Learn Bayes

2024-11-07

Table of contents

1

1 **3**

1.1 A & B 4

1.2 A|B 4

1.3 5

2 **7**

2.1 7

2.2 8

2.3 10

2.4 11

3 **13**

3.1 13

3.2 14

3.3 18

3.4 19

3.5 21

4 Prior Distributions **23**

4.1 Prior Distributions 25

4.2 Batch updating 30

5 **33**

5.1 prior 35

5.2 36

6 Minimum, Maxium, and Mixture **39**

Quarto http://allendowney.github.io/ThinkBayes2

Chapter 1

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Penguins

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/penguins.csv")
df = df.dropna()
df.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

```
df.shape
```

```
(333, 7)
```

```python
set(df.species), set(df.island), set(df.sex)
```

```
({'Adelie', 'Chinstrap', 'Gentoo'},
 {'Biscoe', 'Dream', 'Torgersen'},
 {'FEMALE', 'MALE'})
```

(‘Adelie’, ‘Chinstrap’, ‘Gentoo’) (‘FEMALE’, ‘MALE’)

A FEMALE B Adelie

1.1 A & B

$P(A\&B)$:

```python
def prob_sex_and_species(df, sex_str, species_str):
    subset = df[(df['sex'] == sex_str) & (df['species'] == species_str)]
    return len(subset) / len(df)
```

```python
prob_sex_and_species(df, sex_str='FEMALE', species_str='Adelie')
```

```
0.21921921921921922
```

1.2 A|B

$P(A|B)$ P(Female|Adelie):

```python
def prob_sex_given_species(df, sex_str, species_str):
    species_subset = df[df.species == species_str]
    sex_subset_within_species_subset = species_subset[species_subset.sex == sex_str]
    return len(sex_subset_within_species_subset)/len(species_subset)
```

```python
prob_sex_given_species(df, 'FEMALE', 'Adelie')
```

```
0.5
```

1.3

$$P(A|B) = \frac{P(A\&B)}{P(B)}$$

```python
def prob_species(df, species_str):
    subset = df[df.species == species_str]
    return len(subset)/len(df)
```

```python
prob_species(df, 'Adelie')
```

```
0.43843843843843844
```

```python
prob_sex_given_species(
    df, 'FEMALE', 'Adelie') == prob_sex_and_species(
    df, 'FEMALE', 'Adelie')/prob_species(df, 'Adelie')
```

```
True
```

$$P(A\&B) = P(B\&A)$$

$$P(A\&B) = P(A|B)P(B)$$

$$P(B\&A) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(A\&B)}{P(B)} = \frac{P(B\&A)}{P(B)} = \frac{P(A)P(B|A)}{P(B)}$$

Chapter 2

2.1

1	30	10
2	20	20

$$\frac{30}{30+20} = \frac{3}{5}$$

$$P(\ _1|\ ) = \frac{P(\ _1)P(\ |\ _1)}{P(\ )} \quad = \frac{\frac{1}{2}\cdot\frac{3}{4}}{\frac{5}{8}} = \frac{3}{5}$$

$$P(h|d) = \frac{P(h)P(d|h)}{P(d)}$$

h hypothesis d $P(h)$ prior ( ) $P(d|h)$ likelihood ( ) $P(h|d)$ posterior.

- hypothesis:
- data:

- $P(h)$ (Prior) h hypothesis $P(\ )=0.5$
- $P(d|h)$ (Likelihood) $P(\ |\ )=0.75$
- $P(d)$

$$P(\ )=P(\ _1)P(\ |\ _1)+P(\ _2)P(\ |\ _2)$$

$$P(\ )=\frac{1}{2}\cdot\frac{3}{4}+\frac{1}{2}\cdot\frac{1}{2}=\frac{5}{8}$$

- posterior: $P(h|d)$

$$P(\ _1|\ )=\frac{P(\ _1)P(\ |\ _1)}{P(\ )}$$

$$P(\ _1|\ )=\frac{\frac{1}{2}\cdot\frac{3}{4}}{\frac{5}{8}}=\frac{3}{5}$$

2.2

```python
import pandas as pd
table = pd.DataFrame(index = [" 1", " 2"])

table['prior'] = 1/2, 1/2
table
```

	prior
1	0.5
2	0.5

hypothesis data prior $p(\ _1)$ likelihood $p(\ |\ _1)$ posterior $p(\ _1|\ )$

```
table['likelihood'] = 3/4, 1/2
table
```

	prior	likelihood
1	0.5	0.75
2	0.5	0.50

$prior * likelihood$ $P(\ _1)P(\ |\ _1)$:

```
table['unnorm'] = table['prior'] * table['likelihood']
table
## unnorm   unnormalized posteriors
```

	prior	likelihood	unnorm
1	0.5	0.75	0.375
2	0.5	0.50	0.250

$p(d)$ $p(\ )$ $\frac{5}{8}$

```
table['posterior'] = table['unnorm'] / (5/8)
table
```

	prior	likelihood	unnorm	posterior
1	0.5	0.75	0.375	0.6
2	0.5	0.50	0.250	0.4

3/5 $p(\ _2|\ ) = \frac{2}{5}$

`unnorm` `posterior` `unnorm` `unnorm`

```
table['posterior_again'] = table['unnorm'] / table['unnorm'].sum()
table
```

	prior	likelihood	unnorm	posterior	posterior_again
1	0.5	0.75	0.375	0.6	0.6
2	0.5	0.50	0.250	0.4	0.4

2.3

6 8 12 A 1 A B 6

$p(dice = 6|number = 1)$

- prior: $p(dice = 6)$
- likelihood: $p(number = 1|dice = 6)$
- data: $p(number = 1)$
- posterior: $p(dice = 6|number = 1)$

```
table2 = pd.DataFrame(index = [6, 8, 12])
from fractions import Fraction
table2['prior'] = Fraction(1,3)
table2['likelihood'] = Fraction(1,6), Fraction(1,8), Fraction(1, 12)
table2
```

	prior	likelihood
6	1/3	1/6
8	1/3	1/8
12	1/3	1/12

```
table2['unnorm'] = table2['prior'] * table2['likelihood']
table2
```

	prior	likelihood	unnorm
6	1/3	1/6	1/18
8	1/3	1/8	1/24
12	1/3	1/12	1/36

"unnorm" posterior

```
table2['posterior'] = table2['unnorm']/table2['unnorm'].sum()
table2
```

	prior	likelihood	unnorm	posterior
6	1/3	1/6	1/18	4/9
8	1/3	1/8	1/24	1/3
12	1/3	1/12	1/36	2/9

2.4

A
C A B

1.
2.
3.

A B

1/2

hypothesis prior $p($ $)$ A, B, C

Data: C

Likelihood $p(\text{Data}|h)$

Posterior $p(h|\text{Data})$

prior

```
table3 = pd.DataFrame(index = ['A', 'B', 'C'])
table3['prior'] = Fraction(1, 3)
table3
```

	prior
A	1/3
B	1/3
C	1/3

Likelihood Likelihood hypothesis data

hypothesis A, C 1/2

hypothesis B A C data 1

hypothesis C C 0

```
table3['likelihood'] = Fraction(1, 2), 1, 0
table3['unnorm'] = table3['prior'] * table3['likelihood']
table3['posterior'] = table3['unnorm']/(table3['unnorm'].sum())
table3
```

	prior	likelihood	unnorm	posterior
A	1/3	1/2	1/6	1/3
B	1/3	1	1/3	2/3
C	1/3	0	0	0

B

Chapter 3

3.1

$$\text{Posterior distribution} \propto \text{Prior distribution} \times \text{Likelihood distribution}$$

$\propto$ proportional to a b prior likelihood posterior

Section 2.1

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

```python
def normalize_array(arr):
    return np.array([i/sum(arr) for i in arr])
```

```python
prior = np.array([0.5, 0.5])
prior
```

```
array([0.5, 0.5])
```

```python
# Likelihood:  hypothesis     data
#   hypothesis    data
likelihood_red = np.array([0.75, 0.5])
posterior = normalize_array(prior * likelihood_red)
posterior
```

```
array([0.6, 0.4])
```

prior likelihood
posterior

prior 0.6 0.4

prior posterior likelihood

```
posterior *= likelihood_red
posterior = normalize_array(posterior)
posterior
```

```
array([0.69230769, 0.30769231])
```

prior posterior Likelihood `likelihood_white`

```
likelihood_white = np.array([0.25, 0.5])
posterior *= likelihood_white
posterior = normalize_array(posterior)
posterior
```

```
array([0.52941176, 0.47058824])
```

3.2

101 0 100

- 0 0%
- 1 1%
- 2 2%

…

- 99 99%
- 100 100%

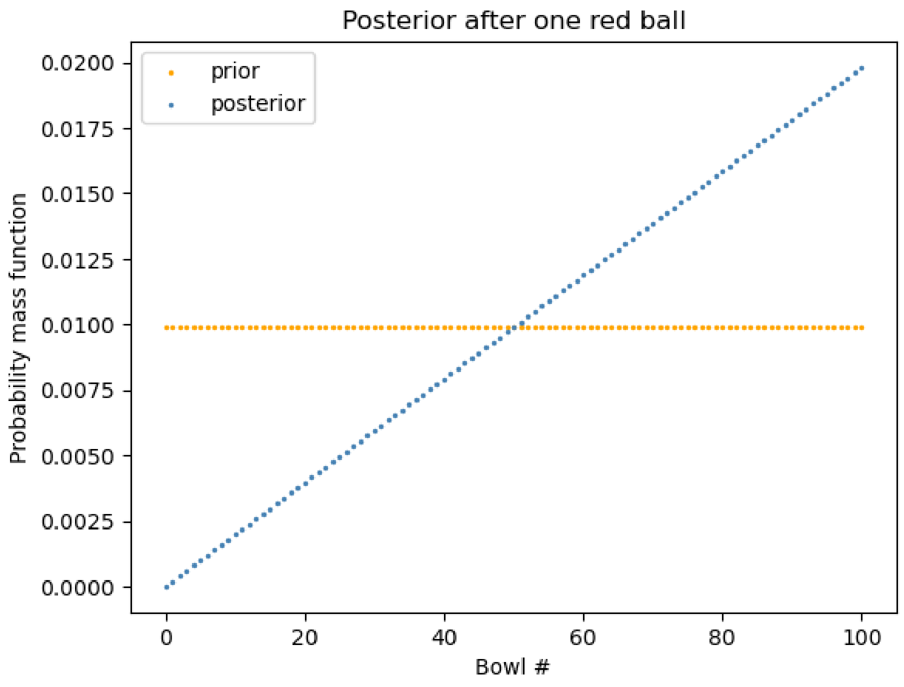
x

- hypothesis: x
- data:
- prior: $p(h)$
- likelihood: $p(d|h)$
- posterior: $p(h|d)$

```
n = 101
# uniform prior:
all_ones = [1]*n
prior = normalize_array(all_ones)
# likelihood array:
likelihood_red = np.array([i/(n-1) for i in range(n)])
posterior = normalize_array(prior * likelihood_red)
posterior[0:5]
```

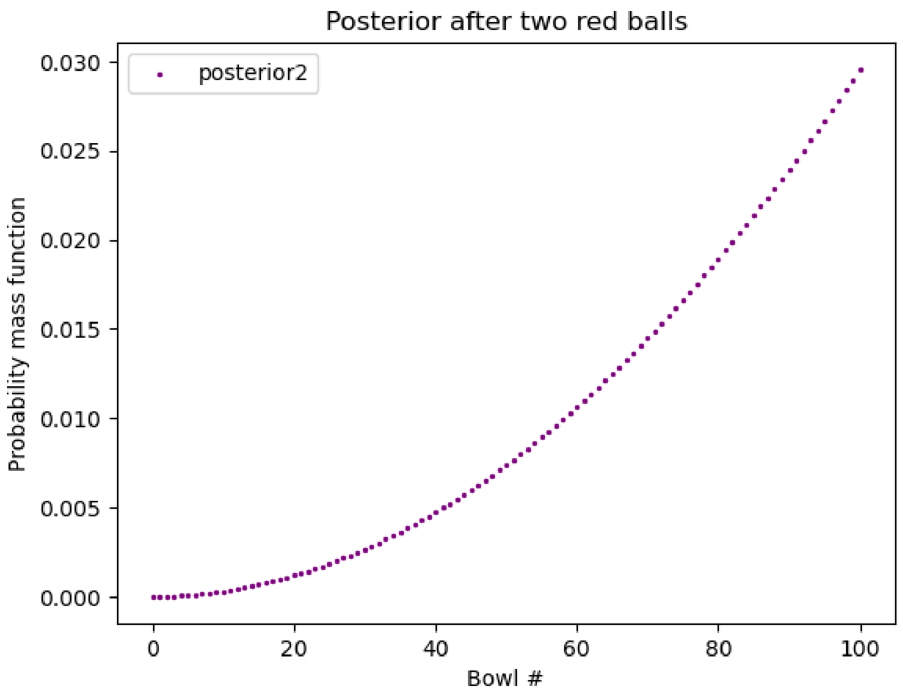
```
array([0.        , 0.00019802, 0.00039604, 0.00059406, 0.00079208])
```

```
x_axis = range(n)
plt.scatter(x = x_axis, y = prior,
            label="prior", color="orange", s = 2)
plt.scatter(x = x_axis, y = posterior,
            label="posterior", color="steelblue", s = 2)
plt.xlabel("Bowl #")
plt.ylabel("Probability mass function")
plt.legend()
plt.title("Posterior after one red ball")
plt.show()
```



x

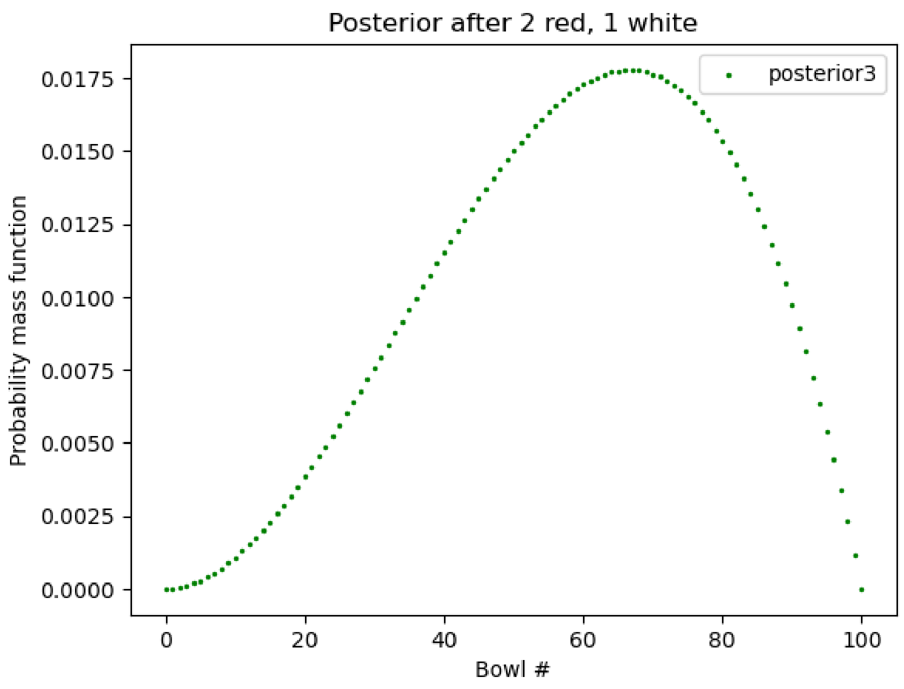
```
posterior2 = normalize_array(posterior * likelihood_red)
plt.scatter(x = x_axis, y = posterior2,
            label="posterior2", color="purple", s = 2)
plt.xlabel("Bowl #")
plt.ylabel("Probability mass function")
plt.legend()
plt.title("Posterior after two red balls")
plt.show()
```



x

prior posterior prior

```
likelihood_white = np.array([1 - x for x in likelihood_red])
posterior3 = normalize_array(posterior2 * likelihood_white)
plt.scatter(x = x_axis, y = posterior3,
            label="posterior3", color="green", s = 2)
plt.xlabel("Bowl #")
plt.ylabel("Probability mass function")
plt.legend()
plt.title("Posterior after 2 red, 1 white")
plt.show()
```



PMF (probability mass function) “maximum a posterior probability” (MAP).

```
max_index = np.argmax(posterior3)
print("     #", max_index)
```

```
# 67
```

3.3

posterior3

```
all_ones = [1]*n
prior = normalize_array(all_ones)
posterior = normalize_array(prior * likelihood_red)
posterior2 = normalize_array(posterior * likelihood_red)
posterior3 = normalize_array(posterior2 * likelihood_white)
```

normalize_array()

```python
all_ones = [1]*n
prior = normalize_array(all_ones)
```

`prior`, `likelihood_red`, `likelihood_white`, `posterior` (array)

`--` `--`

posterior3

```
posterior3_new = all_ones * likelihood_white * likelihood_red * likelihood_red #
posterior3_new = normalize_array(posterior3_new) #
```

```python
posterior3[0:5]
```

```
array([0.00000000e+00, 1.18811881e-05, 4.70447045e-05, 1.04770477e-04,
       1.84338434e-04])
```

```python
posterior3_new = all_ones * likelihood_white * likelihood_red * likelihood_red
posterior3_new = normalize_array(posterior3_new)
posterior3_new[0:5]
```

```
array([0.00000000e+00, 1.18811881e-05, 4.70447045e-05, 1.04770477e-04,
       1.84338434e-04])
```

```python
sum(np.isclose(posterior3, posterior3_new)) == len(prior)
```

```
True
```

`--` `--` `--`

3.4

$n + 1$

- 0 0/n
- 1 1/n

- 2 2/n

  …
- n n/n

T

m r w

T

```python
def update_bowls_pmf(n, r, w):
    """
    n:
    r:
    w:
    """
    priors = np.array([1]*n)
    priors = normalize_array(priors)
    likelihood_red = np.array([i/(n-1) for i in range(n)])
    likelihood_white = np.array([1- i for i in likelihood_red])
    likelihood = {
        "red": likelihood_red,
        "white": likelihood_white
    }
    dataset = ["red"]*r + ["white"]*w
    for data in dataset:
        priors *= likelihood[data]

    posterior = normalize_array(priors)
    return posterior
```

```python
new_posterior = update_bowls_pmf(n=101, r=2, w=1)
new_posterior[0:5]
```

```
array([0.00000000e+00, 1.18811881e-05, 4.70447045e-05, 1.04770477e-04,
       1.84338434e-04])
```

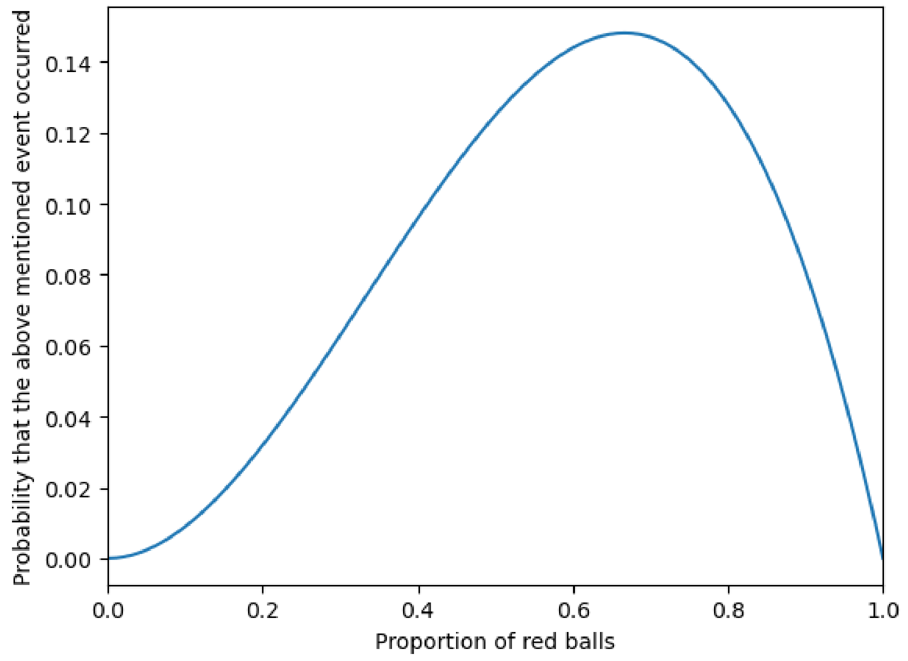
```python
#       - -
new_posterior = update_bowls_pmf(n=101, r=2, w=1)
sum(np.isclose(new_posterior, posterior3_new)) == len(prior)
```

```
True
```

3.5

```python
def y(x):
    #     x
    return x**2*(1-x)
```

```python
x = np.linspace(0, 1, 100)
plt.plot(x, y(x))
plt.xlabel("Proportion of red balls")
plt.ylabel("Probability that the above mentioned event occurred")
plt.xlim(0, 1)
plt.show()
```



```python
from scipy.optimize import minimize_scalar
def y(x):
    return -x**2*(1-x)
result = minimize_scalar(y, bounds=(0,1), method="bounded")
result
```

```
 message: Solution found.
 success: True
  status: 0
     fun: -0.14814814814787028
       x: 0.666666139518174
     nit: 10
    nfev: 10
```

```python
#
max_value = -result.fun
optimal_x = result.x
optimal_x, max_value
```

```
(0.666666139518174, 0.14814814814787028)
```

66.7% $\frac{2}{3}$

67% 67 67%

Chapter 4

Prior Distributions

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import binom
```

250 140 110

Figure 4.1:

Chapter 3

$n + 1$

- 0 0/n
- 1 1/n

- 2 2/n

…

- n n/n

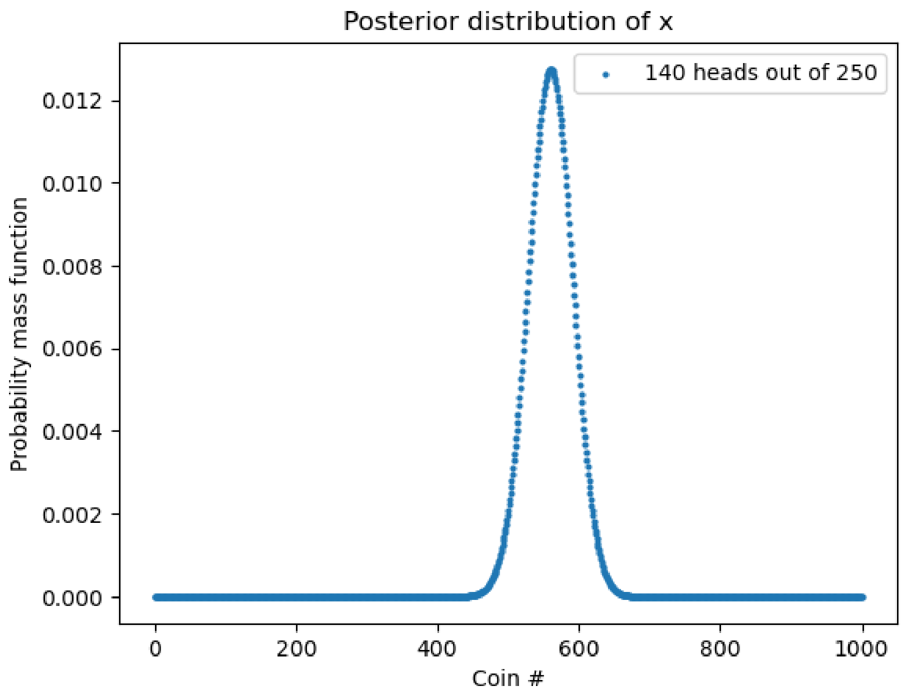
250 140 110

uniform prior

```python
def normalize_array(arr):
    return np.array([i/sum(arr) for i in arr])

#       update_bowls_pmf
def update_coins_pmf(n, h, t):
    """
    n:
    h:
    t:
    """
    prior = np.array([1]*n)
    prior = normalize_array(prior)
    likelihood_head = np.array([i/(n-1) for i in range(n)])
    likelihood_tail = np.array([1- i for i in likelihood_head])
    likelihood = {
        "head": likelihood_head,
        "tail": likelihood_tail
    }
    dataset = ["head"]*h + ["tail"]*t
    posterior = prior.copy()
    for data in dataset:
        posterior *= likelihood[data]
    return normalize_array(posterior)
```

```python
posterior = update_coins_pmf(1001, 140, 110)
```

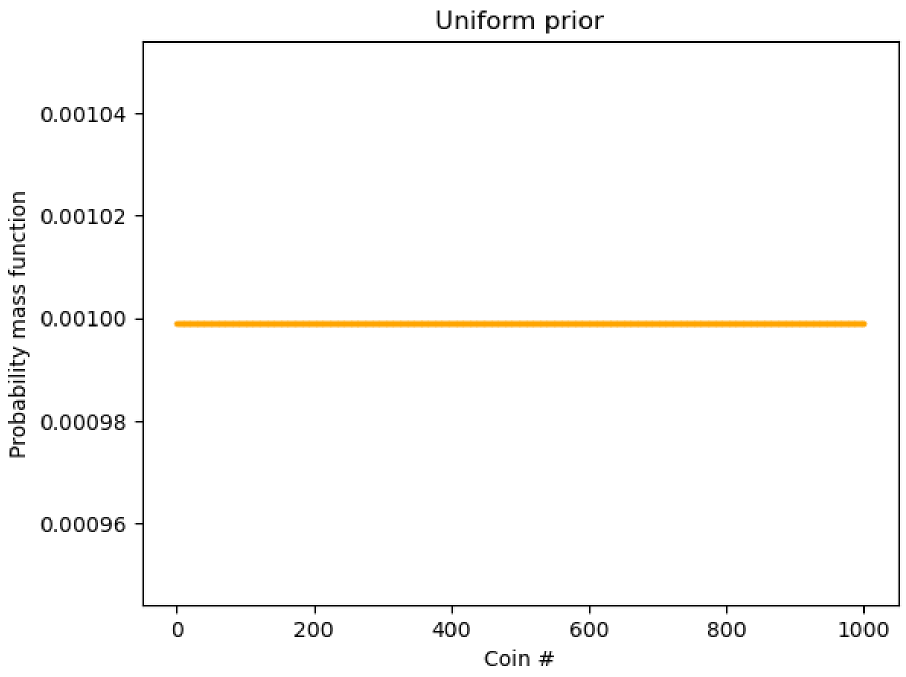
```python
df = pd.DataFrame(posterior, columns=['probs'])
df = df.reset_index(names = "Coin #")
df.plot.scatter(x = "Coin #", y="probs",
                s = 4,
                label = "140 heads out of 250")
plt.ylabel("Probability mass function")
plt.legend()
plt.title("Posterior distribution of x")
plt.show()
```



4.1 Prior Distributions

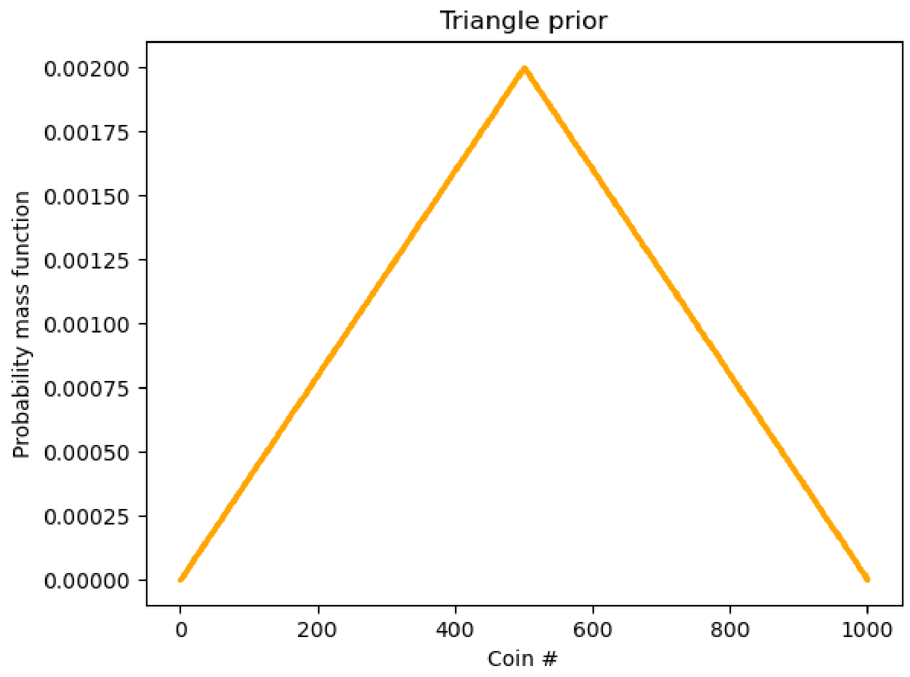
uniform prior

```
n = 1001
prior = np.array([1]*n)
uniform = normalize_array(prior)
x_axis = range(n)
plt.scatter(x = x_axis, y = uniform,
            label="prior", color="orange", s = 2)
plt.xlabel("Coin #")
plt.ylabel("Probability mass function")
plt.title("Uniform prior")
plt.show()
```



prior uniform

```
ramp_up = np.arange(500)
ramp_down = np.arange(500, -1, -1)
prior = np.append(ramp_up, ramp_down)
triangle = normalize_array(prior)
plt.scatter(x = x_axis, y = triangle,
            label="prior", color="orange", s = 2)
plt.xlabel("Coin #")
plt.ylabel("Probability mass function")
plt.title("Triangle prior")
plt.show()
```

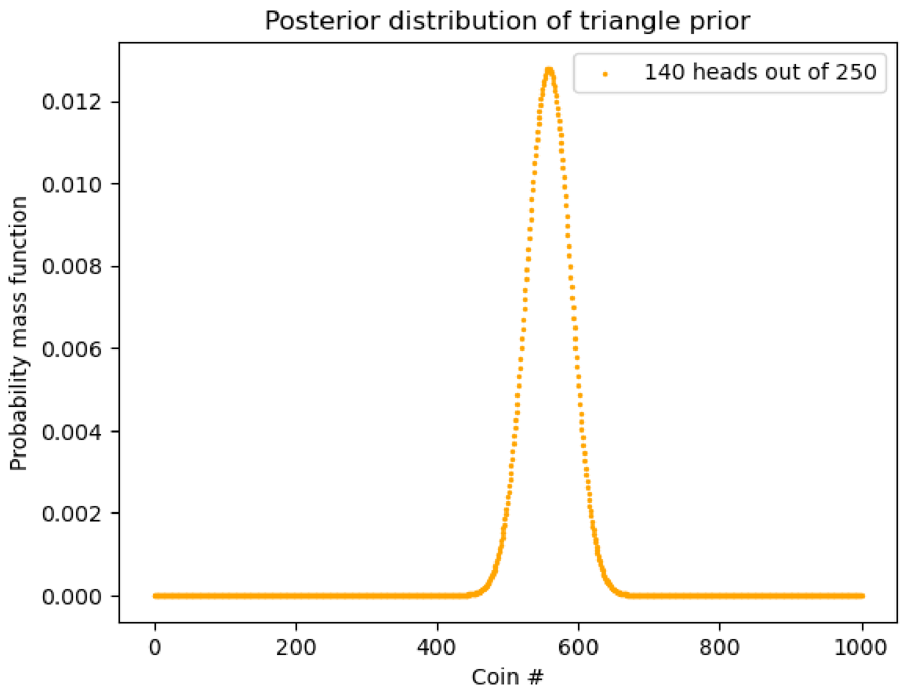


prior

```
#       update_coins_pmf      prior
def update_coins_pmf(n, h, t, prior):
    """
    n:
    h:
    t:
    prior: a normalized array
    """
    likelihood_head = np.array([i/(n-1) for i in range(n)])
    likelihood_tail = np.array([1- i for i in likelihood_head])
    likelihood = {
        "head": likelihood_head,
        "tail": likelihood_tail
    }
    dataset = ["head"]*h + ["tail"]*t
    posterior = prior.copy()
    for data in dataset:
        posterior *= likelihood[data]
    return normalize_array(posterior)
```

```
n = 1001
h = 140
t = 110
ramp_up = np.arange(500)
ramp_down = np.arange(500, -1, -1)
prior = np.append(ramp_up, ramp_down)
prior = normalize_array(prior)

posterior = update_coins_pmf(n, h, t, prior)
```

```
plt.scatter(x = x_axis, y = posterior,
            label="140 heads out of 250", color="orange", s = 2)
plt.xlabel("Coin #")
plt.ylabel("Probability mass function")
plt.title("Posterior distribution of triangle prior")
plt.legend()
plt.show()
```



```
max_index = np.argmax(posterior)
print("    Coin #", max_index)
```

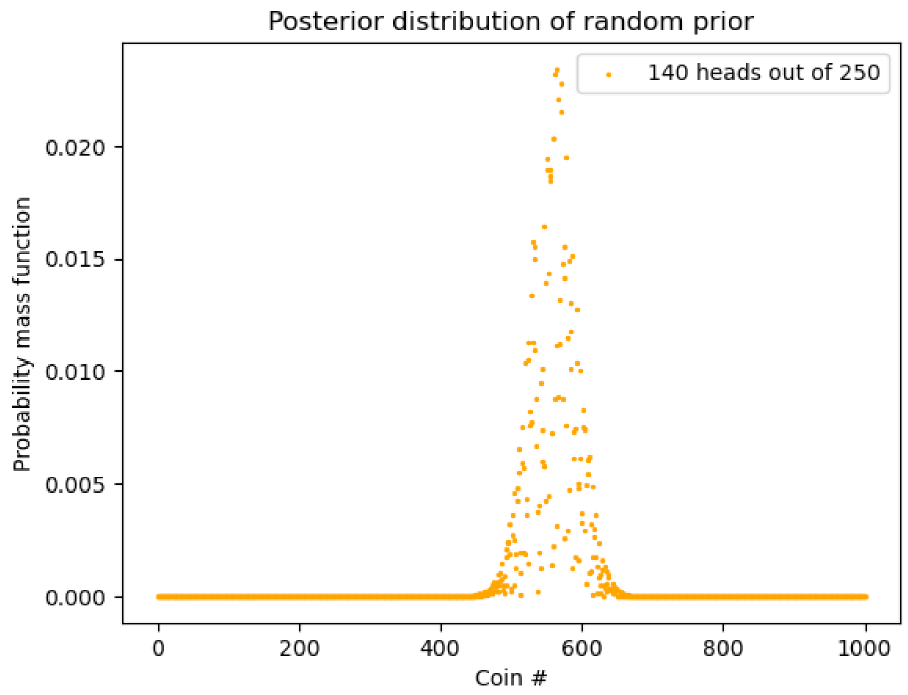
```
    Coin # 558
```

uniform prior prior posterior

prior

```
random_prior_values = np.random.rand(n)
random_prior = normalize_array(random_prior_values)

posterior = update_coins_pmf(n, h, t, prior=random_prior)
```

```
plt.scatter(x = x_axis, y = posterior,
            label="140 heads out of 250", color="orange", s = 2)
plt.xlabel("Coin #")
plt.ylabel("Probability mass function")
plt.title("Posterior distribution of random prior")
plt.legend()
plt.show()
```



```
max_index = np.argmax(posterior)
print("    Coin #", max_index)
```

```
Coin # 565
```

prior posterior

4.2 Batch updating

posterior

```python
for data in dataset:
    posterior *= likelihood[data]
```

250 140 110 Data

- hypothesis:
- data: 250 140 110
- prior: $p(h)$
- likelihood: $p(d|h)$
- posterior: $p(h|d)$

prior likelihood posterior Prior uniform prior triangle prior (array) Likelihood binomial distribution

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

`scipy.stats.binom`

`scipy.stats.binom.pmf(k, n, p)`

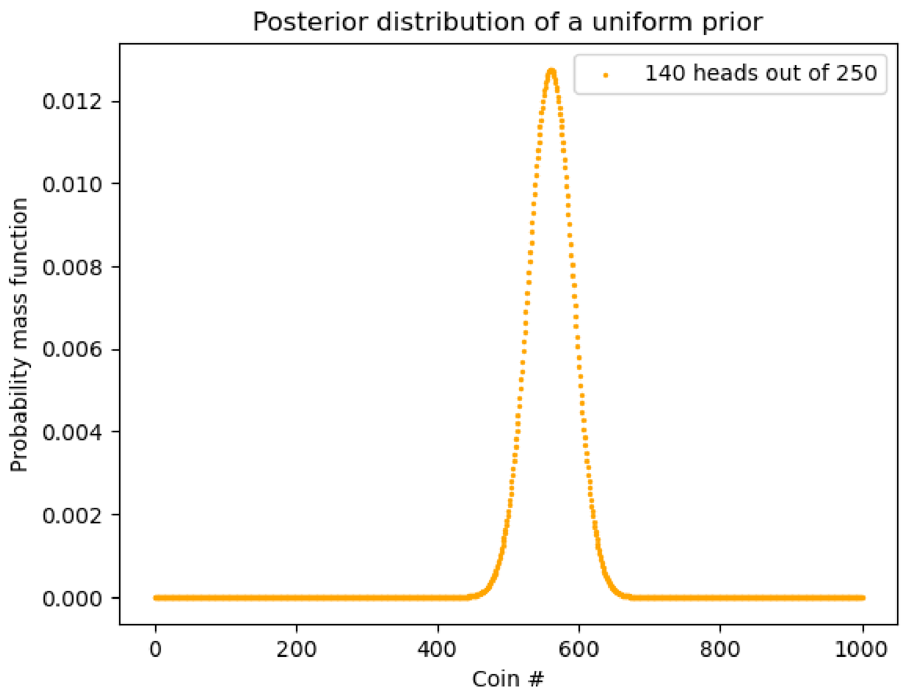
k n p p p p

p prior

```python
def update_binom(n, heads, tosses, prior):
    """
    heads: number of heads
    tosses: total tosses
    prior: prior distribution; should be a empiricaldist.pmf object (a Series)
    """
    # 0/n, 1/n, 2/n ...
    likelihood_head = np.array([i/(n-1) for i in range(n)])
    coin_head_probabilities = likelihood_head
    likelihood = binom.pmf(k = heads, n = tosses, p = coin_head_probabilities)
    posterior = prior.copy()
    posterior *= likelihood
    return normalize_array(posterior)
```

```
# n: number of coins
n = 1001
tosses = 250
# number of heads out of 250 tosses
heads = 140
prior = np.array([1]*n)
uniform = normalize_array(prior)
posterior = update_binom(n, heads, tosses, uniform)
```

```
plt.scatter(x = x_axis, y = posterior,
            label="140 heads out of 250", color="orange", s = 2)
plt.xlabel("Coin #")
plt.ylabel("Probability mass function")
plt.title("Posterior distribution of a uniform prior")
plt.legend()
plt.show()
```



Chapter 5

1 n 60

Figure 5.1:

1 n 1 1 2 2 n n 60

- hypothesis:
- data: 60
- prior: $p(h)$
- likelihood: $p(d|h)$
- posterior: $p(h|d)$

prior   1000   uniform probabilities

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```python
def normalize_array(arr):
    return np.array([i/sum(arr) for i in arr])
```

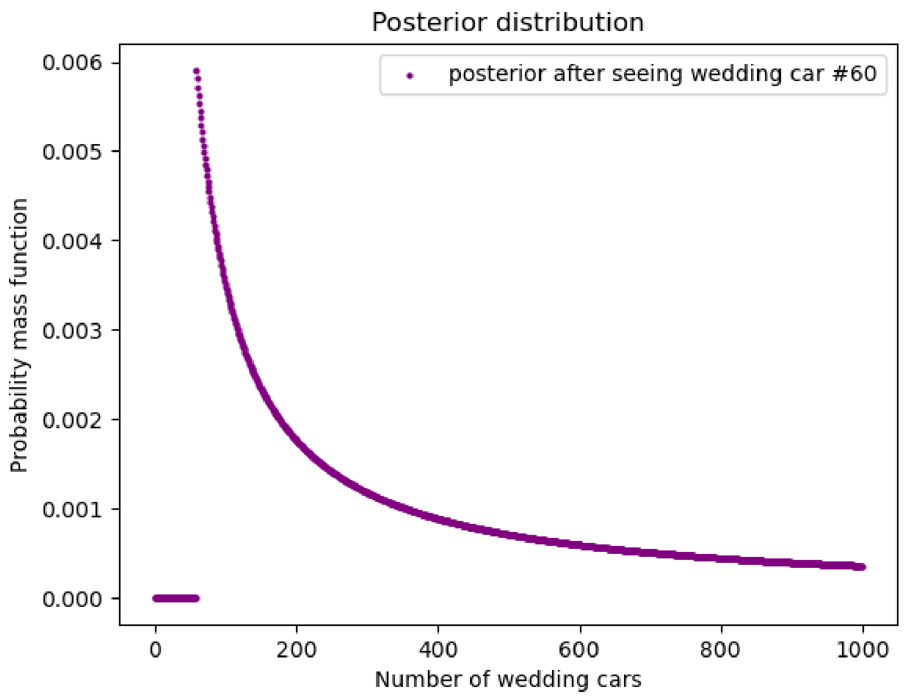
```python
n = 1000
prior = np.array([1]*n)
prior = normalize_array(prior)
```

```python
def update_posterior(n, prior, data):
    '''to upate posterior in this example
    prior: empiricaldist.Pmf object
    data: e.g., 60
    '''
    # likelihood is 1/hypos because, say num 60 is only 1/100 chance in dice #100
    hypos = np.arange(1, n+1)
    likelihood = 1 / hypos
    # dice #1 to dice #data are impossible
    likelihood[data > hypos] = 0
    posterior = prior.copy()
    posterior *= likelihood
    return normalize_array(posterior)
```

```python
posterior = update_posterior(n, prior, data = 60)
```

```python
def draw_posterior(posterior, xlabel, c, legend_text, s=4):
    """
    posterior: posterior, a empiricaldist.Pmf object
    xlabel: xlabel you want to see in the plot
    c: color of the dots
    legend_text: text for the legend
    s: size of the dots
    """
    df = pd.DataFrame(posterior, columns=['probs'])
    df = df.reset_index(names = xlabel)
    df.plot.scatter(
        x = xlabel,
        y = 'probs',
        color = c,
        s = s,
        label = legend_text
    )
    plt.ylabel("Probability mass function")
    plt.legend()
    plt.title("Posterior distribution")
    plt.show()
```

```
draw_posterior(
    posterior, 'Number of wedding cars',
    c = 'purple', legend_text="posterior after seeing wedding car #60")
```



```
max_index = np.argmax(posterior)
print("    Wedding car #", max_index + 1)
```

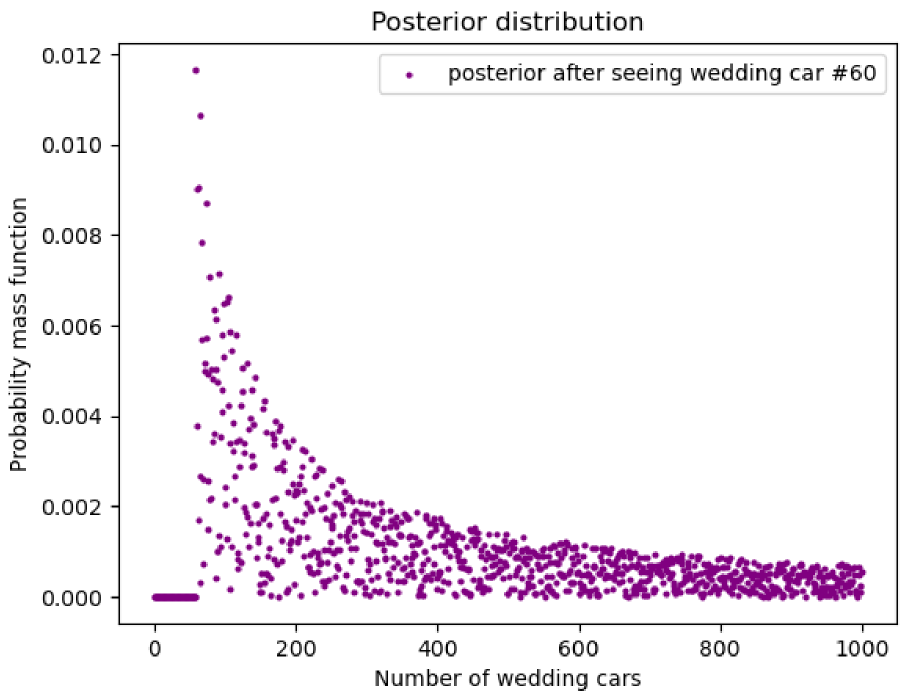
```
Wedding car # 60
```

5.1 prior

prior

```
random_prior_values = np.random.rand(n)
random_prior = normalize_array(random_prior_values)
posterior = update_posterior(n, random_prior, data = 60)
```

```
draw_posterior(
    posterior, 'Number of wedding cars',
    c = 'purple', legend_text="posterior after seeing wedding car #60")
```



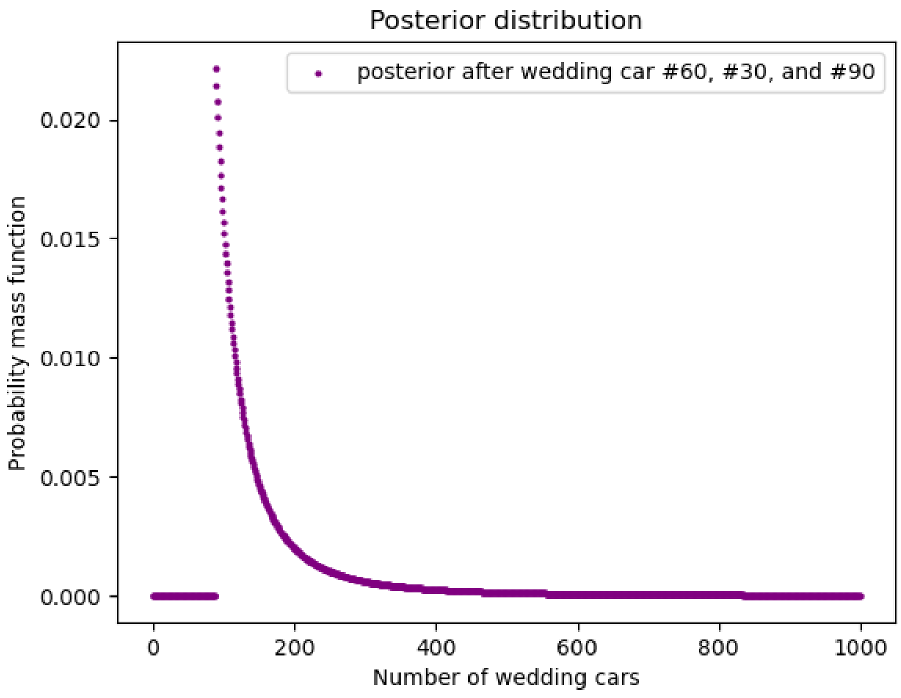
prior posterior

5.2

60 30 90 60 30
90 posterior prior posterior likelihood

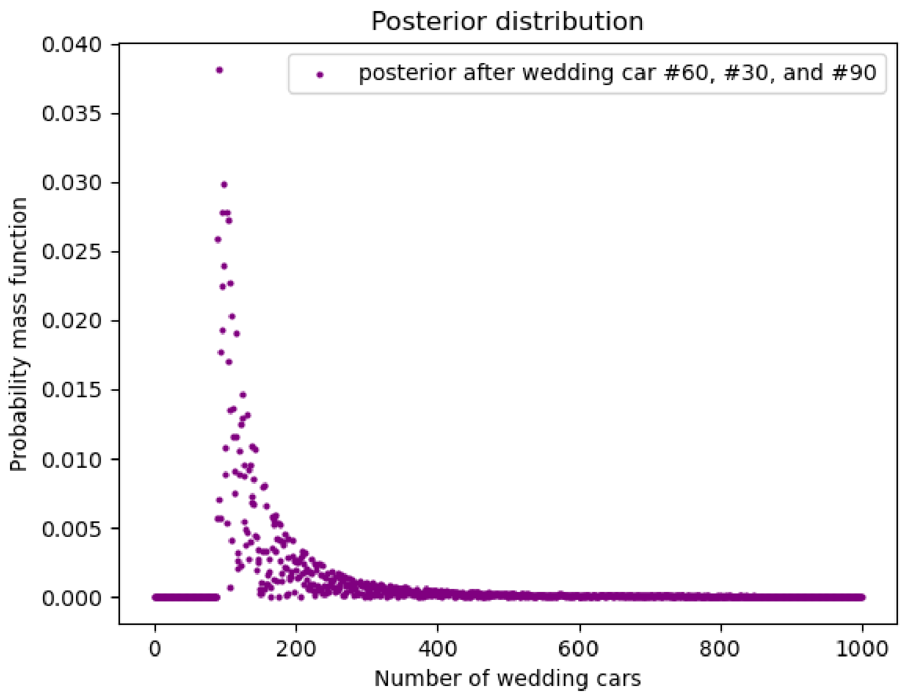
prior posterior

```
dataset = [60, 30,90]
posterior = prior.copy()
for data in dataset:
    posterior = update_posterior(n, posterior, data=data)
draw_posterior(
    posterior, 'Number of wedding cars',
    c = 'purple', legend_text="posterior after wedding car #60, #30, and #90")
```



prior

```
dataset = [60, 30,90]
posterior = random_prior.copy()
for data in dataset:
    posterior = update_posterior(n, posterior, data=data)
draw_posterior(
    posterior, 'Number of wedding cars',
    c = 'purple', legend_text="posterior after wedding car #60, #30, and #90")
```



(60) uniform prior

Chapter 6

Minimum, Maxium, and Mixture

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import binom
```

Cumulative Distriburtion Function (CDF)   Chapter 4

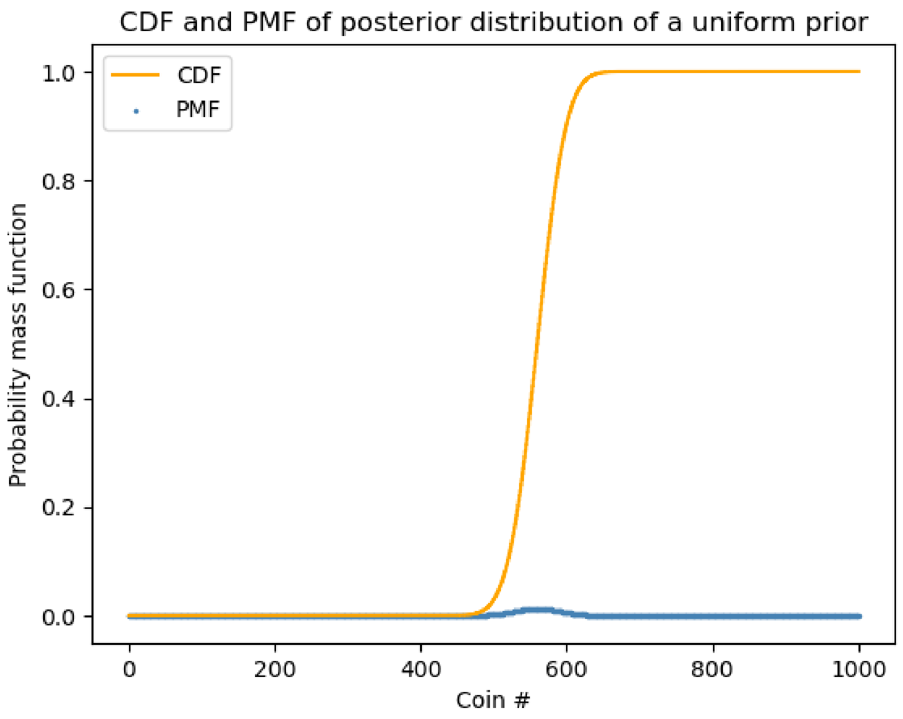
```python
def normalize_array(arr):
    return np.array([i/sum(arr) for i in arr])

def update_binom(heads, tosses, prior):
    """
    heads: number of heads
    tosses: total tosses
    prior: prior distribution; should be a empiricaldist.pmf object (a Series)
    """
    # 0/n, 1/n, 2/n ...
    likelihood_head = np.array([i/(n-1) for i in range(n)])
    coin_head_probabilities = likelihood_head
    likelihood = binom.pmf(k = heads, n = tosses, p = coin_head_probabilities)
    posterior = prior.copy()
    posterior *= likelihood
    return normalize_array(posterior)

# n: number of coins
```

```
n = 1001
x_axis = range(n)
tosses = 250
# number of heads out of 250 tosses
heads = 140
prior = np.array([1]*n)
uniform = normalize_array(prior)
posterior = update_binom(heads, tosses, uniform)
```

```
def get_cdf(arr):
    """Get cumulative distribution function
    """
    total_sum = np.sum(arr)
    res = []
    sum = 0
    for x in arr:
        sum += x
        # normaize to make sure the max in res is 1
        res.append(sum/total_sum)
    return res
```

```
cdf = get_cdf(posterior)
```

```
plt.step(x=x_axis, y=cdf, label="CDF", color='orange', where='post')
# plt.scatter(x=x_axis, y = cdf, label="CDF", color = 'orange', s = 2)
plt.scatter(x = x_axis, y = posterior,
            label="PMF", color="steelblue", s = 2)
plt.xlabel("Coin #")
plt.ylabel("Probability mass function")
plt.title("CDF and PMF of posterior distribution of a uniform prior")
plt.legend()
plt.show()
```


CDF and PMF of posterior distribution of a uniform prior
CDF
PMF
Probability mass function
Coin #
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000