Statistics in Data Science

overview

1. **Introduction to Statistics in Data Science**
2. **Distributions**
3. **Point Estimates**
4. **Distribution Estimators: MoM, MLE, KDE**
5. **Statistical Hypothesis Testing**
6. **Correlation**
7. **Practical Examples**

3. Estimates

- Explain the concept of estimation in statistics.
- Discuss point estimates and distribution estimates.
- Introduce the idea of sample statistics as estimators of population parameters:
    - Sample mean as an estimator of population mean.
    - Sample standard deviation as an estimator of population standard deviation.
- Discuss the properties of good estimators.

4.Distribution Estimators: MoM, MLE, KDE

- Introduce the concept of distribution estimation.
- Explain the Method of Moments (MoM) as an estimation technique based on sample moments.
- Discuss Maximum Likelihood Estimation (MLE) as an alternative technique for estimating distribution parameters.
- Mention Kernel Density Estimation (KDE) as a non-parametric method for estimating probability densities.
- Provide examples and use cases for each estimation method.

- **Distribution estimation**, also known as statistical distribution fitting or parameter estimation, is the process of determining the probability distribution that best describes a given dataset. In statistics, datasets are often assumed to follow specific probability distributions, and distribution estimation aims to identify the parameters of these distributions to model and analyze the data effectively.

- Here are the key aspects of distribution estimation:

1. Selection of Probability Distribution:

   Before performing distribution estimation, you need to choose a probability distribution that you believe is an appropriate model for your dataset. The choice of distribution is often guided by the nature of the data and its characteristics.

   Eg: normal (Gaussian) / exponential / Poisson / gamma distribution, and others.

2. Parameter Estimation:

Each probability distribution has one or more parameters that determine its shape, location, and scale. For example, the normal distribution is characterized by two parameters: mean (μ) and standard deviation (σ).

- The goal of distribution estimation is to estimate the values of these parameters based on the available data.

Two common methods for parameter estimation are:

Method of Moments (MoM): This approach equates sample moments (e.g., sample mean and sample variance) with the moments of the chosen distribution to estimate the parameters.

Maximum Likelihood Estimation (MLE): MLE finds the parameter values that maximize the likelihood of observing the given data under the assumed distribution

3. Goodness of Fit:

Once you have estimated the distribution parameters, you should assess how well the selected distribution fits the observed data. This is often done by comparing the empirical data's histogram or density plot with the theoretical distribution's probability density function (PDF) or probability mass function (PMF).

Statistical tests, such as the Kolmogorov-Smirnov test or the chi-squared goodness-of-fit test, can help assess the goodness of fit.

4. Application:

Once you have successfully estimated the distribution parameters and confirmed the goodness of fit, you can use the distribution model for various statistical analyses and predictions.

5. Model Selection:

In some cases, you may need to compare different candidate distributions to determine which one best describes the data.

Distribution Estimators: MoM, MLE, KDE

Two common methods for parameter estimation are:

Method of Moments (MoM): This approach equates sample moments (e.g., sample mean and sample variance) with the moments of the chosen distribution to estimate the parameters.

Maximum Likelihood Estimation (MLE): MLE finds the parameter values that maximize the likelihood of observing the given data under the assumed distribution

Kernel Density Estimation (KDE) is a non-parametric method for estimating probability densities.

MoM: Method of Moments

- The Method of Moments (MoM) is a statistical technique used to estimate the parameters of a probability distribution or statistical model. It's based on equating the sample moments (typically the mean, variance, skewness, etc.) to their theoretical counterparts in the chosen distribution. The main idea is to set up equations using moments and solve for the parameters of interest.

MoM: Method of Moments

Step 1: Define the Population Moment(s)

First, you need to identify the moments of the population or distribution you want to estimate. These moments could include the mean, variance, skewness, kurtosis, or any other higher moments.

Step 2: Compute the Sample Moments

Next, you calculate the corresponding sample moments from your observed data. For example, if you want to estimate the mean and variance, you calculate the sample mean and sample variance from your dataset.

MoM: Method of Moments

Step 3: Set Up Equations

Now, set up equations equating the sample moments to the population moments. Each moment equation represents a constraint on the parameter(s) of the distribution. The number of equations depends on the number of parameters you want to estimate.

Step 4: Solve for Parameters

Solve the system of equations to find the parameter values that satisfy the moment conditions. These parameter values are your estimates for the population or distribution parameters.

MoM: Method of Moments

Example 1: Exponential Distribution

Suppose you have a dataset of waiting times between events, and you want to estimate the rate parameter (λ) of an exponential distribution.

Population Moment: The mean of an exponential distribution is 1/λ.

Sample Moment: Calculate the sample mean of your dataset.

Set up the equation:

Sample Mean = 1/λ.

Solve for λ:

λ = 1 / Sample Mean.

In this case, λ is your estimate for the rate parameter.

MoM: Method of Moments

Example 2: Normal Distribution

Suppose you have a dataset of exam scores, and you want to estimate the mean (μ) and variance ($σ^2$) of a normal distribution.

Population Moments: The mean of a normal distribution is μ, and the variance is $σ^2$.

Sample Moments: Calculate the sample mean and sample variance of your dataset.

Set up two equations:

Sample Mean = μ.

Sample Variance = $σ^2$.

Solve for μ and $σ^2$ directly using the sample moments.

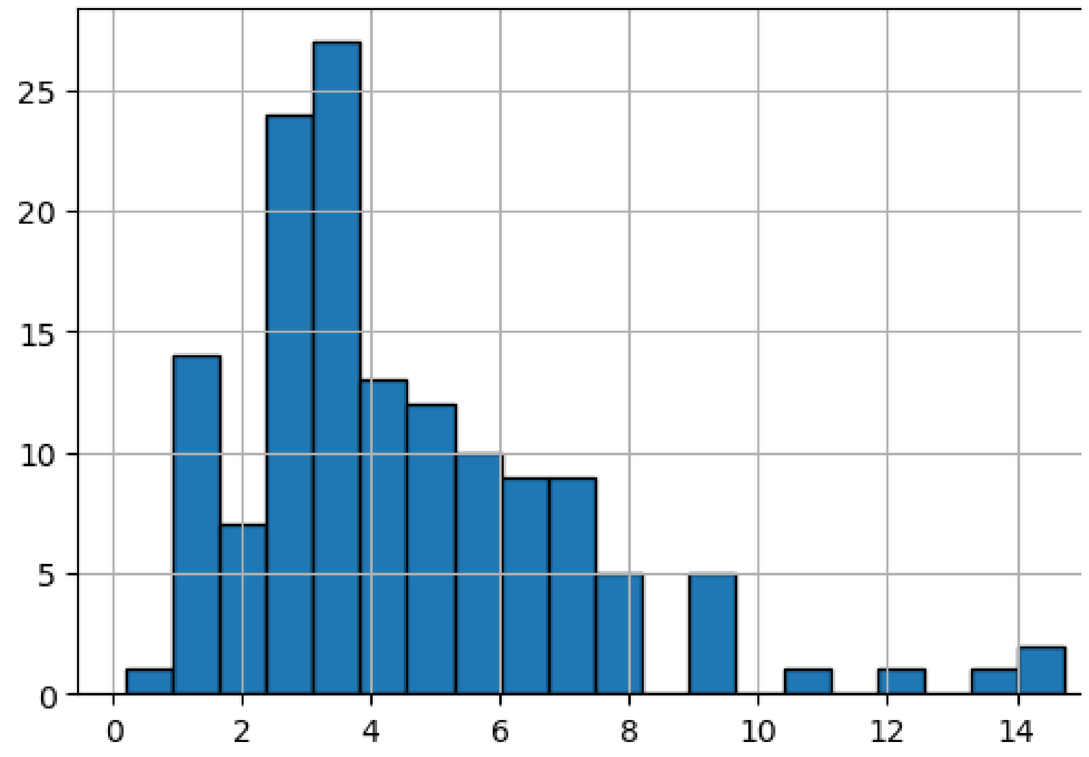
MoM: Method of Moments

Example: Nashville Precipitation

The dataset nashville_precip.txt

contains NOAA precipitation data for Nashville measured since 1871.

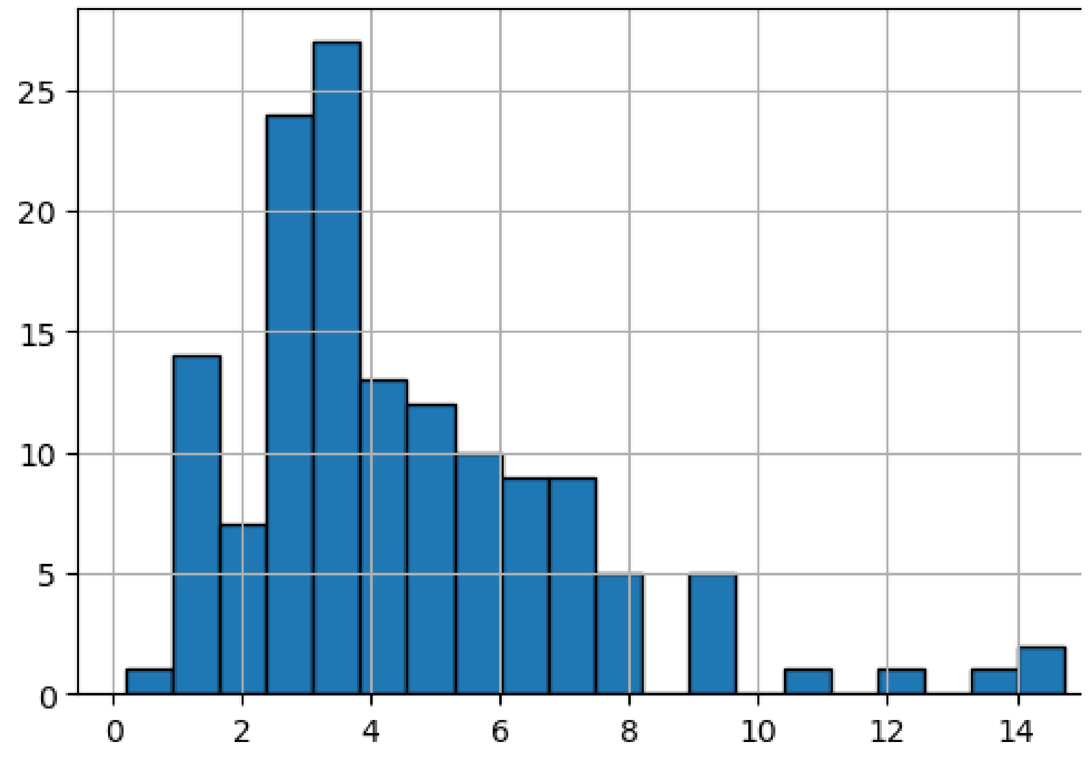
- Histogram of Jan Rainfall



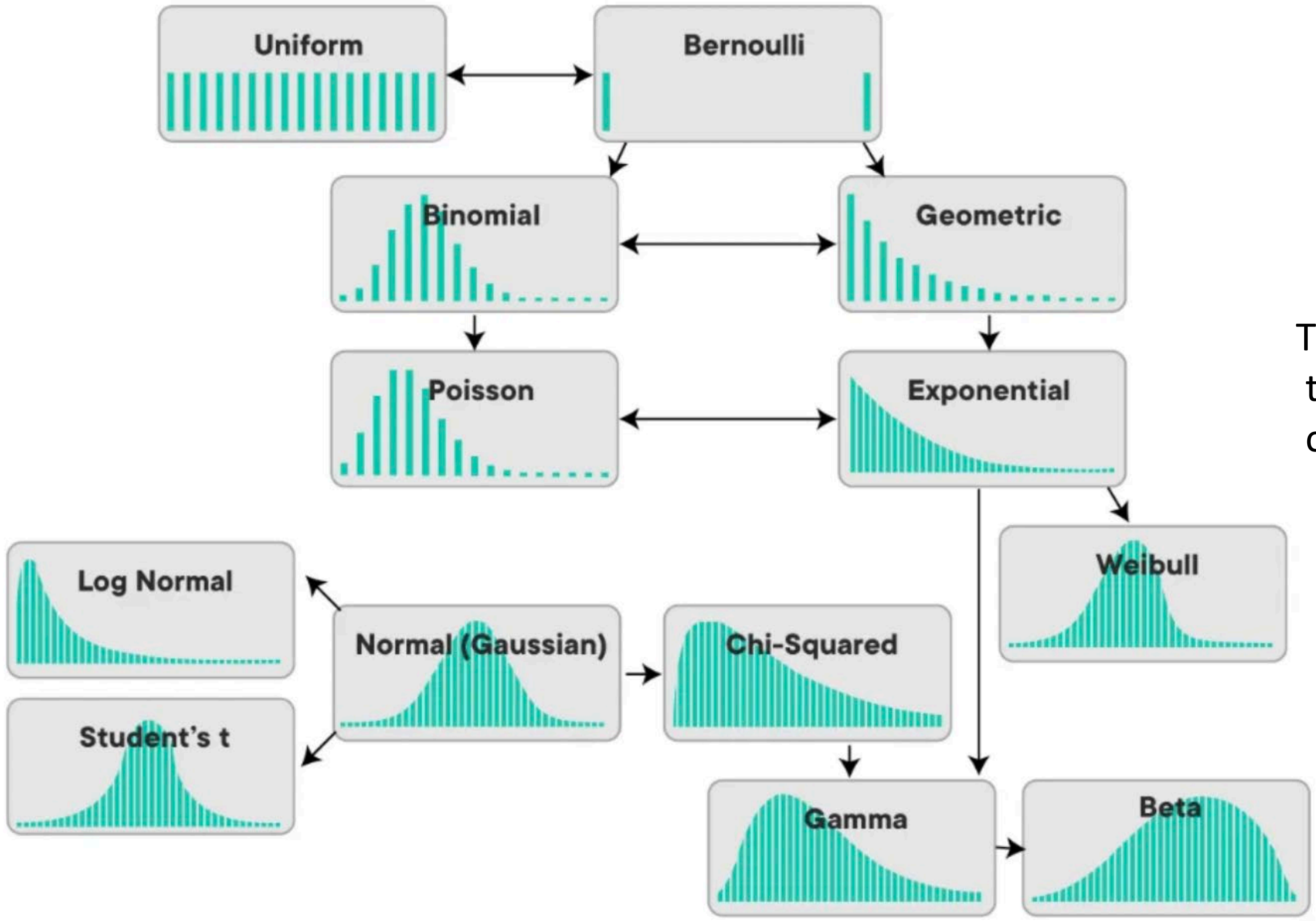
MoM: Method of Moments

The first step is recognizing what sort of distribution to fit our data to. A couple of observations:

1. The data are skewed, with a longer tail to the right than to the left.
2. The data are positive-valued, since they are measuring rainfall.
3. The data are continuous.



What distribution can be used here?



Common types of distributions

The gamma distribution is often a good fit to aggregated rainfall data, and will be our candidate distribution in this case.

Gamma distribution:

$$f(x;\alpha,\beta) \sim P(\mu,\sigma) \sim \text{Gamma}(\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$$

Estimator 1 - Method of Moments (MoM)

The ***method of moments*** simply ***assigns the empirical (sample) mean and variance to their theoretical counterparts, so that we can solve for the parameters.***

- ***Method of moments*** - chooses the parameters so that the sample moments (typically the sample mean and variance) match the theoretical moments of our chosen distribution.

So, for the gamma distribution, the mean and variance are:

$$\hat{\mu} = \bar{X} = \alpha\beta$$
$$\hat{\sigma}^2 = S^2 = \alpha\beta^2$$

So, if we solve for these parameters, we can use a gamma distribution to describe our data:

$$\alpha = \frac{\bar{X}^2}{S^2},\ \beta = \frac{S^2}{\bar{X}}$$

MoM applied to Nashville Precipitation

- Deal with missing value:

```
precip.fillna(value={'Oct': precip.Oct.mean()}, inplace=True)
```

- Calculate the sample moments

the means ($\hat{\mu}$)

and variances ($\hat{\sigma}$^2) by month

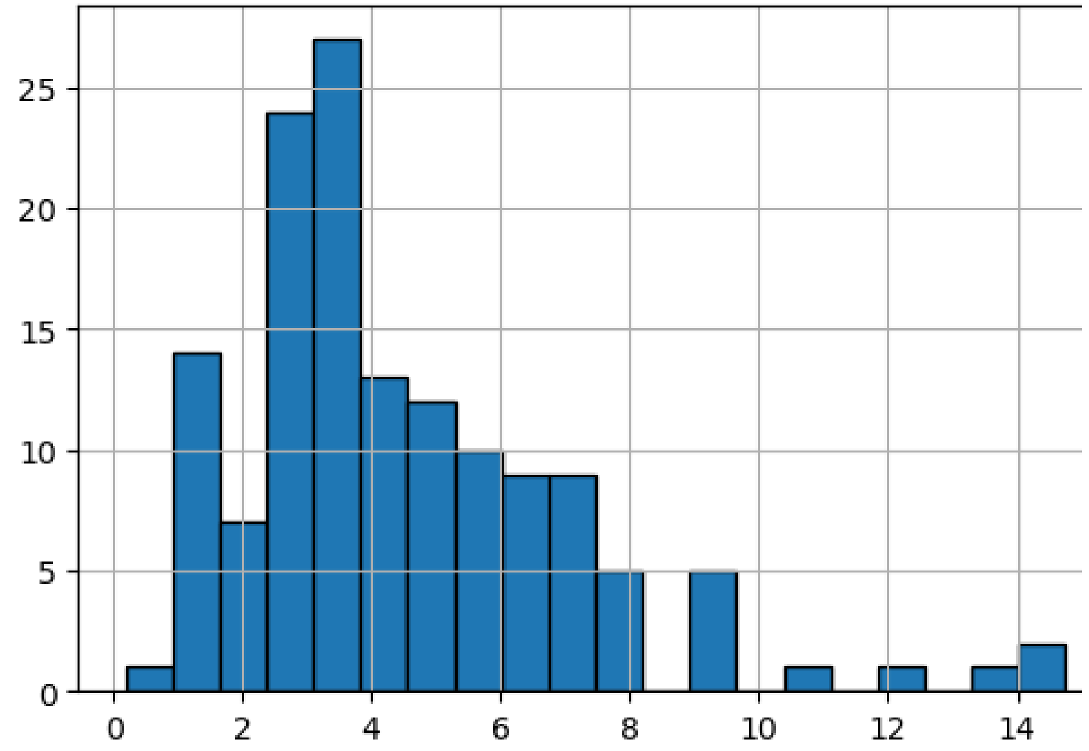
```
precip_mean = precip.mean()
```

```
precip_var = precip.var()
```

- use these moments to

estimate α and β for each month

```
alpha_mom = precip_mean ** 2 / precip_var
beta_mom = precip_var / precip_mean
```



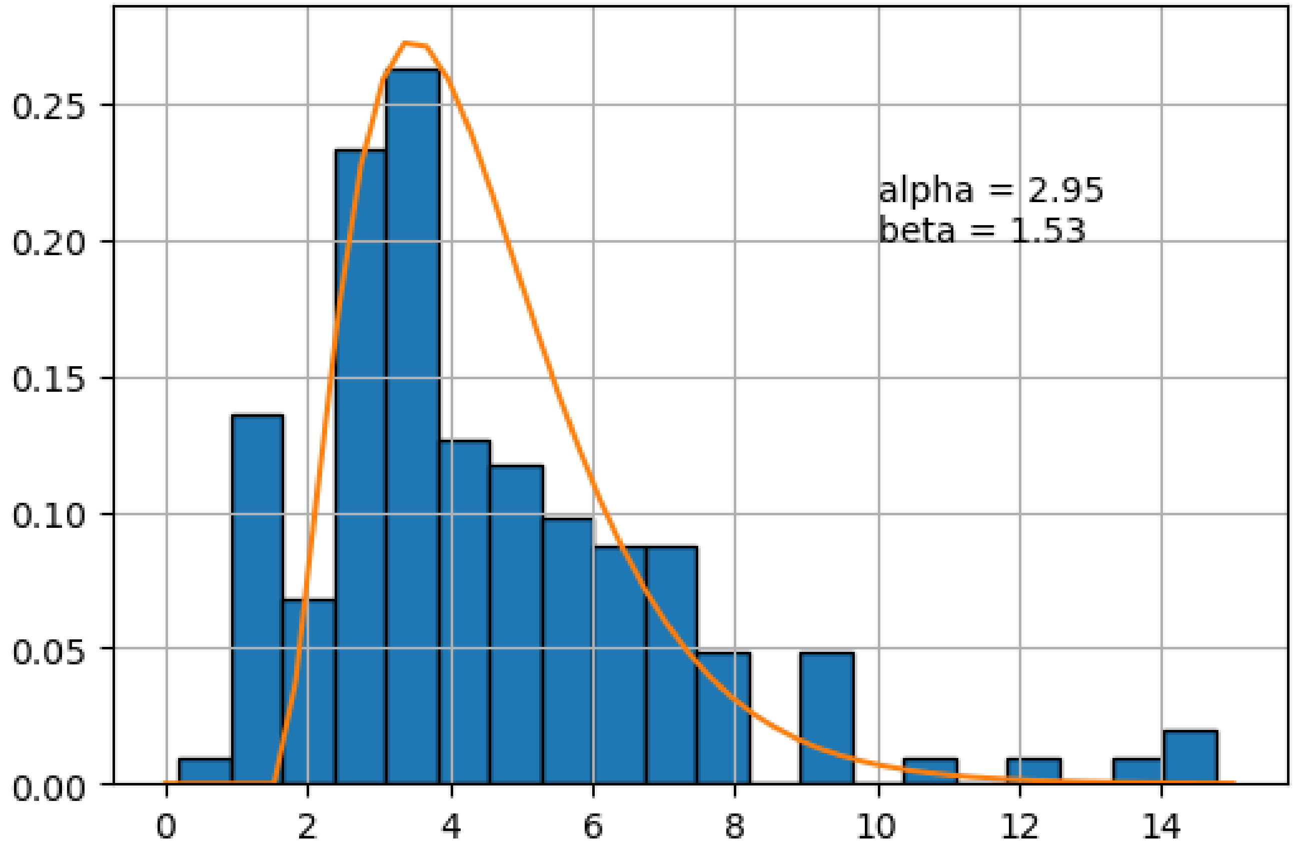
Now that we have our α and β, lets plug back into the formula:

$$f(x;\alpha,\beta) \sim \text{Gamma}(\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$$

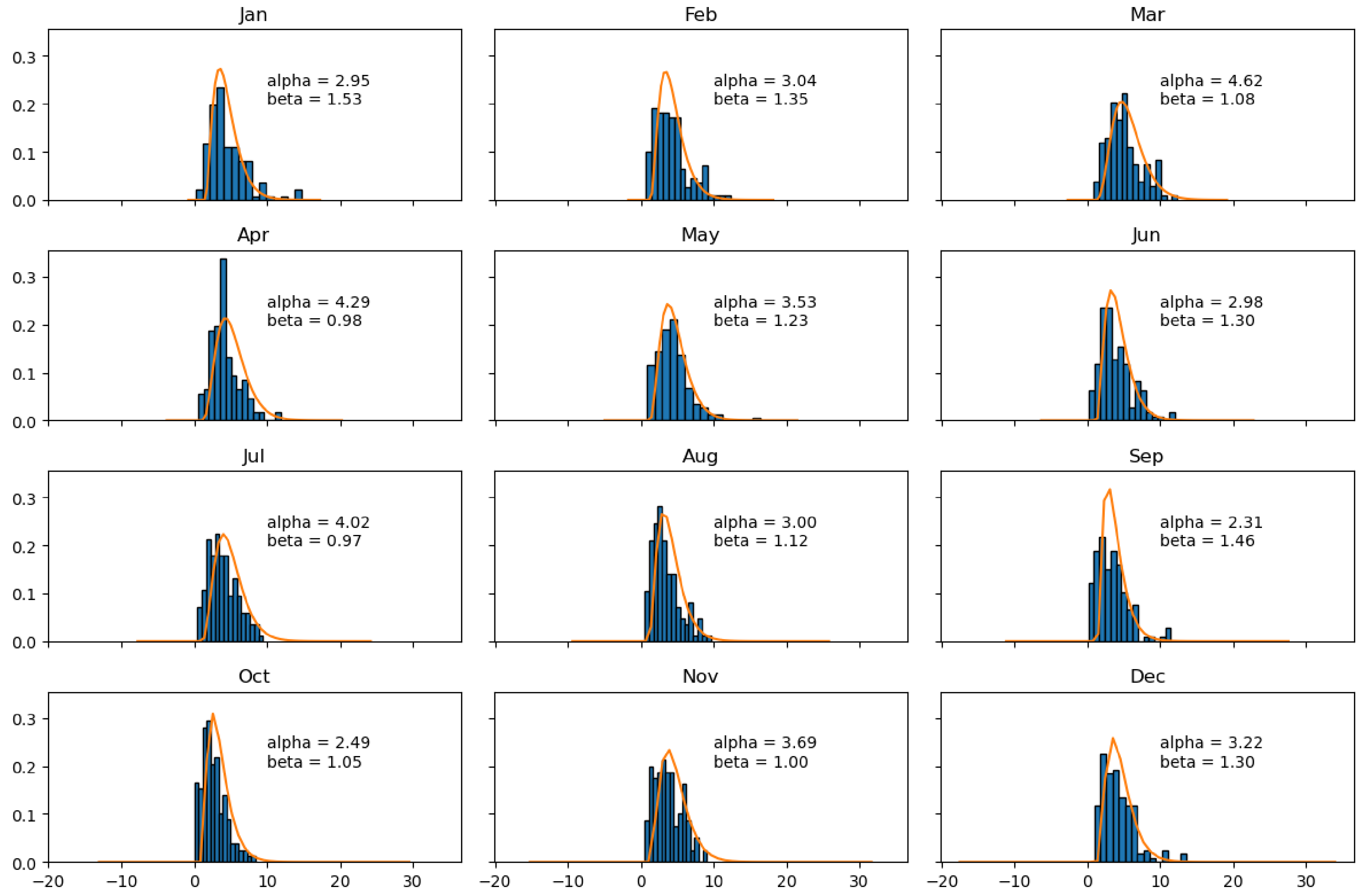
As it is a continious function we can evaluate `pdf()` at each point to get the distribution.

We can use the `gamma.pdf()` function in `scipy.stats.distributions` to plot the distributions implied by the calculated alphas and betas. For example, here is January:

```python
from scipy.stats.distributions impo

precip.Jan.hist(density=True, bins=
label = 'alpha = {0:.2f}\nbeta = {1
plt.annotate(label, xy=(10, 0.2))
plt.plot(np.linspace(0, 15), gamma.
```



- Looping over all months, we can create a grid of plots for the distribution of rainfall, using the gamma distribution:



MLE: Maximum Likelihood Estimation

- Maximum Likelihood Estimation (MLE) is a method for estimating the parameters of a statistical model. It's based on the principle of finding the parameter values that maximize the likelihood function, which measures how well the model explains the observed data. MLE aims to find the parameter values that make the observed data the most probable under the assumed statistical model.

MLE: Maximum Likelihood Estimation

Step 1: Define the Likelihood Function

The likelihood function, denoted as L(θ; x), represents the probability of observing the given data (x) under a specific set of parameter values (θ) in the statistical model.

Step 2: Take the Natural Logarithm

To simplify calculations, it's common to work with the log-likelihood function, denoted as ln L(θ; x). Taking the natural logarithm doesn't change the location of the maximum likelihood, but it simplifies computations.

MLE: Maximum Likelihood Estimation

Step 3: Formulate the Likelihood Function

The likelihood function depends on the specific statistical model. For example:

- In a normal distribution, the likelihood function involves the probability density function (PDF) of the normal distribution.
- In a binomial distribution, the likelihood function is based on the binomial probability formula.

Step 4: Maximize the Log-Likelihood

To find the MLE estimates, you maximize the log-likelihood function by setting its derivative with respect to the parameters (θ) to zero. This yields a system of equations.

Step 5: Solve for Parameters

Solve the system of equations to find the parameter values that maximize the log-likelihood function. These parameter values are your MLE estimates.

MLE: Maximum Likelihood Estimation

Let's look at some examples to illustrate MLE:

Example 1: MLE for Normal Distribution

Suppose you have a dataset of exam scores, and you want to estimate the mean (μ) and variance (σ^2) of a normal distribution.

Likelihood Function: The likelihood function involves the PDF of the normal distribution with parameters μ and σ^2.

Log-Likelihood: Take the natural logarithm of the likelihood function.

Set up two equations:

$\partial \ln L/\partial \mu = 0$ (partial derivative of log-likelihood with respect to μ).

$\partial \ln L/\partial \sigma^2 = 0$ (partial derivative of log-likelihood with respect to σ^2).

Solve these equations to find MLE estimates for μ and σ^2.

Our θ is a parameter which estimates `x = [4, 5, 7, 8, 8, 9, 10, 5, 2, 3, 5, 4, 8, 9]` which we are assuming comes from a normal distribution PDF.

$$f(x_i; \mu, \sigma) = \text{Norm}(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

- We want to maximize the likelihood our parameter θ comes from this distribution.
- To make things simpler we're going to take the `log` of the equation.
- Overall we are tyring to define a function $f()$ where x is most **likely** to appear.
  - due to the `log` we call it `log likelihood`

$$log\ (f\ (x_i; \mu, \sigma^2)) = -\frac{n}{2} log(2\pi) - \frac{n}{2} log(\sigma^2) - \frac{1}{2\sigma^2} \sum (x_i - \mu)^2$$

Now we want to substitute θ in for μ and σ in our likelihood function. Let's call them θ_mu and θ_sigma.

To maximize our equation with respect to each of our parameters, we need to take the ***derivative*** and set the equation to zero.

- for μ
- for σ

First, let's estimate θ_mu from our `Log Likelihood Equation` above:

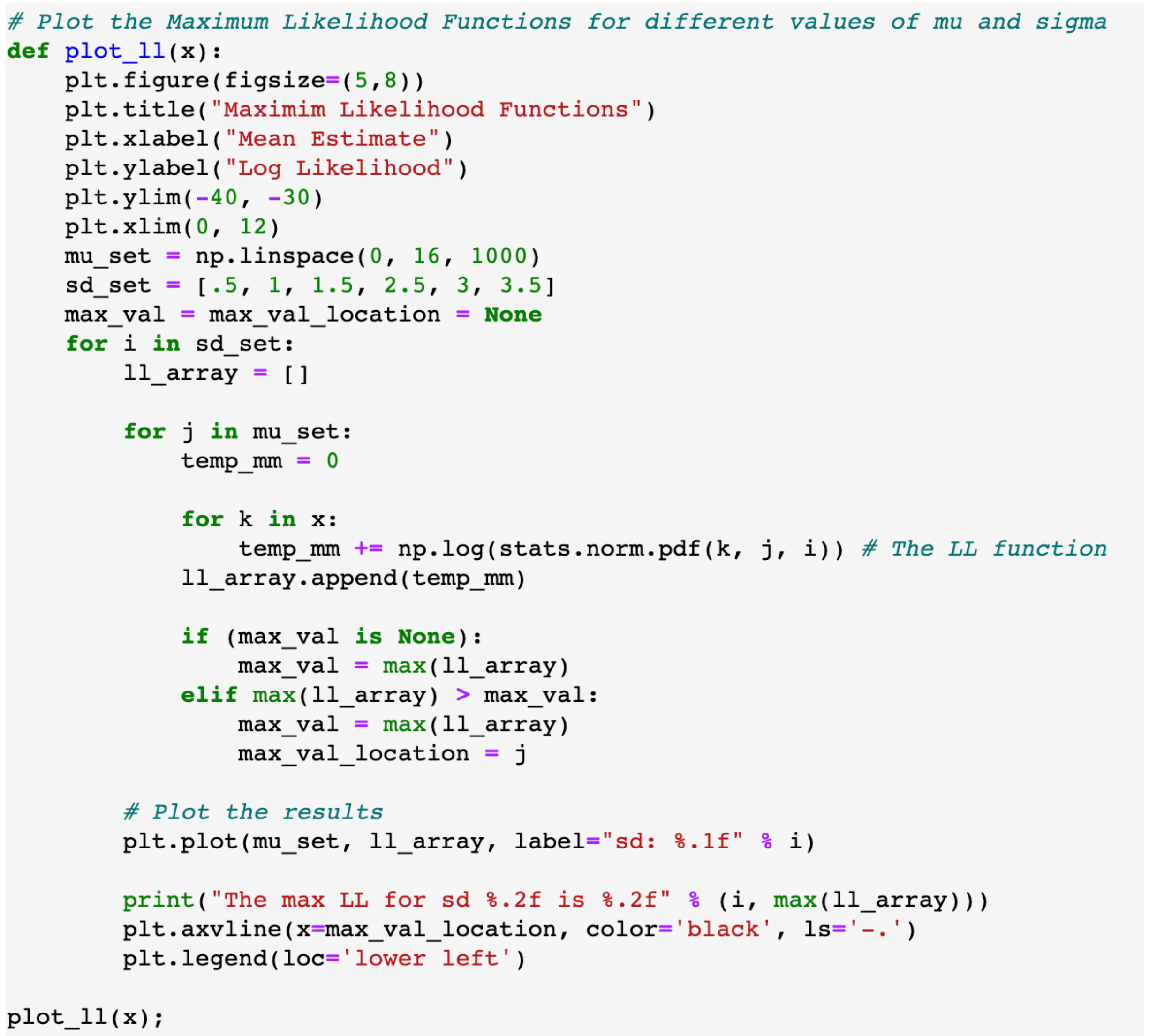
$$\frac{\delta x}{\delta \theta\_mu} = 0 \Rrightarrow \theta\_mu = \mu = \frac{\sum_i^n x_i}{n}$$

Now we can be certain the maximum likelihood estimate for θ_mu is the sum of our observations, divided by the number of observations.

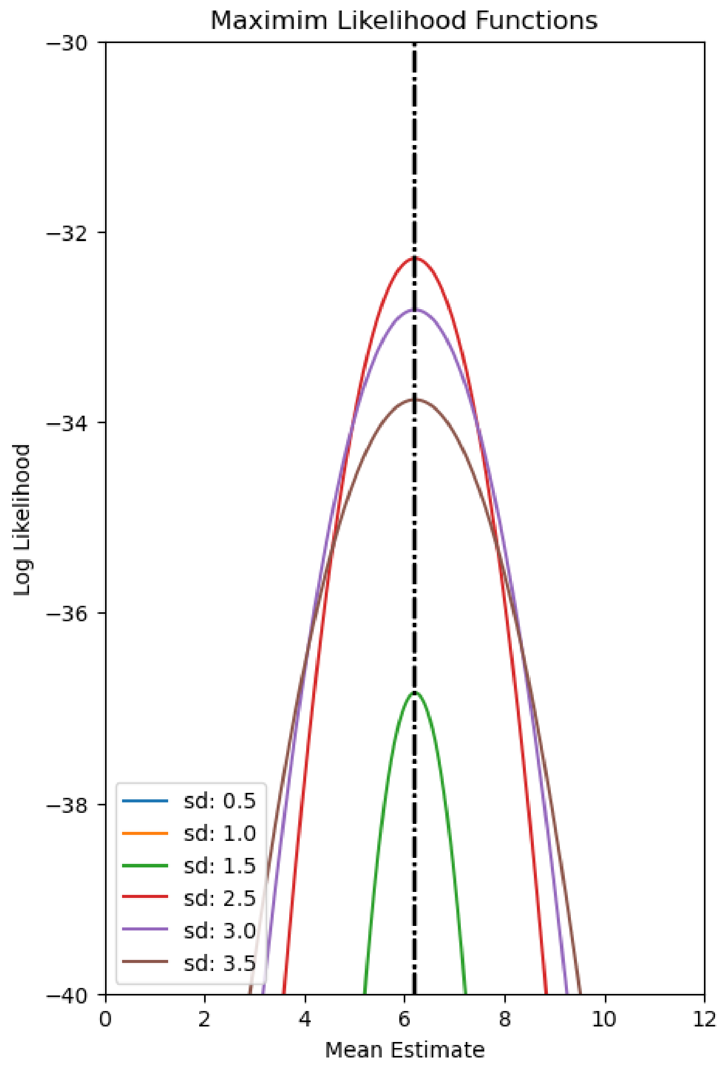
And let's do the same for θ_sigma.

$$\frac{\delta x}{\delta \theta\_sigma} = 0 \Rrightarrow \theta\_sigma = \sigma^2 = \frac{\sum_i^n (x_i - \mu)^2}{n}$$

- Now we know how to estimate both these parameters from the observations we have.
- Let’s look at the visualization of how the MLE for θ_mu and θ_sigma is determined.



```python
# Plot the Maximum Likelihood Functions for different values of mu and sigma
def plot_ll(x):
    plt.figure(figsize=(5,8))
    plt.title("Maximim Likelihood Functions")
    plt.xlabel("Mean Estimate")
    plt.ylabel("Log Likelihood")
    plt.ylim(-40, -30)
    plt.xlim(0, 12)
    mu_set = np.linspace(0, 16, 1000)
    sd_set = [.5, 1, 1.5, 2.5, 3, 3.5]
    max_val = max_val_location = None
    for i in sd_set:
        ll_array = []

        for j in mu_set:
            temp_mm = 0

            for k in x:
                temp_mm += np.log(stats.norm.pdf(k, j, i)) # The LL function
            ll_array.append(temp_mm)

            if (max_val is None):
                max_val = max(ll_array)
            elif max(ll_array) > max_val:
                max_val = max(ll_array)
                max_val_location = j

        # Plot the results
        plt.plot(mu_set, ll_array, label="sd: %.1f" % i)

        print("The max LL for sd %.2f is %.2f" % (i, max(ll_array)))
        plt.axvline(x=max_val_location, color='black', ls='-.')
        plt.legend(loc='lower left')

plot_ll(x);
```



MLE: Maximum Likelihood Estimation

Example 2: MLE for Bernoulli Distribution

Suppose you have a dataset of coin tosses (0 for tails, 1 for heads), and you want to estimate the probability (p) of getting heads.

Likelihood Function: The likelihood function involves the Bernoulli probability formula.

Log-Likelihood: Take the natural logarithm of the likelihood function.

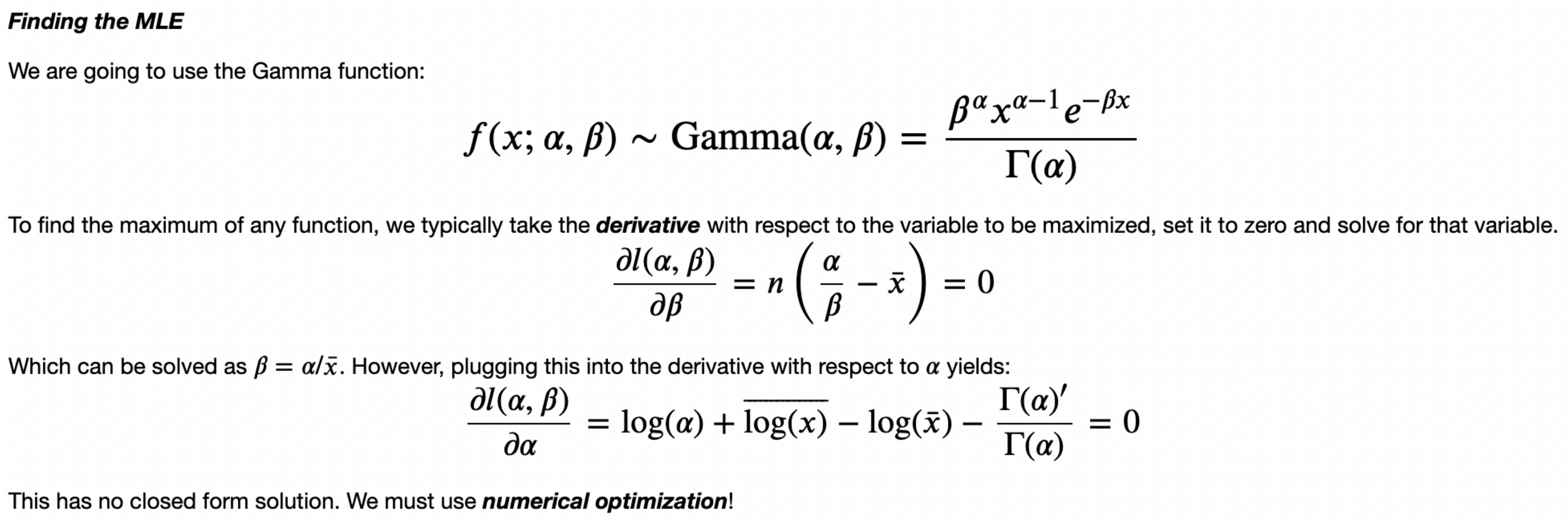
Set up one equation:

$\partial \ln L/\partial p = 0$ (partial derivative of log-likelihood with respect to p).

Solve this equation to find the MLE estimate for p.

MLE: Maximum Likelihood Estimation

Example 3: Back to our Nashville Example



Finding the MLE

We are going to use the Gamma function:

$$f(x;\alpha,\beta) \sim \mathrm{Gamma}(\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$$

To find the maximum of any function, we typically take the ***derivative*** with respect to the variable to be maximized, set it to zero and solve for that variable.

$$\frac{\partial l(\alpha,\beta)}{\partial\beta} = n\left(\frac{\alpha}{\beta} - \bar{x}\right) = 0$$

Which can be solved as $\beta = \alpha/\bar{x}$. However, plugging this into the derivative with respect to α yields:

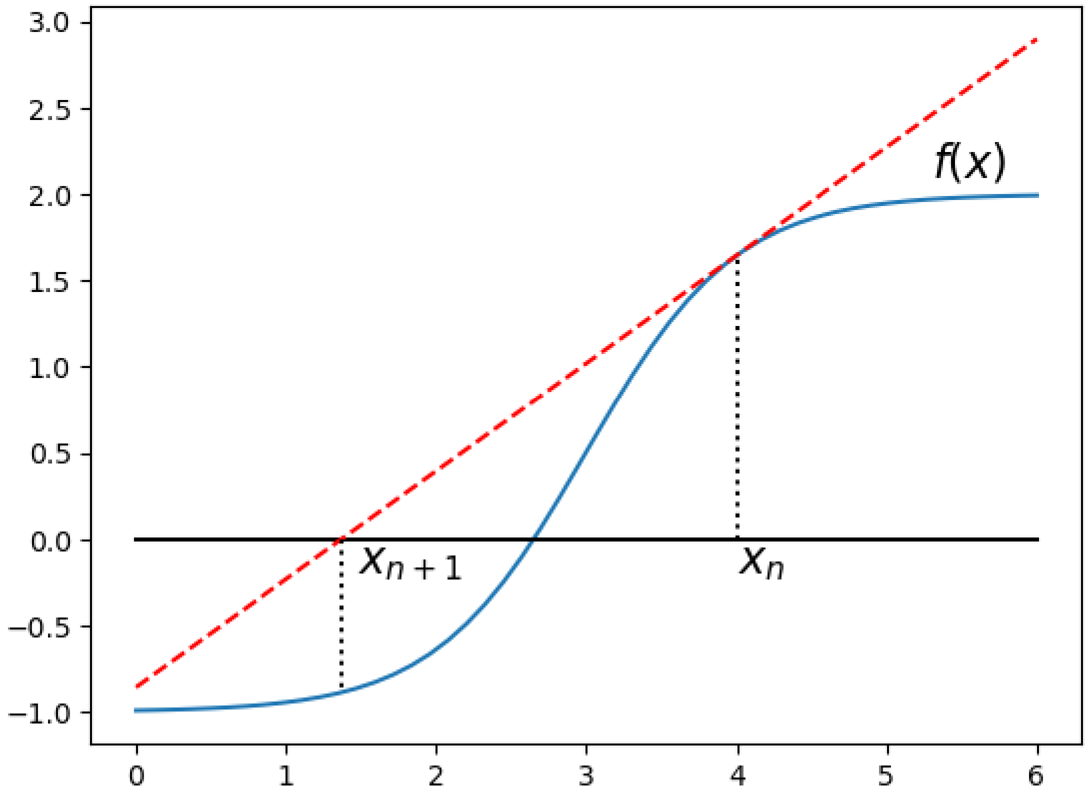
$$\frac{\partial l(\alpha,\beta)}{\partial\alpha} = \log(\alpha) + \overline{\log(x)} - \log(\bar{x}) - \frac{\Gamma(\alpha)'}{\Gamma(\alpha)} = 0$$

This has no closed form solution. We must use ***numerical optimization***!

Numerical optimization alogarithms take an initial "guess" at the solution, and iteratively improve the guess until it gets "close enough" to the answer.

Here, we will use Newton-Raphson algorithm:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



To apply the Newton-Raphson algorithm, we need a function that returns a vector containing the **first and second derivatives** of the function with respect to the variable of interest. In our case, this is:

Slide Type Skip

```
from scipy.special import psi, polygamma

dlgamma = lambda m, log_mean, mean_log: np.log(m) - psi(m) - log_mean + mean_log
dl2gamma = lambda m, *args: 1./m - polygamma(1, m)
```

Slide Type Skip

where `log_mean` and `mean_log` are $\log \bar{x}$ and $\overline{\log(x)}$, respectively. `psi` and `polygamma` are complex functions of the Gamma function that result when you take first and second derivatives of that function.

Slide Type Skip

```
# Calculate statistics
log_mean = precip.mean().apply(np.log)
mean_log = precip.apply(np.log).mean()
```

Slide Type Skip

Time to optimize!

Slide Type Skip

```
# Alpha MLE for December
alpha_mle = newton(dlgamma, 2, dl2gamma, args=(log_mean[-1], mean_log[-1]))
alpha_mle
```

```
3.5189679152399616
```

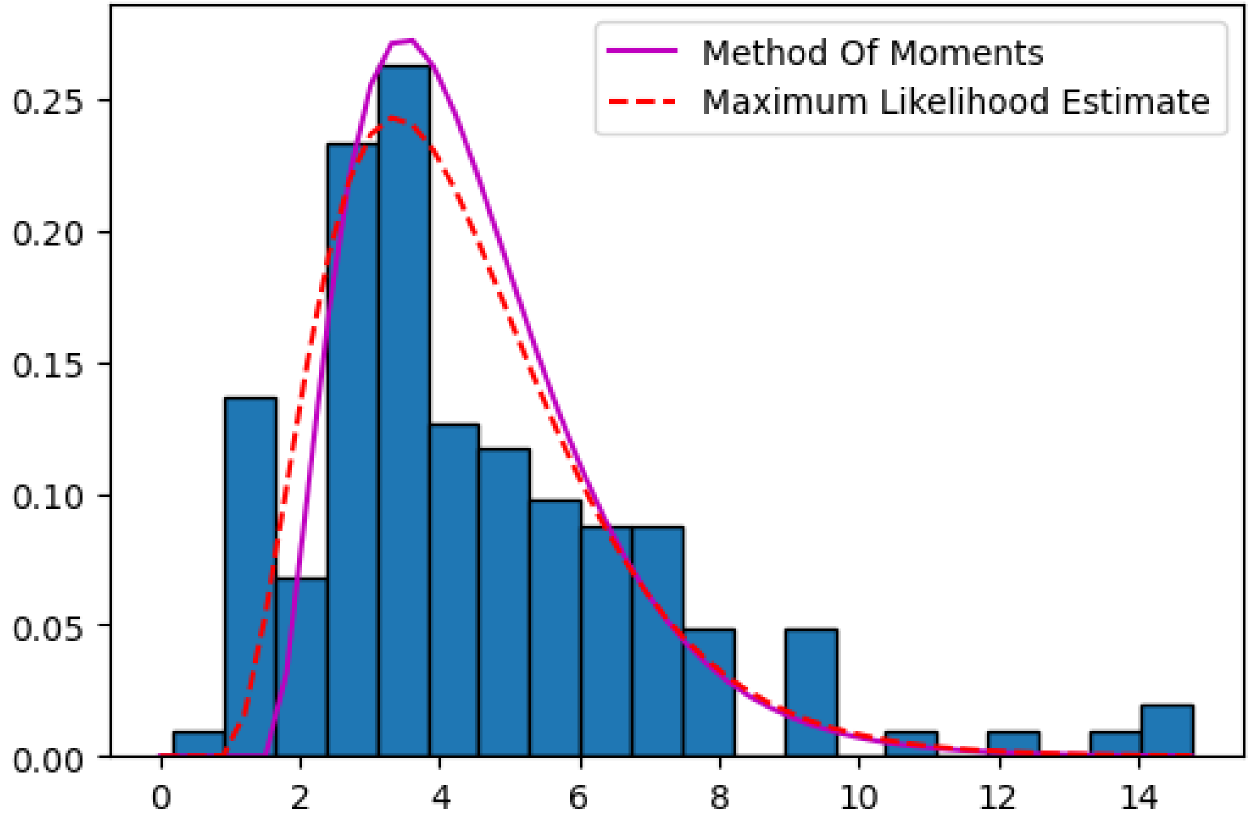
And now plug this back into the solution for beta:

$$\beta = \frac{\alpha}{\bar{X}}$$

```
beta_mle = alpha_mle/precip.mean()[-1]
beta_mle
```

```
0.8426160754841373
```

- We can compare the fit of the estimates derived from MLE to those from the method of moments:



KDE: Kernel Density Estimation

- Kernel Density Estimation (KDE) is a non-parametric technique used for estimating the probability density function (PDF) of a continuous random variable. Unlike parametric methods (such as Method of Moments or Maximum Likelihood Estimation), KDE makes no assumptions about the underlying distribution of the data. Instead, it provides a smooth estimate of the PDF based on the data itself.

KDE: Kernel Density Estimation

1. **Data Smoothing:** KDE involves smoothing the observed data points to estimate the PDF.
2. **Kernel Function:** A kernel function (usually Gaussian) is centered at each data point. The kernel function represents a small "bump" or probability distribution.
3. **Summation:** The contributions of all kernel functions are summed up to create a continuous and smooth PDF estimate.
4. **Bandwidth Parameter:** The bandwidth parameter (h) determines the width of the kernels and affects the smoothing level. It's a crucial parameter in KDE. A small bandwidth results in over-smoothing, while a large bandwidth results in under-smoothing.

KDE: Kernel Density Estimation

KDE Example: Suppose you have a dataset of exam scores and want to estimate the PDF of the scores.

- KDE involves placing a kernel (e.g., Gaussian) at each data point.
- The width of the Gaussian kernel (bandwidth) determines how wide each bump is.
- The PDF estimate is created by summing the contributions of all these Gaussian bumps.

KDE: Kernel Density Estimation

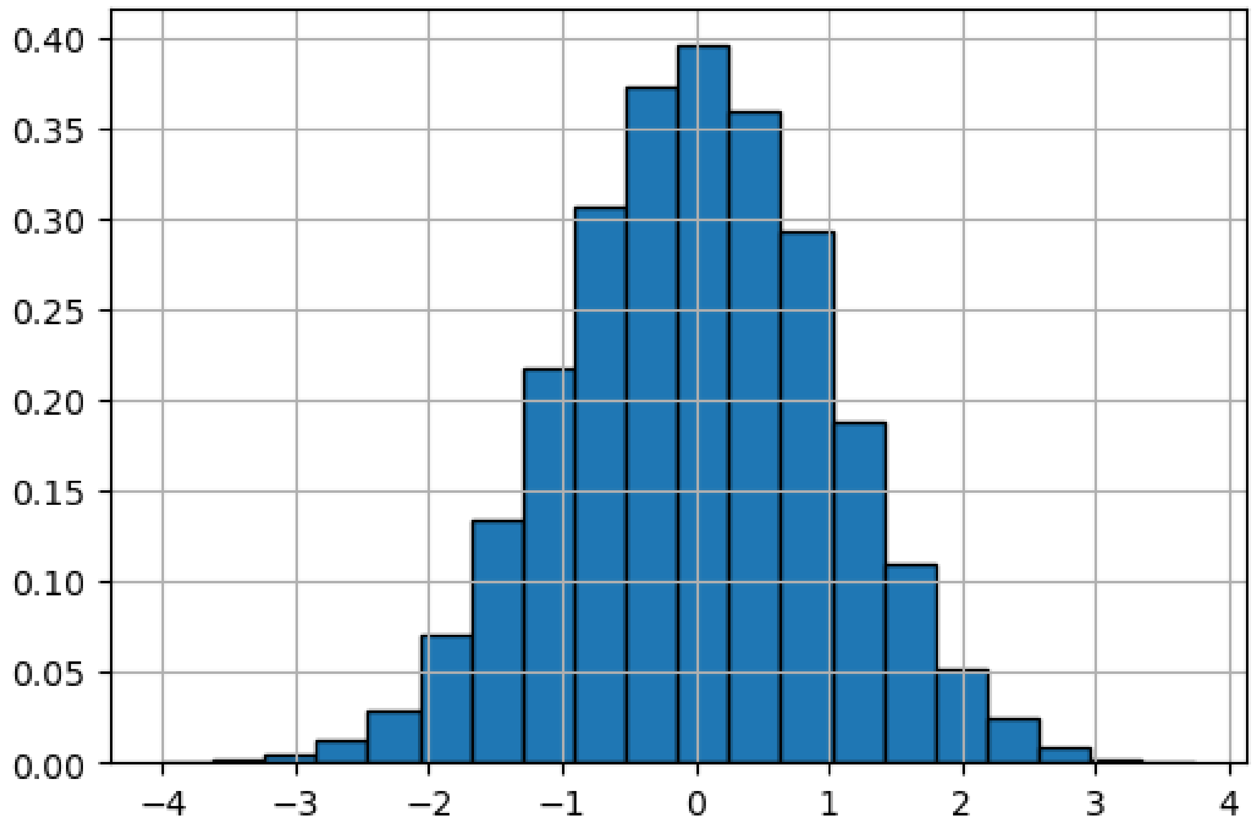
- A ***histogram*** is the simplest non-parametric density estimator and the one that is mostly frequently encountered. Basically used for data smoothing. To construct a histogram,
- we divide the interval covered by the data values and then into equal sub-intervals, known as **bins**.
- Every time, a data value falls into a particular sub-interval, then a block, of size equal 1 by the ***binwidth***, is placed on top of it.

- When we construct a histogram, we need to consider these two main points:

  the **size of the bins (the binwidth)** and

  the **end points of the bins**.

```
n = np.random.randn(100000)
df = pd.Series(np.random.randn(10000))
df.hist(bins=20, density=True, ec='black', figsize=(6,4))
```

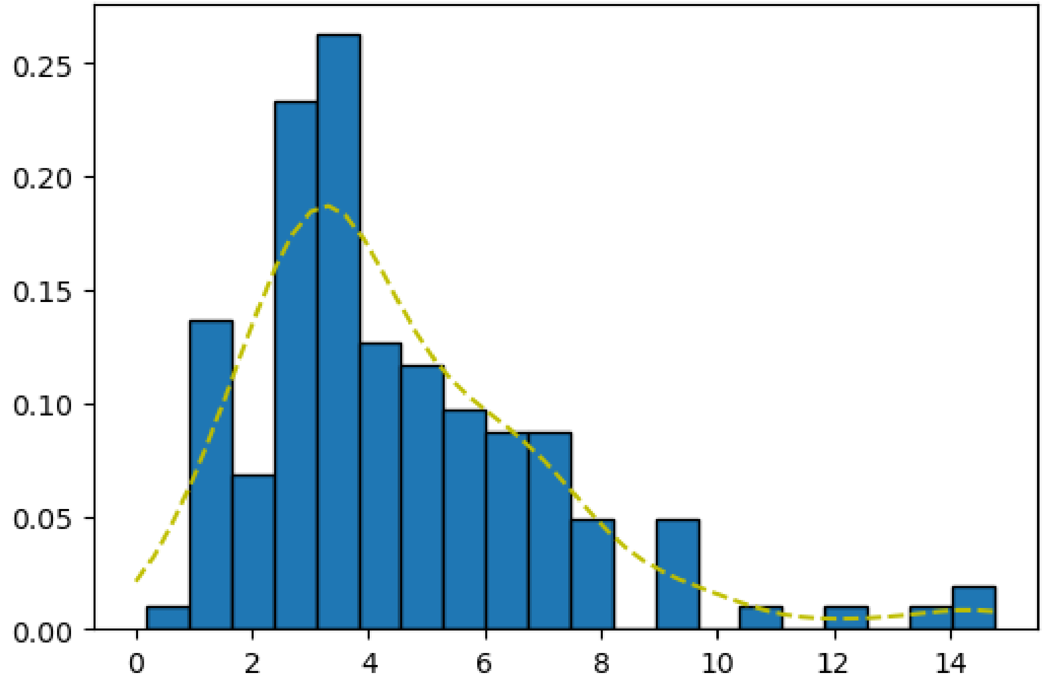


The example has illustrated the properties of histograms:

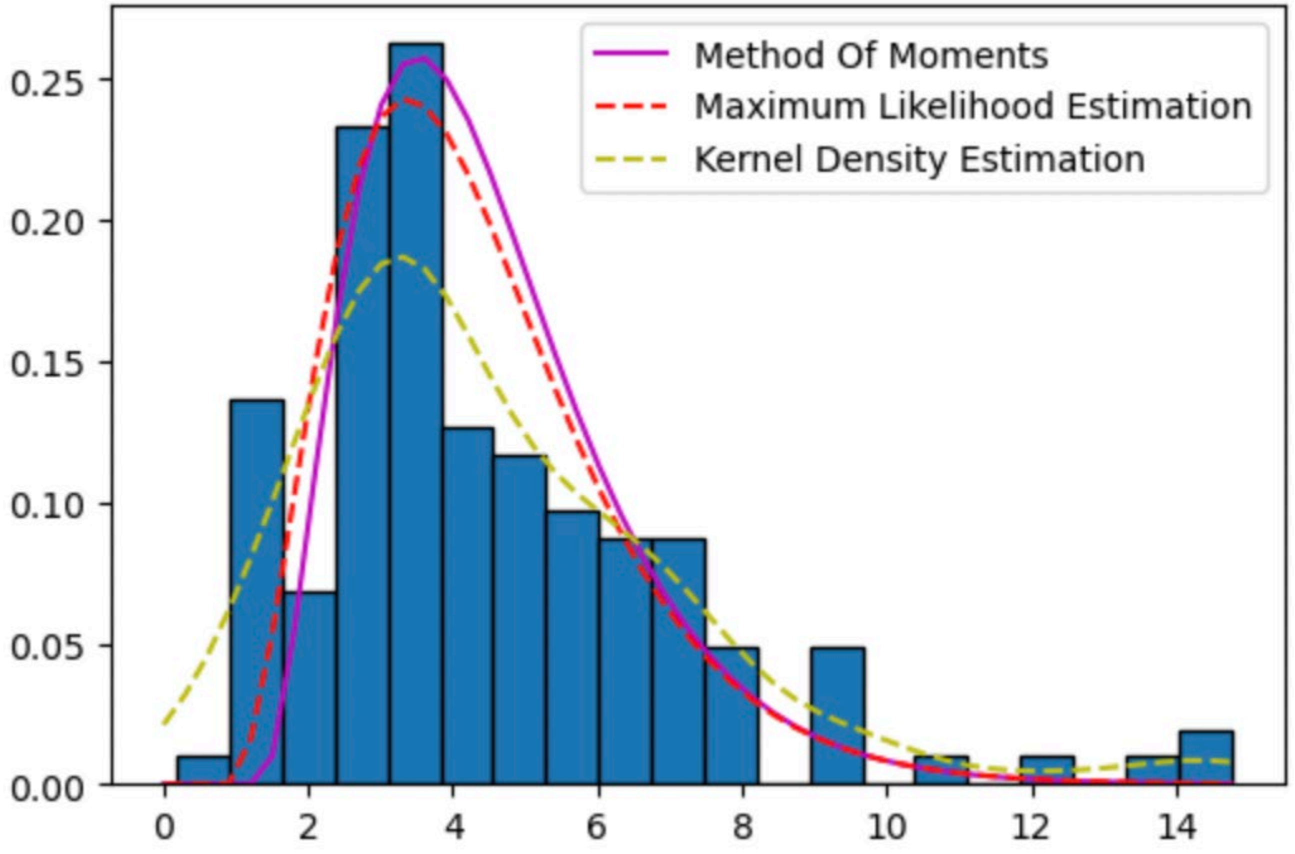
- not smooth
- depend on end points of bins
- depend on width of bins

- Back to Nashville Precipitation Example

```
from scipy.stats import gaussian_kde

jan = precip.Jan
jan.hist(density=True, bins=20, grid=False, ec='black', figsize=(6,4))
x = np.linspace(0, jan.max())
density = gaussian_kde(jan, bw_method=None)
xgrid = np.linspace(x.min(), x.max(), 100)

plt.plot(x, density(x), 'y--')
```



Comparing MoM, MLE, and KDE



- Online resource for introduction of KDE.

https://www.youtube.com/watch?v=x5zLaWT5KPs&ab_channel=webelod

MoM versus MLE versus KDE

- **MoM:**
  - Advantages:
    - Simple
    - easy to use for normal curves.
    - Mean is the best statistic for locating this curve
  - Disadvantages:
    - Not robust/generalizable
    - Depends on sample size
- **MLE:**
  - Advantages:
    - More robust/generalizable - Can be applied to a large set of problems
    - Greater efficiency
  - Disadvantages:
    - Evaluating and maximizing likelihood function is often challenging
    - Difficult to write down complete statistical model of the joint distribution of the data
    - Heavily biased for small samples
- **KDE:**
  - Advantages:
    - works well when there is a lot of data.
    - non-parametric, no fixed structure and depend upon all the data points to reach an estimate
  - Disadvantages:
    - Figuring out bandwidth is a hard problem.

In-Class Exercise:
Comparing Distribution Estimation Methods (MoM, MLE, KDE)

1. Generate a dataset from a normal distribution. (You can also try exponential or poisson distribution)
2. MoM: use sample moments (e.g., sample mean and variance) to estimate distribution parameters.
3. MLE: use the scipy.stats module to fit the distribution using MLE
4. KDE: use seaborn or scipy.stats.gaussian_kde to plot the KED
5. Comparison and Discussion.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm

# Step 1: Generate some sample data from a normal distribution
np.random.seed(42)
sample_data = np.random.normal(loc=10, scale=2, size=1000)

# Step 2: Estimate Parameters Using MoM
mom_mean = np.mean(sample_data)
mom_var = np.var(sample_data, ddof=1)
mom_std = np.sqrt(mom_var)

# Step 3: Estimate Parameters Using MLE
mle_mean, mle_std = norm.fit(sample_data)

# Step 4: Plot the Results
# Plot the KDE
sns.kdeplot(sample_data, label='KDE Estimate', fill=True, color='blue')
```

```
# Step 4: Plot the Results
# Plot the KDE
sns.kdeplot(sample_data, label='KDE Estimate', fill=True, color='blue')

# Plot the MoM estimate
x = np.linspace(min(sample_data), max(sample_data), 1000)
plt.plot(x, norm.pdf(x, mom_mean, mom_std), label='MoM Estimate', color='green')

# Plot the MLE estimate
plt.plot(x, norm.pdf(x, mle_mean, mle_std), label='MLE Estimate', color='red')

# Step 5: Add Labels, Title, and Legend
plt.title('Comparison of MoM, MLE, and KDE')
plt.xlabel('Data Values')
plt.ylabel('Density')
plt.legend()

# Show the plot
plt.show()
```

Explore for Poisson Distribution

- MoM and MLE will both estimate the rate parameter λ (which is the mean of the distribution).
- KDE is less appropriate for discrete distributions, but we can visualize a smoothed KDE-like estimate.

```
from scipy.stats import poisson

# Step 1: Generate some sample data from a Poisson distribution
np.random.seed(42)
sample_data = np.random.poisson(lam=5, size=1000)  # Poisson distribution with lambda=5

# Step 2: Estimate Lambda using MoM
mom_lambda = np.mean(sample_data)

# Step 3: Estimate Lambda Using MLE. For a Poisson distribution, the MLE for lambda is also the sample mean
mle_lambda = np.mean(sample_data)

# Step 4: Plot the KDE: for visualization, we can use a KDE-like estimate
sns.histplot(sample_data, stat="density", bins=30, kde=False, label='Data Histogram', color='blue')

# Plot the MoM
x = np.arange(0, max(sample_data))
plt.plot(x, poisson.pmf(x, mom_lambda), label=f'MoM Estimate (λ={mom_lambda:.2f})', color='green')

# Plot the MLE
plt.plot(x, poisson.pmf(x, mle_lambda), label=f'MLE Estimate (λ={mle_lambda:.2f})', color='red')

# Step 5: Add Labels, Title, and Legend, and show the plot
plt.title('Comparison of MoM and MLE for Poisson Distribution')
plt.xlabel('Data Values')
plt.ylabel('Density')
plt.legend()
plt.show()
```