ENGG1003 - Thursday Week 9

Introduction to random numbers from normal distributions

Steve Weller

University of Newcastle

6 May 2021

Last compiled: May 5, 2021 6:57pm +10:00

Lecture overview

1. **standard** normal distribution (bell curve)
   - pdf, mean $\mu = 0$ and $\sigma = 1$
   - generate using Python
   - histogram
2. using integration to compute probabilities using standard normal distribution
   - area (needs integration) and probability

1) Standard normal distribution

- Straight into it, generate $100,000$ random numbers generated using `normal` function in numpy's random library
- mean $= 0$, std $= 1$

filename.py

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(1)
x = np.random.normal(0.0, 1.0, size=100000)

plt.hist(x, 10)
plt.show()
```

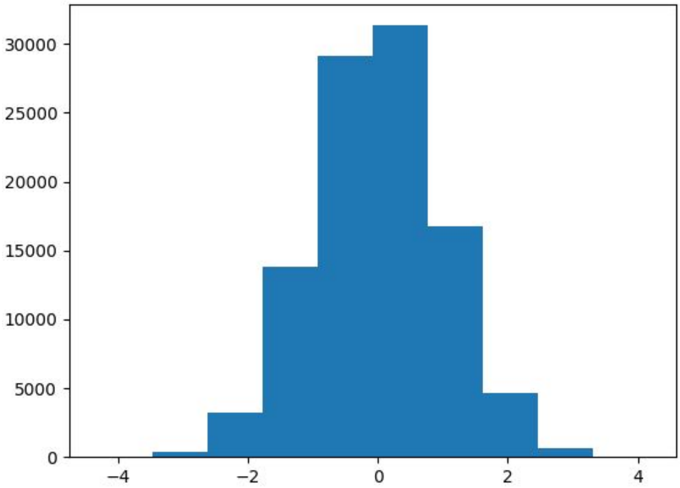
- code commentary
- normal random numbers aka Gaussian distribution
- general form of call to normal()
- explain hist()
- live demo
- nothing much to see in plot of numbers themselves
  "noise"

Histogram

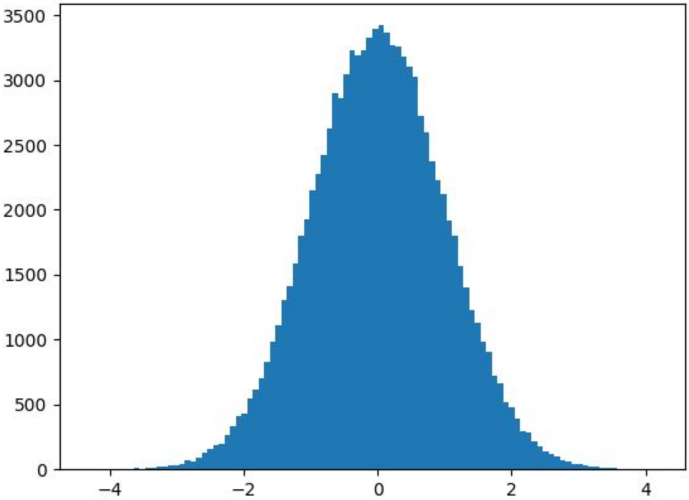
- interpret histogram
- bins, counts, examples
- call hist to return bins—too hard?

- A histogram is a graph showing frequency distributions
- It is a graph showing the number of observations within each given interval.
- To visualize the data set we can draw a histogram with the data we collected
- We will use the Python module Matplotlib to draw a histogram

10 bins

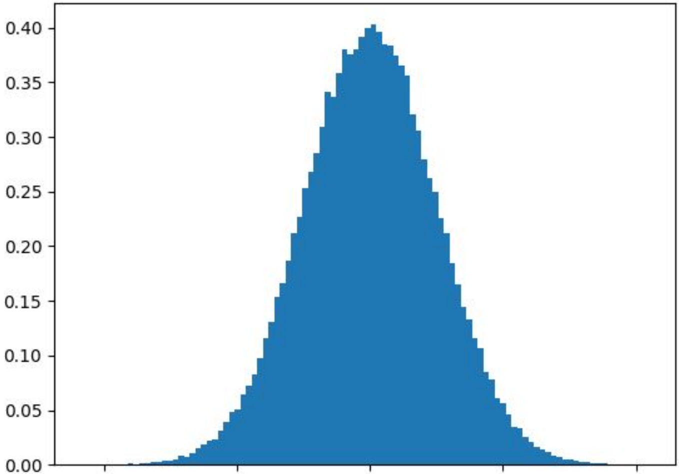


Identical data, but now 100 bins



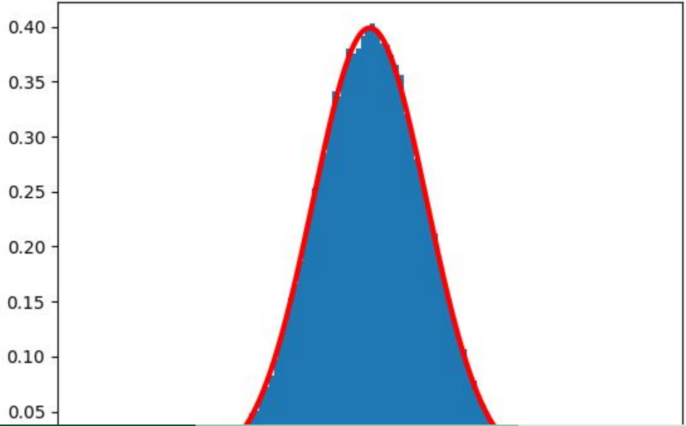
Identical data, same 100 bins, now area under histogram is normalized to 1

```
plt.hist(x, 100, density=True)
```



Now with standard normal pdf as red line

```
x = np.linspace(-5,5,num=1000)
f = 1/(np.sqrt(2 * np.pi)) * np.exp(-x**2 / 2)

plt.hist(d, 100, density=True)
plt.plot(x, f, color='r', linewidth=3)
```



Standard normal distribution

Equation of red function

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

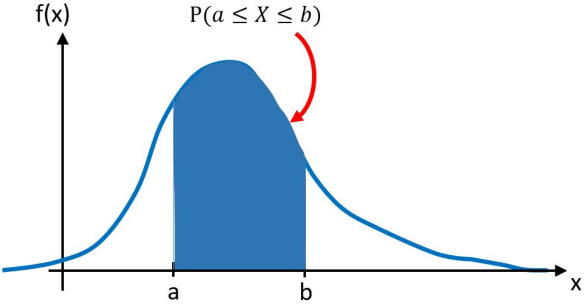
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5,5,num=1000)
f = 1/(np.sqrt(2 * np.pi)) * np.exp(-x**2 / 2)
plt.plot(x, f, color='r', linewidth=3)
```

$f(x)$ is an example of a *probability density function (pdf)*

Probability density functions

- probability that X takes a value in interval $[a, b]$ is

$$\mathrm{P}(a \leq X \leq b) = \int_a^b f(x)dx$$



pdf properties

To qualify as a PDF, function f must be non-negative, and must have the normalization property. Thsi means the entire area under the graph of f must be equal to 1

- area under $f(x)$ is 1
  - reason for the $1/\sqrt{2\pi}$ factor

- $f(x) \geq 0$ for all x, since probability can't be negative

- total area under pdf is 1, since X must take some value

$$\int^{\infty} f(x)dx = \mathrm{P}(-\infty \leq X \leq \infty) = 1$$

2) Integration

the story so far ...

- PDF of random numbers following *standard normal distribution* is a "bell curve"

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

- probability of a random number drawn from standard normal distribution taking value in interval $[a, b]$ is

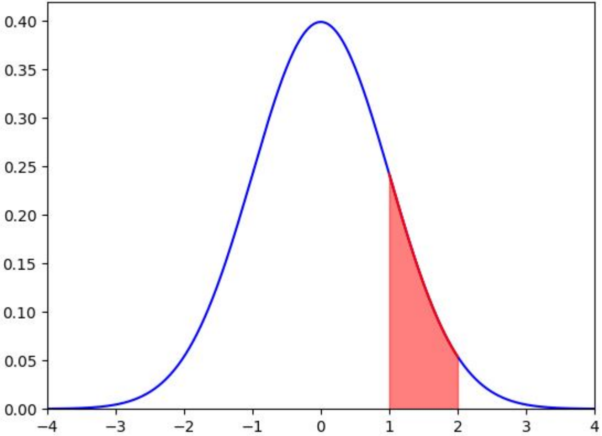
$$\mathrm{P}(a \leq X \leq b) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-x^2/2} dx$$

Example: “engineering” application

- exact expression doesn't exist for $\int_a^b e^{-x^2/2} dx$
- need to use numerical integration

Example

- $a = 1$, $b = 2$
- calculated probability using `standardnormal.py` is 0.1359
  - uses trapezoidal method with 100 sub-intervals on $[1, 2]$



red shaded area is

$$\int_1^2 e^{-x^2/2} dx \approx 0.1359$$

Python code

standardnormal.py

```python
# standardnormal

import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return 1/(np.sqrt(2 * np.pi)) * np.exp(-x**2 / 2)

def trapezoidal(f, a, b, n):
    h = (b-a)/n
    f_sum = 0
    for i in range(1, n, 1):
        x = a + i*h
        f_sum = f_sum + f(x)
    return h*(0.5*f(a) + f_sum + 0.5*f(b))
```

Python code

standardnormal.py—continued

```
a = 1
b = 2

prob_ab = trapezoidal(f, a, b, 100)
print('Probability X in range [{},{}] is: {:.4f}'.format(a, b,
    prob_ab))

x = np.linspace(-4, 4, 1000)
xab = np.linspace(a, b, 100)

plt.plot(x, f(x), 'b')              # standard normal pdf
plt.plot(xab,f(xab),'r')
plt.fill_between(xab,f(xab),color='r',alpha=0.5) #alpha=
    transparency
plt.axis([-4, 4, 0, 0.42])

plt.show()
```

Demo of standard normal generation

- generate 10^6 random numbers
- expect $10^6 \times 0.1359 = 135,900$ in range $[1, 2]$
- live demo
- results

Python code

standardnormaldemo.py

```
# standardnormaldemo

import numpy as np
import matplotlib.pyplot as plt

N = 1000000
x = np.random.normal(0.0, 1.0, size=N)
a = 1
b = 2
num_ab = 0
for k in range(0, len(x)):
    #print(k)
    if a <= x[k] <= b:
        num_ab += 1

print('{} standard normal random numbers'.format(N))
print('#random numbers in range [{},{}] = {}'.format(a, b, num_ab
    ))
```

Lecture summary

1. xxx
2. xxx
3. what's next