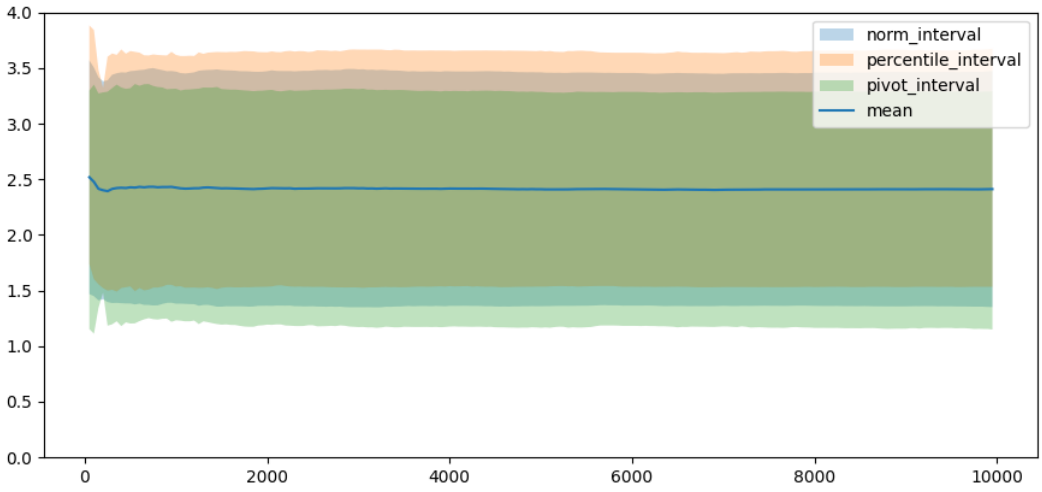
Exersice 6

8.2

三种置信区间：

- The Normal Interval: (1.3520851570311345, 3.4704111747696205)
- Pivotal Intervals: (1.5345884553366111, 3.6728612103969427)
- Percentile Intervals: (1.1496351214038123, 3.2879078764641436)

三种置信区间随bootstrap次数的变化：



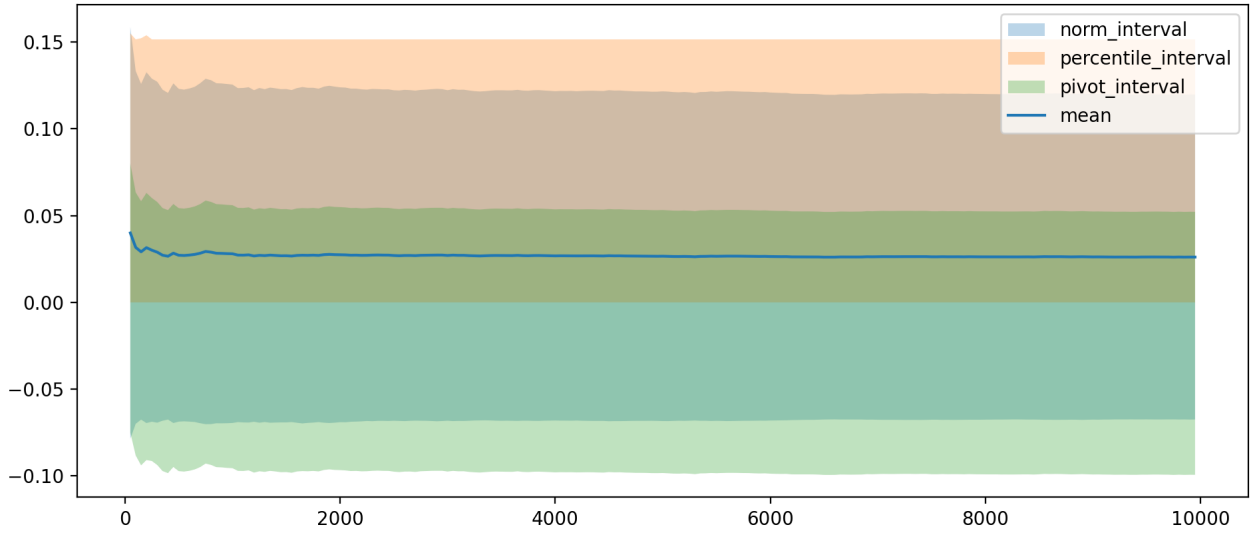
8.3

(i) Normal interval: (-0.06748383058986744, 0.11965159927169523)
Length: 0.18713542986156267

(ii) Percentile interval: (0.0, 0.15146199659020868)
Length: 0.15146199659020868

(iii) Pivotal interval: (-0.09929422790838088, 0.052167768681827785)
Length: 0.15146199659020868

三种置信区间随bootstrap次数的变化：



8.4

对n个真实样本进行bootstrap，可以理解为如下的过程：

1. 准备n个空位，每个空位代表bootstrap采样得到的一个样本
2. 将n个空位分为n组，允许存在没有空位的组
3. 将第i组的所有空位填充为第i个真实样本

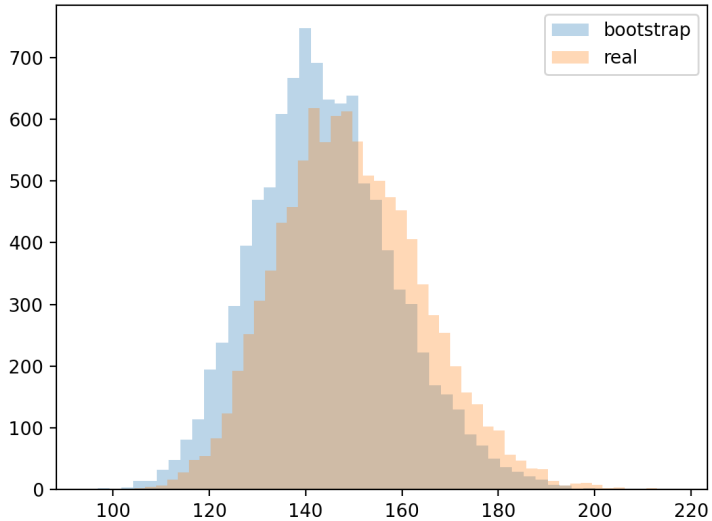
也就是说，bootstrap所有可能得采样结果数，等于将n个空位分为n组的方案数，即$C_{2n-1}^{n-1} = C_{2n-1}^{n}$。

8.6

(a) $\widehat{se} = 28.82$

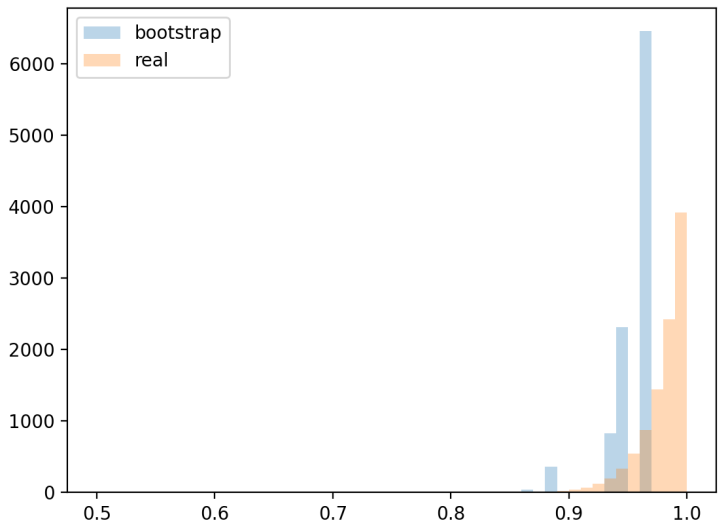
95%置信区间为: 115.324, 172.965

(b) bootstrap采样和真实采样估计的直方图如下



8.7

(a) bootstrap采样和真实采样估计的直方图如下



代码

8.2

```python
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

def mu_hat(X):
    return np.mean(X)

def sigma_hat(X):
    return np.std(X)

def F_hat(X):
    return np.mean((X-mu_hat(X))**3)/(sigma_hat(X)**3)

real_X = np.exp(np.random.normal(0, 1, (50,)))
def bootstrap(real_X, B=10000):
    F_hat_ = np.zeros(B)
    for b in range(B):
        X = np.random.choice(real_X, len(real_X), replace=True)
        F_hat_[b] = F_hat(X)
    return F_hat_

def norm_interval(F_hat_, alpha=0.05):
    se = np.std(F_hat_)
    mean = np.mean(F_hat_)
    z = norm.ppf(1-alpha/2)
    return mean - z*se, mean + z*se
```

```
27
28  def percentile_interval(F_hat, alpha=0.05):
29        return np.percentile(F_hat, 100*alpha/2), np.percentile(F_hat, 100*(1-
    alpha/2))
30
31  def pivot_interval(F_hat_, alpha=0.05):
32        return 2*np.mean(F_hat_) - np.percentile(F_hat_, 100*(1-alpha/2)),
    2*np.mean(F_hat_) - np.percentile(F_hat_, 100*alpha/2)
33
34  F_hat_ = bootstrap(real_X)
35  print(np.mean(F_hat_))
36  print(norm_interval(F_hat_))
37  print(percentile_interval(F_hat_))
38  print(pivot_interval(F_hat_))
39
40  def plot_intervals(interval_func: callable):
41        interval_upper = []
42        interval_lower = []
43        for end in range(50, 10000, 50):
44              upper, lower = interval_func(F_hat_[:end])
45              interval_upper.append(upper)
46              interval_lower.append(lower)
47        plt.fill_between(range(50, 10000, 50), interval_upper, interval_lower,
    alpha=0.3, label=interval_func.__name__)
48
49  def plot_mean():
50        mean = []
51        for end in range(50, 10000, 50):
52              mean.append(np.mean(F_hat_[:end]))
53        plt.plot(range(50, 10000, 50), mean, label='mean')
54
55  plot_intervals(norm_interval)
56  plot_intervals(percentile_interval)
57  plot_intervals(pivot_interval)
58  plot_mean()
59
60  plt.legend()
61  plt.ylim(0, 4)
62  plt.show()
```

8.3

```
1  import numpy as np
2  from scipy.stats import norm
3  import matplotlib.pyplot as plt
4
5  def F_hat(X):
```

```
6       return (np.percentile(X, 0.75) - np.percentile(X, 0.25))/1.34
7
8   def bootstrap(real_X, B=10000):
9       F_hat_ = np.zeros(B)
10      for b in range(B):
11          X = np.random.choice(real_X, len(real_X), replace=True)
12          F_hat_[b] = F_hat(X)
13      return F_hat_
14
15  def norm_interval(F_hat_, alpha=0.05):
16      se = np.std(F_hat_)
17      mean = np.mean(F_hat_)
18      z = norm.ppf(1-alpha/2)
19      return mean - z*se, mean + z*se
20
21  def percentile_interval(F_hat, alpha=0.05):
22      return np.percentile(F_hat, 100*alpha/2), np.percentile(F_hat, 100*(1-
    alpha/2))
23
24  def pivot_interval(F_hat_, alpha=0.05):
25      return 2*np.mean(F_hat_) - np.percentile(F_hat_, 100*(1-alpha/2)),
    2*np.mean(F_hat_) - np.percentile(F_hat_, 100*alpha/2)
26
27  real_X = np.random.standard_t(3, (25,))
28  F_hat_ = bootstrap(real_X)
29
30  print(np.mean(F_hat_))
31  print(norm_interval(F_hat_))
32  print(pivot_interval(F_hat_))
33  print(percentile_interval(F_hat_))
34
35  def plot_intervals(interval_func: callable):
36      interval_upper = []
37      interval_lower = []
38      for end in range(50, 10000, 50):
39          upper, lower = interval_func(F_hat_[:end])
40          interval_upper.append(upper)
41          interval_lower.append(lower)
42      plt.fill_between(range(50, 10000, 50), interval_upper, interval_lower,
    alpha=0.3, label=interval_func.__name__)
43
44  def plot_mean():
45      mean = []
46      for end in range(50, 10000, 50):
47          mean.append(np.mean(F_hat_[:end]))
48      plt.plot(range(50, 10000, 50), mean, label='mean')
49
```

```
plot_intervals(norm_interval)
plot_intervals(percentile_interval)
plot_intervals(pivot_interval)
plot_mean()

plt.legend()
plt.show()
```

8.6

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

def mu_hat(X):
      return np.mean(X)

def F_hat(X):
      return np.exp(mu_hat(X))

def bootstrap(real_X, B=10000):
      F_hat_ = np.zeros(B)
      for b in range(B):
            X = np.random.choice(real_X, len(real_X), replace=True)
            F_hat_[b] = F_hat(X)
      return F_hat_

def norm_interval(F_hat_, alpha=0.05):
      se = np.std(F_hat_)
      mean = np.mean(F_hat_)
      z = norm.ppf(1-alpha/2)
      return mean - z*se, mean + z*se

def percentile_interval(F_hat, alpha=0.05):
      return np.percentile(F_hat, 100*alpha/2), np.percentile(F_hat, 100*(1-
alpha/2))

def pivot_interval(F_hat_, alpha=0.05):
      return 2*np.mean(F_hat_) - np.percentile(F_hat_, 100*(1-alpha/2)),
2*np.mean(F_hat_) - np.percentile(F_hat_, 100*alpha/2)

real_X = np.random.normal(5, 1, (100,))
F_hat_ = bootstrap(real_X)
print(np.std(F_hat_))
print(np.mean(F_hat_))
print(norm_interval(F_hat_))
print(percentile_interval(F_hat_))
```

```
print(pivot_interval(F_hat_))

def plot_intervals(F_hat_, interval_func: callable):
      interval_upper = []
      interval_lower = []
      for end in range(50, 10000, 50):
            upper, lower = interval_func(F_hat_[:end])
            interval_upper.append(upper)
            interval_lower.append(lower)
      plt.fill_between(range(50, 10000, 50), interval_upper, interval_lower,
alpha=0.3, label=interval_func.__name__)

def plot_mean(F_hat_):
      mean = []
      for end in range(50, 10000, 50):
            mean.append(np.mean(F_hat_[:end]))
      plt.plot(range(50, 10000, 50), mean, label='mean')

def plot_se(F_hat_):
      se = []
      for end in range(50, 10000, 50):
            se.append(np.std(F_hat_[:end]))
      plt.plot(range(50, 10000, 50), se, label='se')

plot_intervals(F_hat_, norm_interval)
plot_intervals(F_hat_, percentile_interval)
plot_intervals(F_hat_, pivot_interval)
plot_mean(F_hat_)

plt.legend()
plt.show()

plt.hist(F_hat_, bins=50, alpha=0.3, label='bootstrap')
plt.hist(
      np.exp(np.random.normal(5, 1, (100,10000)).mean(axis=0)),
      bins=50, alpha=0.3, label='real')
plt.legend()
plt.show()
```

8.7

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

def F_hat(X):
      return np.max(X)
```

```
7
8  def bootstrap(real_X, B=10000):
9        F_hat_ = np.zeros(B)
10       for b in range(B):
11             X = np.random.choice(real_X, len(real_X), replace=True)
12             F_hat_[b] = F_hat(X)
13       return F_hat_
14
15 def norm_interval(F_hat_, alpha=0.05):
16       se = np.std(F_hat_)
17       mean = np.mean(F_hat_)
18       z = norm.ppf(1-alpha/2)
19       return mean - z*se, mean + z*se
20
21 def percentile_interval(F_hat, alpha=0.05):
22       return np.percentile(F_hat, 100*alpha/2), np.percentile(F_hat, 100*(1-
   alpha/2))
23
24 def pivot_interval(F_hat_, alpha=0.05):
25       return 2*np.mean(F_hat_) - np.percentile(F_hat_, 100*(1-alpha/2)),
   2*np.mean(F_hat_) - np.percentile(F_hat_, 100*alpha/2)
26
27 real_X = np.random.uniform(0, 1, (50,))
28 F_hat_ = bootstrap(real_X)
29 print(np.std(F_hat_))
30 print(np.mean(F_hat_))
31 print(norm_interval(F_hat_))
32 print(percentile_interval(F_hat_))
33 print(pivot_interval(F_hat_))
34
35 def plot_intervals(F_hat_, interval_func: callable):
36       interval_upper = []
37       interval_lower = []
38       for end in range(50, 10000, 50):
39             upper, lower = interval_func(F_hat_[:end])
40             interval_upper.append(upper)
41             interval_lower.append(lower)
42       plt.fill_between(range(50, 10000, 50), interval_upper, interval_lower,
   alpha=0.3, label=interval_func.__name__)
43
44 def plot_mean(F_hat_):
45       mean = []
46       for end in range(50, 10000, 50):
47             mean.append(np.mean(F_hat_[:end]))
48       plt.plot(range(50, 10000, 50), mean, label='mean')
49
50 def plot_se(F_hat_):
```

```
      se = []
      for end in range(50, 10000, 50):
            se.append(np.std(F_hat_[:end]))
      plt.plot(range(50, 10000, 50), se, label='se')

plot_intervals(F_hat_, norm_interval)
plot_intervals(F_hat_, percentile_interval)
plot_intervals(F_hat_, pivot_interval)
plot_mean(F_hat_)

plt.legend()
plt.show()

plt.hist(F_hat_, bins=50, range=(0.5,1), alpha=0.3, label='bootstrap')
plt.hist(
      np.random.uniform(0, 1, (50,10000)).max(axis=0),
      bins=50, range=(0.5,1) , alpha=0.3, label='real')
plt.legend()
plt.show()
```