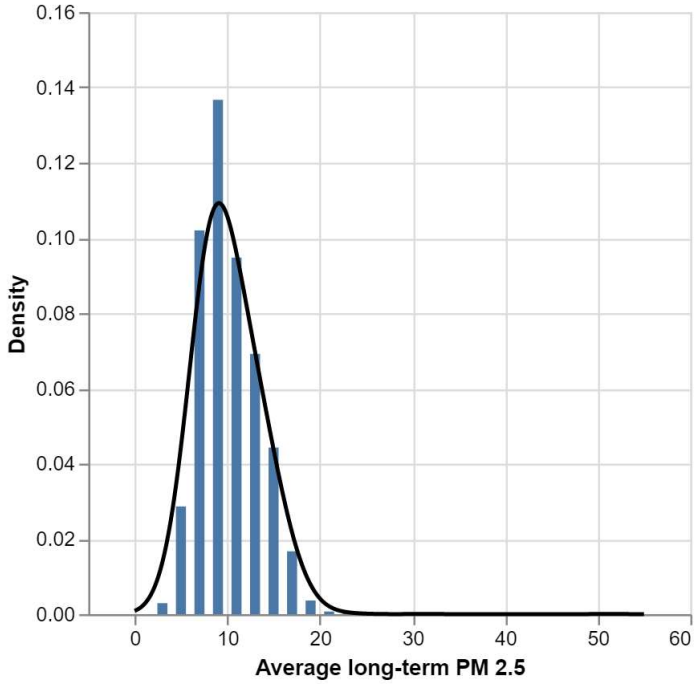
This can be seen by comparing the type of smooth curve we saw earlier with the density scale histogram.

In [26]:
```
# filter, bin, count, convert scale, and plot
kde1b =  alt.Chart(pmdata).transform_density(
    as_ = ['Average long-term PM 2.5', 'Density'],
    density = 'PM25_mean',
    extent = [0, 55],
    bandwidth = 2,
    steps = 500
```

```
).mark_line(color = 'black').encode(
    x = 'Average long-term PM 2.5:Q',
    y = 'Density:Q'
)
```

In [27]:
```
hist1b.properties(height = 300, width = 300) + kde1b
```

Out[27]:




# How KDE works

Technically KDE is a convolution filtering (if that means anything to you).

We can try to understand it in more intuitive terms by developing the idea constructively from the density histogram in two steps.

1. Make a 'local' histogram: relax the binning scheme of the histogram.
2. Change the 'kernel' function: replace counts by weighted aggregation.

# A preliminary: indicator functions

In what follows we're going to express the histogram mathematically as a function of data values.

This will require the use of **indicator functions**, which are simply ***functions that are 1 or 0 depending on a condition***. They are denoted like this:

$$\mathbf{1}\{\text{condition}\} = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{if condition is false} \end{cases}$$

Indicators are useful for expressing counts because the sum of an indicator gives the number of times the condition is met.

For example, given values $x_1, x_2, \ldots, x_n$:

$$\sum_i \mathbf{1}\{x_i > 0\} = \#\text{ of values that are positive}$$

# The usual (count) histogram

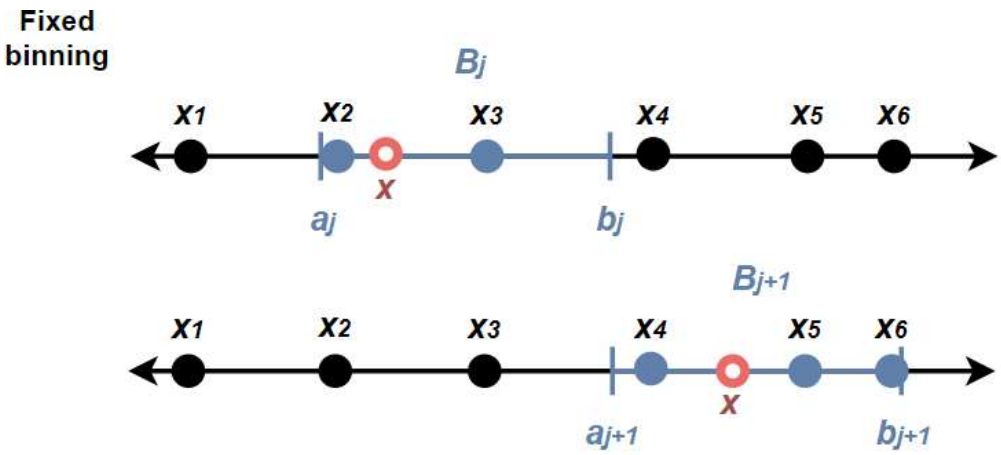
The value (height) of the density scale histogram at an arbitrary point $x$ is

$$\text{hist}(x) = \frac{1}{nb} \sum_{i=1}^{n} \sum_j \mathbf{1}\{x \in B_j\} \mathbf{1}\{x_i \in B_j\}$$

Here's what those indicators do:

$$\mathbf{1}\{x \in B_j\} \quad \text{picks out a bin}, \qquad \mathbf{1}\{x_i \in B_j\} \quad \text{picks out the data points in the bin}$$

Here's a picture:

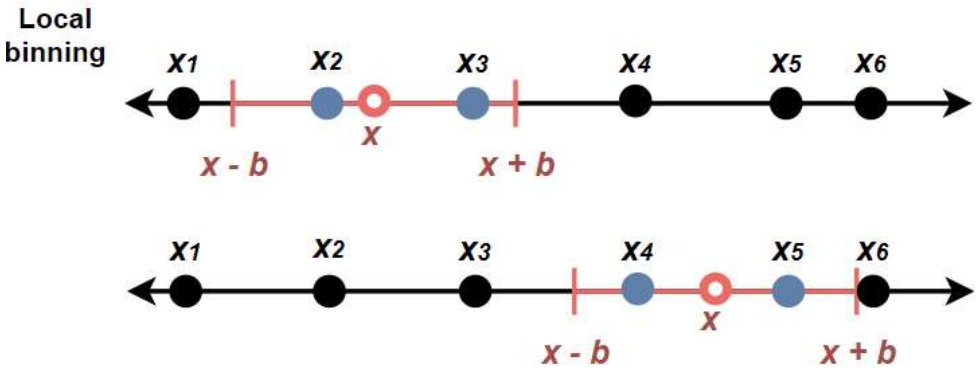



# Step 1: 'local' histogram

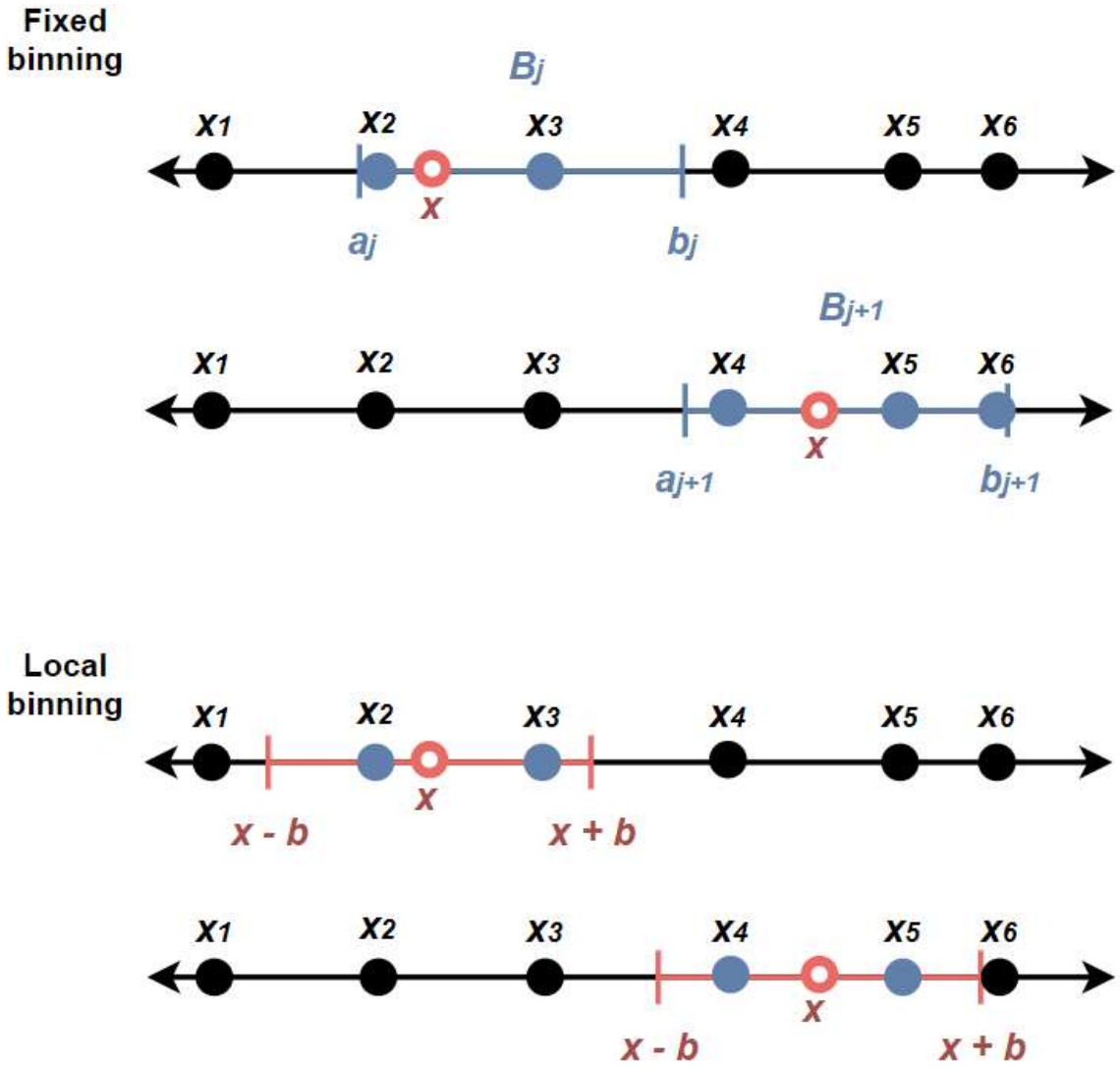
One could do something more adaptive by allowing the height at $x$ to be a normalization of the count *in a neighborhood of* $x$ rather than in one of a fixed set of bins:

$$\text{localhist}(x) = \frac{1}{nb} \sum_{i=1}^{n} \mathbf{1}\left\{|x_i - x| < \frac{b}{2}\right\} = \frac{1}{nb} \#\{x_i \in B(x)\}$$

Let's call this a **local histogram**, because the height at any point $x$ is determined relative to the exact location of $x$.




## Local vs. fixed binning




## Drawing a local histogram

$$\text{localhist}(x) = \frac{1}{nb}\sum_{i=1}^{n} \mathbf{1}\left\{|x_i - x| < \frac{b}{2}\right\} = \frac{1}{nb}\#\{x_i \in B(x)\}$$

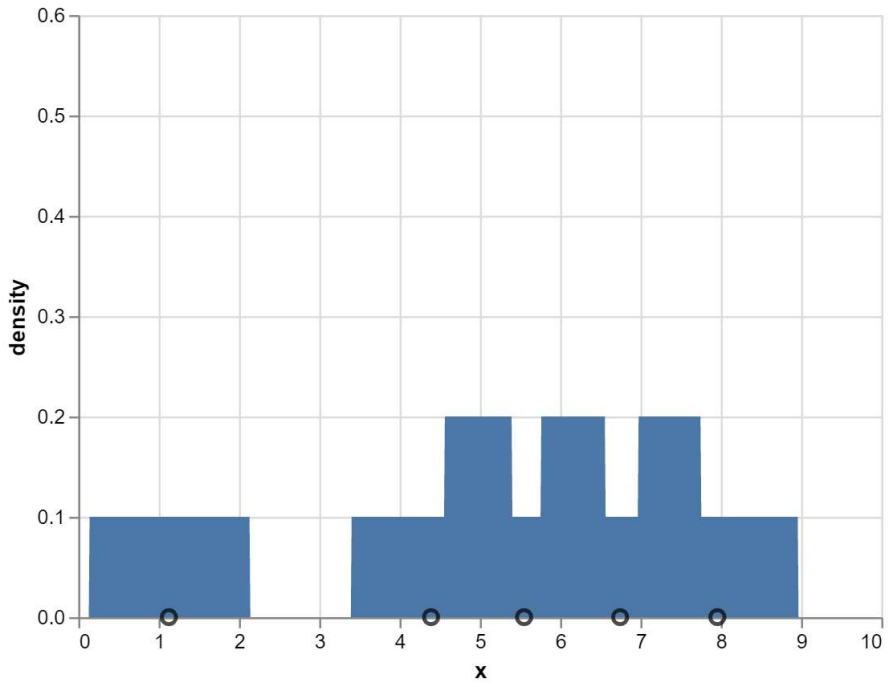
In [21]:
```
np.random.seed(42221)
samp = pd.DataFrame(data = {'x': np.random.uniform(size = 5, low = 0, high = 10)})
drawing = alt.Chart(samp).transform_calculate(
    y = '0'
).mark_point(color = 'black', size = 50).encode(
    x = 'x',
    y = alt.Y('y:Q', title = 'density', scale = alt.Scale(domain = (0, 0.6)))
)


grid1 = np.linspace(0, 10, 1000)
dens1 = []
for i in range(1000):
     dens1.append((np.abs((grid1[i - 1] - samp.x)) < 1).sum()/(2*5))
curve1 = pd.DataFrame(data = {'x': grid1, 'y': dens1})

localhist1 = alt.Chart(curve1).mark_area().encode(
    x = 'x',
    y = 'y'
)
```

In [24]:
```
localhist1 + drawing
```

Out[24]:




# Local histogram of the PM 2.5 data

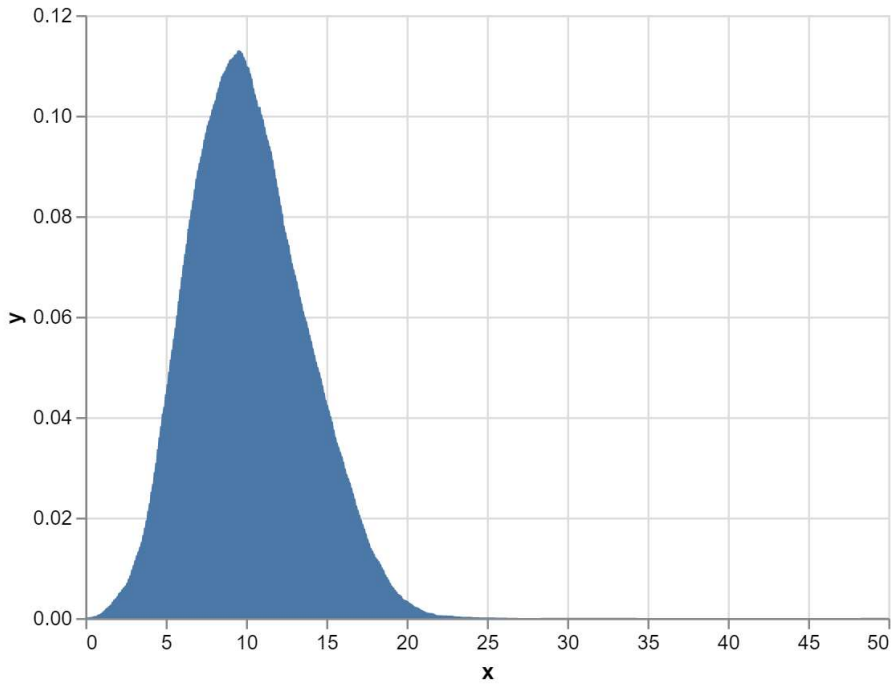
Here's what that would look like with $b = 6$:

In [29]:
```
grid = np.linspace(0, 50, 1000)
dens = []
for i in range(1000):
     dens.append((np.abs((grid[i - 1] - pmdata.dropna().loc[:, 'PM25_mean'])/1) < 3
curve = pd.DataFrame(data = {'x': grid, 'y': dens})

localhist = alt.Chart(curve).mark_area().encode(
    x = 'x',
    y = 'y'
)
```

In [30]:
```
localhist
```

Out[30]:



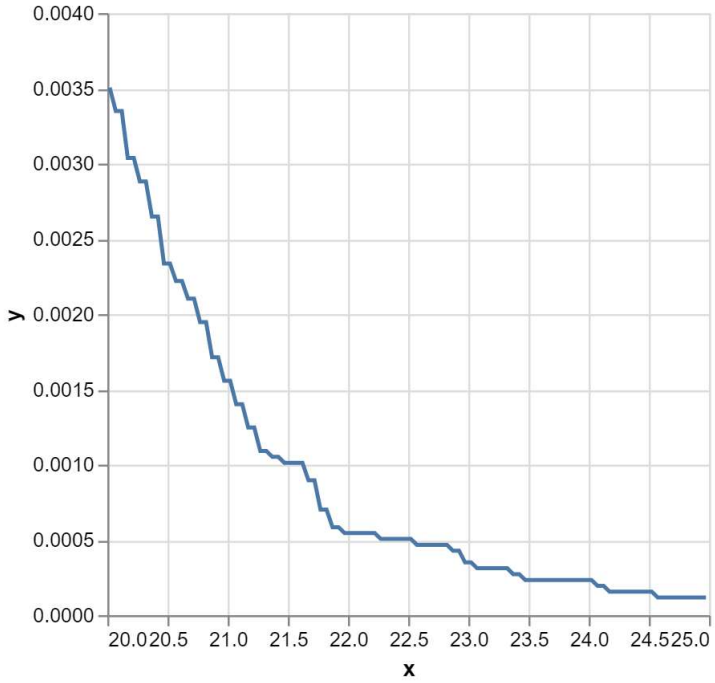

# Local histogram of the PM 2.5 data

Zooming in reveals that this is a step function:

In [31]:
```
localhist_detail = alt.Chart(curve).transform_filter(
    alt.FieldRangePredicate('x', range = [20, 25])
).mark_line().encode(
    x = 'x',
    y = 'y'
).properties(width = 300, height = 300)
```

In [32]:
```
localhist_detail
```

Out[32]:




# The kernel function

The local histogram approach is:

$$\frac{1}{nb}\sum_{i=1}^{n}\mathbf{1}\left\{|x_i - x| < \frac{b}{2}\right\}$$

The portion in blue is known as a **kernel function**, because it ***gives the core components that are aggregated***.

Since the kernel is either a 1 or 0, summing the kernel over the data points returns a count for any $x$.

# Step 2: change the kernel

The local histogram is actually a kernel density estimate with a discrete kernel function.

Let's replace that with the function $\varphi$:

$$\hat{f}\,(x) = \frac{1}{nb}\sum_{i=1}^{n}\varphi\left(\frac{x_i - x}{b}\right)$$

Where $\varphi$ is the standard Gaussian density:

$$\varphi(z) = \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{z^2}{2}\right\}$$

This is the (Gaussian) **kernel density estimate** (KDE).

In effect, the KDE curve at any point $x$ is a weighted aggregation of *all* the data with weights proportional to their distance from $x$.

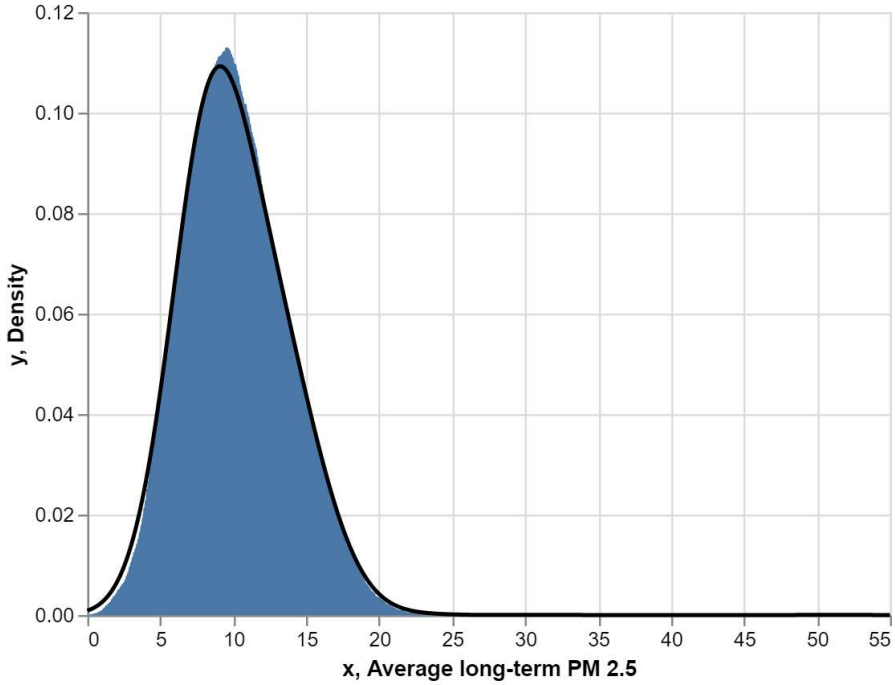
# Effect of the kernel

The Gaussian kernel function is really what's responsible for the smoothness of a KDE.

Let's compare the KDE with the local histogram:

In [34]:
```
localhist + kde1b
```

Out[34]:




Almost identical, except the KDE is smooth.

# Smoothing bandwidth

There is a bandwidth parameter, above denoted $b$, that controls how wiggly the KDE curve is.

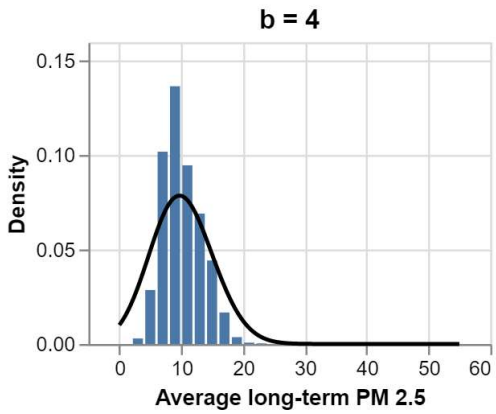
In [35]:
```
# filter, bin, count, convert scale, and plot
kde1b_lowb =  alt.Chart(pmdata.dropna()).transform_density(
    as_ = ['Average long-term PM 2.5', 'Density'],
    density = 'PM25_mean',
    extent = [0, 55],
    bandwidth = 0.25,
    steps = 500
).mark_line(color = 'black').encode(
    x = 'Average long-term PM 2.5:Q',
    y = 'Density:Q'
```

```
).properties(title = 'b = 0.5')

kde1b_highb =  alt.Chart(pmdata.dropna()).transform_density(
    as_ = ['Average long-term PM 2.5', 'Density'],
    density = 'PM25_mean',
    extent = [0, 55],
    bandwidth = 4,
    steps = 500
).mark_line(color = 'black').encode(
    x = 'Average long-term PM 2.5:Q',
    y = 'Density:Q'
).properties(title = 'b = 4')
```

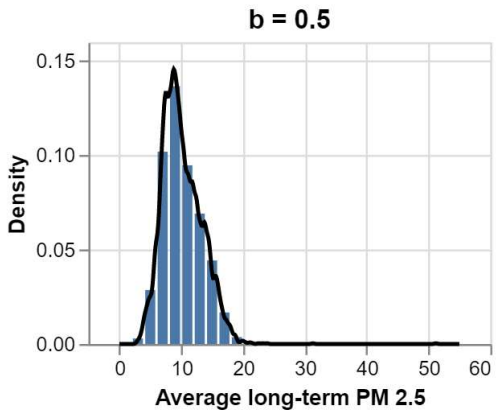
In [37]:
```
(hist1b + kde1b_highb)
```

Out[37]:




In [38]:
```
(hist1b + kde1b_lowb)
```

Out[38]:




The left panel is *too* smooth -- the KDE curve doesn't get the shape right.

But the right panel is too wiggly -- the KDE curve hardly smooths out the shape at all.

There are methods for determining 'optimal' choices of bandwidth, but usually it's picked by trial-and-error.

- Advantage: you can explore the effect of bandwidth choice on the impression the plot makes.

# Returning to PM 2.5

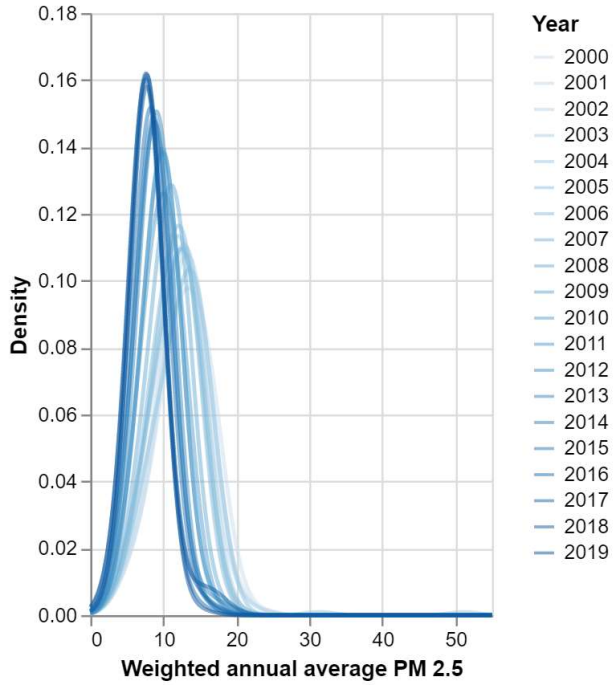
It's important to be aware that the smoothing bandwidth can change the visual impression of what's going on in a plot and affect choices later in analysis.

In [39]:
```
density_higherb = alt.Chart(pmdata).transform_density(
    density = 'PM25_mean',
    as_ = ['Weighted annual average PM 2.5', 'Density'],
    groupby = ['Year'],
    bandwidth = 2,
    extent = [0, 55],
    steps = 500
).mark_line(opacity = 0.5).encode(
    x = alt.X('Weighted annual average PM 2.5:Q'),
    y = alt.Y('Density:Q'),
    color = 'Year:O'
).properties(height = 300, width = 200)

density_lowerb = alt.Chart(pmdata).transform_density(
    density = 'PM25_mean',
    as_ = ['Weighted annual average PM 2.5', 'Density'],
    groupby = ['Year'],
    bandwidth = 0.7,
    extent = [0, 55],
    steps = 500
).mark_line(opacity = 0.5).encode(
    x = alt.X('Weighted annual average PM 2.5:Q'),
    y = alt.Y('Density:Q'),
    color = 'Year:O'
).properties(height = 300, width = 200)
```

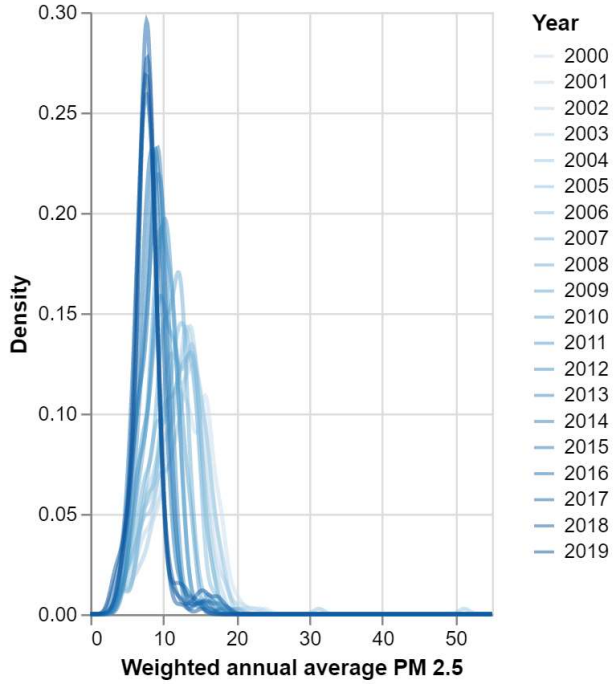
In [42]:
```
density_higherb
```

Out[42]:




In [43]:
```
density_lowerb
```

Out[43]:




Here they mostly tell the same story about what happens over time, but the higher bandwidth obscures a few outliers.

Next question: which cities/years are the outliers? What, if anything, explains the abnormally high PM2.5?

# Connections with probability

- What's a probability distribution again?
- EDA and conditional distributions

# What's a probability distribution again?

In PSTAT 120A, you learned about random variables.

At their essence, random variables are models of the natural variation we see in life.

All those *probability distributions* you covered -- the binomial, Gaussian/normal, Poisson, uniform, gamma, etc. -- are theoretical constructs describing different types of variation. In other words, models.

Their density/mass functions show the distribution of values of a random variable. You can think of this as an abstraction of the histogram.

# Conditional distributions

The *conditional distribution* of $Y$ given $X$ is the distribution of values of a random variable $Y$ when another random variable $X$ is fixed at some value.

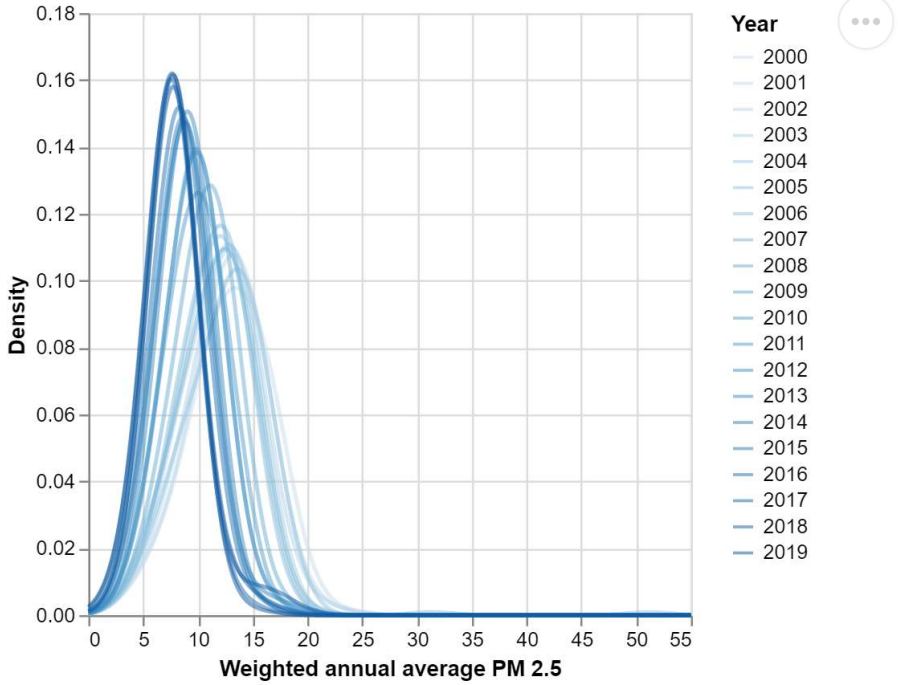
When $X, Y$ are dependent, the conditional distribution of $(Y|X = x)$ changes with $x$.

# Connection with EDA

In our PM2.5 example, we were examining how the distribution of PM2.5 concentration differs by year.

In [44]:
```
density
```

Out[44]:




In other words, we were looking at the conditional distribution of PM2.5, given year.

# Most EDA is really about conditional distributions

When exploring a dataset -- seeking to understand variation and co-variation -- usually the analyst is investigating conditional distributions (whether they know it or not!).

- PM2.5 given year
- Gender gap given test subject and socioeconomic status
- General health given adverse childhood experiences
- Life expectancy given national wealth

Think about this as we move next week into discussing techniques for exploring *co*variation.