Question 1: Chapter 11 #2

Let $X_1, \dots, X_n \sim \text{Normal}(\mu, 1)$.

(a) Simulate a data set (using $\mu = 5$) consisting of $n = 100$ observations.

(b) Take $f(\mu) = 1$ and find the posterior density. Plot the density.

(c) Simulate 1,000 draws from the posterior. Plot a histogram of the simulated values and compare the histogram to the answer in (b).

(a)

Simulation. The data generated is at the end of the file.

(b)

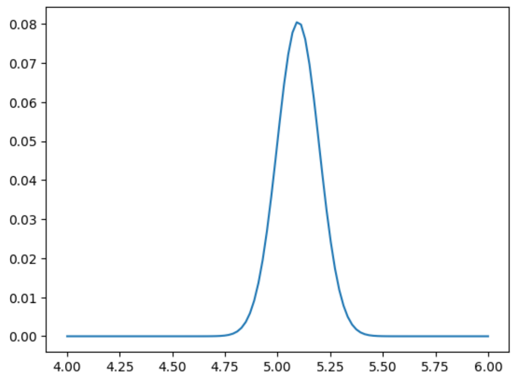


Figure 1: Posterior Density

(c)

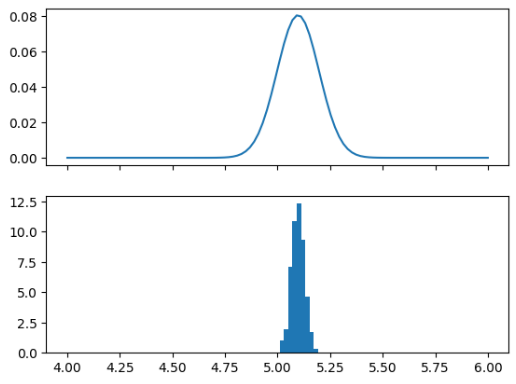


Figure 2: Comparing sampling from posterior distribution to the density

Question 2: Chapter 11 #3

Let $X_1, \dots, X_n \sim \text{Uniform}(0, \theta)$. Let $f(\theta) \propto \frac{1}{\theta}$. Find the posterior density.

Proof. The posterior density is proportional to the likelihood times the prior like follows:

$$f(\theta) \propto \frac{1}{\theta} \tag{1}$$

$$= \frac{1}{\theta} \prod_{i=1}^{n} f(X_i; \theta) \tag{2}$$

$$= \frac{1}{\theta} \prod_{i=1}^{n} \frac{\mathbb{1}(X_i \leq \theta)}{\theta} \tag{3}$$

Where $\mathbb{1}(X_i \leq \theta)$ is the identity function:

$$\mathbb{1}(X_i \leq \theta) = \begin{cases} 1 & \text{where } x_i \leq \theta \\ 0 & \text{else} \end{cases}$$

Continuing with this, we can notice that the product over all of these X_i's is only not 0 when all of $\{X_i\}_{i=1}^n$ are less than θ:

$$\prod_{i=1}^{n} \frac{\mathbb{1}(X_i \leq \theta)}{\theta} = \frac{\mathbb{1}(\max(X_1, \dots, X_n) \leq \theta)}{\theta^n}$$

Continuing from (3), we get that:

$$\frac{1}{\theta}\prod_{i=1}^{n}\frac{\mathbb{1}(X_i \leq \theta)}{\theta} = \frac{1}{\theta}\cdot\frac{\mathbb{1}(\max(X_1,\ldots,X_n)\leq\theta)}{\theta^n} \tag{4}$$

$$= \frac{\mathbb{1}(\max(X_1,\ldots,X_n)\leq\theta)}{\theta^{n+1}} \tag{5}$$

$$= \begin{cases}\frac{1}{\theta^{n+1}} & \text{where } \max(X_1,\ldots,X_n)\leq\theta \\ 0 & \text{else}\end{cases} \tag{6}$$

(6) follows exactly from the definition of the identity function. From here, we have to integrate the function to normalize the posterior density so that it equals 1:

$$1 = \int_{-\infty}^{\infty} f(\theta)d\theta \tag{7}$$

We know that the posterior density: $f(\theta)$ is proportional as:

$$f(\theta) = c\cdot\begin{cases}\frac{1}{\theta^{n+1}} & \text{where } \max(X_1,\ldots,X_n)\leq\theta \\ 0 & \text{else}\end{cases}$$

When we integrate this, we are integrating over θ. We notice that this is only non zero where $\max(X_1,\ldots,X_n)\leq\theta$, so we want to integrate for θ values that are greater than this max.

$$1 = \int_{-\infty}^{\infty} f(\theta)d\theta = \int_{max(X_1,\ldots,X_n)}^{\infty} c\cdot\frac{1}{\theta^{n+1}}d\theta \tag{8}$$

$$1 = \int_{max(X_1,\ldots,X_n)}^{\infty} c\cdot\frac{1}{\theta^{n+1}}d\theta \tag{9}$$

$$1 = c\int_{max(X_1,\ldots,X_n)}^{\infty}\frac{1}{\theta^{n+1}}d\theta \tag{10}$$

$$1 = c\left[\frac{1}{\theta^n}/-n\right]_{\max(X_1,\ldots,X_n)}^{\infty} \tag{11}$$

$$1 = c\left[0+\max(X_1,\ldots,X_n)^{-n}/n\right] \tag{12}$$

$$1 = \frac{c\max(X_1,\ldots,X_n)^{-n}}{n} \tag{13}$$

We know can solve this for c and we get

$$c = n\max(X_1, \ldots, X_n)^n$$

We can use this value for c to derive the posterior density:

$$f(\theta) = n\max(X_1, \ldots, X_n)^n \cdot \frac{1}{\theta^{n+1}}$$

□

1 Question 3: Chapter 20 #4

Proposition: The risk can be written as

$$R(g, g_n) = \int b^2(x)\,dx + \int v(x)\,dx \tag{20.3}$$

where

$$b(x) = \mathbb{E}[g_n(x)] - g(x) \tag{20.4}$$

is the bias of $g_n(x)$ at a fixed x, and

$$v(x) = \mathrm{Var}(g_n(x)) = \mathbb{E}\left[(g_n(x) - \mathbb{E}[g_n(x)])^2\right] \tag{20.5}$$

is the variance of $g_n(x)$ at a fixed x.

Proof. We can start by defining a few terms. The risk is defined as

$$R(g, \hat{g}) = \mathbb{E}(L(g, \hat{g}))$$

where L, the loss function, is the integrated squared error defined as

$$L(g, \hat{g}_n) = \int (g(u) - \hat{g}_n(u))^2 du.$$

We want to show that (20.3) is equivalent to $\mathbb{E}(L(g, \hat{g}))$ in order to prove the lemma. We will start with (20.3) and show that we can manipulate the terms to reach our desired conclusion:

$$\int b^2(x)dx + \int v(x)dx = \int (b^2(x) + v(x))dx \tag{14}$$

$$= \int (\mathbb{E}[g_n(x) - g(x)]^2 + v(x))dx \qquad \text{from (20.4)} \tag{15}$$

$$= \int (\mathbb{E}[g_n(x) - g(x)]^2 + \mathbb{E}\left[(g_n(x) - \mathbb{E}[g_n(x)])^2\right])dx \qquad \text{from (20.5)} \tag{16}$$

We know that the second moment of the random variable $(g_n(x) - g(x))$ can be expressed as:

$$\mathbb{E}\left[(g_n(x) - g(x))^2\right] = (\mathbb{E}\left[g_n(x) - g(x)\right])^2 + \mathbb{E}\left[(g_n(x) - \mathbb{E}[g_n(x)])^2\right]$$

Thus, we can say that

$$\mathbb{E}\left[(g_n(x) - \mathbb{E}[g_n(x)])^2\right] = \mathbb{E}\left[(g_n(x) - g(x))^2\right] - \mathbb{E}\left[g_n(x) - g(x)\right]^2$$

Continuing from (16) and substituting our result in

$$= \int \mathbb{E}[g_n(x) - g(x)]^2 + \mathbb{E}\left[(g_n(x) - g(x))^2\right] - \mathbb{E}\left[g_n(x) - g(x)\right]^2 dx \tag{17}$$

$$= \int \mathbb{E}\left[(g_n(x) - g(x))^2\right] dx \tag{18}$$

$$= \mathbb{E}\left[\int (g_n(x) - g(x))^2 \, dx\right] \qquad \because \text{ expectation is linear} \tag{19}$$

$$= \mathbb{E}\left[L(g, \hat{g}_n)\right] \qquad \text{by def of } L(g, \hat{g}_n) \tag{20}$$

We have shown that the risk, $\int b^2(x)dx + \int v(x)dx$, is equivalent to $L(g, \hat{g}_n)$. We have proved that the lemma is correct. $\square$

homework10

December 5, 2024

[19]:
```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc > /content/
  ↪drive/MyDrive/installs.txt
!pip install pypandoc > /content/drive/MyDrive/pandoc_installs.txt
```

```
Mounted at /content/drive
```

1 Supplemental Question

1.1 Plotting Functions

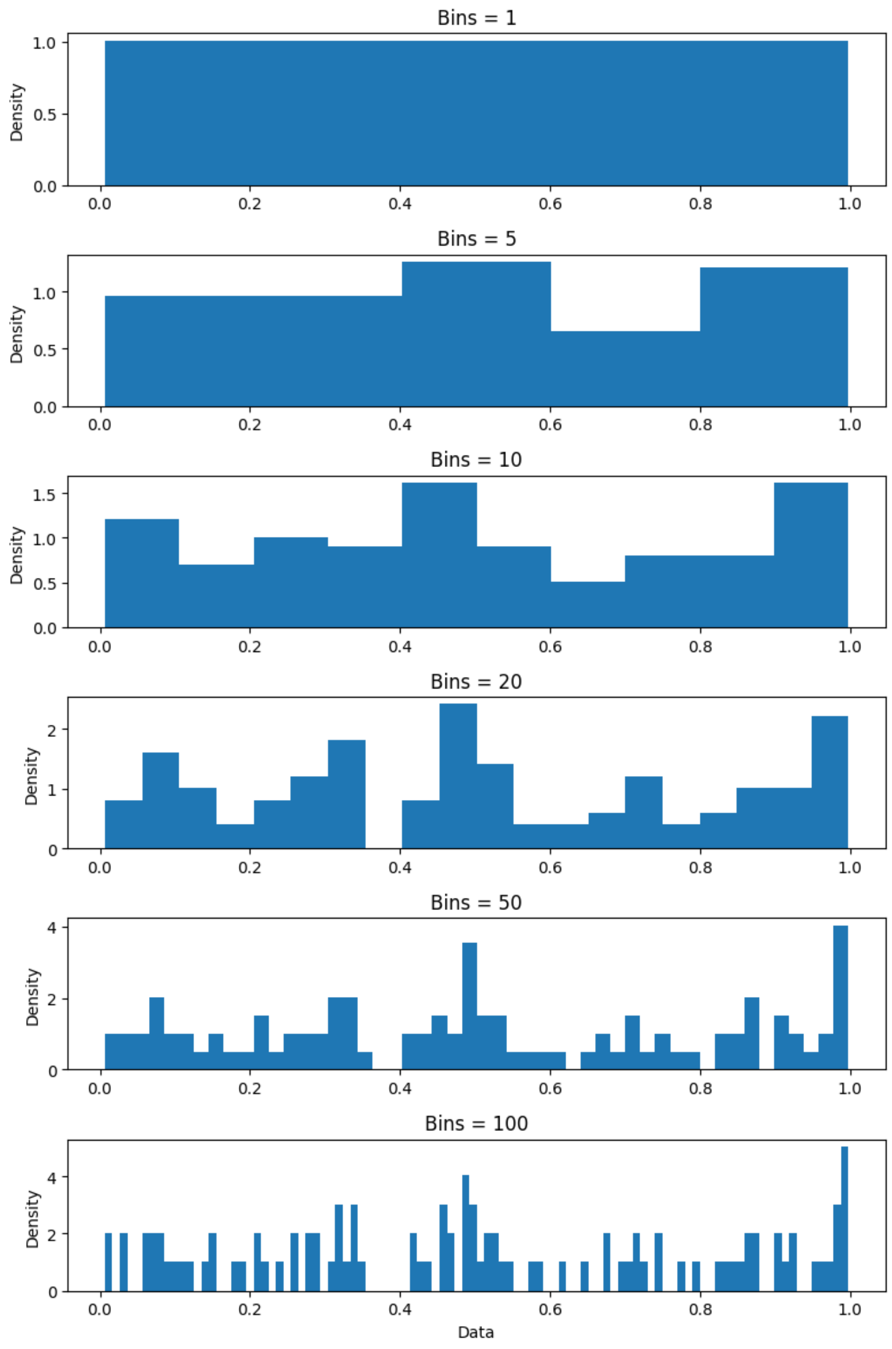
These are the functions that we will use to plot the histogram estimator and the KDE. Both take in the data and either bin_sizes or bandwidths, to plot the estimators given the data at different bin sizes or bandwidths.

[5]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import uniform
from scipy.stats import gaussian_kde

def plot_histogram(data, bin_sizes):
    fig, axes = plt.subplots(len(bin_sizes), 1, figsize=(8, 12))
    for i, bins in enumerate(bin_sizes):
        axes[i].hist(data, density=True, bins=bins)
        axes[i].set_title(f"Bins = {bins}")
        axes[i].set_ylabel("Density")

    axes[-1].set_xlabel("Data")
    plt.tight_layout()
    plt.show()

def plot_kde(data, bandwidths):
    fig, axes = plt.subplots(len(bandwidths), 1, figsize=(8, 12))
    for i, bandwidth in enumerate(bandwidths):
        kde = gaussian_kde(data, bw_method=bandwidth)
        x_eval = np.linspace(data.min() - 1, data.max() + 1, 200)
        density = kde(x_eval)
        axes[i].plot(x_eval, density)
        axes[i].set_title(f"Bandwidth = {bandwidth:.2f}")
        axes[i].set_ylabel("Density")
    axes[-1].set_xlabel("Data")
    plt.tight_layout()
    plt.show()
```

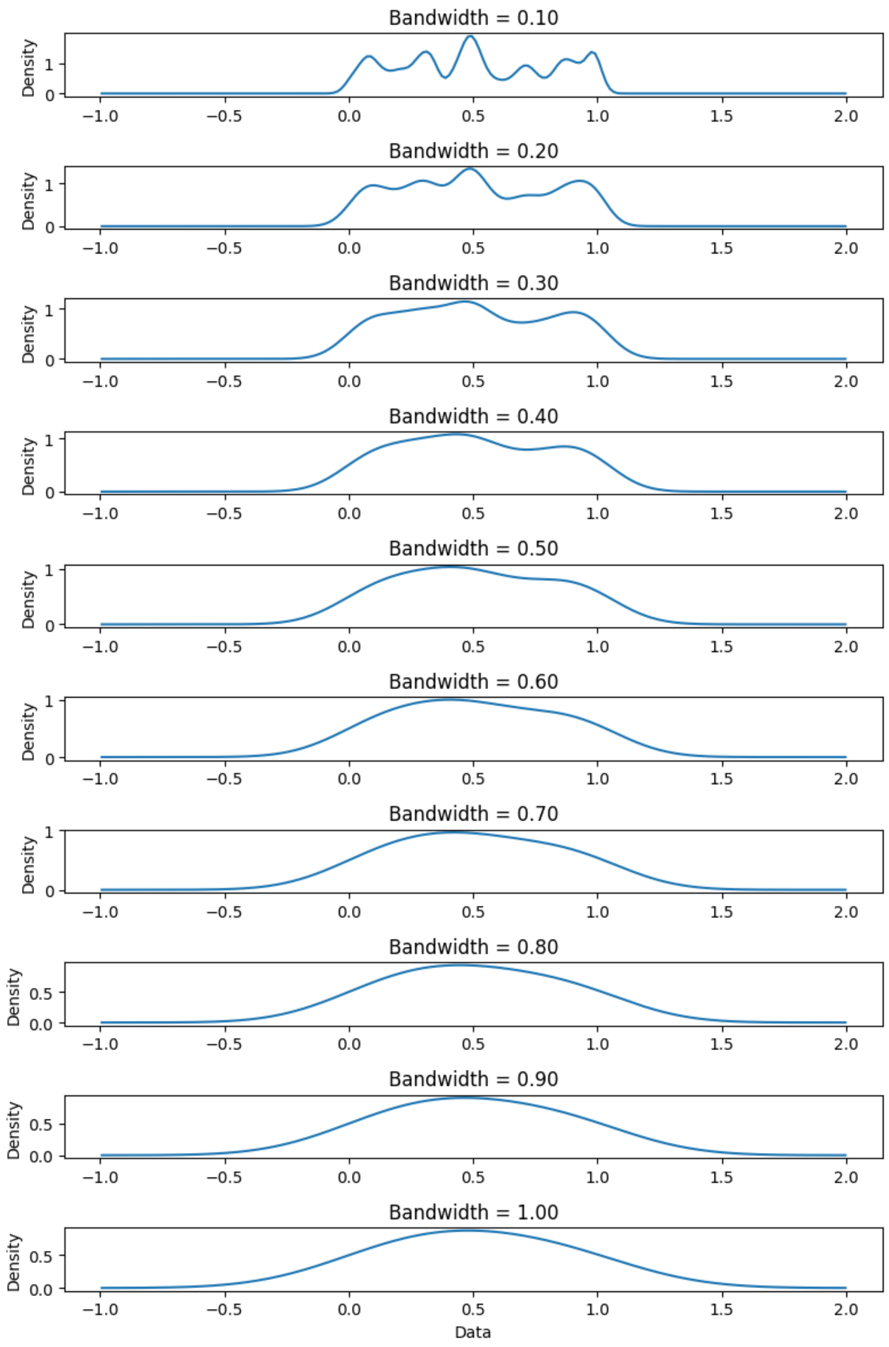
1.2 Part A

[6]:
```
x_a = np.random.uniform(low=0, high=1, size=(1, 100))
```

[7]:
```
bin_sizes = [1, 5, 10, 20, 50, 100]

plot_histogram(x_a[0], bin_sizes)
```


Bins = 1
Bins = 5
Bins = 10
Bins = 20
Bins = 50
Bins = 100
Density
Data
0.0
0.2
0.4
0.6
0.8
1.0

[8]:
```
bandwidths = np.linspace(0.1, 1.0, 10)

plot_kde(x_a[0], bandwidths)
```


Bandwidth = 0.10
Bandwidth = 0.20
Bandwidth = 0.30
Bandwidth = 0.40
Bandwidth = 0.50
Bandwidth = 0.60
Bandwidth = 0.70
Bandwidth = 0.80
Bandwidth = 0.90
Bandwidth = 1.00
Density
Data
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

1.2.1 Histogram Behavior

With our basis of the theory behind histogram estimators, we know that the risk is miminized for some intermediate value of bin sizes. We notice that this is the case in our experiment as well.

We know that the random values that we are sampling are from a unifrom distribution on the interval [0,1], so we want the density function to look simmilar to the density function for a unifrom distribution i.e. flat over the region [0,1].

In terms of the number of bins that creates a density function that looks most similar to the uniform density function, trivially 1 bin looks the best. With 1 bin all values are equally likely to occur becuase there is just one bin which has probability n / n where n is the number of samples. This is exactly the definition of the uniform density function on the interval [0,1].

As we increase the number of bins, we increase the noise that we pick up and the variation increases. We can see this experimentally as there are spikes in the graph for larger numbers of bins that only increase as we increase the number of bins.

1.2.2 KDE Behavior

With our theory of KDEs, we again know that some intermediate value for bandwidth offers the best fit guess for the underlying distribution.

For lower values of bandwidth, we notice that the graph contains more spikes and is less smooth. The lower bandwidth values give more weight to the variations of the random variable. This observation is in line with the theory which says that the varience of the kernel estimator decreases by a factor of $\frac{1}{h}$ where h is the bandwidth. As $h \to 0$, the varience blows up. We see this in the graph as the graph gets much more jagged as the h decreases.

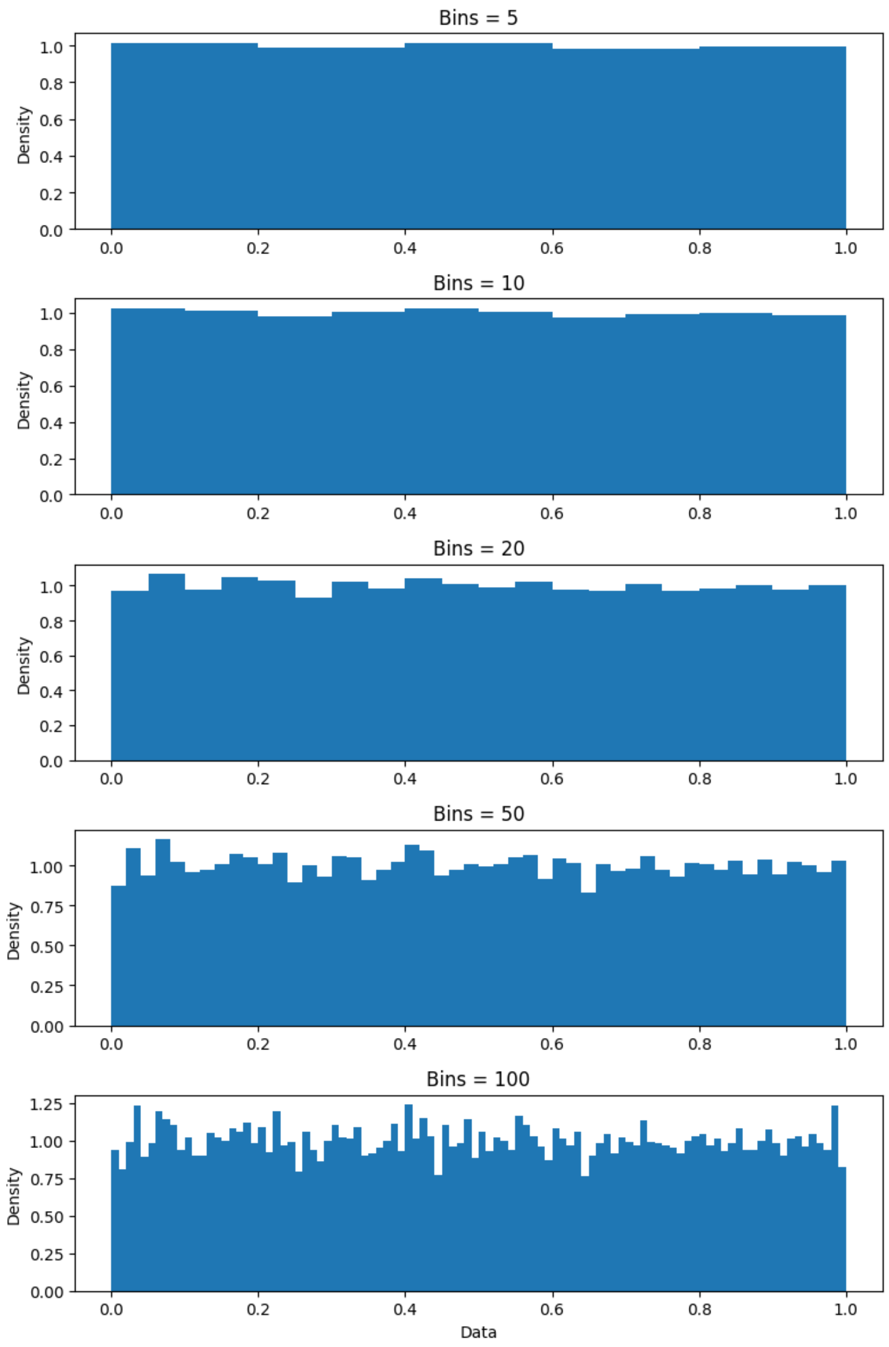
For higher values of bandwidth, we notice htat the graph looks more and more like a Gaussian, the underlying kernel that we use. This also makes sense given the underlying theory. Theory says that the bias of the estimator increases by a factor of h^2. So large h yields more biased estimators.

We notice this in the experiment as larger values of h look more Gaussian than uniform. A bandwidth value somewhere between 0.2 and 0.3 looks most like a unifrom distribition, while smaller bandwidths make the density look more complex and higher bandwidths make the graph smoother and more like a Gaussian.

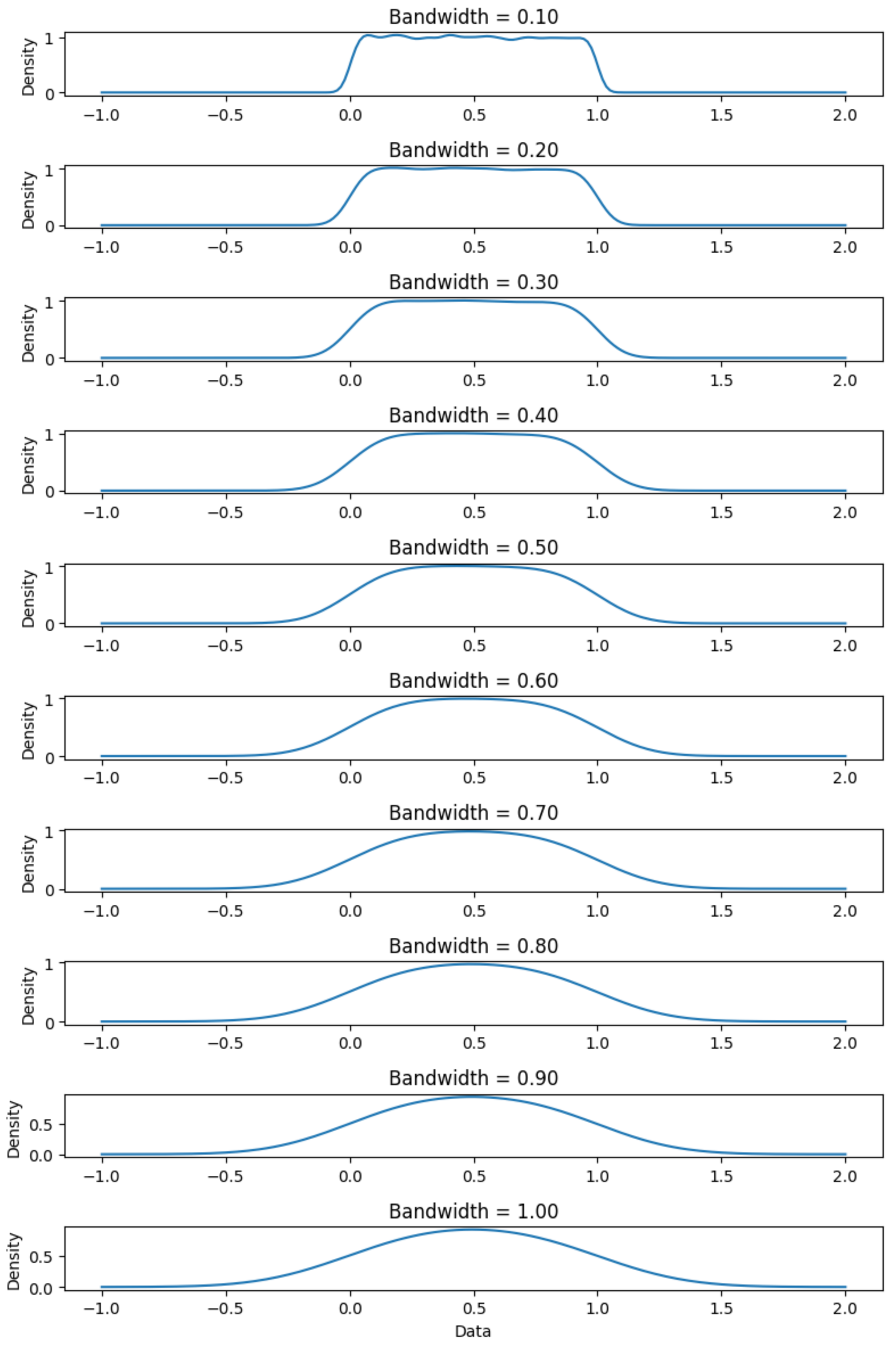
1.3 Part B

[9]:
```
x_b = np.random.uniform(low=0, high=1, size=(1, 10000))
```

[10]:
```
bin_sizes = [5, 10, 20, 50, 100]

plot_histogram(x_b[0], bin_sizes)
```


Bins = 5
Bins = 10
Bins = 20
Bins = 50
Bins = 100
Density
Data

[11]:
```
bandwidths = np.linspace(0.1, 1.0, 10)

plot_kde(x_b[0], bandwidths)
```


Bandwidth = 0.10
Bandwidth = 0.20
Bandwidth = 0.30
Bandwidth = 0.40
Bandwidth = 0.50
Bandwidth = 0.60
Bandwidth = 0.70
Bandwidth = 0.80
Bandwidth = 0.90
Bandwidth = 1.00
Density
Data
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

1.3.1 Behavior

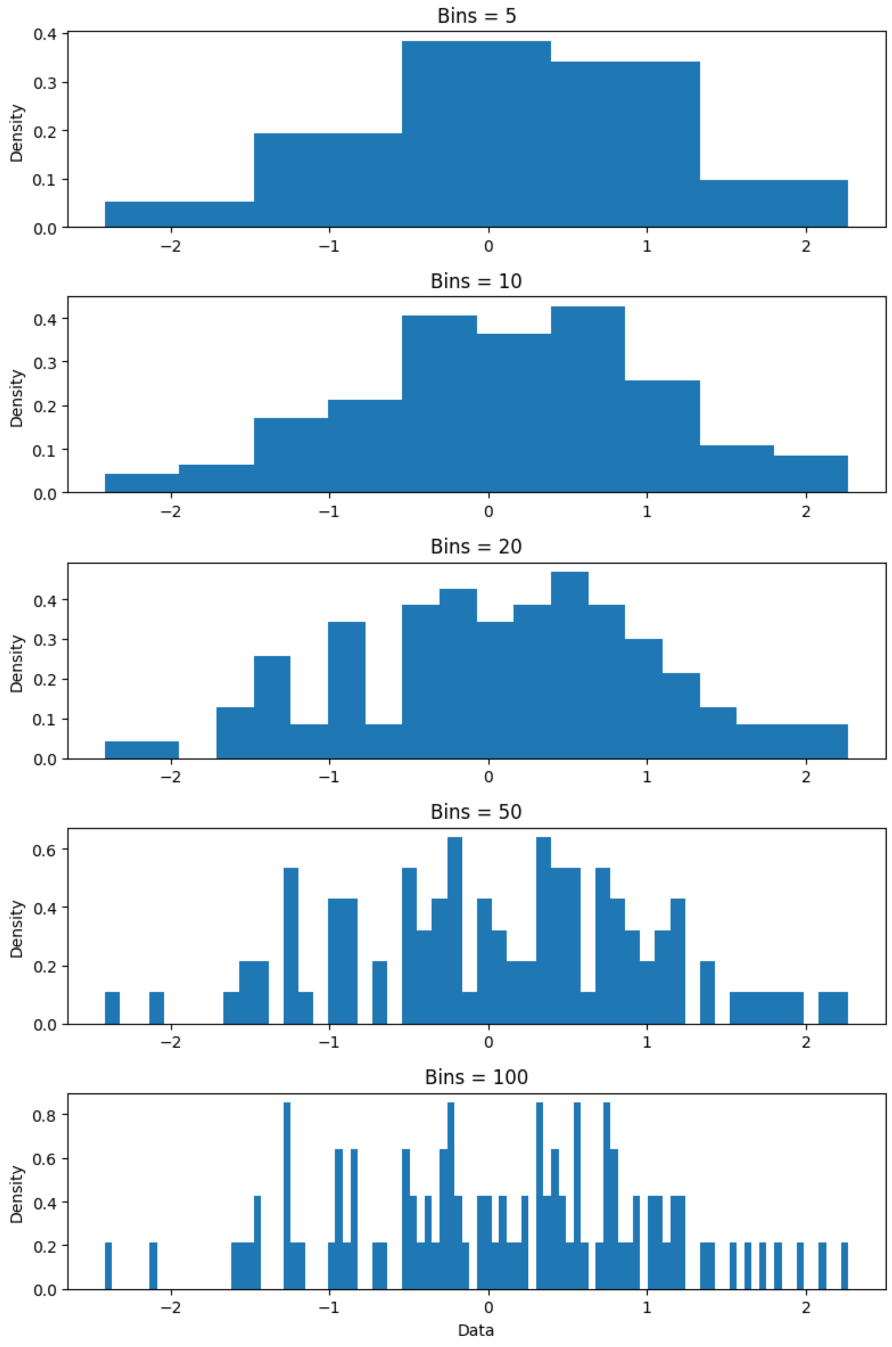
We notice similar trends as in Part A. Smaller number of bins for the histogram decreases the variation and for this case 1 bin is optimal as it exactly resembles a uniform distribution. For the KDE, some intermediate bandwidth value makes the density most resemble a uniform density.

In this experiment, though, we notice an overall improvement of the histogram and the KDE estimators; both look more like unifrom distributions at the optimal bandwidth, bin numbers. This is because the KDE and histogram estimator are consistent i.e. they approach the underlying distribution for the random variable with more samples observed.

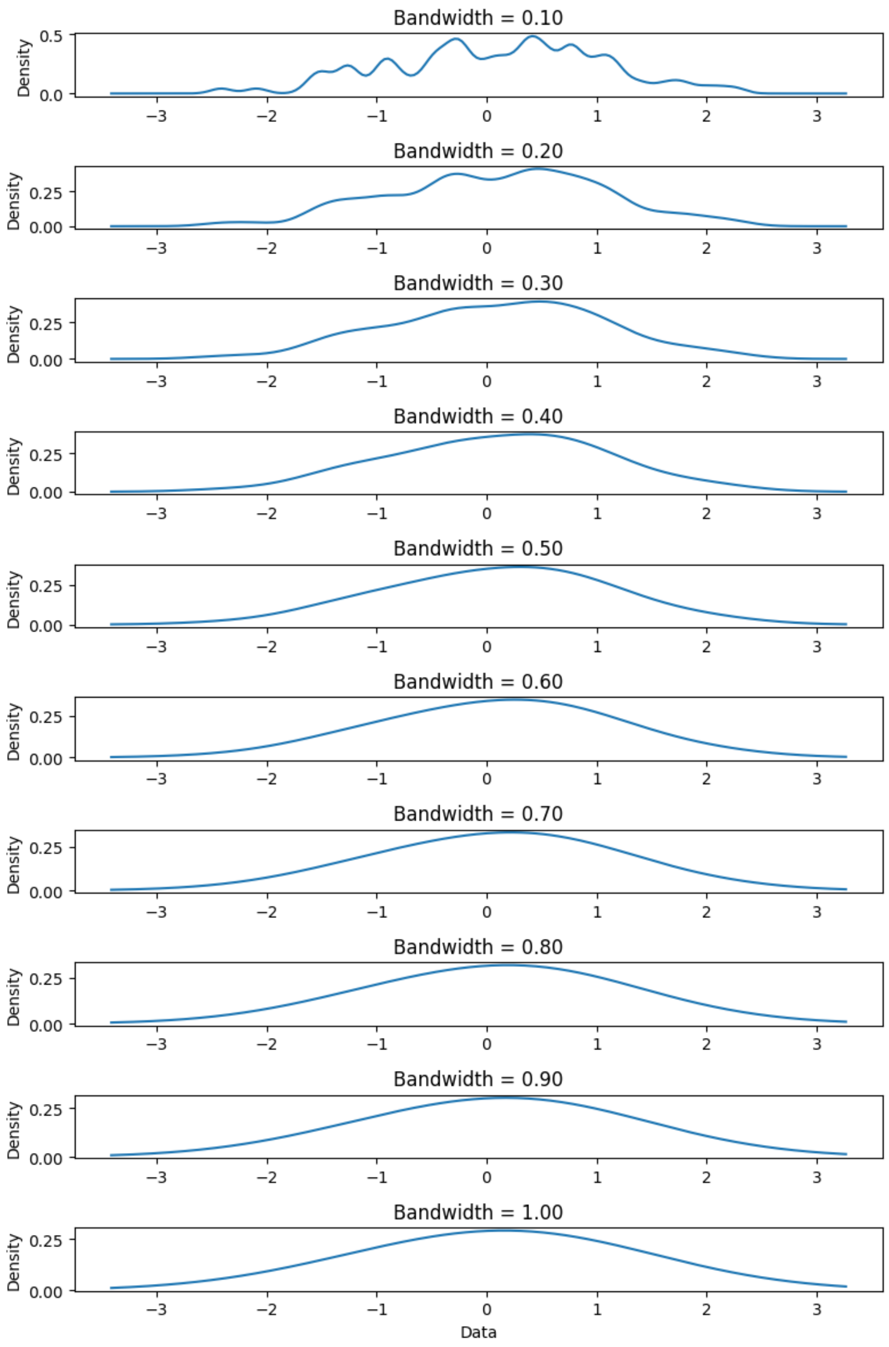
1.4 Part C

[12]:
```
x_c = np.random.normal(loc=0, scale=1, size=(1, 100))
```

[13]:
```
bin_sizes = [5, 10, 20, 50, 100]

plot_histogram(x_c[0], bin_sizes)
```


Bins = 5
Bins = 10
Bins = 20
Bins = 50
Bins = 100
Density
Data
−2
−1
0
1
2

[14]:
```
bandwidths = np.linspace(0.1, 1.0, 10)

plot_kde(x_c[0], bandwidths)
```


Bandwidth = 0.10
Bandwidth = 0.20
Bandwidth = 0.30
Bandwidth = 0.40
Bandwidth = 0.50
Bandwidth = 0.60
Bandwidth = 0.70
Bandwidth = 0.80
Bandwidth = 0.90
Bandwidth = 1.00
Density
0.0
0.5
0.00
0.25
−3
−2
−1
0
1
2
3
Data

1.4.1 Behavior

For Part C and Part D, we know that the data comes from a normal distribution with mean 0 and $\sigma = 1$

1.4.2 Histogram

From theory, we know that an optimal binwidth is not too large and not too small; as the bin width decreases the number of bins increases which increases the variation of the estimator as we pick up more noise of the random variable and as the bin width increases, the estimator becomes more biased as it can't capture the nuances of the underlying distibution and becomes blocky.

We can notice this trend in our experiment as well. About 10-20 bins makes the estimated density look most like a Gaussian with mean 0 and standard deviation 1. Smaller number of bins make the shape more blocky and look less like that smooth Gaussian. Larger number of bins makes the small variations in the random samples jump out and look less Gaussian as well.

1.4.3 KDE

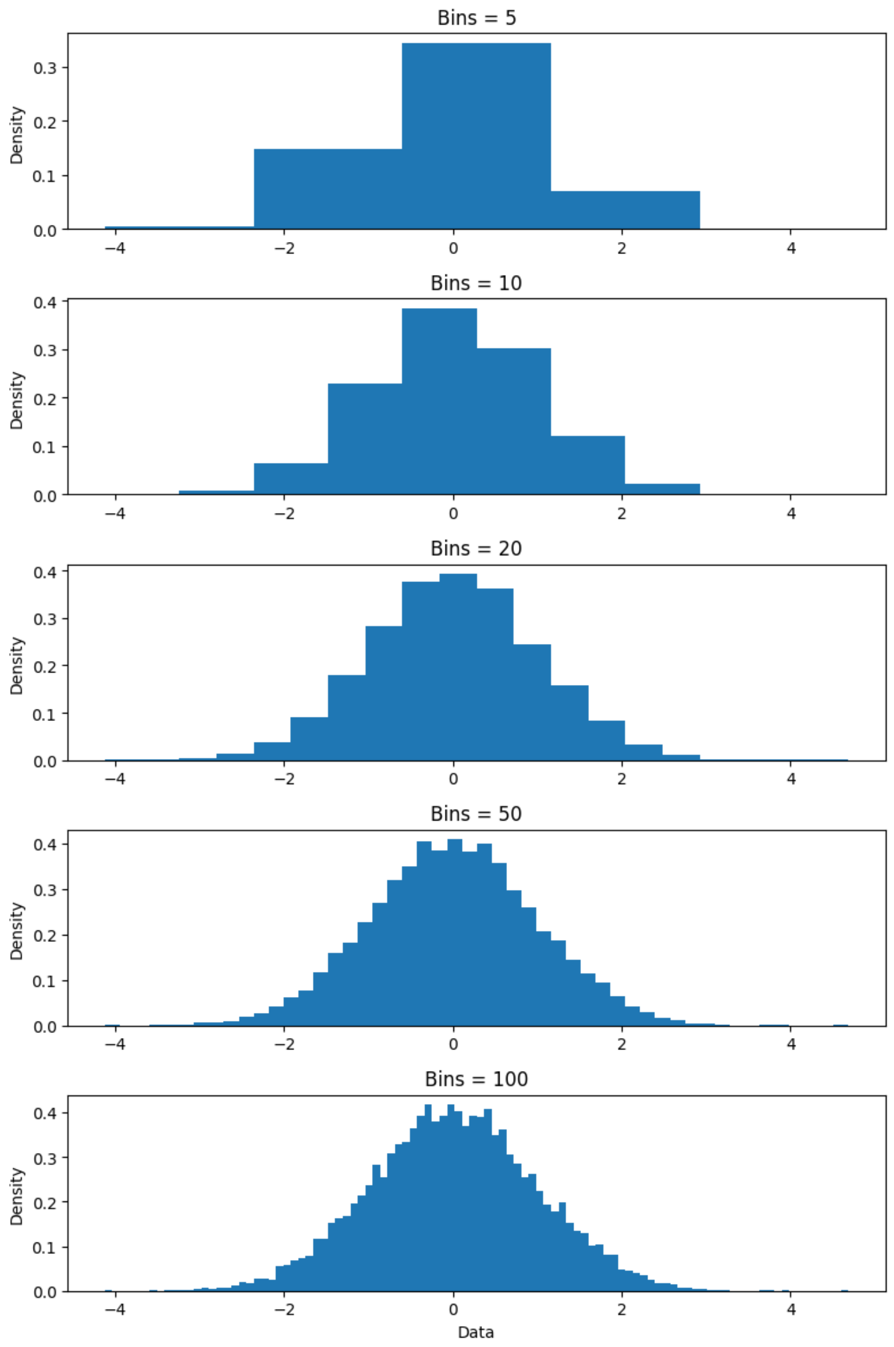
From theory, we know that an optimal bandwidth is not too large or too small. As the bandwidth increases, we smooth out the variations of the variable and make the estimated density function look like the underlying Guassian kernel.

For this, large bandwidths look the most convincing only because the samples are actually drawn from a gaussian that is identical to the kernal. Ideally, the optimal bandwidth should fall somewhere between the extremes.

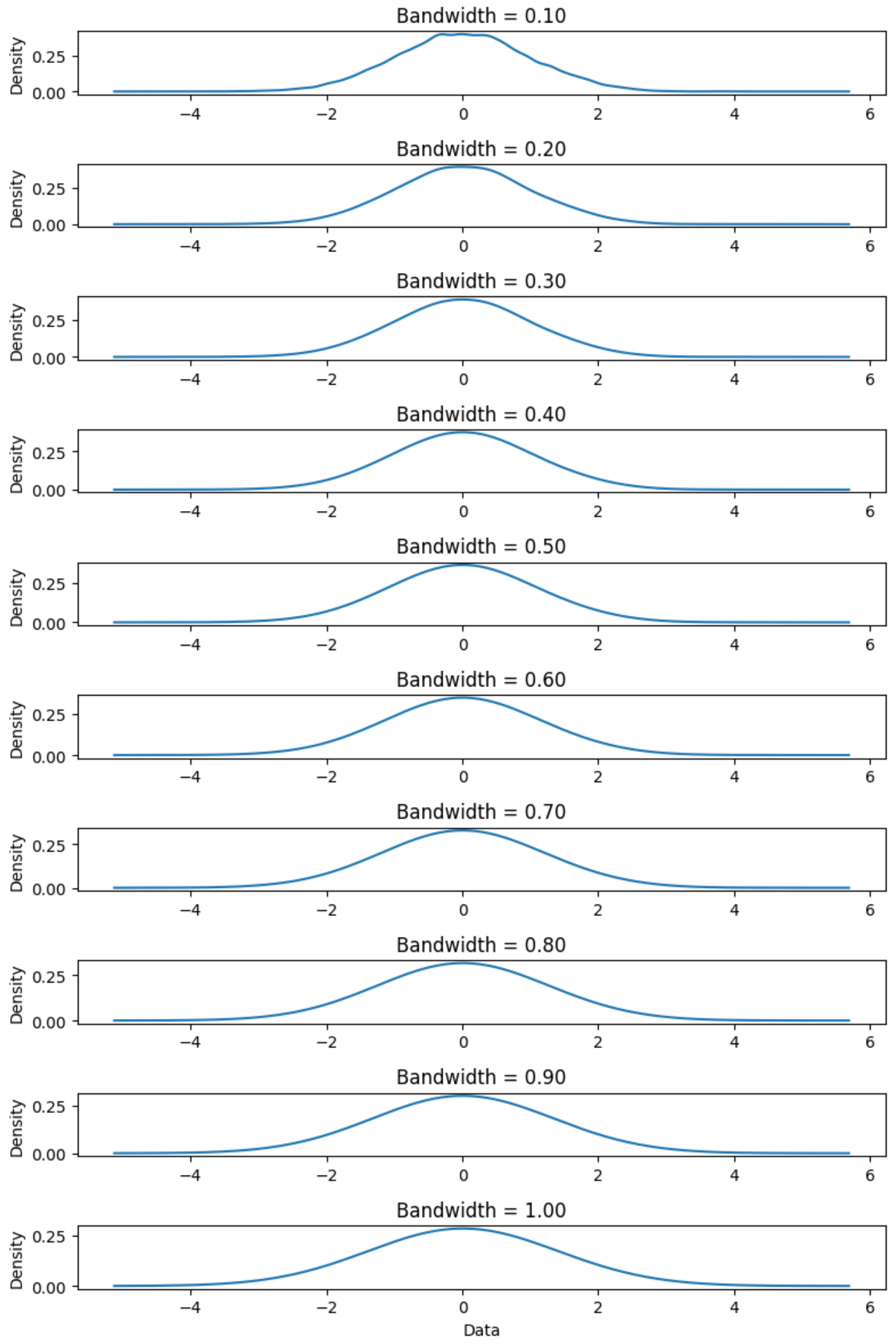
1.5 Part D

[15]:
```
x_d = np.random.normal(loc=0, scale=1, size=(1, 10000))
```

[16]:
```
bin_sizes = [5, 10, 20, 50, 100]

plot_histogram(x_d[0], bin_sizes)
```


Bins = 5
Bins = 10
Bins = 20
Bins = 50
Bins = 100
Density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
Data

```
[17]: bandwidths = np.linspace(0.1, 1.0, 10)

      plot_kde(x_d[0], bandwidths)
```


Bandwidth = 0.10
Bandwidth = 0.20
Bandwidth = 0.30
Bandwidth = 0.40
Bandwidth = 0.50
Bandwidth = 0.60
Bandwidth = 0.70
Bandwidth = 0.80
Bandwidth = 0.90
Bandwidth = 1.00
Density
0.25
0.00
−4
−2
0
2
4
6
Data

1.5.1 Behavior

The behavior is simmilar to that of Part C, but in this case, with more samples, both estimators look more convincing. This is because the risk of the estimator drops off as we increase the number of samples that we see from the random variable.

This is supported by theory as both types of density estimators are consistent and approach looking like the actual underlying density as we increase the number of samples we observe.

2 Turn to PDF

[21]:
```
!cp "./drive/My Drive/Colab Notebooks/homework10.ipynb" ./homework10.ipynb
!jupyter nbconvert --to PDF "homework10.ipynb"
```

```
[NbConvertApp] Converting notebook homework10.ipynb to PDF
[NbConvertApp] Support files will be in homework10_files/
[NbConvertApp] Making directory ./homework10_files
[NbConvertApp] Writing 43790 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 711209 bytes to homework10.pdf
```

hw10_question1

December 5, 2024

```
[ ]: from google.colab import drive
     drive.mount('/content/drive', force_remount=True)
     !apt-get install texlive texlive-xetex texlive-latex-extra pandoc > /content/
       ↪drive/MyDrive/installs.txt
     !pip install pypandoc > /content/drive/MyDrive/pandoc_installs.txt
```

```
Mounted at /content/drive
Extracting templates from packages: 100%
```

1 Question 1

1.1 Part A

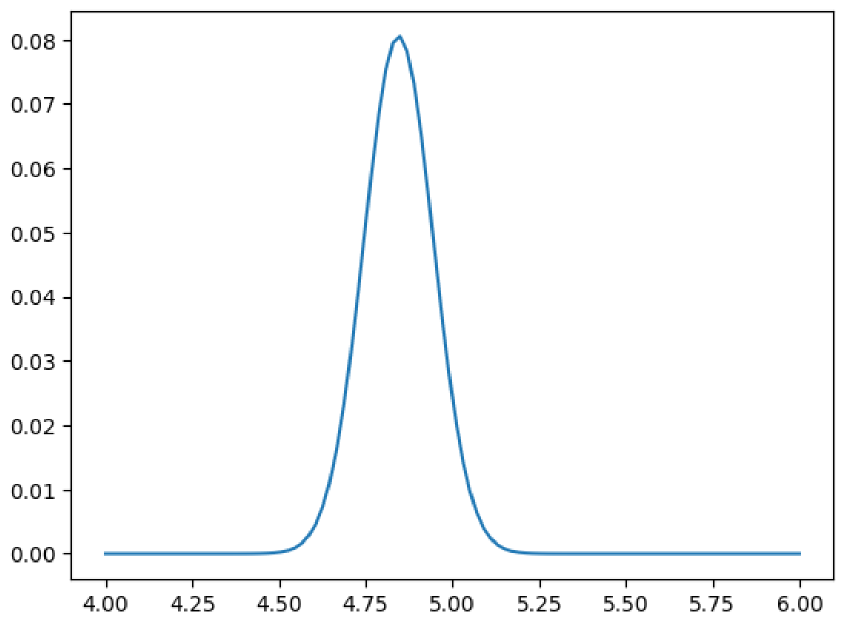
```
[1]: from numpy import random
     import numpy as np
     import matplotlib.pyplot as plt
     from scipy.stats import norm

     x = random.normal(loc=5, scale=1, size=(1, 100))

     print(x[0])
```

```
[3.68757447 5.52661666 5.09497826 4.88684317 3.85297905 3.60173846
 4.27354304 5.83386785 5.56563855 5.10525298 5.16462787 4.37866933
 5.0192959  4.62370623 2.02914245 4.08982215 6.20046321 4.82299019
 4.52947859 6.66729452 4.80429576 2.20467535 4.91045917 6.19706197
 4.59161588 4.39142523 5.78679632 5.91960706 6.94488794 5.14651206
 4.67496675 5.49026471 5.81709553 6.1624702  5.12126365 5.4600119
 4.71167575 7.43893405 3.18316307 5.3367248  3.56563071 6.08489198
 4.55456714 4.58033689 4.33226893 3.87028322 3.60228658 3.16681233
 4.62130507 5.71902801 5.04103773 5.94423745 5.0217606  7.30227426
 4.7234378  4.28395574 2.86754028 5.95629816 4.65991073 5.12478231
 5.42811268 6.46449095 5.53319662 5.23138666 4.75663376 3.71425482
 3.62850803 3.80894065 4.55418397 7.7084876  4.11414491 5.91120615
 4.62604869 3.85929815 5.41488676 4.7861139  5.45336132 4.92662916
 5.19462943 3.18327188 4.27499576 4.85286853 3.59516655 5.00690964
 6.7159407  4.58651958 3.64531492 3.97277654 3.77616091 5.90079125
 4.98884203 5.42680207 4.10550099 4.04435238 5.22818936 4.92845563
 3.5102055  4.31589177 5.6932358  3.34017279]
```

1.2 Part B

Plotting the posterior density

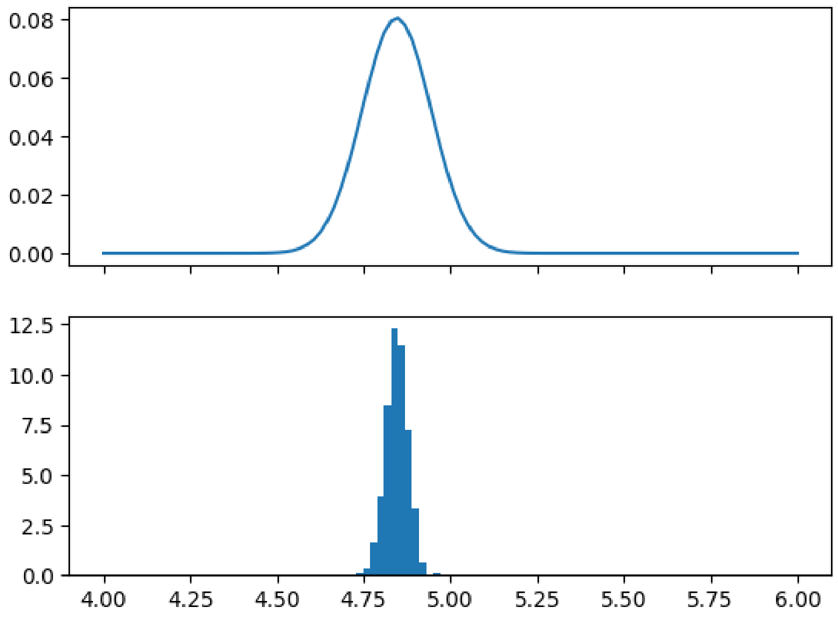
[2]:
```
mu_hat = x.mean()
mu_values = np.linspace(4, 6, 100)

likelihood = np.vectorize(lambda mu_hat: np.exp(np.log(norm.pdf(x, loc=mu_hat,␣
  ↪scale=1)).sum()))
L_i = likelihood(mu_values)

plt.plot(mu_values, L_i / L_i.sum());
plt.show()
```



1.3 Part C

[3]:
```
fig, (ax1,ax2) = plt.subplots(2, 1, sharex='col')
ax1.plot(mu_values, L_i / L_i.sum())

posterior_samples = norm.rvs(loc=mu_hat, scale=1/np.sqrt(1000), size=1000)
```

```
ax2.hist(posterior_samples, density=True, bins=mu_values);
```



```
[ ]: !cp "./drive/My Drive/Colab Notebooks/hw10_question1.ipynb" ./hw10_question1.
     ↪ipynb
     !jupyter nbconvert --to PDF "hw10_question1.ipynb"
```