Python Applications

Modeling!

CS101 Lecture #16

2016-11-16

Modeling

Modeling

- Consider a cup falling from the edge of a table, describe its path and time until it hits the ground.
  - Use analytical equation (if available)
  - Use *finite difference* equation for numerical approximation!

Modeling

Use analytical equation (if available):

$$y(t) = y_0 + v_0 t + 0.5at^2$$
$$y_0 = 1$$
$$v_0 = 0$$
$$a = -9.8$$

Subject to:

$$y(t) \geq 0$$

Modeling

```
Import numpy as np

# parameters of simulation
n = 100
start = 0.0
end = 1.0
a = -9.8

# state variable initialization
t = np.linspace(start, end, n+1)  #time, s

y = 1.0 + a/2*t**2

for i in range(1,n+1):
    if y[i] <= 0:
        y[i] = 0
```

Modeling

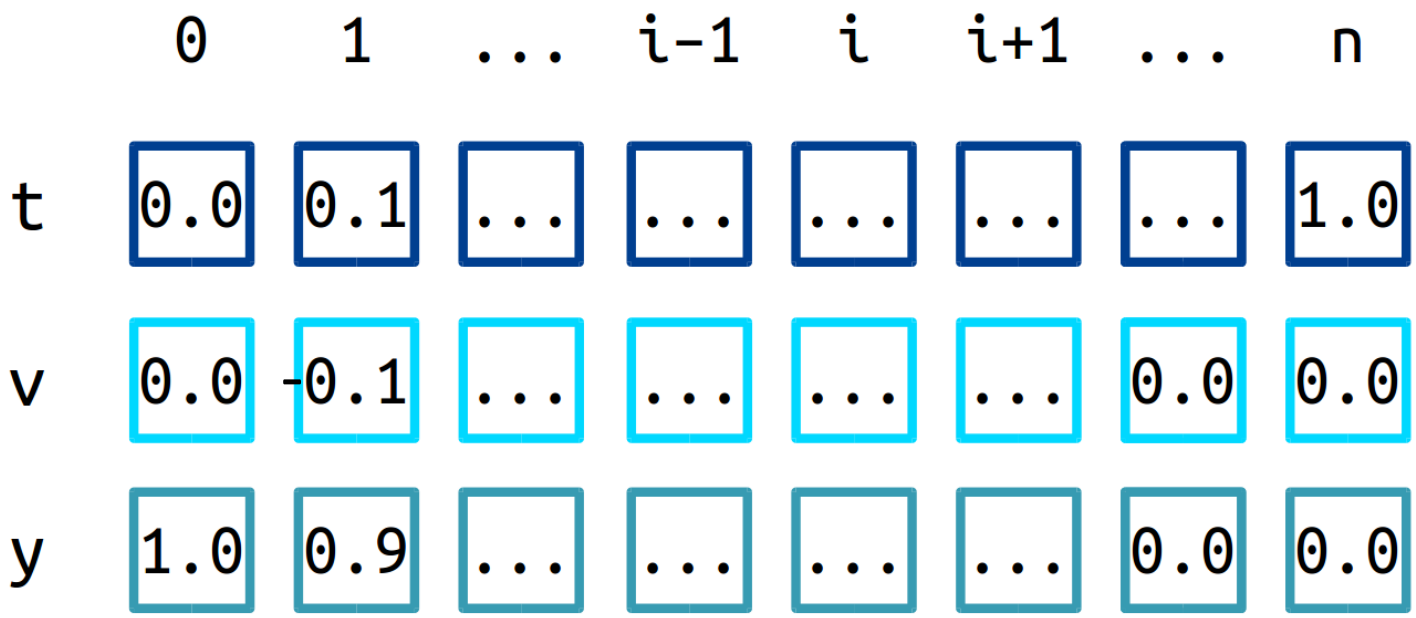
- Use ***finite difference*** equation

$v^0 = 0, \quad y^0 = 1 \quad a = -9.8$ subject to: $y(t) \geq 0$

$$a = \frac{dv}{dt} \approx \frac{v^{n+1} - v^n}{t^{n+1} - t^n} \quad \rightarrow \quad v^{n+1} = v^n + a(t^{n+1} - t^n)$$

$$v(t) = \frac{dy}{dt} \approx \frac{y^{n+1} - y^n}{t^{n+1} - t^n} \quad \rightarrow \quad y^{n+1} = y^n + v(t^{n+1} - t^n)$$

Modeling



Modeling

```
import numpy as np

# parameters of simulation
n = 100
start = 0.0
end = 1.0
a = -9.8

# state variable initialization
t = np.linspace(start, end, n+1)  #time, s
v = np.zeros(n+1)
y = np.zeros(n+1)
y[0] = 1.0

for i in range(1,n+1):
    v[i] = v[i-1] + a*(t[i]-t[i-1])
    y[i] = y[i-1] + v[i-1]*(t[i]-t[i-1])      ##Watch out!

    if y[i] <= 0:
        v[i] = 0
        y[i] = 0
```

Modeling

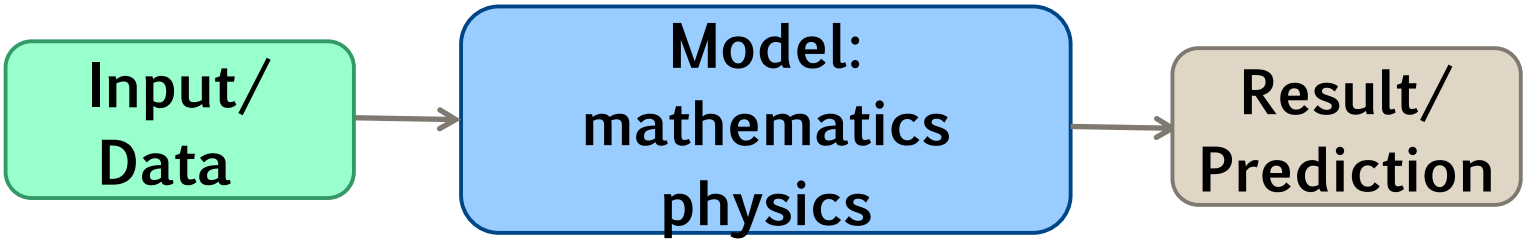
- How would you make the cup bounce?

- How would you include lateral motion and bouncing?

Mathematic Modeling

Explain data with a model

the same story



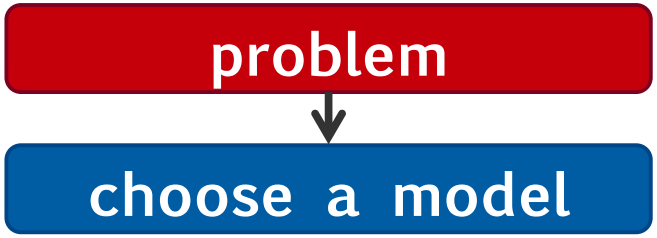
elements of mathematical modeling

the modeling lifecycle

problem

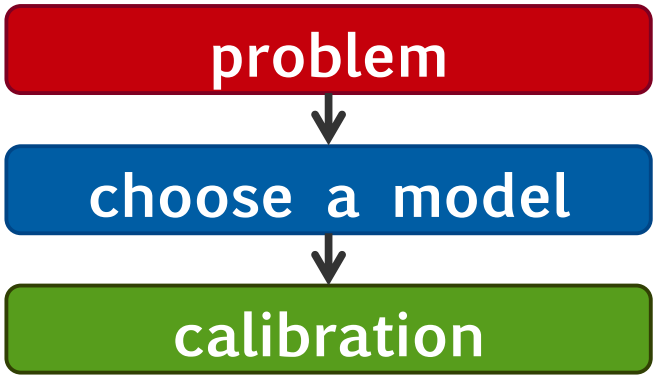
elements of mathematical modeling

the modeling lifecycle



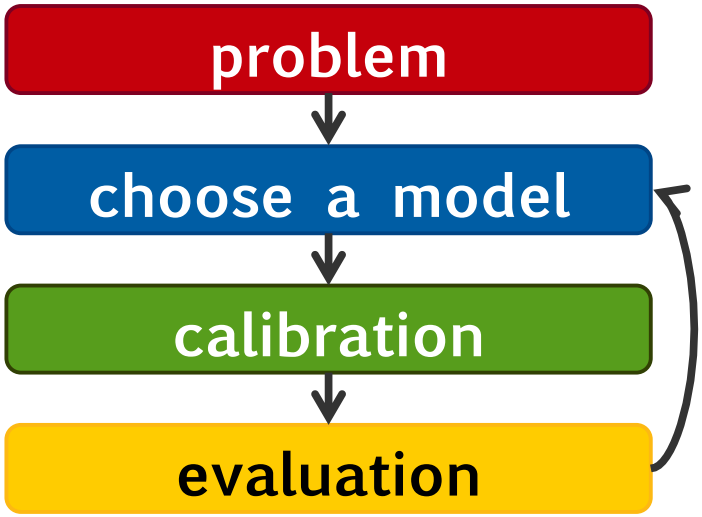
elements of mathematical modeling

the modeling lifecycle



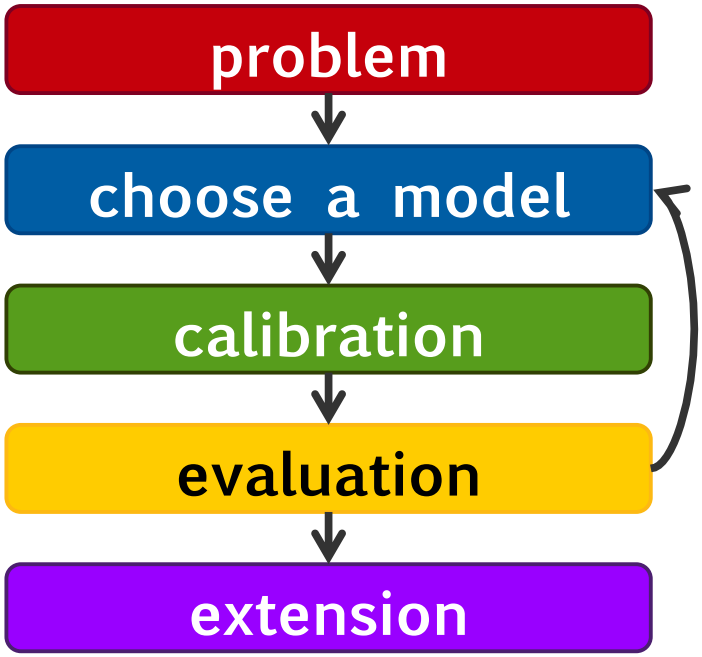
elements of mathematical modeling

the modeling lifecycle

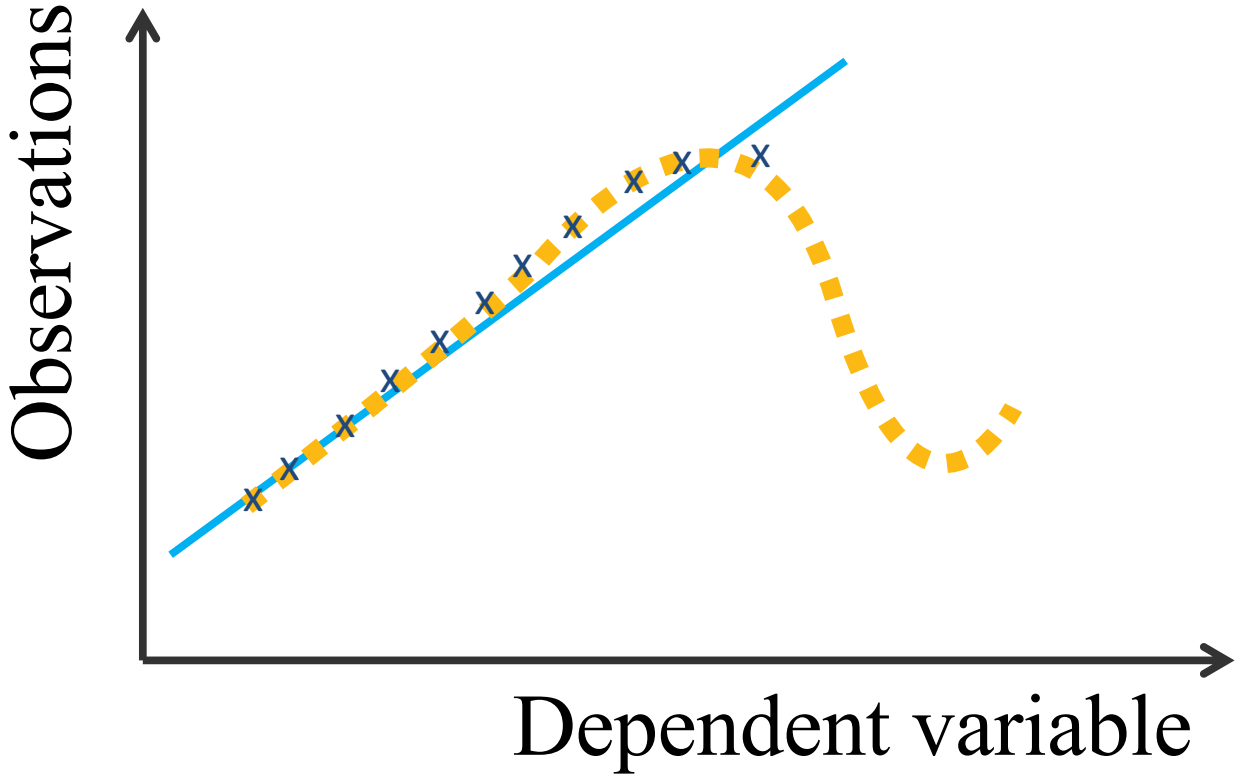


elements of mathematical modeling

the modeling lifecycle



Explain data with a model

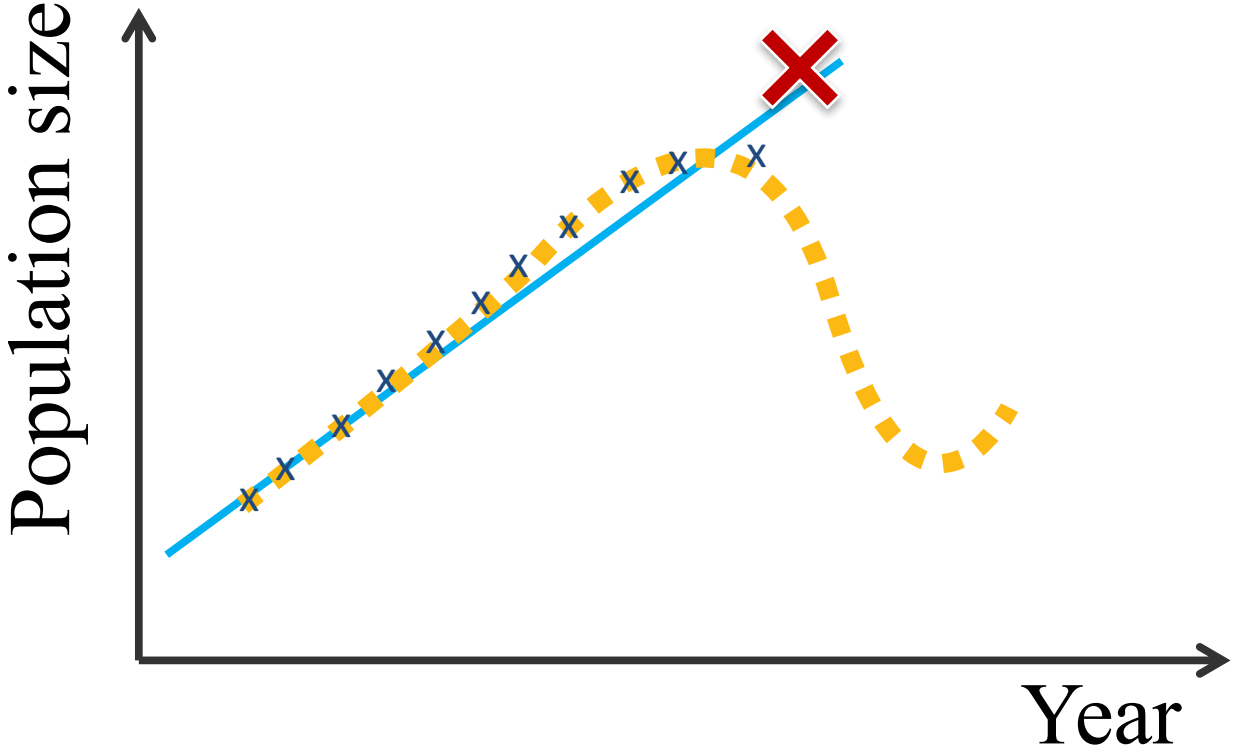


Which model better explains the data?

Ockham's razor

The principle to explain a phenomena by the simplest hypothesis possible

Explain data with a model

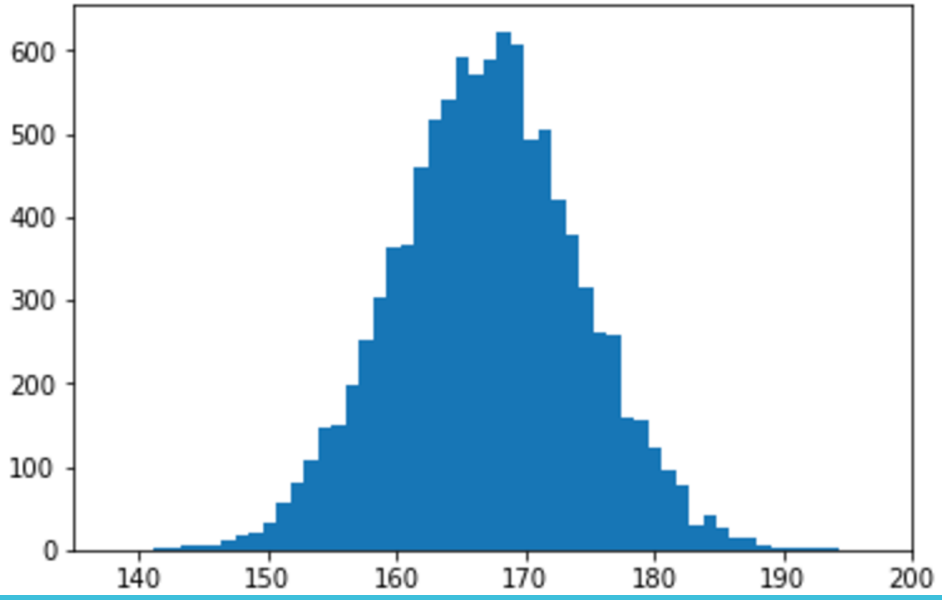


Fit the distribution of population height

- Samples of 10,000 male adults' height measurements
  - Load data from file `populationHeight.csv`

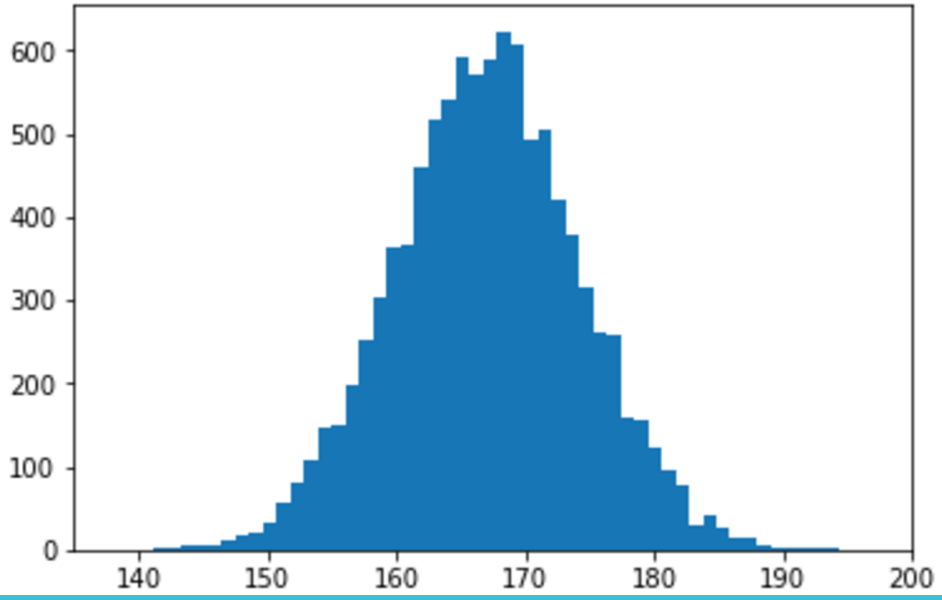
Fit the distribution of population height

- Samples of 10,000 male grownups' height measurements
  - Load data from file *`populationHeight.csv`*
  - Eyeball the distribution using *`matplotlib`*



Fit the distribution of population height

- Samples of 10,000 male grownups' height measurements
  - Load data from file `populationHeight.csv`
  - Eyeball the distribution using `matplotlib`
  - *Find a model to fit the data distribution??*



Fit the distribution of population height

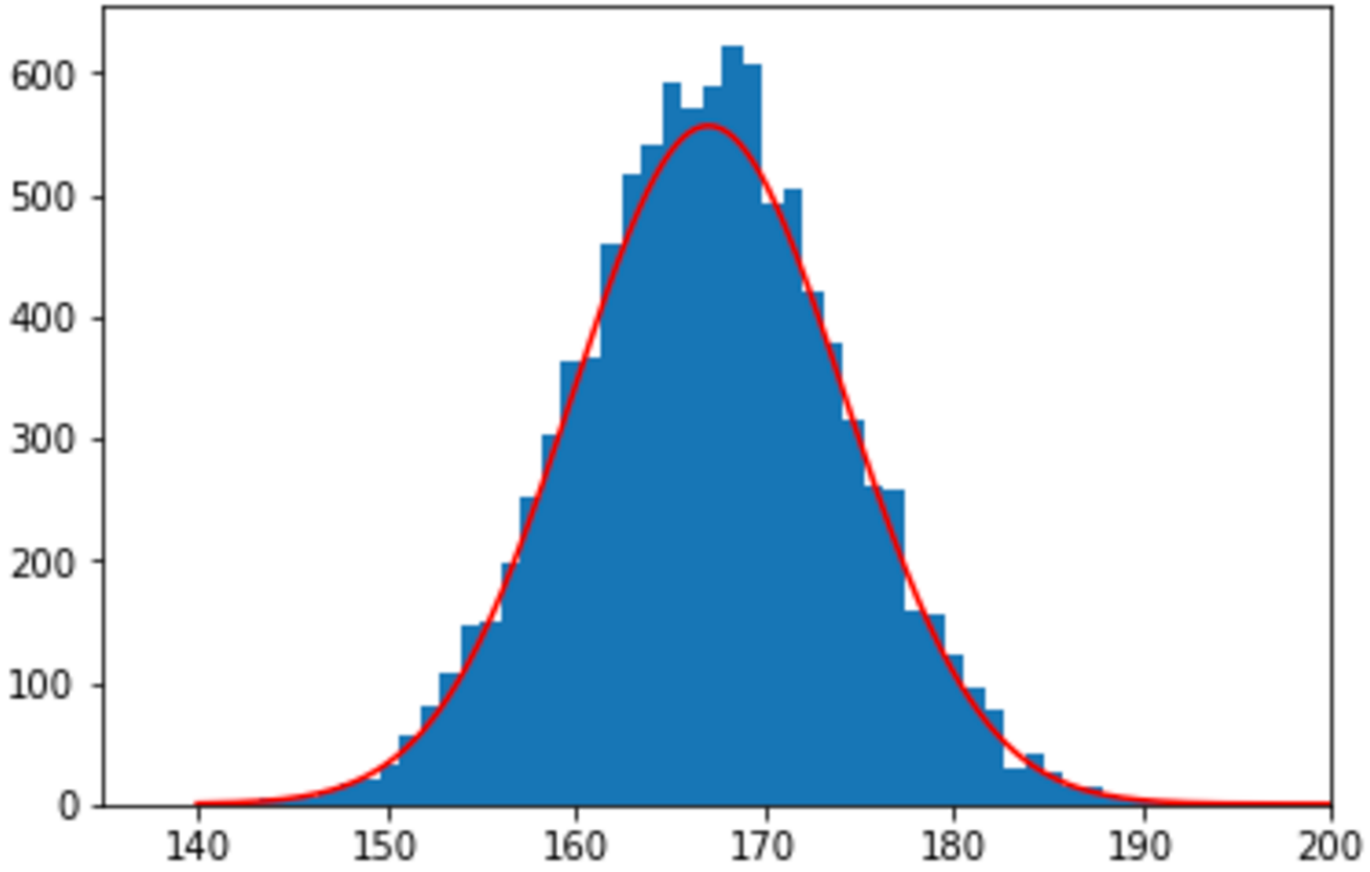
```
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('populationHeight.csv')

# choose a model: Gaussian distribution!
# estimate parameters of Gaussian
mean = data.mean()
std = data.std()

# validate your model
from math import pi
x = np.linspace(140,200,500)  #from 140 to 200, 500 pts
y = 1/((2*pi)**0.5*std)*np.exp(-(x-mean)**2/(2*std**2))

plt.hist(data, 50)
plt.plot(x,y*10000,'r-')
plt.xlabel('height'); plt.ylabel('scaled probability')
plt.show()
```

Fit the distribution of population height



Questions

- How to evaluate the quality of the estimated model?
- What if we have chosen a different/wrong model?

All models are wrong but some are useful

—George Box, statistician