Random variables and random number generation

3F3 laboratory experiment

Theo A. Brown
Selwyn College, University of Cambridge

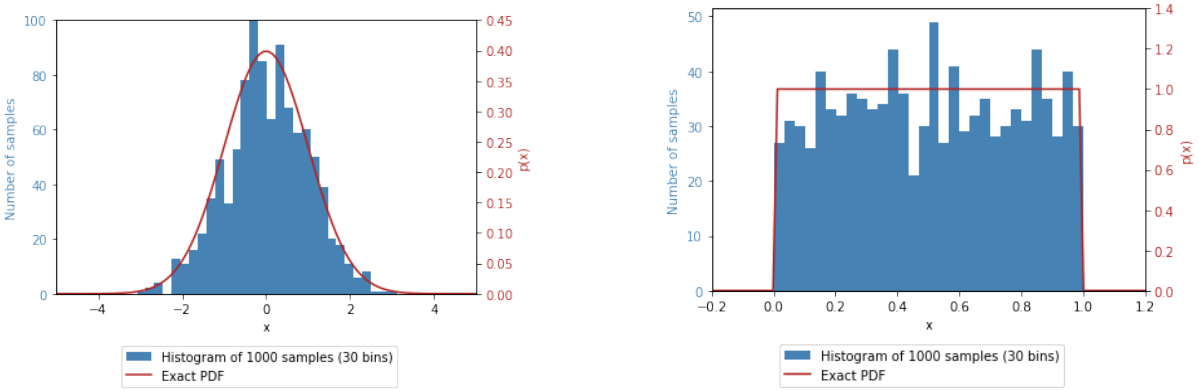
November 9, 2021

Contents

1 Generating random numbers from the Uniform and Normal distributions

1.1 Comparison of histogram with true probability density function

A vector of Gaussian random numbers was generated using `np.random.randn`, and a vector of uniformly distributed random numbers generated using `np.random.rand`. Histograms of the samples are plotted in Figure 1, overlaid with the exact probability density function (PDF). The histograms closely follow the shape of the PDF, showing that the generation of the random numbers for these distributions is accurate.



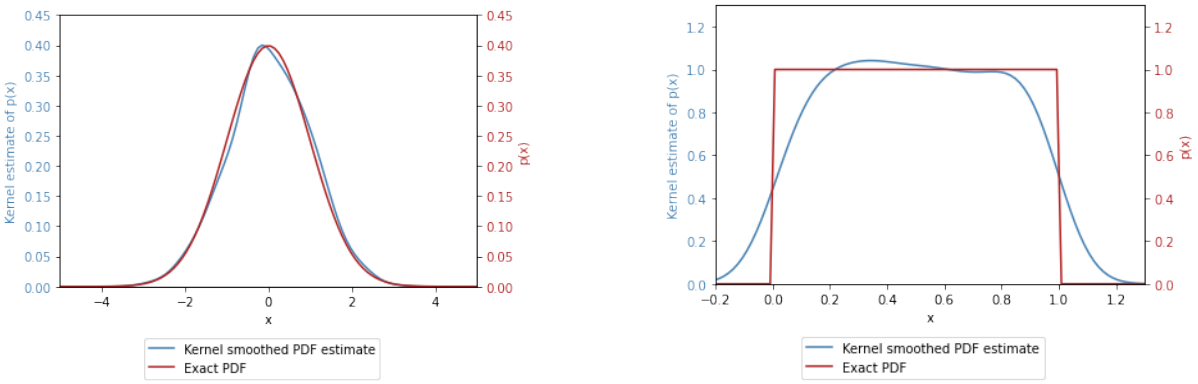
(a) Gaussian distribution

(b) Uniform distribution

Figure 1: Histogram of samples drawn from a distribution and true probability density function of distribution

1.2 Kernel density smoothing

A smooth estimate of the probability density function is calculated using the kernel smoothing method with a Gaussian kernel $\mathcal{N}(0, 1)$. The result is plotted for Gaussian and Uniform distributions in Figure 2.



(a) Gaussian distribution

(b) Uniform distribution

Figure 2: Plot of true PDF and estimate of PDF generated using Gaussian kernel smoothing

The kernel method takes an average of neighbouring values, weighted using a Gaussian distribution. This is advantageous as it can smooth random irregularities in the samples: there may be histogram bins with a disproportionately high sample count, which would give an incorrectly high estimate of the probability of this bin; the kernel smooths out this local peak by taking neighbouring values into account. However, this also means that the kernel smoothing method becomes inaccurate when there is a sudden change or discontinuity in the probability density function. In the Uniform distribution there is zero probability for values outside the range $0 < x < 1$, but as the kernel averages over a window of values it does not capture the step change at $x = 0$ and $x = 1$, instead decaying smoothly.

1.3 Multinomial distribution theory

For N samples, let B be the random variable representing the number of samples in bin j. Let the probability that a sample falls in bin j be p_j. By definition, the expected number of samples in bin j is $\mu_j = \mathbb{E}[B] = Np_j$, and the variance in the number of samples in bin j is $\sigma_j^2 = Var[B] = Np_j(1 - p_j)$. For the Uniform distribution between 0 and 1, the pdf $p(x)$ is defined as:

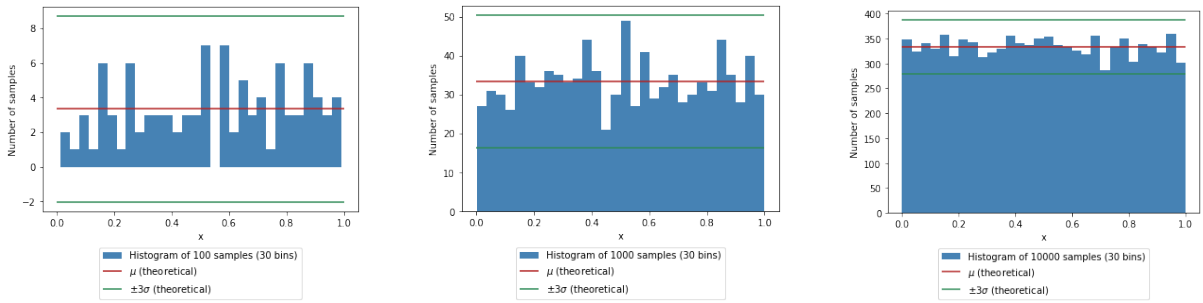
$$\begin{aligned} p(x) &= \frac{1}{x_{max} - x_{min}} \\ &= 1 \end{aligned}$$

Hence, for a histogram with bin width δ and bin center c_j:

$$\begin{aligned} p_j &= \int_{c_j - \delta/2}^{c_j + \delta/2} 1 \, dx \\ &= \delta \\ \mu_j &= N\delta \\ \sigma_j &= \sqrt{N\delta(1 - \delta)} \end{aligned}$$

As N becomes large, the estimate improves. The standard deviation increases less rapidly than the mean with increasing N ($\sigma \propto \sqrt{N}$ whereas $\mu \propto N$). Consequently an increase in N will result in the standard deviation becoming smaller compared to the mean ($\frac{\sigma}{\mu} \propto \frac{1}{\sqrt{N}}$). This is shown in Figure 3, where the standard deviation reduces as N increases.

In Figure 3 it can be seen that the sample count lies within the 3σ interval for all bins, so it is concluded that the results are consistent with the multinomial distribution theory.



(a) $N = 100$ (b) $N = 1000$ (c) $N = 10000$

Figure 3: Histogram of uniformly distributed samples for increasing N, showing mean and ±3 standard deviation

2 Functions of random variables

2.1 $f = ax + b$

For $x \sim \mathcal{N}(0, 1)$, let $y = f(x) = ax + b$. Then:

$$
\begin{aligned}
f^{-1}(y) &= \frac{y-b}{a} \\
\left|\frac{dy}{dx}\right| &= |a| \\
p_Y(y) &= \frac{1}{|a|} p_X\left(\frac{y-b}{a}\right) \\
&= \frac{1}{|a|} \times \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y-b}{a}\right)^2\right)
\end{aligned}
$$

Comparing with the equation for a general Gaussian distribution, it is clear that $p_Y(y)$ is a Gaussian distribution with mean b and variance a. This is shown in Figure 4.

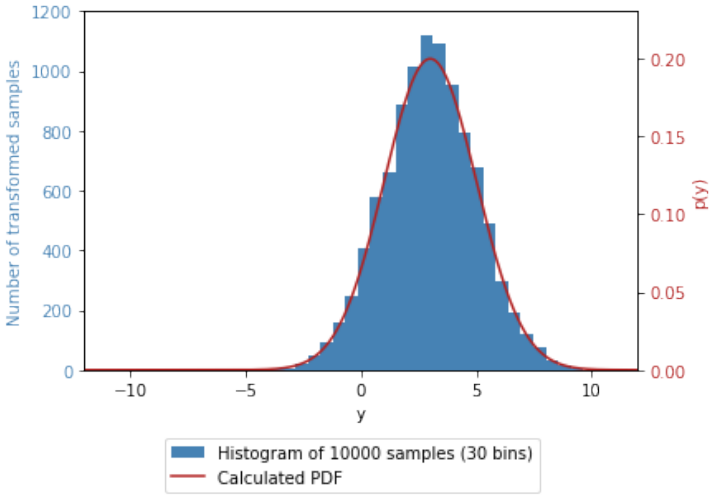


Figure 4: Histogram of linear function ($f(x^{(i)}) = 2x^{(i)} + 3$) of Gaussian samples overlaid with calculated PDF

2.2 $f = x^2$

For $x \sim \mathcal{N}(0,1)$, let $y = f(x) = x^2$. Then:

$$\begin{aligned} f^{-1}(y) &= \pm\sqrt{y} \\ \left|\frac{dy}{dx}\right| &= |2x| \\ p_Y(y) &= \sum_{i=1}^{2} \frac{1}{|2\sqrt{y}|} p_X\left(\sqrt{y}\right) \\ &= \frac{1}{\sqrt{2\pi y}} \exp\left(-\frac{1}{2}y\right) \end{aligned}$$

The result is plotted in Figure 5.

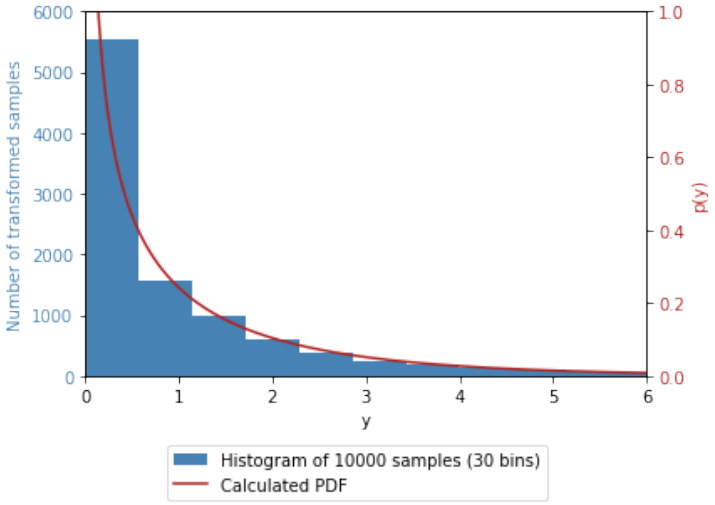


Figure 5: Histogram of quadratic function $\left(f(x^{(i)}) = \left(x^{(i)}\right)^2\right)$ of Gaussian samples overlaid with calculated PDF

3 Functions of random variables

For the exponential distribution with mean one:

$$\begin{aligned} &\text{PDF: } p(y) = \exp(-y), \ y \geq 0 \\ &\text{CDF: } F(y) = \int_0^y p(y')dy' = 1 - \exp(-y) \\ &\text{iCDF: } F^{-1}(x) = -\ln(1-x) \end{aligned}$$

Figure 6 shows that samples from the exponential distribution generated using the iCDF follow the true PDF.

4 Simulation from non-standard densities

Using the iCDF method, we can generate samples that are distributed according to:

$$p(u) = \frac{\alpha^2}{2} \exp\left(-\frac{\alpha^2}{2}u\right)$$

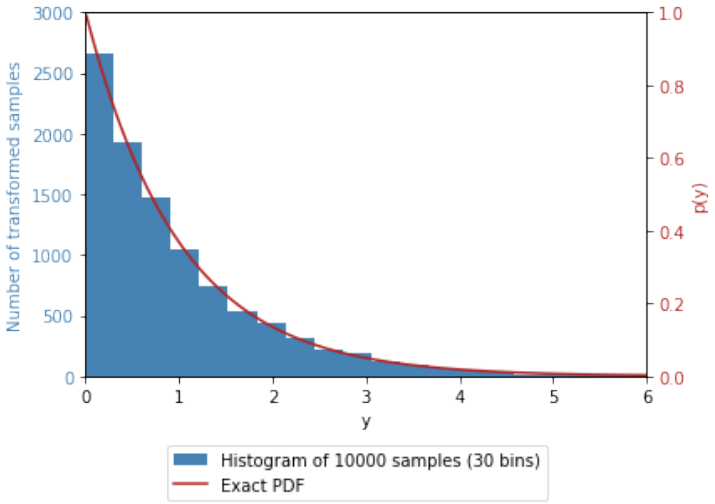


Figure 6: Histogram of samples drawn from the uniform distribution and transformed using the iCDF method to follow the exponential distribution, overlaid with the true exponential PDF

This distribution has:

$$\text{CDF: } F(u) = \int_0^u p(u')du' = 1 - \exp\left(-\frac{\alpha^2}{2}u\right)$$

$$\text{iCDF: } F^{-1}(v) = -\frac{2}{\alpha^2}\ln(1-v)$$

Samples $u^{(i)}$ can therefore be generated from uniformly distributed samples $v^(i)$. The samples $u^{(i)}$ are then used to set the variance of the random variable X:

$$p(x) = int_0^{\inf}\mathcal{N}(x\,|0,u)p(u)du$$

The limit of the integral is from zero to infinity because the standard deviation must be positive. A histogram of samples of $x^{(i)}$ is plotted in Figure 7 for different values of α. It appears that α sets the standard deviation of the distribution.

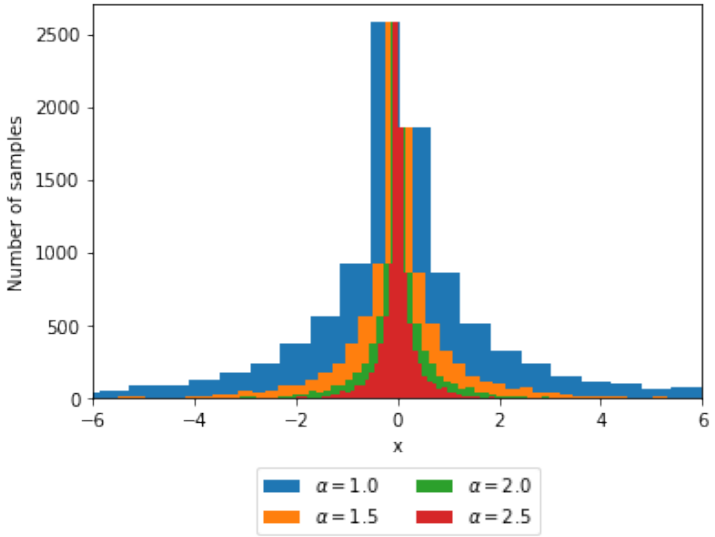
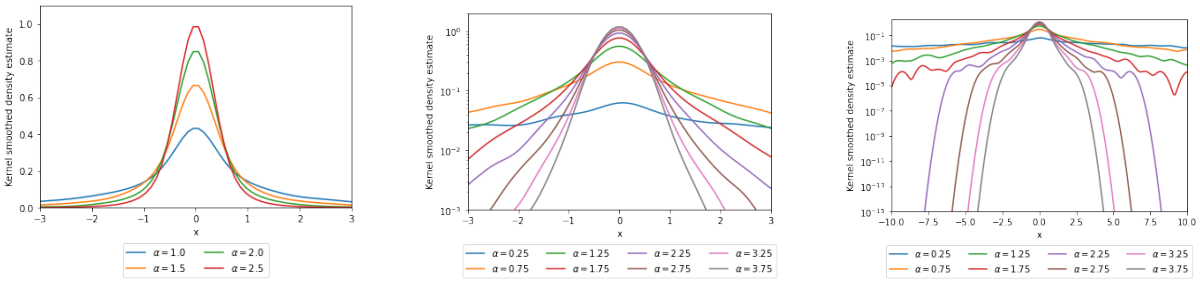


Figure 7: 100-bin histogram of samples drawn from the example distribution for different values of α

The kernel method was applied to the data, and the smoothed density was plotted on linear and log axes at different scales (Figure 8). From the shape of the distribution seems to be a sharper version of

a Gaussian distribution, with significantly higher probability density in the centre and tails that last for longer. Closer examination of the logarithmic plots suggests that the distribution behaves more like a Gaussian close to $x = 0$, where the logarithmic graph resembles an inverted parabola, and more like an exponential further from $x = 0$, where the logarithmic graph becomes linear.



(a) Linear scale (b) $10^{-3} < \ln(p(x)) < 1$ (c) $10^{-15} < \ln(p(x)) < 1$

Figure 8: Kernel smoothed probability density estimates for the example distribution

Jupyter Notebook

November 9, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
def gaussian_pdf(x, mu=0, sigma=1):
        u = (x - mu) / abs(sigma)
        y = (1 / (np.sqrt(2 * np.pi) * abs(sigma)))
        return y * np.exp(-0.5 * u**2)
```

[3]:
```
def uniform_pdf(x):
    return 1 * ((x >= 0) & (x <= 1))
```

[4]:
```
def kernel_smoothed_density(x_values, samples, width=0.3,␣
↪kernel_function=gaussian_pdf):
    # Generate an array of kernel values centred on the samples
    kernel_values = [kernel_function(x_value, samples, width) for x_value in␣
↪x_values]
    return np.average(kernel_values, axis=1)
```

[5]:
```
# General global variables
color1 = 'steelblue'
color2 = 'firebrick'
color3 = 'seagreen'
color4 = 'darkorange'
```

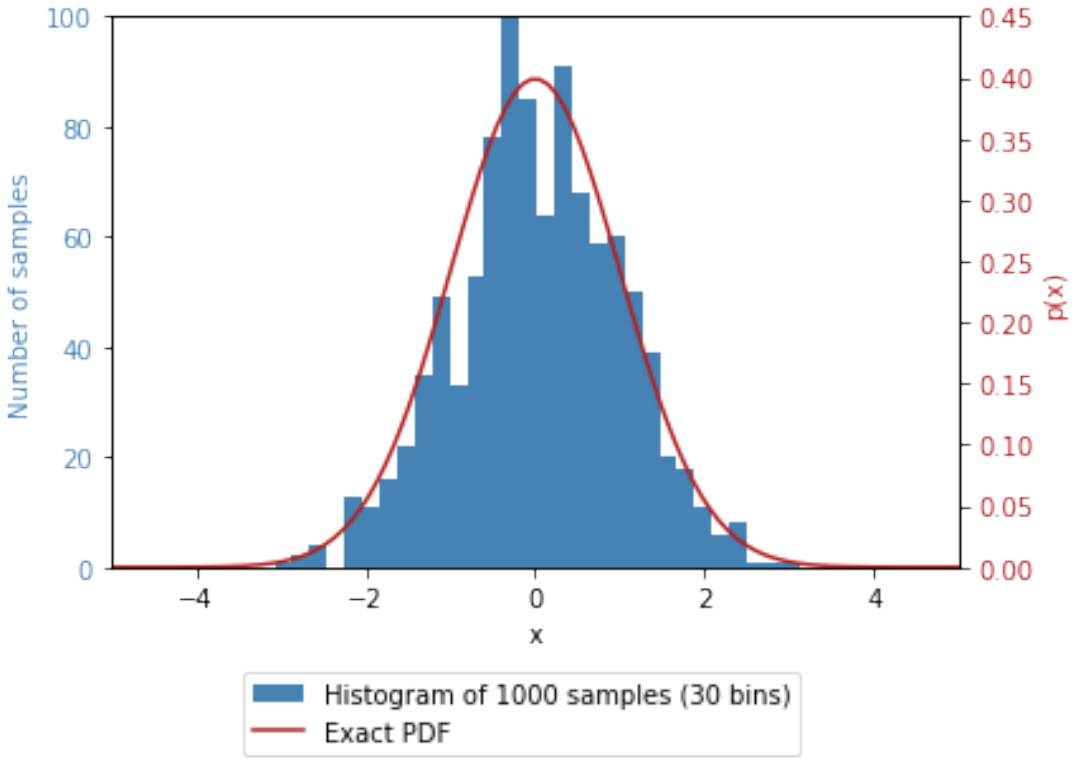
0.0.1 Section 1: Uniform and normal random variables

[6]:
```
X_gaussian = np.random.randn(10000)
X_uniform = np.random.rand(10000)
```

Histograms of random samples compared with true pdf

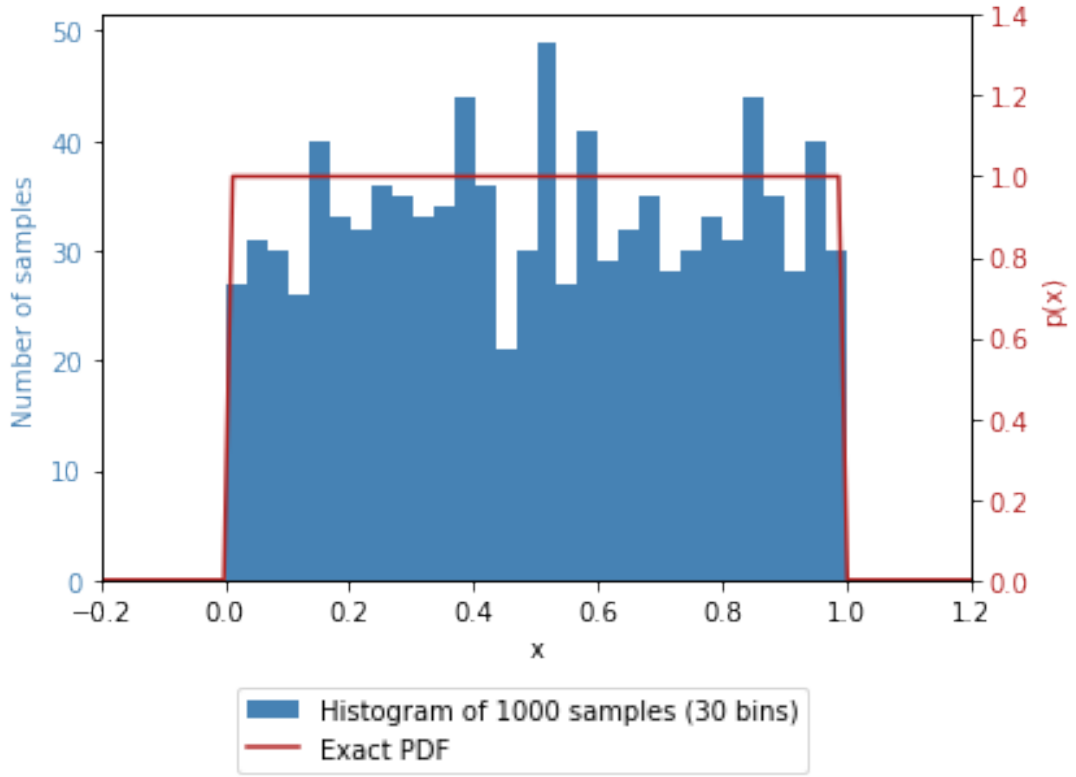
[7]:
```
fig, ax1 = plt.subplots()

ax1.hist(X_gaussian[:1000], bins=30, color=color1, label='Histogram of 1000␣
↪samples (30 bins)')
ax1.set_xlabel('x')
ax1.set_ylabel('Number of samples', color=color1)
```

```
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 100)


ax2 = ax1.twinx()
x = np.linspace(-5, 5, 100)
ax2.plot(x, gaussian_pdf(x), color=color2, label='Exact PDF')
ax2.set_ylabel('p(x)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 0.45)
ax2.set_xlim(-5, 5)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/gaussian_histogram_and_pdf.png', bbox_inches='tight')
```



[8]:
```
fig, ax1 = plt.subplots()

x = np.linspace(-0.2, 1.2, 100)

ax1.hist(X_uniform[:1000], bins=30, color=color1, label='Histogram of 1000␣
↪samples (30 bins)')
```

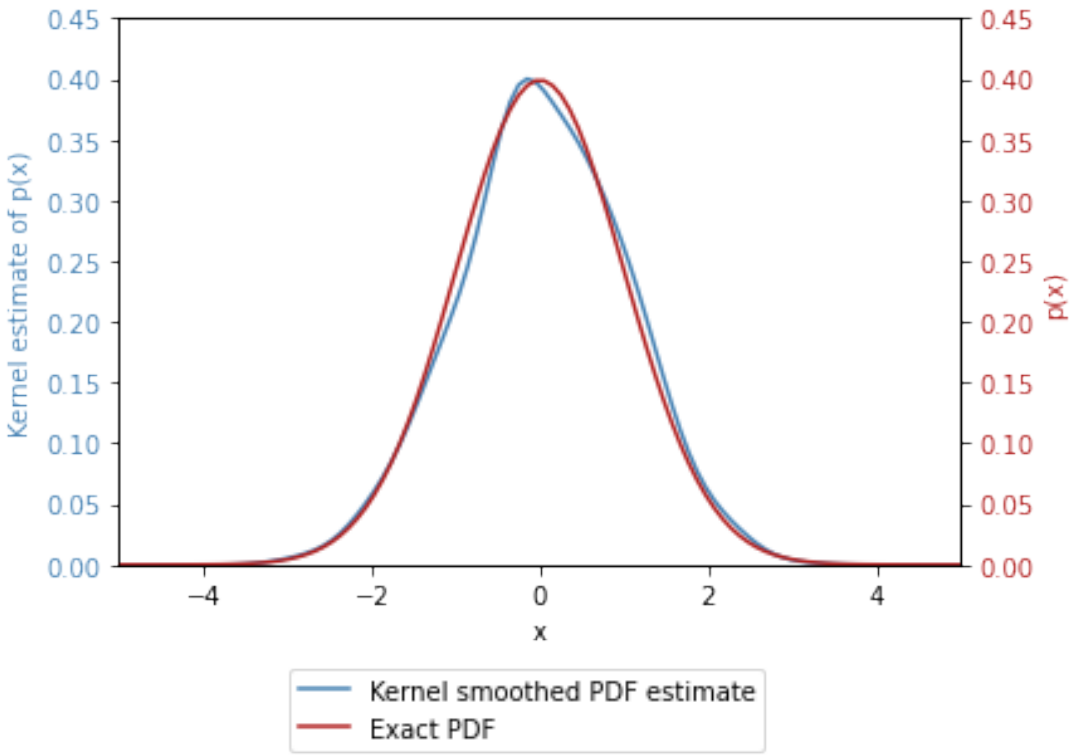
```
ax1.set_xlabel('x')
ax1.set_ylabel('Number of samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)

ax2 = ax1.twinx()
ax2.plot(x, uniform_pdf(x), color=color2, label='Exact PDF')
ax2.set_ylabel('p(x)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1.4)
ax2.set_xlim(-0.2, 1.2)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/uniform_histogram_and_pdf.png', bbox_inches='tight')
```



Kernel smoothing of random samples compared with true pdf

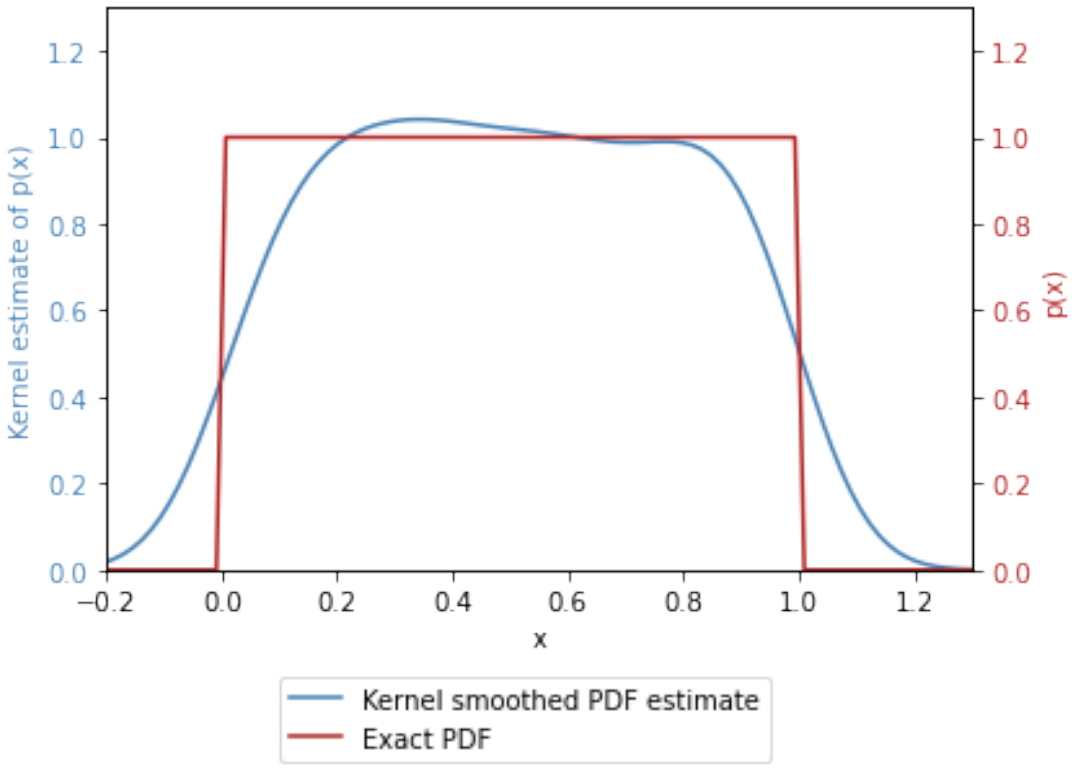
[9]:
```
fig, ax1 = plt.subplots()

x = np.linspace(-5, 5, 100)
```

```
ax1.plot(x, kernel_smoothed_density(x, X_gaussian[:1000]), color=color1,␣
 ↪label='Kernel smoothed PDF estimate')
ax1.set_xlabel('x')
ax1.set_ylabel('Kernel estimate of p(x)', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 0.45)

ax2 = ax1.twinx()
ax2.plot(x, gaussian_pdf(x), color=color2, label='Exact PDF')
ax2.set_ylabel('p(x)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 0.45)
ax2.set_xlim(-5, 5)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))
plt.savefig('figures/gaussian_kernel_smoothed.png', bbox_inches='tight')
```



[10]:
```
fig, ax1 = plt.subplots()

x = np.linspace(-0.3, 1.3, 100)
```

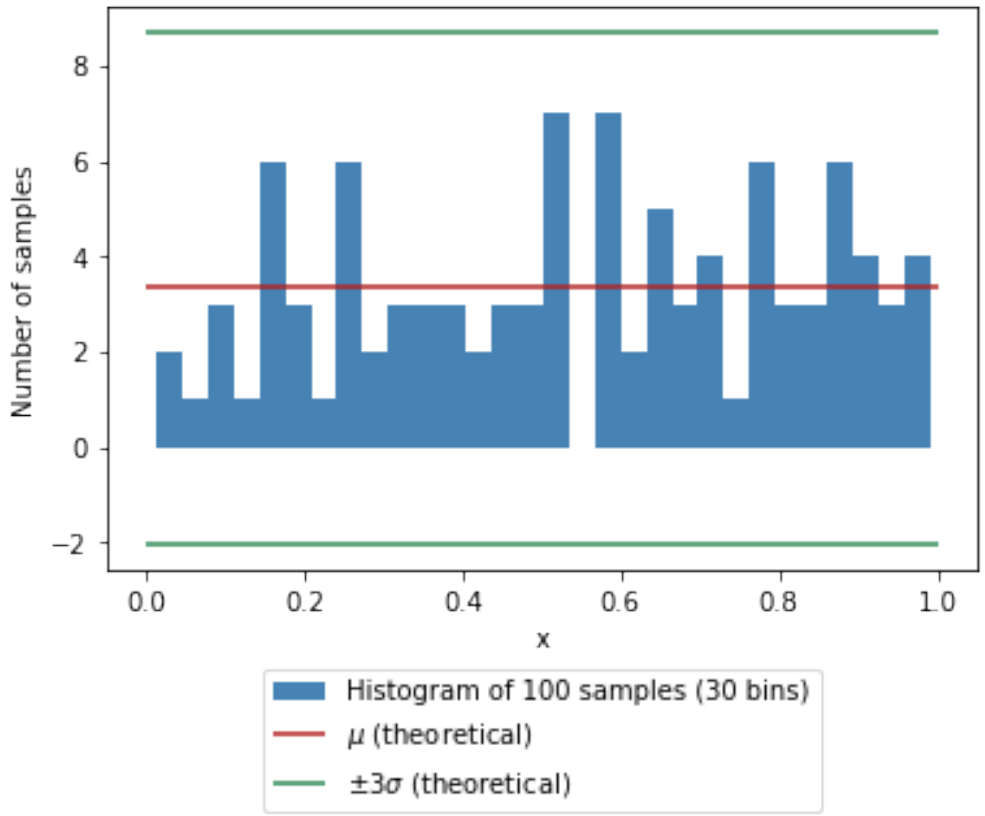
```
ax1.plot(x, kernel_smoothed_density(x, X_uniform[:1000], width=0.1),␣
 ↪color=color1, label='Kernel smoothed PDF estimate')
ax1.set_xlabel('x')
ax1.set_ylabel('Kernel estimate of p(x)', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 1.3)


ax2 = ax1.twinx()
ax2.plot(x, uniform_pdf(x), color=color2, label='Exact PDF')
ax2.set_ylabel('p(x)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1.3)
ax2.set_xlim(-0.2, 1.3)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))
plt.savefig('figures/uniform_kernel_smoothed.png', bbox_inches='tight')
```



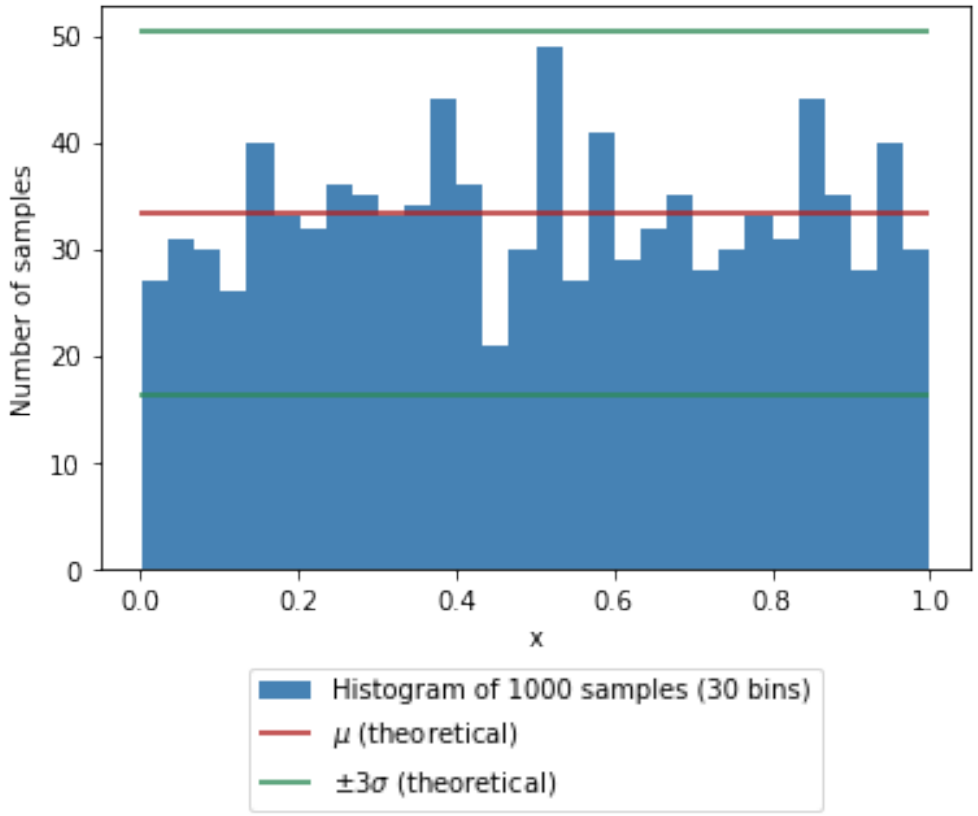
[11]:
```
def plot_uniform_histogram_mean_sd(N, nbins=30):
    plt.figure()
```

```
    plt.hist(X_uniform[:N], bins=nbins, color=color1, label=f'Histogram of {N}␣
↪samples ({nbins} bins)')

    bin_width = 1 / nbins
    mean = N * bin_width
    sd = np.sqrt(N * bin_width * (1 - bin_width))
    plt.hlines(y=mean, xmin=0, xmax=1, color=color2, label=r'$\mu$␣
↪(theoretical)')
    plt.hlines(y=[mean + 3 * sd, mean - 3 * sd], xmin=0, xmax=1, color=color3,␣
↪label='$\pm 3 \sigma$ (theoretical)')

    plt.xlabel('x')
    plt.ylabel('Number of samples')
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15))
    plt.savefig(f'figures/uniform_histogram_{N}.png', bbox_inches='tight')
```

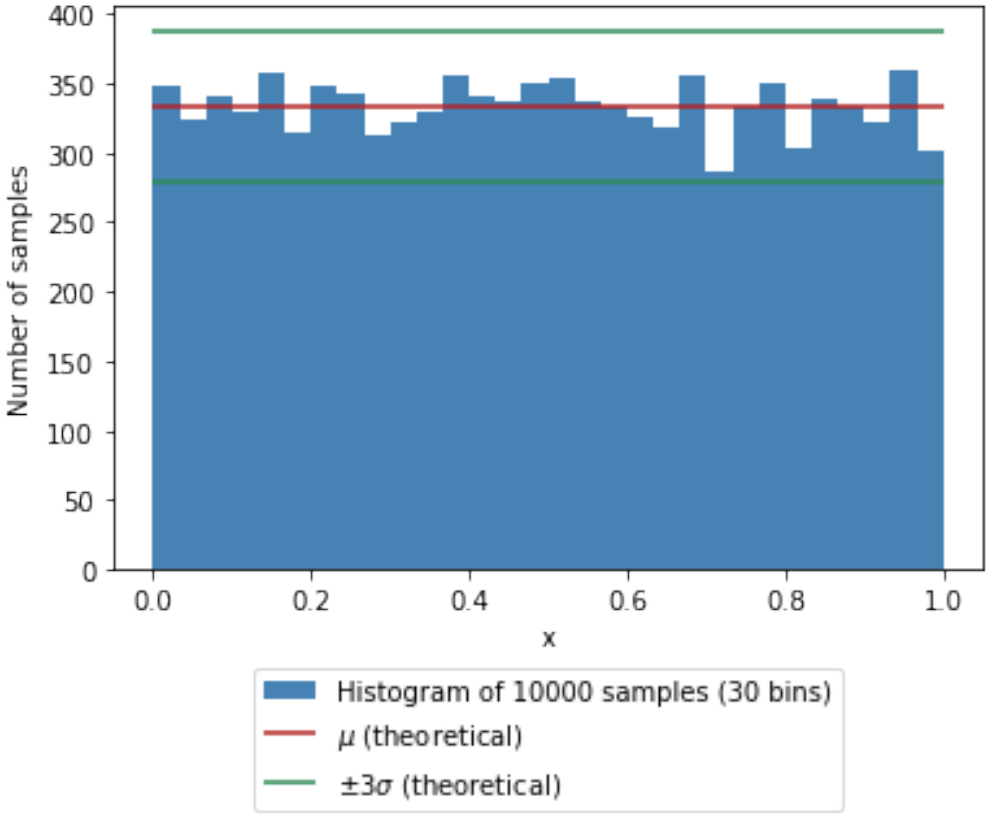
[12]:
```
plot_uniform_histogram_mean_sd(100)
```



[13]:
```
plot_uniform_histogram_mean_sd(1000)
```



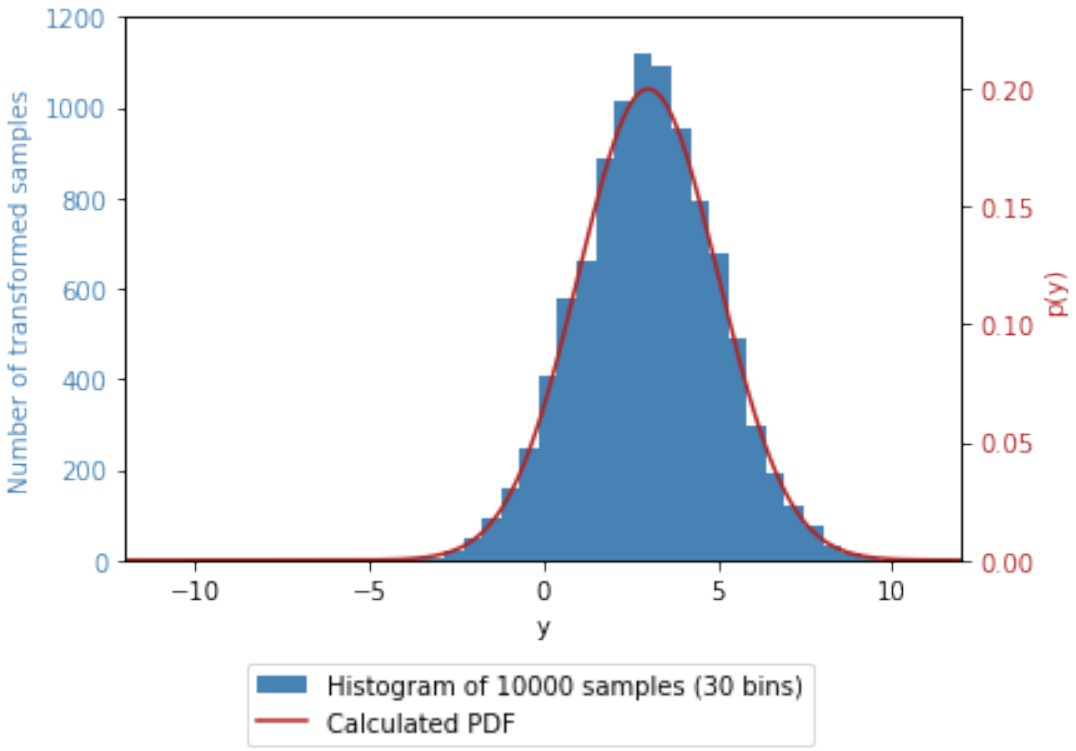
[14]:
```
plot_uniform_histogram_mean_sd(10000)
```



0.0.2 Section 2: Functions of Random Variables

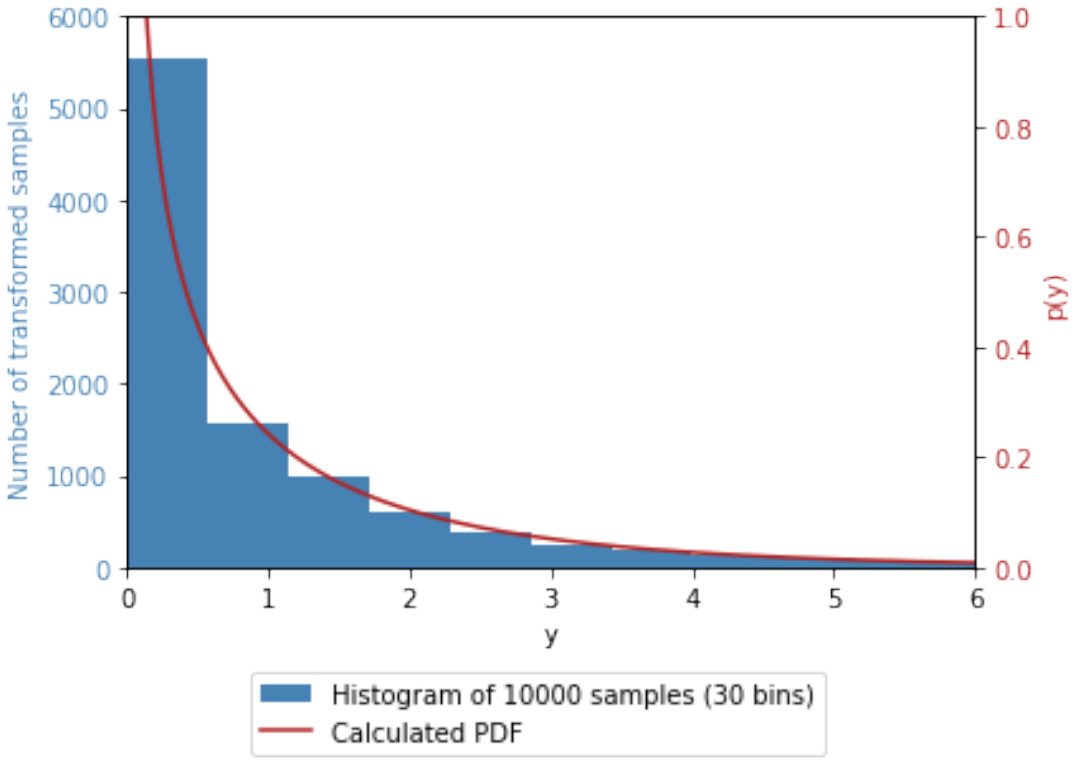
[15]:
```
fig, ax1 = plt.subplots()

Y = 2 * X_gaussian[:10000] + 3
ax1.hist(Y, bins=30, color=color1, label='Histogram of 10000 samples (30 bins)')
ax1.set_xlabel('y')
ax1.set_ylabel('Number of transformed samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 1200)


ax2 = ax1.twinx()
x = np.linspace(-12, 12, 1000)
ax2.plot(x, gaussian_pdf(x, mu=3, sigma=2), color=color2, label='Calculated␣
 ↪PDF')
ax2.set_ylabel('p(y)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 0.23)
ax2.set_xlim(-12, 12)
```

```
fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/linear_function_of_gaussian.png', bbox_inches='tight')
```



[16]:
```
fig, ax1 = plt.subplots()

Y = X_gaussian[:10000] ** 2
ax1.hist(Y, bins=30, color=color1, label='Histogram of 10000 samples (30 bins)')
ax1.set_xlabel('y')
ax1.set_ylabel('Number of transformed samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 6000)


ax2 = ax1.twinx()
x = np.linspace(0.01, 12, 1000)
pdf = np.exp(-0.5*x) / np.sqrt(2*np.pi*x)
ax2.plot(x, pdf, color=color2, label='Calculated PDF')
ax2.set_ylabel('p(y)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1)
```

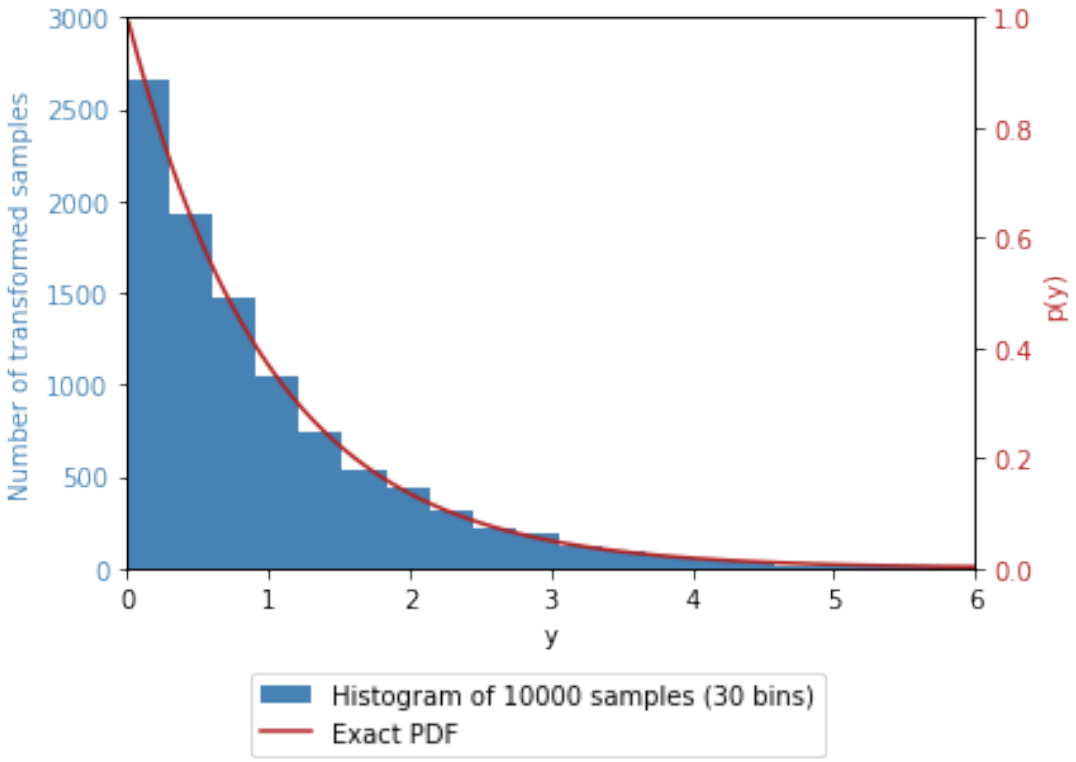
```
ax2.set_xlim(0, 6)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/quadratic_function_of_gaussian.png', bbox_inches='tight')
```



0.0.3 Section 3: iCDF method

[24]:
```
fig, ax1 = plt.subplots()

Y = -np.log(1 - X_uniform[:10000])

ax1.hist(Y, bins=30, color=color1, label='Histogram of 10000 samples (30 bins)')
ax1.set_xlabel('y')
ax1.set_ylabel('Number of transformed samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 3000)


ax2 = ax1.twinx()
x = np.linspace(0.01, 12, 1000)
```

```
pdf = np.exp(-x)
ax2.plot(x, pdf, color=color2, label='Exact PDF')
ax2.set_ylabel('p(y)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1)
ax2.set_xlim(0, 6)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/icdf_exponential.png', bbox_inches='tight')
```

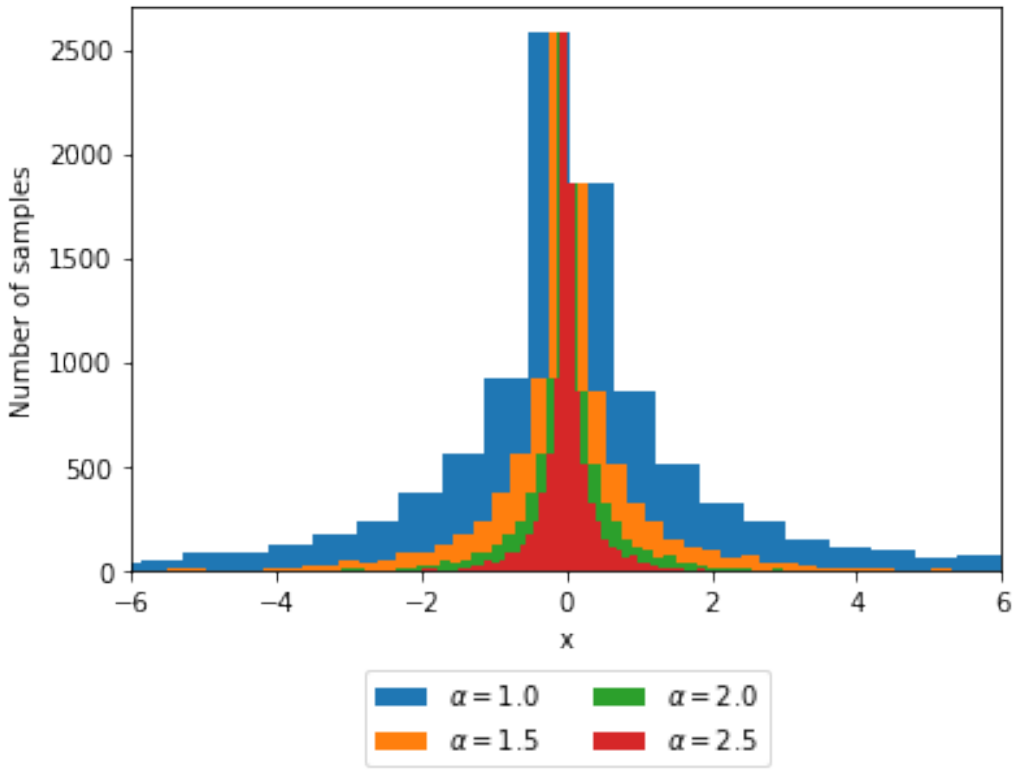


0.0.4 Section 4: Simulation from non-standard densities

[18]:
```
def nonstandard_samples(N, alpha=1.5):
    exponential_samples = (-2 / (alpha ** 2)) * np.log(1 - X_uniform[:N])
    return X_gaussian[:N] * exponential_samples
```

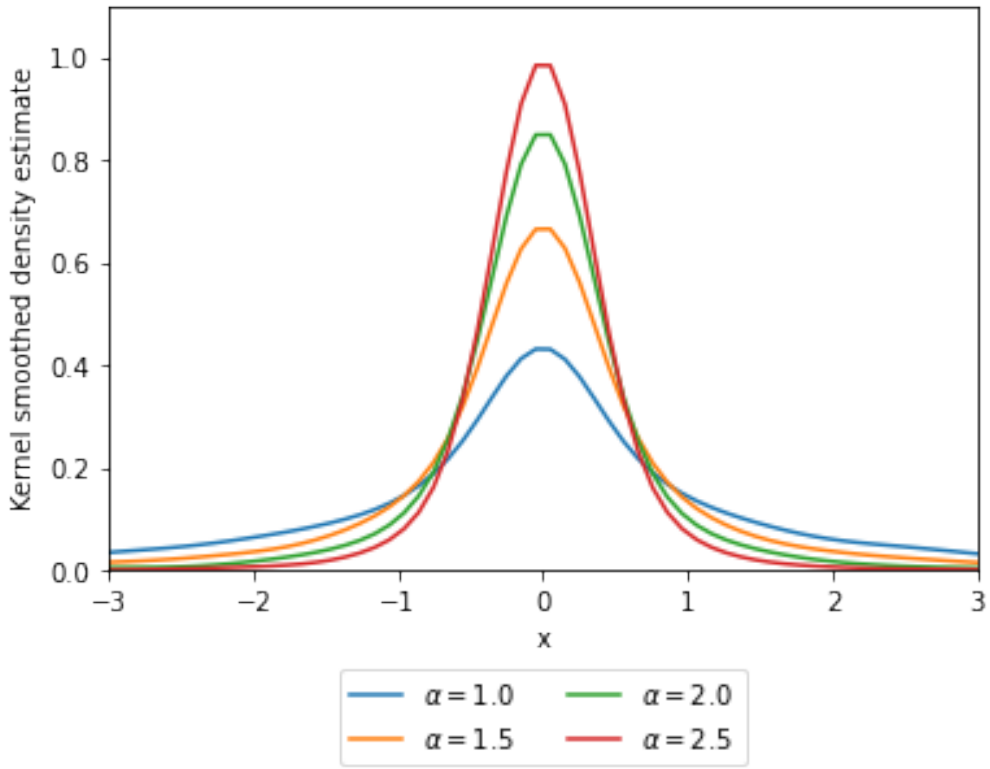
[19]:
```
plt.figure()
N = 100000
for i in np.arange(0, 4):
    alpha = (i/2) + 1
```

```
    plt.hist(nonstandard_samples(N, alpha), bins=100, label=r'$\alpha = {}$'.
 ↪format(alpha))

plt.xlabel('x')
plt.ylabel('Number of samples')
plt.xlim(-6, 6)
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=2)
plt.savefig(f'figures/nonstandard_distribution.png', bbox_inches='tight')
```



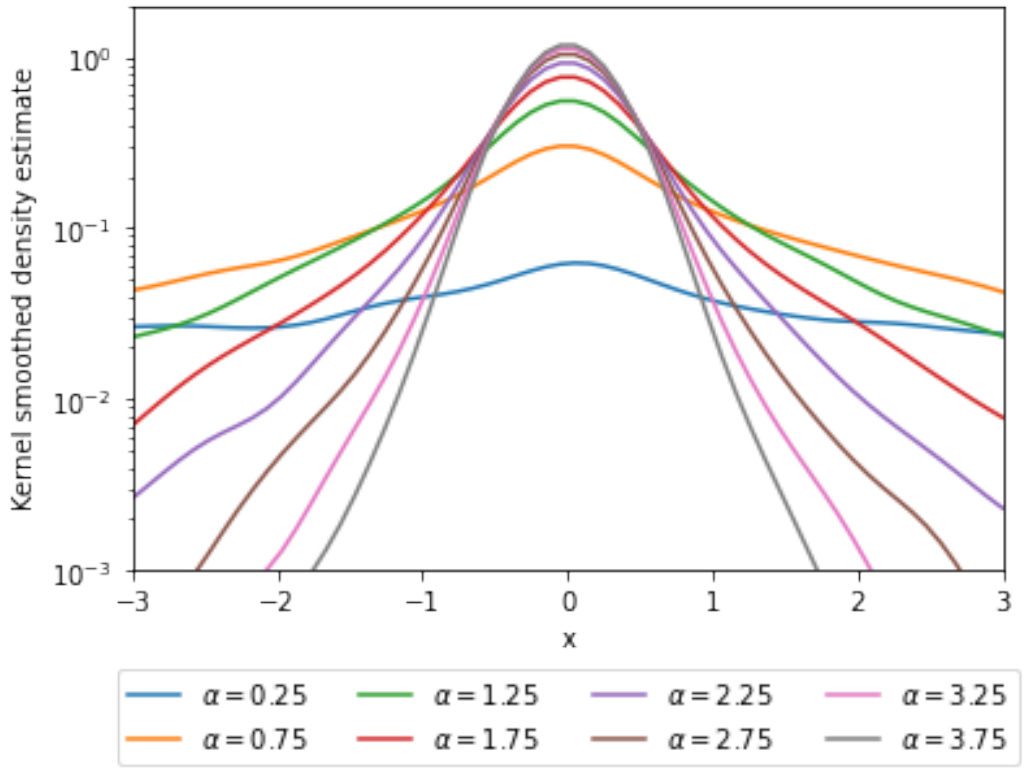
[20]:
```
plt.figure()

N = 100000
x = np.linspace(-5, 5, 100)

for i in np.arange(0, 4):
    alpha = i/2 + 1
    plt.plot(x, kernel_smoothed_density(x, nonstandard_samples(N, alpha)),␣
 ↪label=r'$\alpha = {}$'.format(alpha))

plt.xlabel('x')
plt.ylabel('Kernel smoothed density estimate')
```

```
plt.xlim(-3, 3)
plt.ylim(0, 1.1)
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=2)
plt.savefig(f'figures/nonstandard_distribution_ksdensity.png',␣
 ↪bbox_inches='tight')
```

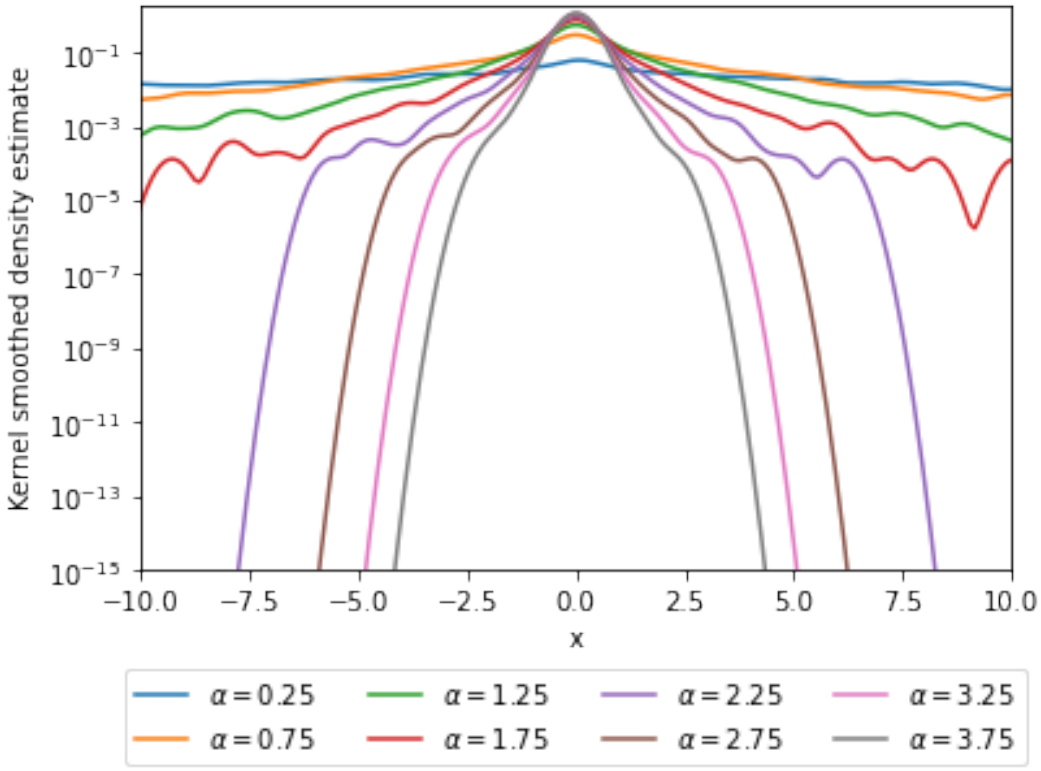


[21]:
```
plt.figure()

N = 100000
x = np.linspace(-5, 5, 100)

for i in np.arange(0, 8):
    alpha = i/2 + 0.25
    plt.semilogy(x, kernel_smoothed_density(x, nonstandard_samples(N, alpha)),␣
 ↪label=r'$\alpha = {}$'.format(alpha))

plt.xlabel('x')
plt.ylabel('Kernel smoothed density estimate')
plt.xlim(-3, 3)
plt.ylim(1e-3, 2)
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=4)
```

```
plt.savefig(f'figures/nonstandard_distribution_ksdensity_log_close.png',␣
 ↪bbox_inches='tight')
```



[22]:
```
plt.figure()

N = 100000
x = np.linspace(-50, 50, 1000)

for i in np.arange(0, 8):
    alpha = i/2 + 0.25
    plt.semilogy(x, kernel_smoothed_density(x, nonstandard_samples(N, alpha)),␣
 ↪label=r'$\alpha = {}$'.format(alpha))

plt.xlabel('x')
plt.ylabel('Kernel smoothed density estimate')
plt.xlim(-10, 10)
plt.ylim(1e-15, 2)
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=4)
plt.savefig(f'figures/nonstandard_distribution_ksdensity_log_far.png',␣
 ↪bbox_inches='tight')
```


Kernel smoothed density estimate
10^{-1}
10^{-3}
10^{-5}
10^{-7}
10^{-9}
10^{-11}
10^{-13}
10^{-15}
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x
$\alpha = 0.25$
$\alpha = 0.75$
$\alpha = 1.25$
$\alpha = 1.75$
$\alpha = 2.25$
$\alpha = 2.75$
$\alpha = 3.25$
$\alpha = 3.75$